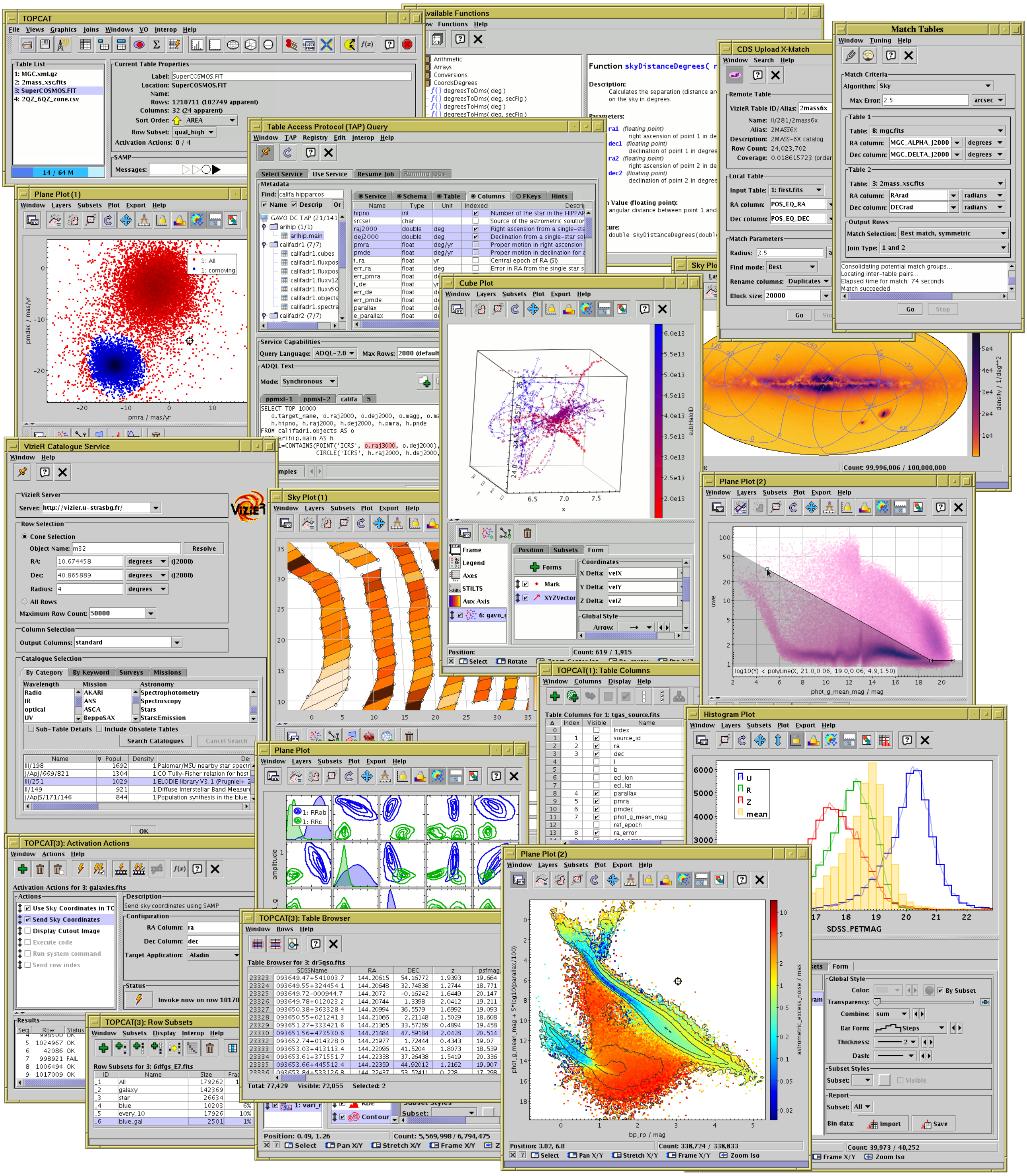
TOPCAT - Tool for OPerations on Catalogues And Tables

Version 4.10-3



Starlink User Note253
Mark Taylor
7 March 2025

Abstract

TOPCAT is an interactive graphical viewer and editor for tabular data. It has been designed for use with astronomical tables such as object catalogues, but is not restricted to astronomical applications. It understands a number of different astronomically important formats, and more formats can be added. It is designed to cope well with large tables; a million rows by a hundred columns should not present a problem even with modest memory and CPU resources.

It offers a variety of ways to view and analyse the data, including a browser for the cell data themselves, viewers for information about table and column metadata, tools for joining tables using flexible matching algorithms, and extensive 2- and 3-d visualisation facilities. Using a powerful and extensible Java-based expression language new columns can be defined and row subsets selected for separate analysis. Selecting a row can be configured to trigger an action, for instance displaying an image of the catalogue object in an external viewer. Table data and metadata can be edited and the resulting modified table can be written out in a wide range of output formats.

A number of options are provided for loading data from external sources, including Virtual Observatory (VO) services, thus providing a gateway to many remote archives of astronomical data. It can also interoperate with other desktop tools using the SAMP protocol.

TOPCAT is written in pure Java (except for a few optional libraries) and is available under the GNU General Public Licence. Its underlying table processing facilities are provided by STIL, the Starlink Tables Infrastructure Library.

Contents

Abstract.......... 1
1 Introduction.......... 11
2 Quick Start Guide.......... 13
3 Apparent Table.......... 15
3.1 Row Subsets.......... 15
3.1.1 Defining Subsets.......... 16
3.2 Row Order.......... 17
3.3 Column Set.......... 17
4 Table I/O.......... 19
4.1 Table Formats.......... 19
4.1.1 Input Formats.......... 19
4.1.1.1 FITS.......... 19
4.1.1.2 Column-oriented FITS.......... 20
4.1.1.3 VOTable.......... 21
4.1.1.4 CDF.......... 22
4.1.1.5 CSV.......... 22
4.1.1.6 ECSV.......... 23
4.1.1.7 ASCII.......... 25
4.1.1.8 IPAC.......... 26
4.1.1.9 PDS4.......... 27
4.1.1.10 MRT.......... 28
4.1.1.11 Parquet.......... 29
4.1.1.12 HAPI.......... 30
4.1.1.13 Feather.......... 31
4.1.1.14 GBIN.......... 31
4.1.1.15 TST.......... 33
4.1.1.16 World Data Center.......... 33
4.1.2 Output Formats.......... 33
4.1.2.1 FITS.......... 34
4.1.2.2 VOTable.......... 36
4.1.2.3 CSV.......... 37
4.1.2.4 ECSV.......... 37
4.1.2.5 ASCII.......... 38
4.1.2.6 IPAC.......... 39
4.1.2.7 Parquet.......... 40
4.1.2.8 Feather.......... 42
4.1.2.9 Text.......... 42

4.1.2.10 HTML 43
4.1.2.11 LaTeX 43
4.1.2.12 Tab-Separated Table 44
4.1.2.13 Mirage Format 44
4.1.3 Non-standard FITS conventions 45
4.1.3.1 FITS-plus 45
4.1.3.2 Wide FITS 46
4.1.4 Custom I/O Formats 48
4.2 Input Locations 48
4.3 Input Schemes 50
4.3.1 `skysim` 50
4.3.2 `attractor` 50
4.3.3 `jdbc` 52
4.3.4 `loop` 52
4.3.5 `test` 53
4.3.6 `class` 54
4.3.7 `hapi` 54
5 Joins and Matches 56
5.1 Concatenating Tables 56
5.2 Matching Rows Between Tables 56
5.3 Matching Rows Within a Table 58
5.4 Multi-Object Matches 58
5.5 Plotting Match Results 58
5.6 Notes on Matching 61
5.7 Matching against a Remote Table 61
6 Virtual Observatory Access 62
6.1 The Registry 62
6.2 Data Access Services 62
6.3 Authentication 63
7 Algebraic Expression Syntax 65
7.1 Referencing Cell Values 65
7.2 Referencing Row Subset Flags 66
7.3 Referencing Table Parameters 67
7.4 Special Tokens 67
7.5 Null Values 68
7.6 Operators 69
7.7 Functions 70
7.7.1 Technical Note 71
7.8 Instance Methods 71
7.9 Examples 72
7.10 Adding User-Defined Functions 74
8 Activation Actions 76
9 Tool Interoperability 77
9.1 SAMP control 77
9.2 PLASTIC control 78
9.3 Messages Transmitted 79
9.4 Messages Received 80
10 Invoking TOPCAT 83
10.1 TOPCAT Command-line Arguments 83
10.2 Java Options 86
10.2.1 Class Path 86
10.2.2 Memory Size 87
10.2.3 System properties 87
10.3 JDBC Configuration 90
10.4 Tips for Large Tables 92

10.5 Examples........ 93
Appendix A: TOPCAT Windows........ 95
A.1 Common Window Features........ 95
A.1.1 Toolbar........ 95
A.1.2 Menus........ 95
A.1.3 JTables........ 97
A.1.4 Column Selector........ 98
A.1.5 STILTS Window........ 98
A.2 Control Window........ 101
A.2.1 Table List........ 102
A.2.2 Current Table Properties panel........ 102
A.2.3 Memory Monitor........ 103
A.2.4 SAMP Panel........ 104
A.2.5 Toolbar Buttons........ 104
A.2.6 Menu Items........ 106
A.3 Table View Windows........ 110
A.3.1 Data Window........ 110
A.3.2 Parameters Window........ 112
A.3.3 Columns Window........ 114
A.3.4 Subsets Window........ 118
A.3.5 Statistics Window........ 121
A.3.6 DataLink Window........ 125
A.4 Plot Windows........ 127
A.4.1 Differences From Old-Style Plot Windows........ 128
A.4.2 Plot Window Overview........ 129
A.4.2.1 Navigation........ 134
A.4.2.2 Table Data Layer Controls........ 135
A.4.2.3 Defining Subsets Graphically........ 138
A.4.2.4 Distance Measurement........ 144
A.4.2.5 Plot Export Window........ 146
A.4.3 Fixed Controls........ 148
A.4.3.1 Frame Control........ 148
A.4.3.2 Legend Control........ 150
A.4.3.3 Axes Control........ 151
A.4.3.4 Stilts Control........ 151
A.4.3.5 Aux Axis Control........ 155
A.4.4 Layer Controls........ 158
A.4.4.1 Position Layer Control........ 158
A.4.4.2 Pair Position Layer Control........ 160
A.4.4.3 Quad Position Layer Control........ 162
A.4.4.4 Matrix Layer Control........ 165
A.4.4.5 Healpix Layer Control........ 169
A.4.4.6 Histogram Layer Control........ 173
A.4.4.7 Area Layer Control........ 175
A.4.4.8 Spectrogram Layer Control........ 179
A.4.4.9 XYArray Layer Control........ 181
A.4.4.10 Function Layer Control........ 184
A.4.4.11 SkyGrid Layer Control........ 187
A.4.4.12 SphereGrid Layer Control........ 189
A.4.5 Plot Forms........ 191
A.4.5.1 Mark Form........ 191
A.4.5.2 Size Form........ 192
A.4.5.3 SizeXY Form........ 194
A.4.5.4 Vector Form........ 196
A.4.5.5 SkyVector Form........ 198

A.4.5.6 Error Bars Form.. 200
A.4.5.7 XYEllipse Form.. 202
A.4.5.8 SkyEllipse Form.. 204
A.4.5.9 XYCorr Form.. 206
A.4.5.10 SkyCorr Form.. 208
A.4.5.11 Polygon Form.. 210
A.4.5.12 Line Form.. 211
A.4.5.13 Line3d Form.. 213
A.4.5.14 Linear Fit Form.. 214
A.4.5.15 Quantile Form.. 217
A.4.5.16 Label Form.. 219
A.4.5.17 Contour Form.. 221
A.4.5.18 Grid Form.. 223
A.4.5.19 SkyDensity Form.. 226
A.4.5.20 Fill Form.. 229
A.4.5.21 Histogram Form.. 230
A.4.5.22 KDE Form.. 233
A.4.5.23 KNN Form.. 235
A.4.5.24 Densogram Form.. 238
A.4.5.25 Gaussian Form.. 241
A.4.5.26 Link2 Form.. 243
A.4.5.27 Mark2 Form.. 244
A.4.5.28 Poly4 Form.. 245
A.4.5.29 Mark4 Form.. 247
A.4.5.30 Area Form.. 248
A.4.5.31 Central Form.. 250
A.4.5.32 AreaLabel Form.. 251
A.4.5.33 Lines Form.. 254
A.4.5.34 Marks Form.. 255
A.4.5.35 Handles Form.. 256
A.4.5.36 YErrors Form.. 257
A.4.5.37 XYErrors Form.. 259
A.4.5.38 StatLine Form.. 260
A.4.5.39 StatMark Form.. 262
A.4.5.40 ArrayQuantile Form.. 263
A.4.6 Shading Modes.. 265
A.4.6.1 Flat Mode.. 266
A.4.6.2 Translucent Mode.. 266
A.4.6.3 Transparent Mode.. 267
A.4.6.4 Auto Mode.. 268
A.4.6.5 Density Mode.. 269
A.4.6.6 Aux Mode.. 271
A.4.6.7 Weighted Mode.. 273
A.4.6.8 PAux Mode.. 274
A.4.6.9 PWeighted Mode.. 277
A.4.7 Colour Maps.. 279
A.4.8 Histogram Plot Window.. 280
A.4.8.1 Histogram Navigation.. 282
A.4.8.2 Histogram Axes Control.. 283
A.4.8.3 Bins Control.. 283
A.4.9 Plane Plot Window.. 286
A.4.9.1 Plane Navigation.. 289
A.4.9.2 Plane Axes Control.. 289
A.4.10 Sky Plot Window.. 294
A.4.10.1 Sky Navigation.. 297

A.4.10.2 Sky Axes Control.... 298
A.4.11 Cube Plot Window.... 301
A.4.11.1 Cube Navigation.... 304
A.4.11.2 Cube Axes Control.... 305
A.4.12 Sphere Plot Window.... 310
A.4.12.1 Sphere Navigation.... 313
A.4.12.2 Sphere Axes Control.... 313
A.4.13 Corner Plot Window.... 316
A.4.13.1 Corner Navigation.... 319
A.4.13.2 Corner Axes Control.... 320
A.4.14 Time Plot Window.... 324
A.4.14.1 Plot Zones.... 327
A.4.14.2 Time Navigation.... 328
A.4.14.3 Time Axes Control.... 328
A.5 Old-Style Plot Windows.... 333
A.5.1 Common Features.... 334
A.5.1.1 Dataset Selectors.... 334
A.5.1.2 Axis Configuration and Zooming.... 336
A.5.1.3 Error Bars.... 338
A.5.1.4 Point Labels.... 340
A.5.1.5 Auxiliary Axes.... 342
A.5.1.6 Defining Subsets by Region.... 344
A.5.1.7 Exporting Graphics.... 346
A.5.2 Histogram (old-style).... 347
A.5.2.1 Histogram Style Editor.... 350
A.5.3 2D Plot (old-style).... 351
A.5.3.1 Plot Style Editor.... 354
A.5.4 Stacked Line Plot (old-style).... 357
A.5.4.1 Lines Style Editor.... 361
A.5.5 3D Plot (old-style).... 362
A.5.5.1 3D Plot Style Editor.... 366
A.5.6 Spherical Plot (old-style).... 368
A.5.7 Density Map (old-style).... 372
A.5.7.1 Density Style Editor.... 376
A.6 Load Window.... 377
A.6.1 Filestore Browser.... 379
A.6.2 System Browser.... 381
A.6.3 Hierarchy Browser.... 381
A.6.3.1 Navigation.... 383
A.6.3.2 Table Searches.... 384
A.6.4 SQL Query.... 384
A.6.5 Cone Search.... 385
A.6.6 SIA Query.... 385
A.6.7 SSA Query.... 385
A.6.8 TAP Query.... 385
A.6.9 VizieR Catalogue Service Query.... 385
A.6.10 HAPI Query.... 390
A.6.11 Virgo-Millennium Simulation Query.... 393
A.6.12 BaSTI Theory Database Query.... 395
A.6.13 Example Tables.... 397
A.7 Save Window.... 397
A.7.1 Content Panel.... 399
A.7.1.1 Current Table.... 399
A.7.1.2 Multiple Tables.... 399
A.7.1.3 Session.... 399

A.7.2 Destination Panel.......... 400
A.7.2.1 Enter Location.......... 401
A.7.2.2 Filestore Browser.......... 401
A.7.2.3 System Browser.......... 402
A.7.2.4 SQL Output Dialogue.......... 402
A.8 Match Windows.......... 403
A.8.1 Common Features.......... 403
A.8.1.1 Match Criteria.......... 403
A.8.1.2 Column Selection Boxes.......... 406
A.8.1.3 Tuning.......... 407
A.8.2 Pair Match Window.......... 408
A.8.2.1 Output Rows Selector Box.......... 410
A.8.3 Internal Match Window.......... 413
A.8.3.1 Internal Match Action Box.......... 415
A.8.4 Multiple Match Window.......... 416
A.8.4.1 Multiple Match Action Box.......... 417
A.9 VO Data Access Windows.......... 417
A.9.1 Common Features.......... 418
A.9.1.1 Registry Query Panel.......... 418
A.9.1.2 Single Positional Search Panel.......... 420
A.9.1.3 Multiple Positional Search Panel.......... 421
A.9.2 Cone Search.......... 423
A.9.3 Simple Image Access (SIA) Query.......... 425
A.9.4 Simple Spectral Access (SSA) Query.......... 426
A.9.5 Multiple Cone Search.......... 428
A.9.6 Multiple SIA Query.......... 429
A.9.7 Multiple SSA Query.......... 431
A.9.8 Table Access Protocol (TAP) Query.......... 433
A.9.8.1 Select Service tab.......... 436
A.9.8.2 Use Service tab.......... 438
A.9.8.3 Running Jobs tab.......... 446
A.9.8.4 Resume Job tab.......... 449
A.10 Activation Window.......... 450
A.10.1 Activation Actions.......... 453
A.10.1.1 Use Sky Coordinates in TOPCAT.......... 453
A.10.1.2 Send Sky Coordinates.......... 453
A.10.1.3 Display HiPS Cutout.......... 454
A.10.1.4 Send HiPS Cutout.......... 456
A.10.1.5 Display Image.......... 457
A.10.1.6 Display Image Region.......... 458
A.10.1.7 Send FITS Image.......... 459
A.10.1.8 Send Spectrum.......... 459
A.10.1.9 Send VOTable.......... 460
A.10.1.10 Send Row Index.......... 460
A.10.1.11 Load Table.......... 461
A.10.1.12 Plot Table.......... 462
A.10.1.13 View in Web Browser.......... 463
A.10.1.14 Download URL.......... 465
A.10.1.15 Delay.......... 466
A.10.1.16 Execute Code.......... 466
A.10.1.17 Run System Command.......... 468
A.10.1.18 SAMP Message.......... 469
A.10.1.19 Display Cutout Image.......... 470
A.10.1.20 Invoke Service.......... 471
A.10.1.21 Invoke Datalink Row.......... 472

A.10.1.22 View Datalink Table.... 473
A.10.2 Viewer Windows.... 475
A.10.2.1 Image Viewer Applications.... 475
A.10.2.2 Spectrum Viewers.... 476
A.10.2.3 Web Browsers.... 476
A.10.3 Activation Security.... 477
A.11 Other Windows.... 479
A.11.1 CDS Upload X-Match Window.... 479
A.11.2 Concatenation Window.... 483
A.11.3 SAMP Window.... 484
A.11.4 Help Window.... 487
A.11.5 Column Search Window.... 488
A.11.6 New Parameter Window.... 490
A.11.7 Synthetic Column Window.... 491
A.11.8 Sky Coordinates Window.... 492
A.11.9 Algebraic Subset Window.... 493
A.11.10 Multi Algebraic Subset Window.... 494
A.11.11 Column Classification Window.... 495
A.11.12 Available Functions Window.... 498
A.11.13 Log Window.... 499
Appendix B: Algebraic Functions.... 502
B.1 General Functions.... 502
B.1.1 Arithmetic.... 502
B.1.2 Arrays.... 506
B.1.3 Bits.... 515
B.1.4 Conversions.... 517
B.1.5 CoordsDegrees.... 522
B.1.6 CoordsRadians.... 525
B.1.7 Coverage.... 530
B.1.8 Distances.... 532
B.1.9 Fluxes.... 535
B.1.10 Formats.... 538
B.1.11 Gaia.... 539
B.1.12 Json.... 548
B.1.13 KCorrections.... 551
B.1.14 Lists.... 555
B.1.15 Maths.... 557
B.1.16 Randoms.... 561
B.1.17 Shapes.... 563
B.1.18 Sky.... 564
B.1.19 Strings.... 566
B.1.20 Tilings.... 573
B.1.21 Times.... 577
B.1.22 TrigDegrees.... 584
B.1.23 URLs.... 585
B.1.24 VO.... 589
B.2 Activation Functions.... 591
B.2.1 BasicImageDisplay.... 591
B.2.2 Browsers.... 591
B.2.3 Image.... 592
B.2.4 Mgc.... 593
B.2.5 Output.... 593
B.2.6 Sdss.... 594
B.2.7 SuperCosmos.... 595
B.2.8 System.... 597

B.2.9 TwoQZ.. 598
Appendix C: Release Notes.. 600
C.1 Acknowledgements... 600
C.2 Version History.. 603

1 Introduction

TOPCAT is an interactive graphical program which can examine, analyse, combine, edit and write out tables. A table is, roughly, something with columns and rows; each column contains objects of the same type (for instance floating point numbers) and each row has an entry for each of the columns (though some entries might be blank). A common astronomical example of a table is an object catalogue.

TOPCAT can read in tables in a number of formats from various sources, allow you to inspect and manipulate them in various ways, and if you have edited them optionally write them out in the modified state for later use, again in a variety of formats. Here is a summary of its main capabilities:

- View/edit table data in a scrollable browser
- View/edit table metadata (parameters)
- View/edit column metadata (column names, units, UCDs...)
- Re-order and hide/reveal columns
- Insert 'synthetic' columns defined by algebraic expression
- Sort rows on the values in given columns or expressions
- Define row subsets in various ways, including algebraically and graphically
- Plot columns against each other in 1, 2 and 3 dimensions, distinguishing different subsets
- Calculate statistics on each column for some or all rows
- Trigger a configurable action (e.g. object image display) when a column is selected
- Perform flexible matching of rows in the same or different tables
- Concatenate the rows of existing tables to create new ones
- Acquire tables from web services, external filestores or other customisable sources
- Perform single and multiple queries to Virtual Observatory services including TAP services
- Interoperate with other desktop tools using SAMP
- Write modified tables out in original or different format

Considerable effort has gone into making it work with large tables; a few million rows and hundreds of columns is usually quite manageable.

The general idea of the program is quite straightforward. At any time, it has a list of tables it knows about - these are displayed in the Control Window which is the first thing you see when you start up the program. You can add to the list by loading tables in, or by some actions which create new tables from the existing ones. When you select a table in the list by clicking on it, you can see general information about it in the control window, and you can also open more specialised view windows which allow you to inspect it in more detail or edit it. Some of the actions you can take, such as changing the current Sort Order, Row Subset or Column Set change the Apparent Table (Section 3), which is a view of the table used for things such as saving it and performing row matches. Changes that you make do not directly modify the tables on disk (or wherever they came from), but if you want to save the changes you have made, you can write the modified table(s) to a new location.

The main body of this document explains these ideas and capabilities in more detail, and Appendix A gives a full description of all the windows which form the application. While the program is running, this document is available via the online help system - clicking the **Help** (?) toolbar button in any window will pop up a help browser open at the page which describes that window. This document is heavily hyperlinked, so you may find it easier to read in its HTML form than on paper.

Recent news about the program can be found on the TOPCAT web page (http://www.starlink.ac.uk/topcat/). It was initially developed within the now-terminated Starlink and then AstroGrid projects, and has subsequently been supported by the UK's PPARC and STFC research councils, various Euro-VO and FP7 projects, GAVO and ESA. The underlying table handling facilities are supplied by the Starlink Tables Infrastructure Library STIL (http://www.starlink.ac.uk/stil/), which is documented more fully in SUN/252. The software is written in pure Java (except for some compression codecs used by the Parquet I/O handlers), and should run on any J2SE platform version 1.8 or later. This makes it highly portable, since it can run on any machine which has a suitable Java installation, which is available for MS Windows, Mac OS X and most flavours of Unix amongst others. Some of the external viewer applications it talks to rely on non-Java code however so one or two facilities, such as displaying spectra, may be absent in some cases.

The TOPCAT application is available under the terms of the GNU General Public License, since some of the libraries it uses are also GPL. However, all of the original code and many of the libraries it uses may alternatively be used under more permissive licenses such as the GNU Lesser General Public License, see documentation of the STILTS package for more details.

2 Quick Start Guide

This manual aims to give detailed tutorial and reference documentation on most aspects of TOPCAT's capabilities, and reading it is an excellent way to learn about the program. However, it's quite a fat document, and if you feel you've got better things to do with your time than read it all, you should be able to do most things by playing around with the software and dipping into the manual (or equivalently the online help) when you can't see how to do something or the program isn't behaving as expected. This section provides a short introduction for the impatient, explaining how to get started.

To start the program, you will probably type `topcat` or something like `java -jar topcat-full.jar` (see Section 10 for more detail). To view a table that you have on disk, you can either give its name on the command line or load it using the **Load** button from the GUI. FITS, VOTable, ECSV, CDF, PDS4, feather, Parquet and GBIN files are recognised automatically; if your data is in another format such as ASCII (see Section 4.1.1) you need to tell the program (e.g. `-f ascii` on the command line). If you just want to try the program out, `topcat -demo` will start with a couple of small tables for demonstration purposes.

The first thing that you see is the Control Window (Appendix A.2). This has a list of the loaded table(s) on the left. If one of these is highlighted by clicking on it, information about it will be shown on the right; some of this (table name, sort order) you can change here. Along the top is a toolbar with a number of buttons, most of which open up new windows. These correspond to some of the things you might most often want to do in TOPCAT, and fall into a few groups:

Load/Save/Send Table(s).

Display various aspects of information about the table's data and metadata.

Open plotting/visualisation windows of various kinds.

Join tables in various ways including spatial crossmatching, and access remote databases.

Help and information

Exit

The menus provide alternative ways to open up these windows, and also list a number of other, less commonly-used, options. The **Help** (?) button appears in most windows - if you click it a help browser will be displayed showing an appropriate part of this manual. The **Help** menu gives you a few more options along the same lines, including displaying the help information in your usual web browser rather than in TOPCAT's (somewhat scrappy) help viewer. All the windows follow roughly this pattern. For some of the toolbar buttons you can probably guess what they do from their icons, for others probably not - to find out you can hover with the mouse to see the tooltip, look in the menus, read the manual, or just push it and see.

Some of the windows allow you to make changes of various sorts to the tables, such as performing sorts, selecting rows, modifying data or metadata. None of these affect the table on disk (or database, or wherever), but if you subsequently save the table the changes will be reflected in the table that you save.

A notable point to bear in mind concerns memory. TOPCAT is fairly efficient in use of memory, but in some cases when dealing with large tables you might see an OutOfMemoryError. It is usually possible to work round this by using the `-Xmx`*NNN*`M` flag on startup - see Section 10.2.2.

Finally, if you have queries, comments or requests about the software, and they don't appear to be addressed in the manual, consult the TOPCAT web page, use the topcat-user mailing list, or contact the author - user feedback is always welcome.

3 Apparent Table

The **Apparent Table** is a particular view of a table which can be influenced by some of the viewing controls.

When you load a table into TOPCAT it has a number of characteristics like the number of columns and rows it contains, the order of the rows that make up the data, the data and metadata themselves, and so on. While manipulating it you can modify the way that the table appears to the program, by changing or adding data or metadata, or changing the order or selection of columns or rows that are visible. For each table its "apparent table" is a table which corresponds to the current state of the table according to the changes that you have made.

In detail, the apparent table consists of the table as it was originally imported into the program plus any of the following changes that you have made:

- Selection of rows changed by changing the current Row Subset (Section 3.1)
- Changes to the current Row Order (Section 3.2) caused by doing a sort
- Changes to the current Column Set (Section 3.3) caused by adding, hiding or moving columns
- Changes to cell data by editing cells in the Data window
- Changes to table metadata by editing cells in the Parameter window
- Changes to column metadata by editing cells in the Columns window

The apparent table is used in the following contexts:

Data Window

The Data window always shows the rows and columns of the apparent table, so if you are in doubt about what form a table will get exported in, you can see what it looks like there.

Exports

When you save a table, or export it by dragging it off the Table List panel in the Control Window, or create a duplicate table, it is the apparent table which is copied. So for instance if you define a subset containing only the first ten rows of a table and then save it to a new table, or create a duplicate within TOPCAT using the **Duplicate Table** ( ) toolbar button, the resulting table will contain only those ten rows.

Joins

When you use the Match Window or Concatenation Window to construct a new table on the basis of one or more existing ones, the new table will be built on the basis of the apparent versions of the tables being operated on. The same applies to the join-like functionality provided by table uploads in the TAP window, CDS Upload X-Match window and the multiple positional search (Cone, SIA, SSA) windows.

Some of the other table view windows are affected too, for instance the Columns window displays its columns in the order that they appear in the Apparent Table.

3.1 Row Subsets

An important feature of TOPCAT is the ability to define and use **Row Subsets**. A Row Subset is a selection of the rows within a whole table being viewed within the application, or equivalently a new table composed from some subset of its rows. You can define these and use them in several different ways; the usefulness comes from defining them in one context and using them in another. The Subset Window displays the currently defined Row Subsets and permits some operations on them.

At any time each table has a **current** row subset, and this affects the Apparent Table. You can always see what it is by looking at the "Row Subset" selector in the Control Window when that table is selected; by default it is one containing all the rows. You can change it by choosing from this selector or as a result of some other actions.

Other contexts in which subsets can be used are picking a selection of rows from which to calculate in the Statistics Window and marking groups of rows to plot using different markers in the various plotting (and old-style plotting) windows.

Tables always have two special subsets:

- **All**: contains all the rows in the table
- **Activated**: contains a single row if one has been activated, e.g. by clicking on a table row or plotted point representing it

Other subsets can be defined by user actions as described below.

3.1.1 Defining Subsets

You can define a Row Subset in one of the following ways:

Defining an algebraic expression

From the Subset Window using the **Add New Subset** (✚) button will pop up the Algebraic Subset Window which allows you to define a new subset using an algebraic expression based on the values of the cells in each row. The format of such expressions is described in Section 7. The Subsets Window also provides some variants on this option for convenience, like selecting the first *N* (✚), last *N* (✚), or every *N* th (✚) rows, or the complement () of an existing subset.

Graphical selection

There are several ways to indicate a region graphically in the plotting area of the plotting windows. which can be used to define subsets. The options are **Subset From Visible** (), **Algebraic Subset From Visible** (), **Draw Subset Blob** () and **Draw Algebraic Subset** (), though not all are available for all plot types.

Classifying by value

The Column Classification Window lets you define multiple mutually exclusive subsets based on the value in a given column (or other algebraic expression).

Boolean columns

Any column which has a boolean (true/false) type value can be used as a subset; rows in which it has a true value are in the subset and others are not. Any boolean column in a table is made available as a row subset with the same name when the table is imported.

Selecting rows in the browser

You can select a single row in the Data Window by clicking on it, or select a group of adjacent rows by dragging the mouse over them. You can add more rows to the selection by keeping the <Control> button pressed while you do it. Once you have a set of rows selected you can use the **Subset From Selected Rows** () or **Subset From Unselected Rows** () buttons to create a new subset based on the set of highlighted rows or their complement. Combining this with sorting the rows in the table can be useful; if you do a Sort Up on a given column and then drag out the top few rows of the table you can easily create a subset consisting of the highest values of a given column.

In all these cases you will be asked to assign a name for the subset. As with column names, it is a good idea to follow a few rules for these names so that they can be used in algebraic expressions. They should be:

- Distinct from other subset and column names, even apart from upper/lower case distinctions
- In the form of a java identifier (starts with a letter, continues with alphanumerics including underscore, no spaces)
- Not too long

When you choose a name, you can either type one in, or select one from the drop-down list, which gives the names of all the existing subsets. This allows you to redefine existing subsets. Note if you do select or type in one of the existing names, any previous content of that subset will be lost.

In the first subset definition method above, the **current** subset will be set immediately to the newly created one. In other cases the new subset may be highlighted appropriately in other windows, for instance by being plotted in scatter plot windows.

3.2 Row Order

You can sort the rows of each table according to the values in the table. Normally you will want to sort on a numeric column, but other values may be sortable too, for instance a String column will sort alphabetically. Some kinds of columns (e.g. array ones) don't have any well-defined order, and it is not possible to select these for sorting on.

At any time, each table has a **current** row order, and this affects the Apparent Table. You can always see what it is by looking under the **Sort Order** item in the Control Window when that table is selected; by default it is blank, which means the rows have the same order as that of the table they were loaded in from. The little arrow (⇧/⇩) indicates whether the sense of the sort is up or down. You can change the sort order by selecting a column name or entering an algebraic expression in this selector, and change the sense by clicking on the arrow. To sort on multiple columns, use the **Add/Remove Selector** (➕ ➖) buttons to change the number of selectors; selectors to the left are more significant, and ones to the right are used in case of a tie in earlier values. The sort order can also be changed by using menu items in the Columns Window or right-clicking popup menus in the Data Window.

Selecting values to sort by calculates the new row order by performing a sort on the cell values there and then. If the table data change somehow (e.g. because you edit cells in the table) then it is possible for the sort order to become out of date. If that happens you can resort by making a new selection and then changing it back again.

The current row order affects the Apparent Table, and hence determines the order of rows in tables which are exported in any way (e.g. written out) from TOPCAT. You can always see the rows in their currently sorted order in the Data Window.

3.3 Column Set

When each table is imported it has a list of columns. Each column has header information which determines the kind of data which can fill the cells of that column as well as a name, and maybe some additional information like units and Unified Content Descriptor. All this information can be viewed, and in some cases modified, in the Columns Window.

During the lifetime of the table within TOPCAT, this list of columns can be changed by adding new columns, hiding (and perhaps subsequently revealing) existing columns, and changing their order.

The current state of which columns are present and visible and what order they are in is collectively known as the **Column Set**, and affects the Apparent Table. The current Column Set is always reflected in the columns displayed in the Data Window and Statistics Window. The Columns Window shows all the known columns, including hidden ones; whether they are currently visible is indicated by the checkbox in the **Visible** column. By default, the **Columns Window** and **Statistics Window** also reflect the current column set order, though there are options to change this view.

You can affect the current Column Set in the following ways:

Hide/Reveal columns

In the Columns Window you can toggle columns between hidden and visible by clicking on their checkbox in the **Visible** column. To make a group of columns hidden or visible at once, select the corresponding rows (drag the mouse over them to select a contiguous group; hold the Control button down to add more single rows or contiguous groups to the selection) and hit the **Hide Selected** (□) or **Reveal Selected** (☑) button in the toolbar or menu. Note when selecting rows, don't drag the mouse over the Visible column, do it somewhere in the middle of the table. The **Hide All** (□□□) and **Reveal All (☑☑☑)** buttons set all columns in the table invisible or visible - a useful convenience if you've got a very wide table.

You can also hide a column by right-clicking on it in the Data Window, which brings up a popup menu - select the **Hide** option. To make it visible again you have to go to the Columns Window as above.

Move Columns

In the Data Window you can move columns around by dragging the grey column header left or right to a new position (as usual in a JTable). Alternatively, you can drag the rows in the Columns Window by grabbing the grey row header (numbered cell at the left) and dragging it up or down (though note that may not work if the JTable is currently sorted). Either of these affects the Column Set, as you can see by looking at one window while moving columns in the other.

Add Columns

You can use the **New Synthetic Column** (✚) or **New Sky Coordinate Columns** (🌐) buttons in the Columns Window or the (right-click) popup menu in the Data Window to add new columns derived from exsiting ones.

Replace a Column

If a column is selected in the Columns Window or from the Data Window popup menu you can use the **Replace Column with Synthetic** (↪) button. This is similar to the **Add a Synthetic Column** described in the previous item, but it pops up a new column dialogue with similar characteristics (name, units etc) to those of the column that's being replaced, and when completed it slots the new column in to the table hiding the old one.

Add a Subset Column

If you have defined a Row Subset somehow and you want it to appear explicitly in the table (for instance so that when you write the table out the selection is saved) you can select that subset in the Subsets Window and use the **To Column** (▦) button, which will add a new boolean column to the table with the value **true** for rows part of that subset and **false** for the other rows.

4 Table I/O

Tables can be loaded into TOPCAT using the Load Window or from the command line, or acquired from VO services, and saved using the Save Window. This section describes the file formats supported for input and output, as well as the syntax to use when specifying a table by name, either as a file/URL or using a *scheme specification*.

4.1 Table Formats

TOPCAT supports a number of different serialization formats for table data; some have better facilities for storing table data and metadata than others.

Since you can load a table from one format and save it in a different one, TOPCAT can be used to convert a table from one format to another. If this is all you want to do however, you may find it more convenient to use the `tcopy` or `tpipe` command line utilities in the STILTS package.

The following subsections describe the available formats for reading and writing tables. The two operations are separate, so not all the supported input formats have matching output formats and vice versa.

4.1.1 Input Formats

Loading table into TOPCAT from files or URLs is done either using the Load Table dialogue or one of its sub-windows, or from the command line when you start the program. For some file formats (e.g. FITS, VOTable, CDF), the format can be automatically determined by looking at the file content, regardless of filename; for others (e.g. CSV files with a "`.csv`" extension), TOPCAT may be able to use the filename as a hint to guess the format (the details of these rules are given in the format-specific subsections below). In other cases though, you will have to specify the format that the file is in. In the Load Window, there is a selection box from which you can choose the format, and from the command line you use the `-f` flag - see Section 10 for details. You can always specify the format rather than using automatic detection if you prefer - this is slightly more efficient, and may give you a more detailed error message if a table fails to load.

In either case, table locations may be given as filenames or as URLs, and any data compression (gzip, unix compress and bzip2) will be automatically detected and dealt with - see Section 4.2.

Some of the formats (e.g. FITS, VOTable) are capable of storing more than one table; usually loading such files loads all the tables into TOPCAT. If you want to specify only a single table, you can give a position indicator either after a "`#`" sign at the end of the filename or using the **Position in file** field in the Filestore Browser or similar. The details of the syntax for this is given in the relevant format description below.

The following sections describe the table formats which TOPCAT can read.

4.1.1.1 FITS

FITS is a very well-established format for storage of astronomical table or image data (see https://fits.gsfc.nasa.gov/). This reader can read tables stored in binary (`XTENSION='BINTABLE'`) and ASCII (`XTENSION='TABLE'`) table extensions; any image data is ignored. Currently, binary table extensions are read much more efficiently than ASCII ones.

When a table is stored in a BINTABLE extension in an uncompressed FITS file on disk, the table is 'mapped' into memory; this generally means very fast loading and low memory usage. FITS tables are thus usually efficient to use.

Limited support is provided for the semi-standard HEALPix-FITS convention; such information about HEALPix level and coordinate system is read and made available for application usage and user examination.

A private convention is used to support encoding of tables with more than 999 columns (not possible in standard FITS); see Section 4.1.3.2.

Header cards in the table's HDU header will be made available as table parameters. Only header cards which are not used to specify the table format itself are visible as parameters (e.g. NAXIS, TTYPE* etc cards are not). HISTORY and COMMENT cards are run together as one multi-line value.

Any 64-bit integer column with a non-zero integer offset (`TFORMn='K'`, `TSCALn=1`, `TZEROn<>0`) is represented in the read table as Strings giving the decimal integer value, since no numeric type in Java is capable of representing the whole range of possible inputs. Such columns are most commonly seen representing unsigned long values.

Where a multi-extension FITS file contains more than one table, a single table may be specified using the position indicator, which may take one of the following forms:

- The numeric index of the HDU. The first extension (first HDU after the primary HDU) is numbered 1. Thus in a compressed FITS table named "`spec23.fits.gz`" with one primary HDU and two BINTABLE extensions, you would view the first one using the name "`spec23.fits.gz`" or "`spec23.fits.gz#1`" and the second one using the name "`spec23.fits.gz#2`". The suffix "`#0`" is never used for a legal FITS file, since the primary HDU cannot contain a table.
- The name of the extension. This is the value of the `EXTNAME` header in the HDU, or alternatively the value of `EXTNAME` followed by "`-`" followed by the value of `EXTVER`. This follows the recommendation in the FITS standard that `EXTNAME` and `EXTVER` headers can be used to identify an HDU. So in a multi-extension FITS file "`cat.fits`" where a table extension has `EXTNAME='UV_DATA'` and `EXTVER=3`, it could be referenced as "`cat.fits#UV_DATA`" or "`cat.fits#UV_DATA-3`". Matching of these names is case-insensitive.

Files in this format may contain multiple tables; depending on the context, either one or all tables will be read. Where only one table is required, either the first one in the file is used, or the required one can be specified after the "`#`" character at the end of the filename.

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading FITS tables, regardless of the filename.

There are actually two FITS input handlers, `fits-basic` and `fits-plus`. The `fits-basic` handler extracts standard column metadata from FITS headers of the HDU in which the table is found, while the `fits-plus` handler reads column and table metadata from VOTable content stored in the primary HDU of the multi-extension FITS file. FITS-plus is a private convention effectively defined by the corresponding output handler; it allows de/serialization of much richer metadata than can be stored in standard FITS headers when the FITS file is read by fits-plus-aware readers, though other readers can understand the unenhanced FITS file perfectly well. It is normally not necessary to worry about this distinction; TOPCAT will determine whether a FITS file is FITS-plus or not based on its content and use the appropriate handler, but if you want to force the reader to use or ignore the enriched header, you can explicitly select an input format of "`FITS-plus`" or "`FITS`". The details of the FITS-plus convention are described in Section 4.1.3.1.

4.1.1.2 Column-oriented FITS

As well as normal binary and ASCII FITS tables, STIL supports FITS files which contain tabular data stored in column-oriented format. This means that the table is stored in a BINTABLE extension HDU, but that BINTABLE has a single row, with each cell of that row holding a whole column's worth of data. The final (slowest-varying) dimension of each of these cells (declared via the `TDIMn` headers) is the same for every column, namely, the number of rows in the table that is represented. The point of this is that all the cells for each column are stored contiguously, which for very large, and especially very wide tables means that certain access patterns (basically, ones which access only a small proportion of the columns in a table) can be much more efficient since they require less I/O overhead in reading data blocks.

Such tables are perfectly legal FITS files, but general-purpose FITS software may not recognise them as multi-row tables in the usual way. This format is mostly intended for the case where you have a large table in some other format (possibly the result of an SQL query) and you wish to cache it in a way which can be read efficiently by a STIL-based application.

For performance reasons, it is advisable to access colfits files uncompressed on disk. Reading them from a remote URL, or in gzipped form, may be rather slow (in earlier versions it was not supported at all).

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading colfits-basic tables, regardless of the filename.

Like the normal (row-oriented) FITS handler, two variants are supported: with (`colfits-plus`) or without (`colfits-basic`) metadata stored as a VOTable byte array in the primary HDU. For details of the FITS-plus convention, see Section 4.1.3.1.

4.1.1.3 VOTable

VOTable is an XML-based format for tabular data endorsed by the International Virtual Observatory Alliance; while the tabular data which can be encoded is by design close to what FITS allows, it provides for much richer encoding of structure and metadata. Most of the table data exchanged by VO services is in VOTable format, and it can be used for local table storage as well.

Any table which conforms to the VOTable 1.0, 1.1, 1.2, 1.3 or 1.4 specifications can be read. This includes all the defined cell data serializations; cell data may be included in-line as XML elements (TABLEDATA serialization), included/referenced as a FITS table (FITS serialization), or included/referenced as a raw binary stream (BINARY or BINARY2 serialization). The handler does not attempt to be fussy about input VOTable documents, and it will have a good go at reading VOTables which violate the standards in various ways.

Much, but not all, of the metadata contained in a VOTable document is retained when the table is read in. The attributes `unit`, `ucd`, `xtype` and `utype`, and the elements `COOSYS`, `TIMESYS` and `DESCRIPTION` attached to table columns or parameters, are read and may be used by the application as appropriate or examined by the user. However, information encoded in the hierarchical structure of the VOTable document, including `GROUP` structure, is not currently retained when a VOTable is read.

VOTable documents may contain more than one actual table (`TABLE` element). To specify a specific single table, the table position indicator is given by the zero-based index of the `TABLE` element in a breadth-first search. Here is an example VOTable document:

```
<VOTABLE>
  <RESOURCE>
    <TABLE name="Star Catalogue"> ... </TABLE>
    <TABLE name="Galaxy Catalogue"> ... </TABLE>
  </RESOURCE>
</VOTABLE>
```

If this is available in a file named "cats.xml" then the two tables could be named as "cats.xml#0" and "cats.xml#1" respectively.

Files in this format may contain multiple tables; depending on the context, either one or all tables will be read. Where only one table is required, either the first one in the file is used, or the required one can be specified after the "`#`" character at the end of the filename.

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading VOTable tables, regardless of the filename.

4.1.1.4 CDF

NASA's Common Data Format, described at https://cdf.gsfc.nasa.gov/, is a binary format for storing self-described data. It is typically used to store tabular data for subject areas like space and solar physics.

CDF does not store tables as such, but sets of variables (columns) which are typically linked to a time quantity; there may be multiple such disjoint sets in a single CDF file. This reader attempts to extract these sets into separate tables using, where present, the `DEPEND_0` attribute defined by the ISTP Metadata Guidelines. Where there are multiple tables they can be identified using a "`#`" symbol at the end of the filename by index ("`<file>.cdf#0`" is the first table) or by the name of the independent variable ("`<file>.cdf#EPOCH`" is the table relating to the `EPOCH` column).

Files in this format may contain multiple tables; depending on the context, either one or all tables will be read. Where only one table is required, either the first one in the file is used, or the required one can be specified after the "`#`" character at the end of the filename.

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading CDF tables, regardless of the filename.

4.1.1.5 CSV

Comma-separated value ("CSV") format is a common semi-standard text-based format in which fields are delimited by commas. Spreadsheets and databases are often able to export data in some variant of it. The intention is to read tables in the version of the format spoken by MS Excel amongst other applications, though the documentation on which it was based was not obtained directly from Microsoft.

The rules for data which it understands are as follows:

- Each row must have the same number of comma-separated fields.
- Whitespace (space or tab) adjacent to a comma is ignored.
- Adjacent commas, or a comma at the start or end of a line (whitespace apart) indicates a null field.
- Lines are terminated by any sequence of carriage-return or newline characters ('\r' or '\n') (a corollary of this is that blank lines are ignored).
- Cells may be enclosed in double quotes; quoted values may contain linebreaks (or any other character); a double quote character within a quoted value is represented by two adjacent double quotes.
- The first line *may* be a header line containing column names rather than a row of data. Exactly the same syntactic rules are followed for such a row as for data rows.

Note that you can *not* use a "`#`" character (or anything else) to introduce "comment" lines.

Because the CSV format contains no metadata beyond column names, the handler is forced to guess the datatype of the values in each column. It does this by reading the whole file through once and guessing on the basis of what it has seen (though see the `maxSample` configuration option). This has the disadvantages:

- Sometimes it guesses a different type than what you want (e.g. 32-bit integer rather than 64-bit integer)
- It's slow to read.

This means that CSV is not generally recommended if you can use another format instead. If you're stuck with a large CSV file that's misbehaving or slow to use, one possibility is to turn it into an ECSV file file by adding some header lines by hand.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`csv(header=true,maxSample=100000)`". The following options are available:

`header = true|false|null`

Indicates whether the input CSV file contains the optional one-line header giving column names. Options are:

- `true`: the first line is a header line containing column names
- `false`: all lines are data lines, and column names will be assigned automatically
- `null`: a guess will be made about whether the first line is a header or not depending on what it looks like

The default value is `null` (auto-determination). This usually works OK, but can get into trouble if all the columns look like string values. (Default: `null`)

`maxSample = <int>`

Controls how many rows of the input file are sampled to determine column datatypes. When reading CSV files, since no type information is present in the input file, the handler has to look at the column data to see what type of value appears to be present in each column, before even starting to read the data in. By default it goes through the whole table when doing this, which can be time-consuming for large tables. If this value is set, it limits the number of rows that are sampled in this data characterisation pass, which can reduce read time substantially. However, if values near the end of the table differ in apparent type from those near the start, it can also result in getting the datatypes wrong. (Default: `0`)

This format cannot be automatically identified by its content, so in general it is necessary to specify that a table is in CSV format when reading it. However, if the input file has the extension "`.csv`" (case insensitive) an attempt will be made to read it using this format.

An example looks like this:

```
RECNO,SPECIES,NAME,LEGS,HEIGHT,MAMMAL
1,pig,Pigling Bland,4,0.8,true
2,cow,Daisy,4,2.0,true
3,goldfish,Dobbin,,0.05,false
4,ant,,6,0.001,false
5,ant,,6,0.001,false
6,queen ant,Ma'am,6,0.002,false
7,human,Mark,2,1.8,true
```

See also ECSV as a format which is similar and capable of storing more metadata.

4.1.1.6 ECSV

The Enhanced Character Separated Values format was developed within the Astropy project and is described in Astropy APE6 (DOI). It is composed of a YAML header followed by a CSV-like body, and is intended to be a human-readable and maybe even human-writable format with rich metadata. Most of the useful per-column and per-table metadata is preserved when de/serializing to this format. The version supported by this reader is currently ECSV 1.0.

There are various ways to format the YAML header, but a simple example of an ECSV file looks like this:

```
# %ECSV 1.0
# ---
# delimiter: ','
# datatype: [
#   { name: index,   datatype: int32   },
#   { name: Species, datatype: string  },
#   { name: Name,    datatype: string  },
#   { name: Legs,    datatype: int32   },
#   { name: Height,  datatype: float64, unit: m },
#   { name: Mammal,  datatype: bool    },
# ]
index,Species,Name,Legs,Height,Mammal
1,pig,Bland,4,,True
2,cow,Daisy,4,2,True
3,goldfish,Dobbin,,0.05,False
4,ant,,6,0.001,False
5,ant,,6,0.001,False
6,human,Mark,2,1.9,True
```

If you follow this pattern, it's possible to write your own ECSV files by taking an existing CSV file and decorating it with a header that gives column datatypes, and possibly other metadata such as units. This allows you to force the datatype of given columns (the CSV reader guesses datatype based on content, but can get it wrong) and it can also be read much more efficiently than a CSV file and its format can be detected automatically.

The header information can be provided either in the ECSV file itself, or alongside a plain CSV file from a separate source referenced using the `header` configuration option. In Gaia EDR3 for instance, the ECSV headers are supplied alongside the CSV files available for raw download of all tables in the Gaia source catalogue, so e.g. STILTS can read one of the gaia_source CSV files with full metadata as follows:

```
   stilts tpipe
      ifmt='ecsv(header=http://cdn.gea.esac.esa.int/Gaia/gedr3/ECSV_headers/gaia_source.heade
      in=http://cdn.gea.esac.esa.int/Gaia/gedr3/gaia_source/GaiaSource_000000-003111.csv.gz
```

The ECSV datatypes that work well with this reader are `bool`, `int8`, `int16`, `int32`, `int64`, `float32`, `float64` and `string`. Array-valued columns are also supported with some restrictions. Following the ECSV 1.0 specification, columns representing arrays of the supported datatypes can be read, as columns with `datatype: string` and a suitable `subtype`, e.g. "`int32[<dims>]`" or "`float64[<dims>]`". Fixed-length arrays (e.g. `subtype: int32[3,10]`) and 1-dimensional variable-length arrays (e.g. `subtype: float64[null]`) are supported; however variable-length arrays with more than one dimension (e.g. `subtype: int32[4,null]`) cannot be represented, and are read in as string values. Null elements of array-valued cells are not supported; they are read as NaNs for floating point data, and as zero/false for integer/boolean data. ECSV 1.0, required to work with array-valued columns, is supported by Astropy v4.3 and later.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`ecsv(header=http://cdn.gea.esac.esa.int/Gaia/gedr3/ECSV_headers/gaia_source.header,colcheck=F`
The following options are available:

`header = <filename-or-url>`
Location of a file containing a header to be applied to the start of the input file. By using this you can apply your own ECSV-format metadata to plain CSV files. (Default: `null`)

`colcheck = IGNORE|WARN|FAIL`

Determines the action taken if the columns named in the YAML header differ from the columns named in the first line of the CSV part of the file. (Default: `WARN`)

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading ECSV tables, regardless of the filename.

4.1.1.7 ASCII

In many cases tables are stored in some sort of unstructured plain text format, with cells separated by spaces or some other delimiters. There is a wide variety of such formats depending on what delimiters are used, how columns are identified, whether blank values are permitted and so on. It is impossible to cope with them all, but the ASCII handler attempts to make a good guess about how to interpret a given ASCII file as a table, which in many cases is successful. In particular, if you just have columns of numbers separated by something that looks like spaces, you should be just fine.

Here are the detailed rules for how the ASCII-format tables are interpreted:

- Bytes in the file are interpreted as ASCII characters
- Each table row is represented by a single line of text
- Lines are terminated by one or more contiguous line termination characters: line feed (0x0A) or carriage return (0x0D)
- Within a line, fields are separated by one or more whitespace characters: space (" ") or tab (0x09)
- A field is either an unquoted sequence of non-whitespace characters, or a sequence of non-newline characters between matching single (') or double (") quote characters - spaces are therefore allowed in quoted fields
- Within a quoted field, whitespace characters are permitted and are treated literally
- Within a quoted field, any character preceded by a backslash character ("\") is treated literally. This allows quote characters to appear within a quoted string.
- An empty quoted string (two adjacent quotes) or the string "`null`" (unquoted) represents the null value
- All data lines must contain the same number of fields (this is the number of columns in the table)
- The data type of a column is guessed according to the fields that appear in the table. If all the fields in one column can be parsed as integers (or null values), then that column will turn into an integer-type column. The types that are tried, in order of preference, are: `Boolean`, `Short` `Integer`, `Long`, `Float`, `Double`, `String`
- Some special values are permitted for floating point columns: `NaN` for not-a-number, which is treated the same as a null value for most purposes, and `Infinity` or `inf` for infinity (with or without a preceding +/- sign). These values are matched case-insensitively.
- Empty lines are ignored
- Anything after a hash character "#" (except one in a quoted string) on a line is ignored as far as table data goes; any line which starts with a "!" is also ignored. However, lines which start with a "#" or "!" at the start of the table (before any data lines) will be interpreted as metadata as follows:
  - The last "#"/"!"-starting line before the first data line may contain the column names. If it has the same number of fields as there are columns in the table, each field will be taken to be the title of the corresponding column. Otherwise, it will be taken as a normal comment line.
  - Any comment lines before the first data line not covered by the above will be concatenated to form the "description" parameter of the table.

If the list of rules above looks frightening, don't worry, in many cases it ought to make sense of a table without you having to read the small print. Here is an example of a suitable ASCII-format table:

```
#
# Here is a list of some animals.
#
# RECNO  SPECIES         NAME         LEGS   HEIGHT/m
  1      pig             "Pigling Bland"  4  0.8
  2      cow             Daisy        4      2
  3      goldfish        Dobbin       ""     0.05
  4      ant             ""           6      0.001
  5      ant             ""           6      0.001
  6      ant             ''           6      0.001
  7      "queen ant"     'Ma\'am'     6      2e-3
  8      human           "Mark"       2      1.8
```

In this case it will identify the following columns:

```
Name       Type
----       ----
RECNO      Short
SPECIES    String
NAME       String
LEGS       Short
HEIGHT/m   Float
```

It will also use the text "`Here is a list of some animals`" as the Description parameter of the table. Without any of the comment lines, it would still interpret the table, but the columns would be given the names `col1`..`col5`.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`ascii(maxSample=5000)`". The following options are available:

`maxSample = <int>`
Controls how many rows of the input file are sampled to determine column datatypes. When reading ASCII files, since no type information is present in the input file, the handler has to look at the column data to see what type of value appears to be present in each column, before even starting to read the data in. By default it goes through the whole table when doing this, which can be time-consuming for large tables. If this value is set, it limits the number of rows that are sampled in this data characterisation pass, which can reduce read time substantially. However, if values near the end of the table differ in apparent type from those near the start, it can also result in getting the datatypes wrong. (Default: `0`)

This format cannot be automatically identified by its content, so in general it is necessary to specify that a table is in ASCII format when reading it. However, if the input file has the extension "`.txt`" (case insensitive) an attempt will be made to read it using this format.

4.1.1.8 IPAC

CalTech's Infrared Processing and Analysis Center use a text-based format for storage of tabular data, defined at http://irsa.ipac.caltech.edu/applications/DDGEN/Doc/ipac_tbl.html. Tables can store column name, type, units and null values, as well as table parameters.

This format cannot be automatically identified by its content, so in general it is necessary to specify that a table is in IPAC format when reading it. However, if the input file has the extension "`.tbl`" or "`.ipac`" (case insensitive) an attempt will be made to read it using this format.

An example looks like this:

```
\Table name = "animals.vot"
```

```
\Description = "Some animals"
\Author = "Mark Taylor"
| RECNO | SPECIES   | NAME          | LEGS | HEIGHT | MAMMAL |
| int   | char      | char          | int  | double | char   |
|       |           |               |      | m      |        |
| null  | null      | null          | null | null   | null   |
  1       pig         Pigling Bland   4      0.8      true
  2       cow         Daisy           4      2.0      true
  3       goldfish    Dobbin          null   0.05     false
  4       ant         null            6      0.001    false
  5       ant         null            6      0.001    false
  6       queen ant   Ma'am           6      0.002    false
  7       human       Mark            2      1.8      true
```

4.1.1.9 PDS4

NASA's Planetary Data System version 4 format is described at https://pds.nasa.gov/datastandards/. This implementation is based on v1.16.0 of PDS4.

PDS4 files consist of an XML *Label* file which provides detailed metadata, and which may also contain references to external data files stored alongside it. This input handler looks for (binary, character or delimited) tables in the Label; depending on the configuration it may restrict them to those in the `File_Area_Observational` area. The Label is the file which has to be presented to this input handler to read the table data. Because of the relationship between the label and the data files, it is usually necessary to move them around together.

If there are multiple tables in the label, you can refer to an individual one using the "`#`" specifier after the label file name by table `name`, `local_identifier`, or 1-based index (e.g. "`label.xml#1`" refers to the first table).

If there are `Special_Constants` defined in the label, they are in most cases interpreted as blank values in the output table data. At present, the following special values are interpreted as blanks: `saturated_constant`, `missing_constant`, `error_constant`, `invalid_constant`, `unknown_constant`, `not_applicable_constant`, `high_instrument_saturation`, `high_representation_saturation`, `low_instrument_saturation`, `low_representation_saturation`.

Fields within top-level Groups are interpreted as array values. Any fields in nested groups are ignored. For these array values only limited null-value substitution can be done (since array elements are primitives and so cannot take null values).

This input handler is somewhat experimental, and the author is not a PDS expert. If it behaves strangely or you have suggestions for how it could work better, please contact the author.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`pds4(checkmagic=false,observational=true)`". The following options are available:

`checkmagic = true|false`
: Determines whether an initial test is made to see whether the file looks like PDS4 before attempting to read it as one. The tests are ad-hoc and look for certain elements and namespaces that are expected to appear near the start of a table-containing PDS4 file, but it's not bulletproof. Setting this true is generally a good idea to avoid attempting to parse non-PDS4 files, but you can set it false to attempt to read an PDS4 file that starts with the wrong sequence. (Default: `true`)

`observational = true|false`
: Determines whether only tables within a `<File_Area_Observational>` element of the PDS4 label should be included. If true, only observational tables are found, if false, other tables will

be found as well. (Default: `false`)

Files in this format may contain multiple tables; depending on the context, either one or all tables will be read. Where only one table is required, either the first one in the file is used, or the required one can be specified after the "`#`" character at the end of the filename.

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading PDS4 tables, regardless of the filename.

4.1.1.10 MRT

The so-called "Machine-Readable Table" format is used by AAS journals, and based on the format of readMe files used by the CDS. There is some documentation at https://journals.aas.org/mrt-standards/, which mostly builds on documentation at http://vizier.u-strasbg.fr/doc/catstd.htx, but the format is in fact quite poorly specified, so this input handler was largely developed on a best-efforts basis by looking at MRT tables actually in use by AAS, and with assistance from AAS staff. As such, it's not guaranteed to succeed in reading all MRT files out there, but it will try its best.

It only attempts to read MRT files themselves, there is currently no capability to read VizieR data tables which provide the header and formatted data in separate files; however, if a table is present in VizieR, there will be options to download it in more widely used formats that can be used instead.

An example looks like this:

```
Title: A search for multi-planet systems with TESS using a Bayesian
       N-body retrieval and machine learning
Author: Pearson K.A.
Table: Stellar Parameters
================================================================================
Byte-by-byte Description of file: ajab4e1ct2_mrt.txt
--------------------------------------------------------------------------------
   Bytes Format Units   Label   Explanations
--------------------------------------------------------------------------------
   1-  9 I9     ---     ID      TESS Input Catalog identifier
  11- 15 F5.2   mag     Tmag    Apparent TESS band magnitude
  17- 21 F5.3   solRad  R*      Stellar radius
  23- 26 I4     K       Teff    Effective temperature
  28- 32 F5.3   [cm/s2] log(g)  log surface gravity
  34- 38 F5.2   [Sun]   [Fe/H]  Metallicity
  40- 44 F5.3   ---     u1      Linear Limb Darkening
  46- 50 F5.3   ---     u2      Quadratic Limb Darkening
--------------------------------------------------------------------------------
231663901 12.35 0.860 5600 4.489  0.00 0.439 0.138
149603524  9.72 1.280 6280 4.321  0.24 0.409 0.140
336732616 11.46 1.400 6351 4.229  0.00 0.398 0.140
231670397  9.85 2.070 6036 3.934  0.00 0.438 0.117
...
```

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`mrt(checkmagic=false,errmode=FAIL)`". The following options are available:

`checkmagic = true|false`
Determines whether an initial test is made to see whether the file looks like MRT before attempting to read it as one; the test is that it starts with the string "`Title: `". Setting this true is generally a good idea to avoid attempting to parse non-MRT files, but you can set it false to attempt to read an MRT file that starts with the wrong sequence. (Default: `true`)

`errmode = IGNORE|WARN|FAIL`
Indicates what action should be taken if formatting errors are detected in the file at read time.

(Default: `warn`)

`usefloat = true|false`

Sets whether this handler will use a 32-bit float type for reading sufficiently narrow floating point fields. This is usually a good idea since it reduces storage requirements when only a few significant figures are provided, but can fail if the column contains any very large absolute values (>~1e38), which cannot be represented in a 32-bit IEEE float. So it's safer to set it false.

If it is set true, then encountering values outside the representable range will be reported in accordance with the current ErrorMode. (Default: `false`)

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading MRT tables, regardless of the filename.

4.1.1.11 Parquet

Parquet is a columnar format developed within the Apache project. Data is compressed on disk and read into memory before use. The file format is described at https://github.com/apache/parquet-format. This software is written with reference to version 2.10.0 of the format.

This input handler will read columns representing scalars, strings and one-dimensional arrays of the same. It is not capable of reading multi-dimensional arrays, more complex nested data structures, or some more exotic data types like 96-bit integers. If such columns are encountered in an input file, a warning will be emitted through the logging system and the column will not appear in the read table. Support may be introduced for some additional types if there is demand.

Parquet files typically do not contain rich metadata such as column units, descriptions, UCDs etc. To remedy that, this reader supports the VOParquet convention (version 1.0), in which metadata is recorded in a DATA-less VOTable stored in the parquet file header. If such metadata is present it will by default be used, though this can be controlled using the `votmeta` configuration option below.

Depending on the way that the table is accessed, the reader tries to take advantage of the column and row block structure of parquet files to read the data in parallel where possible.

Note:

The parquet I/O handlers require large external libraries, which are not always bundled with the library/application software because of their size. In some configurations, parquet support may not be present, and attempts to read or write parquet files will result in a message like:

```
Parquet-mr libraries not available
```

If you can supply the relevant libaries on the classpath at runtime, the parquet support will work. At time of writing, the required libraries are included in the `topcat-extra.jar` monolithic jar file (though not `topcat-full.jar`), and are included if you have the `topcat-all.dmg` file. They can also be found in the starjava github repository (https://github.com/Starlink/starjava/tree/master/parquet/src/lib or you can acquire them from the Parquet MR package. These arrangements may be revised in future releases, for instance if parquet usage becomes more mainstream. The required dependencies are a minimal subset of those required by the Parquet MR submodule `parquet-cli` at version 1.13.1, in particular the files `aircompressor-0.21.jar` `commons-collections-3.2.2.jar` `commons-configuration2-2.1.1.jar` `commons-lang3-3.9.jar` `failureaccess-1.0.1.jar` `guava-27.0.1-jre.jar` `hadoop-auth-3.2.3.jar` `hadoop-common-3.2.3.jar` `hadoop-mapreduce-client-core-3.2.3.jar` `htrace-core4-4.1.0-incubating.jar` `parquet-cli-1.13.1.jar` `parquet-column-1.13.1.jar` `parquet-common-1.13.1.jar` `parquet-encoding-1.13.1.jar` `parquet-format-structures-1.13.1.jar` `parquet-hadoop-1.13.1.jar` `parquet-jackson-1.13.1.jar` `slf4j-api-1.7.22.jar` `slf4j-nop-1.7.22.jar` `snappy-java-1.1.8.3.jar` `stax2-api-4.2.1.jar` `woodstox-core-5.3.0.jar` `zstd-jni-1.5.0-1.jar`.

These libraries support some, but not all, of the compression formats defined for parquet, currently `uncompressed`, `gzip`, `snappy`, `zstd` and `lz4_raw`. Supplying more of the parquet-mr dependencies at runtime would extend this list. Unlike the rest of TOPCAT/STILTS/STIL which is written in pure java, some of these libraries (currently the snappy and zstd compression codecs) contain native code, which means they may not work on all architectures. At time of writing all common architectures are covered, but there is the possibility of failure with a `java.lang.UnsatisfiedLinkError` on other platforms if attempting to read/write files that use those compression algorithms.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`parquet(cachecols=true,nThread=4)`". The following options are available:

`cachecols = true|false|null`
Forces whether to read all the column data at table load time. If `true`, then when the table is loaded, all data is read by column into local scratch disk files, which is generally the fastest way to ingest all the data. If `false`, the table rows are read as required, and possibly cached using the normal STIL mechanisms. If `null` (the default), the decision is taken automatically based on available information. (Default: `null`)

`nThread = <int>`
Sets the number of read threads used for concurrently reading table columns if the columns are cached at load time - see the `cachecols` option. If the value is <=0 (the default), a value is chosen based on the number of apparently available processors. (Default: `0`)

`tryUrl = true|false`
Whether to attempt to open non-file URLs as parquet files. This usually seems to fail with a cryptic error message, so it is not attempted by default, but it's possible that with suitable library support on the classpath it might work, so this option exists to make the attempt. (Default: `false`)

`votmeta = true|false|null`
If true, the content of the parquet extra metadata key-value list item with key `IVOA.VOTable-Parquet.content` will be read to supply the metadata for the input table, following the VOParquet convention. If false, any such VOTable metadata is ignored. If set null, the default, then such VOTable metadata will be used only if it is present and apparently consistent with the parquet data and metadata. (Default: `null`)

`votable = <filename-or-url>`
Location of a UTF-8-encoded data-less VOTable that will supply additional metadata for a parquet table being read, according to the VOParquet convention. This is normally not required, but if present it overrides any such metadata VOTable embedded within the parquet file. This value will only be used if the `votmeta` configuration is not false. (Default: `null`)

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading parquet tables, regardless of the filename.

4.1.1.12 HAPI

HAPI, the Heliophysics Data Application Programmer’s Interface is a protocol for serving streamed time series data. This reader can read HAPI CSV and binary tables if they include header information (the `include=header` request parameter must be present). An example HAPI URL is

```
   https://vires.services/hapi/data?dataset=GRACE_A_MAG&start=2009-01-01&stop=2009-01-02&inclu
```

While HAPI data is normally accessed directly from the service, it is possible to download a HAPI stream to a local file and use this handler to read it from disk.

This format cannot be automatically identified by its content, so in general it is necessary to specify that a table is in HAPI format when reading it. However, if the input file has the extension "`.hapi`" (case insensitive) an attempt will be made to read it using this format.

4.1.1.13 Feather

The Feather file format is a column-oriented binary disk-based format based on Apache Arrow and supported by (at least) Python, R and Julia. Some description of it is available at https://github.com/wesm/feather and https://blog.rstudio.com/2016/03/29/feather/. It can be used for large datasets, but it does not support array-valued columns. It can be a useful format to use for exchanging data with R, for which FITS I/O is reported to be slow.

At present CATEGORY type columns are not supported, and metadata associated with TIME, DATE and TIMESTAMP columns is not retrieved.

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading feather tables, regardless of the filename.

4.1.1.14 GBIN

GBIN format is a special-interest file format used within DPAC, the Data Processing and Analysis Consortium working on data from the Gaia astrometry satellite. It is based on java serialization, and in all of its various forms has the peculiarity that you only stand any chance of decoding it if you have the Gaia data model classes on your java classpath at runtime. Since the set of relevant classes is very large, and also depends on what version of the data model your GBIN file corresponds to, those classes will not be packaged with this software, so some additional setup is required to read GBIN files.

As well as the data model classes, you must provide on the runtime classpath the GaiaTools classes required for GBIN reading. The table input handler accesses these by reflection, to avoid an additional large library dependency for a rather niche requirement. It is likely that since you have to supply the required data model classes you will also have the required GaiaTools classes to hand as well, so this shouldn't constitute much of an additional burden for usage.

In practice, if you have a jar file or files for pretty much any java library or application which is capable of reading a given GBIN file, just adding it or them to the classpath at runtime when using this input handler ought to do the trick. Examples of such jar files are the `MDBExplorerStandalone.jar` file available from https://gaia.esac.esa.int/mdbexp/, or the `gbcat.jar` file you can build from the CU9/software/gbcat/ directory in the DPAC subversion repository.

The GBIN format doesn't really store tables, it stores arrays of java objects, so the input handler has to make some decisions about how to flatten these into table rows.

In its simplest form, the handler basically looks for public instance methods of the form `getXxx()` and uses the `Xxx` as column names. If the corresponding values are themselves objects with suitable getter methods, those objects are added as new columns instead. This more or less follows the practice of the `gbcat` (`gaia.cu1.tools.util.GbinInterogator`) tool. Method names are sorted alphabetically. Arrays of complex objects are not handled well, and various other things may trip it up. See the source code (e.g. `uk.ac.starlink.gbin.GbinTableProfile`) for more details.

If the object types stored in the GBIN file are known to the special metadata-bearing class `gaia.cu9.tools.documentationexport.MetadataReader` and its dependencies, and if that class is on the runtime classpath, then the handler will be able to extract additional metadata as available, including standardised column names, table and column descriptions, and UCDs. An example of a jar file containing this metadata class alongside data model classes is `GaiaDataLibs-18.3.1-r515078.jar`. Note however at time of writing there are some deficiencies with this metadata extraction functionality related to unresolved issues in the upstream gaia class libraries and the relevant interface control document (GAIA-C9-SP-UB-XL-034-01, "External Data Centres ICD"). Currently columns appear in the output table in a more or less random order, units and Utypes are not extracted, and using the GBIN reader tends to cause a 700kbyte file "temp.xml" to be written in the current directory. If the upstream issues are fixed, this behaviour may improve.

Note: support for GBIN files is somewhat experimental. Please contact the author (who is not a GBIN expert) if it doesn't seem to be working properly or you think it should do things differently.

Note: there is a known bug in some versions of GaiaTools (caused by a bug in its dependency library zStd-jni) which in rare cases can fail to read all the rows in a GBIN input file. If this bug is encountered by the reader, it will by default fail with an error mentioning zStd-jni. In this case, the best thing to do is to put a fixed version of zStd-jni or GaiaTools on the classpath. However, if instead you set the config option `readMeta=false` the read will complete without error, though the missing rows will not be recovered.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`gbin(readMeta=false,hierarchicalNames=true)`". The following options are available:

`readMeta = true|false`

Configures whether the GBIN metadata will be read prior to reading the data. This may slow things down slightly, but means the row count can be determined up front, which may have benefits for downstream processing.

Setting this false can prevent failing on an error related to a broken version of the zStd-jni library in GaiaTools. Note however that in this case the data read, though not reporting an error, will silently be missing some rows from the GBIN file. (Default: `true`)

`hierarchicalNames = true|false`

Configures whether column names in the output table should be forced to reflect the compositional hierarchy of their position in the element objects. If set true, columns will have names like "`Astrometry_Alpha`", if false they may just be called "`Alpha`". In case of name duplication however, the hierarchical form is always used. (Default: `false`)

This format can be automatically identified by its content so you do not need to specify the format explicitly when reading GBIN tables, regardless of the filename.

Example: Suppose you have the `MDBExplorerStandalone.jar` file containing the data model classes, you can read GBIN files by starting TOPCAT like this:

```
   topcat -classpath MDBExplorerStandalone.jar ...
```

or like this:

```
   java -classpath topcat-full.jar:MDBExplorerStandalone.jar uk.ac.starlink.topcat.Driver ...
```

4.1.1.15 TST

Tab-Separated Table, or TST, is a text-based table format used by a number of astronomical tools including Starlink's GAIA and ESO's SkyCat on which it is based. A definition of the format can be found in Starlink Software Note 75. The implementation here ignores all comment lines: special comments such as the "`#column-units:`" are not processed.

An example looks like this:

```
Simple TST example; stellar photometry catalogue.

A.C. Davenhall (Edinburgh) 26/7/00.

Catalogue of U,B,V colours.
UBV photometry from Mount Pumpkin Observatory,
see Sage, Rosemary and Thyme (1988).

# Start of parameter definitions.
EQUINOX: J2000.0
EPOCH: J1996.35

id_col: -1
ra_col: 0
dec_col: 1

# End of parameter definitions.
ra<tab>dec<tab>V<tab>B_V<tab>U_B
--<tab>---<tab>-<tab>---<tab>---
5:09:08.7<tab> -8:45:15<tab>  4.27<tab>  -0.19<tab>  -0.90
5:07:50.9<tab> -5:05:11<tab>  2.79<tab>  +0.13<tab>  +0.10
5:01:26.3<tab> -7:10:26<tab>  4.81<tab>  -0.19<tab>  -0.74
5:17:36.3<tab> -6:50:40<tab>  3.60<tab>  -0.11<tab>  -0.47
[EOD]
```

This format cannot be automatically identified by its content, so in general it is necessary to specify that a table is in TST format when reading it.

4.1.1.16 World Data Center

Some support is provided for files produced by the World Data Centre for Solar Terrestrial Physics. The format itself apparently has no name, but files in this format look something like the following:

```
Column formats and units - (Fixed format columns which are single space separated.)
------------------------
Datetime (YYYY mm dd HHMMSS)            %4d %2d %2d %6d      -
                                        %1s
aa index - 3-HOURLY (Provisional)       %3d                  nT

2000 01 01 000000  67
2000 01 01 030000  32
    ...
```

Support for this (obsolete?) format may not be very complete or robust.

This format cannot be automatically identified by its content, so in general it is necessary to specify that a table is in WDC format when reading it.

4.1.2 Output Formats

Writing out tables from TOPCAT is done using the Save Table Window. In general you have to specify the format in which you want the table to be output by selecting from the Save Window's **Table Output Format** selector; the following sections describe the possible choices. In some cases there are variants within each format, also described. If you use the default **(auto)** output format, TOPCAT will try to guess the format based on the filename you provide; the rules for that are described below as well.

The program has no "native" file format, but if you have no particular preference about which format to save tables to, FITS format is a good choice. Uncompressed FITS tables do not in most cases have to be read all the way through (they are 'mapped' into memory), which makes them very fast to load up. The FITS format which is written by default (also known as "FITS-plus") also uses a trick to store extra metadata, such as table parameters and UCDs in a way TOPCAT can read in again later. These files are quite usable as normal FITS tables by other applications, but they will only be able to see the limited metadata stored in the FITS headers. For very large files, in some circumstances the Column-Oriented FITS variant (**colfits**) can be more efficient, though this is unlikely to be understood except by STIL-based code (TOPCAT and STILTS). The FITS output handler with its options and variants is documented in Section 4.1.2.1. If you want to write to a format which retains all metadata in a portable format, then one of the VOTable formats might be better.

4.1.2.1 FITS

FITS is a very well-established format for storage of astronomical table or image data (see https://fits.gsfc.nasa.gov/). This writer stores tables in BINTABLE extensions of a FITS file.

There are a number of variations in exactly how the table data is written to FITS. These can be configured with `name=value` options in brackets as described below, but for most purposes this isn't required; you can just choose `fits` or one of the standard aliases for commonly-used combinations like `colfits` or `fits-basic`.

In all cases the output from this handler is legal FITS, but some non-standard conventions are used:

fits-plus
In "fits-plus" format, the primary HDU contains an array of bytes which stores the full table metadata as the text of a VOTable document, along with headers that indicate this has been done. Most FITS table readers will ignore this altogether and treat the file just as if it contained only the table. When it is re-read by this or compatible applications however, they can read out the metadata and make it available for use. In this way you can store your data in the efficient and widely portable FITS format without losing the additional metadata such as table parameters, column UCDs, lengthy column descriptions etc that may be attached to the table. This variant, which is the default, can be explicitly selected with the `primary=votable` option or `fits-plus` alias (if you don't want it, use `primary=basic` or `fits-basic`). This convention is described in more detail in Section 4.1.3.1.

colfits
In Column-Oriented FITS output, the HDU containing the table data, instead of containing a multi-row table, contains a single-row table in which each cell is an (nrow-element) array containing the data for an entire column. The point of this is to keep all the data for a single row localised on the disk rather than scattered through the whole file. This can be more efficient for certain applications, especially when the table is larger than physical memory, and has many columns of which only a few are needed for a particular task, for instance plotting two columns against each other. The overhead for writing this format is somewhat higher than for normal (row-oriented) FITS however, and other FITS table applications may not be able to work with it, so in most cases normal FITS is a better choice. This variant can be selected with the `col=true` option or the `colfits-plus/colfits-basic` aliases. If you write to a file with the "`.colfits`" extension it is used by default.

wide
A private convention is used where required to support encoding of tables with more than 999 columns, which is not possible in standard FITS. If software unaware of this convention (e.g. CFITSIO) is used to read such tables, it will only see the first 998 columns written as intended,

plus a column 999 containing an undescribed byte buffer where the rest of the column data is stored. This convention is described in more detail in Section 4.1.3.2.

For convenience, and compatibility with earlier versions, these standard aliases are provided:

fits-plus
Alias for `fits` or `fits(primary=votable)`.

fits-basic
Alias for `fits(primary=basic)`.

fits-var
Alias for `fits(primary=basic,var=true)`.

colfits-plus
Alias for `fits(col=true)`.

colfits-basic
Alias for `fits(col=true,primary=basic)`.

fits-healpix
This is a special case. It is used for storing HEALPix pixel data in a way that conforms to the HEALPix-FITS serialization convention. In most ways it behaves the same as `fits-basic`, but it will rearrange and rename columns as required to follow the convention, and it will fail if the table does not contain the required HEALPix metadata (`STIL_HPX_*` parameters).

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`fits(primary=basic,col=false)`". The following options are available:

`primary = basic|votable[n.n]|none`
Determines what is written into the Primary HDU. The Primary HDU (PHDU) of a FITS file cannot contain a table; the following options are available.

`basic`
A minimal PHDU is written with no interesting content

`votable[n.n]`
The PHDU contains the full table metadata as the text of a VOTable document, along with headers to indicate that this has been done. This corresponds to the "**fits-plus**" format. The "`[n.n]`" part is optional, but if included (e.g. "`votable1.5`") indicates the version of the VOTable format to use.

`none`
No PHDU is written. The output is therefore not a legal FITS file, but it can be appended to an existing FITS file that already has a PHDU and perhaps other extension HDUs.

(Default: `votable`)

`col = true|false`
If true, writes data in column-oriented format. In this case, the output is a single-row table in which each cell is an array value holding the data for an entire column. All the arrays in the row have the same length, which is the row count of the table being represented. This corresponds to the "**colfits**" format. (Default: `false`)

`var = FALSE|TRUE|P|Q`
Determines how variable-length array-valued columns will be stored. `True` stores variable-length array values after the main part of the table in the heap, while `false` stores all arrays as fixed-length (with a length equal to that of the longest array in the column) in the body of the table.The options `P` or `Q` can be used to force 32-bit or 64-bit pointers for indexing

into the heap, but it's not usually necessary since a suitable choice is otherwise made from the data. (Default: `FALSE`)

`date = true|false`
If true, the DATE-HDU header is filled in with the current date; otherwise it is not included. (Default: `true`)

Multiple tables may be written to a single output file using this format.

If no output format is explicitly chosen, writing to a filename with the extension "`.fits`", "`.fit`" or "`.fts`" (case insensitive) will select `fits` format for output.

4.1.2.2 VOTable

VOTable is an XML-based format for tabular data endorsed by the International Virtual Observatory Alliance and defined in the VOTable Recommendation. While the tabular data which can be encoded is by design close to what FITS allows, it provides for much richer encoding of structure and metadata. Most of the table data exchanged by VO services is in VOTable format, but it can be used for local table storage as well.

When a table is saved to VOTable format, a document conforming to the VOTable specification containing a single TABLE element within a single RESOURCE element is written. Where the table contains such information (often obtained by reading an input VOTable), column and table metadata will be written out as appropriate to the attributes `unit`, `ucd`, `xtype` and `utype`, and the elements `COOSYS`, `TIMESYS` and `DESCRIPTION` attached to table columns or parameters.

There are various ways that a VOTable can be written; by default the output serialization format is TABLEDATA and the VOTable format version is 1.4, or a value controlled by the `votable.version` system property. However, configuration options are available to adjust these defaults.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`votable(format=BINARY2,version=V13)`". The following options are available:

`format = TABLEDATA|BINARY|BINARY2|FITS`
Gives the serialization type (DATA element content) of output VOTables. (Default: `TABLEDATA`)

`version = V10|V11|V12|V13|V14|V15`
Gives the version of the VOTable format which will be used when writing the VOTable. "`V10`" is version 1.0 etc.

`inline = true|false`
If true, STREAM elements are written base64-encoded within the body of the document, and if false they are written to a new external binary file whose name is derived from that of the output VOTable document. This is only applicable to BINARY, BINARY2 and FITS formats where output is not to a stream. (Default: `true`)

`compact = true|false|null`
Controls whitespace formatting for TABLEDATA output, ignored for other formats. By default a decision will be taken dependent on table width. (Default: `null`)

`encoding = UTF-8|UTF-16|...`
Specifies the XML encoding used in the output VOTable. The default value is UTF-8. Note that certain optimisations are in place for UTF-8 output which means that other encodings may be significantly slower. (Default: `UTF-8`)

`date = true|false`
If true, the output file will contain a comment recording the current date; otherwise it is not included. (Default: `true`)

Multiple tables may be written to a single output file using this format.

If no output format is explicitly chosen, writing to a filename with the extension "`.vot`", "`.votable`" or "`.xml`" (case insensitive) will select `votable` format for output.

4.1.2.3 CSV

Writes tables in the semi-standard Comma-Separated Values format. This does not preserve any metadata apart from column names, and is generally inefficient to read, but it can be useful for importing into certain external applications, such as some databases or spreadsheets.

By default, the first line is a header line giving the column names, but this can be inhibited using the `header=false` configuration option.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`csv(header=true,maxCell=160)`". The following options are available:

`header = true|false`
If true, the first line of the CSV output will be a header containing the column names; if false, no header line is written and all lines represent data rows. (Default: `true`)

`maxCell = <int>`
Maximum width in characters of an output table cell. Cells longer than this will be truncated. (Default: `2147483647`)

If no output format is explicitly chosen, writing to a filename with the extension "`.csv`" (case insensitive) will select `CSV` format for output.

An example looks like this:

```
RECNO,SPECIES,NAME,LEGS,HEIGHT,MAMMAL
1,pig,Pigling Bland,4,0.8,true
2,cow,Daisy,4,2.0,true
3,goldfish,Dobbin,,0.05,false
4,ant,,6,0.001,false
5,ant,,6,0.001,false
6,queen ant,Ma'am,6,0.002,false
7,human,Mark,2,1.8,true
```

4.1.2.4 ECSV

The Enhanced Character Separated Values format was developed within the Astropy project and is described in Astropy APE6 (DOI). It is composed of a YAML header followed by a CSV-like body, and is intended to be a human-readable and maybe even human-writable format with rich metadata. Most of the useful per-column and per-table metadata is preserved when de/serializing to this format. The version supported by this writer is currently ECSV 1.0.

ECSV allows either a space or a comma for delimiting values, controlled by the `delimiter` configuration option. If `ecsv(delimiter=comma)` is used, then removing the YAML header will leave a CSV file that can be interpreted by the CSV inputhandler or imported into other CSV-capable applications.

Following the ECSV 1.0 specification, array-valued columns are supported. ECSV 1.0, required for working with array-valued columns, is supported by Astropy v4.3 and later.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`ecsv(delimiter=comma)`". The following options are available:

`delimiter = comma|space`
: Delimiter character, which for ECSV may be either a space or a comma. Permitted values are "`space`" or "`comma`".

If no output format is explicitly chosen, writing to a filename with the extension "`.ecsv`" (case insensitive) will select `ECSV` format for output.

An example looks like this:

```
# %ECSV 1.0
# ---
# datatype:
# -
#   name: RECNO
#   datatype: int32
# -
#   name: SPECIES
#   datatype: string
# -
#   name: NAME
#   datatype: string
#   description: How one should address the animal in public & private.
# -
#   name: LEGS
#   datatype: int32
#   meta:
#     utype: anatomy:limb
# -
#   name: HEIGHT
#   datatype: float64
#   unit: m
#   meta:
#     VOTable precision: 2
# -
#   name: MAMMAL
#   datatype: bool
# meta:
#   name: animals.vot
#   Description: Some animals
#   Author: Mark Taylor
RECNO SPECIES NAME LEGS HEIGHT MAMMAL
1 pig "Pigling Bland" 4 0.8 True
2 cow Daisy 4 2.0 True
3 goldfish Dobbin "" 0.05 False
4 ant "" 6 0.001 False
5 ant "" 6 0.001 False
6 "queen ant" Ma'am 6 0.002 False
7 human Mark 2 1.8 True
```

4.1.2.5 ASCII

Writes to a simple plain-text format intended to be comprehensible by humans or machines.

The first line is a comment, starting with a "`#`" character, naming the columns, and an attempt is made to line up data in columns using spaces. No metadata apart from column names is written.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g.

`ascii(maxCell=158,maxParam=160)`". The following options are available:

`maxCell = <int>`
Maximum width in characters of an output table cell. Cells longer than this will be truncated. (Default: `158`)

`maxParam = <int>`
Maximum width in characters of an output table parameter. Parameters with values longer than this will be truncated. (Default: `160`)

`params = true|false`
Whether to output table parameters as well as row data. (Default: `false`)

`sampledRows = <int>`
The number of rows examined on a first pass of the table to determine the width of each column. Only a representative number of rows needs to be examined, but if a formatted cell value after this limit is wider than the cells up to it, then such later wide cells may get truncated. If the value is <=0, all rows are examined in the first pass; this is the default, but it can be configured to some other value if that takes too long. (Default: `0`)

If no output format is explicitly chosen, writing to a filename with the extension "`.txt`" (case insensitive) will select `ascii` format for output.

An example looks like this:

```
# RECNO SPECIES   NAME          LEGS HEIGHT MAMMAL
  1     pig       "Pigling Bland" 4    0.8    true
  2     cow       Daisy         4    2.0    true
  3     goldfish  Dobbin        ""   0.05   false
  4     ant       ""            6    0.001  false
  5     ant       ""            6    0.001  false
  6     "queen ant" "Ma\'am"      6    0.002  false
  7     human     Mark          2    1.8    true
```

4.1.2.6 IPAC

Writes output in the format used by CalTech's Infrared Processing and Analysis Center, and defined at http://irsa.ipac.caltech.edu/applications/DDGEN/Doc/ipac_tbl.html. Column name, type, units and null values are written, as well as table parameters.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`ipac(maxCell=1000,maxParam=100000)`". The following options are available:

`maxCell = <int>`
Maximum width in characters of an output table cell. Cells longer than this will be truncated. (Default: `1000`)

`maxParam = <int>`
Maximum width in characters of an output table parameter. Parameters with values longer than this will be truncated. (Default: `100000`)

`params = true|false`
Whether to output table parameters as well as row data. (Default: `true`)

`sampledRows = <int>`
The number of rows examined on a first pass of the table to determine the width of each column. Only a representative number of rows needs to be examined, but if a formatted cell value after this limit is wider than the cells up to it, then such later wide cells may get truncated. If the value is <=0, all rows are examined in the first pass; this is the default, but it

can be configured to some other value if that takes too long. (Default: `0`)

If no output format is explicitly chosen, writing to a filename with the extension "`.tbl`" or "`.ipac`" (case insensitive) will select `IPAC` format for output.

An example looks like this:

```
\Table name = "animals.vot"
\Description = "Some animals"
\Author = "Mark Taylor"
| RECNO | SPECIES    | NAME            | LEGS | HEIGHT | MAMMAL |
| int   | char       | char            | int  | double | char   |
|       |            |                 |      | m      |        |
| null  | null       | null            | null | null   | null   |
  1       pig          Pigling Bland     4      0.8      true
  2       cow          Daisy             4      2.0      true
  3       goldfish     Dobbin            null   0.05     false
  4       ant          null              6      0.001    false
  5       ant          null              6      0.001    false
  6       queen ant    Ma'am             6      0.002    false
  7       human        Mark              2      1.8      true
```

4.1.2.7 Parquet

Parquet is a columnar format developed within the Apache project. Data is compressed on disk and read into memory before use. The file format is described at https://github.com/apache/parquet-format. This software is written with reference to version 2.10.0 of the format.

The parquet file format itself defines only rather limited semantic metadata, so that there is no standard way to record column units, descriptions, UCDs etc. By default, additional metadata is written in the form of a DATA-less VOTable attached to the file footer, as described by the VOParquet convention. This additional metadata can then be retrieved by other VOParquet-aware software.

Note:

The parquet I/O handlers require large external libraries, which are not always bundled with the library/application software because of their size. In some configurations, parquet support may not be present, and attempts to read or write parquet files will result in a message like:

```
Parquet-mr libraries not available
```

If you can supply the relevant libaries on the classpath at runtime, the parquet support will work. At time of writing, the required libraries are included in the `topcat-extra.jar` monolithic jar file (though not `topcat-full.jar`), and are included if you have the `topcat-all.dmg` file. They can also be found in the starjava github repository (https://github.com/Starlink/starjava/tree/master/parquet/src/lib or you can acquire them from the Parquet MR package. These arrangements may be revised in future releases, for instance if parquet usage becomes more mainstream. The required dependencies are a minimal subset of those required by the Parquet MR submodule `parquet-cli` at version 1.13.1, in particular the files `aircompressor-0.21.jar` `commons-collections-3.2.2.jar` `commons-configuration2-2.1.1.jar` `commons-lang3-3.9.jar` `failureaccess-1.0.1.jar` `guava-27.0.1-jre.jar` `hadoop-auth-3.2.3.jar` `hadoop-common-3.2.3.jar` `hadoop-mapreduce-client-core-3.2.3.jar` `htrace-core4-4.1.0-incubating.jar` `parquet-cli-1.13.1.jar` `parquet-column-1.13.1.jar` `parquet-common-1.13.1.jar` `parquet-encoding-1.13.1.jar` `parquet-format-structures-1.13.1.jar` `parquet-hadoop-1.13.1.jar` `parquet-jackson-1.13.1.jar` `slf4j-api-1.7.22.jar` slf4j-nop-1.7.22.jar snappy-java-1.1.8.3.jar stax2-api-4.2.1.jar woodstox-core-5.3.0.jar zstd-jni-1.5.0-1.jar.

These libraries support some, but not all, of the compression formats defined for parquet, currently `uncompressed`, `gzip`, `snappy`, `zstd` and `lz4_raw`. Supplying more of the parquet-mr dependencies at runtime would extend this list. Unlike the rest of TOPCAT/STILTS/STIL which is written in pure java, some of these libraries (currently the snappy and zstd compression codecs) contain native code, which means they may not work on all architectures. At time of writing all common architectures are covered, but there is the possibility of failure with a `java.lang.UnsatisfiedLinkError` on other platforms if attempting to read/write files that use those compression algorithms.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`parquet(votmeta=false,compression=gzip)`". The following options are available:

`votmeta = true|false`
If true, rich metadata for the table will be written out in the form of a DATA-less VOTable that is stored in the parquet extra metadata key-value list under the key `IVOA.VOTable-Parquet.content`, according to the VOParquet convention (version 1.0). This enables items such as Units, UCDs and column descriptions, that would otherwise be lost in the serialization, to be stored in the output parquet file. This information can then be recovered by parquet readers that understand this convention. (Default: `true`)

`compression = uncompressed|snappy|zstd|gzip|lz4_raw`
Configures the type of compression used for output. Supported values are probably `uncompressed`, `snappy`, `zstd`, `gzip` and `lz4_raw`. Others may be available if the relevant codecs are on the classpath at runtime. If no value is specified, the parquet-mr library default is used, which is probably `uncompressed`. (Default: `null`)

`groupArray = true|false`
Controls the low-level detail of how array-valued columns are written. For an array-valued int32 column named IVAL, `groupArray=false` will write it as "`repeated int32 IVAL`" while `groupArray=true` will write it as "`optional group IVAL (LIST) {repeated group list {optional int32 element}}`".

Although setting it `false` may be slightly more efficient, the default is `true`, since if any of the columns have array values that either may be null or may have elements which are null, groupArray-style declarations for all columns are required by the Parquet file format:

"A repeated field that is neither contained by a LIST- or MAP-annotated group nor annotated by LIST or MAP should be interpreted as a required list of required elements where the element type is the type of the field. Implementations should use either LIST and MAP annotations or unannotated repeated fields, but not both. When using the annotations, no unannotated repeated types are allowed."

If this option is set false and an attempt is made to write null arrays or arrays with null values, writing will fail. (Default: `true`)

`usedict = true|false|null`
Determines whether dictionary encoding is used for output. This will work well to compress the output for columns with a small number of distinct values. Even when this setting is true, dictionary encoding is abandoned once many values have been encountered (the dictionary gets too big). If no value is specified, the parquet-mr library default is used, which is probably `true`. (Default: `null`)

If no output format is explicitly chosen, writing to a filename with the extension "`.parquet`" or

.parq" (case insensitive) will select parquet format for output.

4.1.2.8 Feather

The Feather file format is a column-oriented binary disk-based format based on Apache Arrow and supported by (at least) Python, R and Julia. Some description of it is available at https://github.com/wesm/feather and https://blog.rstudio.com/2016/03/29/feather/. It can be used for large datasets, but it does not support array-valued columns. It can be a useful format to use for exchanging data with R, for which FITS I/O is reported to be slow.

This writer is somewhat experimental; please report problems if you encounter them.

If no output format is explicitly chosen, writing to a filename with the extension ".fea" or ".feather" (case insensitive) will select feather format for output.

4.1.2.9 Text

Writes tables in a simple text-based format designed to be read by humans. No reader exists for this format.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "text(maxCell=40,maxParam=160)". The following options are available:

maxCell = <int>
Maximum width in characters of an output table cell. Cells longer than this will be truncated. (Default: 40)

maxParam = <int>
Maximum width in characters of an output table parameter. Parameters with values longer than this will be truncated. (Default: 160)

params = true|false
Whether to output table parameters as well as row data. (Default: true)

sampledRows = <int>
The number of rows examined on a first pass of the table to determine the width of each column. Only a representative number of rows needs to be examined, but if a formatted cell value after this limit is wider than the cells up to it, then such later wide cells may get truncated. If the value is <=0, all rows are examined in the first pass; this is the default, but it can be configured to some other value if that takes too long. (Default: 0)

Multiple tables may be written to a single output file using this format.

An example looks like this:

```
Table name: animals.vot
Description: Some animals
Author: Mark Taylor
+-------+-----------+---------------+------+--------+--------+
| RECNO | SPECIES   | NAME          | LEGS | HEIGHT | MAMMAL |
+-------+-----------+---------------+------+--------+--------+
| 1     | pig       | Pigling Bland | 4    | 0.8    | true   |
| 2     | cow       | Daisy         | 4    | 2.0    | true   |
| 3     | goldfish  | Dobbin        |      | 0.05   | false  |
| 4     | ant       |               | 6    | 0.001  | false  |
| 5     | ant       |               | 6    | 0.001  | false  |
| 6     | queen ant | Ma'am         | 6    | 0.002  | false  |
| 7     | human     | Mark          | 2    | 1.8    | true   |
+-------+-----------+---------------+------+--------+--------+
```

4.1.2.10 HTML

Writes a basic HTML `TABLE` element suitable for use as a web page or for insertion into one.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`html(maxCell=200,standalone=false)`". The following options are available:

`maxCell = <int>`
Maximum width in characters of an output table cell. Cells longer than this will be truncated. (Default: `200`)

`standalone = true|false`
If true, the output is a freestanding HTML document complete with HTML, HEAD and BODY tags. If false, the output is just a TABLE element. (Default: `false`)

Multiple tables may be written to a single output file using this format.

If no output format is explicitly chosen, writing to a filename with the extension "`.html`" or "`.htm`" (case insensitive) will select `HTML` format for output.

An example looks like this:

```
<TABLE BORDER='1'>
<CAPTION><STRONG>animals.vot</STRONG></CAPTION>
<THEAD>
<TR> <TH>RECNO</TH> <TH>SPECIES</TH> <TH>NAME</TH> <TH>LEGS</TH> <TH>HEIGHT</TH> <TH>MAMMAL</T
<TR> <TH>&nbsp;</TH> <TH>&nbsp;</TH> <TH>&nbsp;</TH> <TH>&nbsp;</TH> <TH>(m)</TH> <TH>&nbsp;</
<TR><TD colspan='6'></TD></TR>
</THEAD>
<TBODY>
<TR> <TD>1</TD> <TD>pig</TD> <TD>Pigling Bland</TD> <TD>4</TD> <TD>0.8</TD> <TD>true</TD></TR
<TR> <TD>2</TD> <TD>cow</TD> <TD>Daisy</TD> <TD>4</TD> <TD>2.0</TD> <TD>true</TD></TR>
<TR> <TD>3</TD> <TD>goldfish</TD> <TD>Dobbin</TD> <TD>&nbsp;</TD> <TD>0.05</TD> <TD>false</TD
<TR> <TD>4</TD> <TD>ant</TD> <TD>&nbsp;</TD> <TD>6</TD> <TD>0.001</TD> <TD>false</TD></TR>
<TR> <TD>5</TD> <TD>ant</TD> <TD>&nbsp;</TD> <TD>6</TD> <TD>0.001</TD> <TD>false</TD></TR>
<TR> <TD>6</TD> <TD>queen ant</TD> <TD>Ma&apos;am</TD> <TD>6</TD> <TD>0.002</TD> <TD>false</TD
<TR> <TD>7</TD> <TD>human</TD> <TD>Mark</TD> <TD>2</TD> <TD>1.8</TD> <TD>true</TD></TR>
</TBODY>
</TABLE>
```

4.1.2.11 LaTeX

Writes a table as a LaTeX `tabular` environment, suitable for insertion into a document intended for publication. This is only likely to be useful for fairly small tables.

The handler behaviour may be modified by specifying one or more comma-separated name=value configuration options in parentheses after the handler name, e.g. "`latex(standalone=false)`". The following options are available:

`standalone = true|false`
If true, the output is a freestanding LaTeX document consisting of a `tabular` environment within a `table` within a `document`. If false, the output is just a `tabular` environment. (Default: `false`)

If no output format is explicitly chosen, writing to a filename with the extension "`.tex`" (case insensitive) will select `LaTeX` format for output.

An example looks like this:

```
\begin{tabular}{|r|l|l|r|r|l|}
\hline
  \multicolumn{1}{|c|}{RECNO} &
  \multicolumn{1}{c|}{SPECIES} &
  \multicolumn{1}{c|}{NAME} &
  \multicolumn{1}{c|}{LEGS} &
  \multicolumn{1}{c|}{HEIGHT} &
  \multicolumn{1}{c|}{MAMMAL} \\
\hline
  1 & pig & Pigling Bland & 4 & 0.8 & true\\
  2 & cow & Daisy & 4 & 2.0 & true\\
  3 & goldfish & Dobbin &  & 0.05 & false\\
  4 & ant &  & 6 & 0.001 & false\\
  5 & ant &  & 6 & 0.001 & false\\
  6 & queen ant & Ma'am & 6 & 0.002 & false\\
  7 & human & Mark & 2 & 1.8 & true\\
\hline\end{tabular}
```

4.1.2.12 Tab-Separated Table

Tab-Separated Table, or TST, is a text-based table format used by a number of astronomical tools including Starlink's GAIA and ESO's SkyCat on which it is based. A definition of the format can be found in Starlink Software Note 75.

If no output format is explicitly chosen, writing to a filename with the extension "`.tst`" (case insensitive) will select `TST` format for output.

An example looks like this:

```
animals.vot

# Table parameters
Description: Some animals
Author: Mark Taylor

# Attempted guesses about identity of columns in the table.
# These have been inferred from column UCDs and/or names
# in the original table data.
# The algorithm which identifies these columns is not particularly reliable,
# so it is possible that these are incorrect.
id_col: 2
ra_col: -1
dec_col: -1

# This TST file generated by STIL v4.3-2

RECNO SPECIES NAME LEGS HEIGHT MAMMAL
----- ------- ---- ---- ------ ------
1 pig Pigling Bland 4 0.8 true
2 cow Daisy 4 2.0 true
3 goldfish Dobbin         0.05 false
4 ant         6 0.001 false
5 ant         6 0.001 false
6 queen ant Ma'am 6 0.002 false
7 human Mark 2 1.8 true
[EOD]
```

4.1.2.13 Mirage Format

Mirage was a nice standalone tool for analysis of multidimensional data, from which TOPCAT took some inspiration. It was described in a 2007 paper 2007ASPC..371..391H, but no significant development seems to have taken place since then. This format is therefore probably obsolete, but you can still write table output in Mirage-compatible format if you like.

If no output format is explicitly chosen, writing to a filename with the extension "`.mirage`" (case insensitive) will select `mirage` format for output.

An example looks like this:

```
#
# Written by uk.ac.starlink.mirage.MirageFormatter
# Omitted column 5: MAMMAL(Boolean)
#
# Column names
format var RECNO SPECIES NAME LEGS HEIGHT
#
# Text columns
format text SPECIES
format text NAME
#
# Table data
1 pig Pigling_Bland 4 0.8
2 cow Daisy 4 2.0
3 goldfish Dobbin <blank> 0.05
4 ant <blank> 6 0.001
5 ant <blank> 6 0.001
6 queen_ant Ma'am 6 0.002
7 human Mark 2 1.8
```

4.1.3 Non-standard FITS conventions

STIL, the I/O library underlying TOPCAT, uses a few private conventions when writing and reading FITS files. These are not private in the sense that non-STIL code is prevented from cooperating with them, but STIL does not assume that other code, or FITS tables it encounters, will use these conventions. Instead, they offer (in some cases) added value for tables that were written by STIL and are subsequently re-read by STIL, while causing the minimum of trouble for non-STIL readers.

4.1.3.1 FITS-plus

When writing tables to FITS BINTABLE format, STIL can optionally store additional metadata in the FITS file using a private convention known as "FITS-plus". The table is written exactly as usual in a BINTABLE extension HDU, but the primary HDU (HDU#0) contains a sequence of characters, stored as a 1-d array of bytes using UTF-8 encoding, which forms the text of a DATA-less VOTable document. Note that the Primary HDU cannot be used to store table data, so under normal circumstances has no interesting content in a FITS file used just for table storage. The FITS tables in the subsequent extensions are understood to contain the data.

The point of this is that the VOTable can contain all the rich metadata about the table(s), but the bulk data are in a form which can be read efficiently. Crucially, the resulting FITS file is a perfectly good FITS table on its own, so non-VOTable-aware readers can read it in just the usual way, though of course they do not benefit from the additional metadata stored in the VOTable header.

In practice, STIL normally writes FITS files using this convention (it writes the VOTable metadata into the Primary HDU) and when reading a FITS files it looks for use of this convention (examines the Primary HDU for VOTable metadata and uses it if present). But if an input file does not follow this convention, the metadata is taken directly from the BINTABLE header as normal. Non-FITS-plus-aware (i.e. non-STIL) readers will ignore the Primary HDU, since it has no purpose in a standard FITS file containing only a table, and it doesn't look like anything else that such readers are usually expecting. The upshot is that for nearly all purposes you can forget about use of this convention when writing and reading FITS tables using STIL and other libraries, but STIL may be able to recover rich metadata from files that it has written itself.

To be recognised as a FITS-plus file, the Primary HDU (and hence the FITS file) must begin like this:

```
SIMPLE  =                    T
BITPIX  =                    8
NAXIS   =                    1
NAXIS1  =                  ???
VOTMETA =                    T
```

The sequence and values of the given header cards must be as shown, except for `NAXIS1` which contains the number of bytes in the data block; any comments are ignored.

The content of the Primary HDU must be a VOTable document containing zero or more `TABLE` elements, one for each BINTABLE extension appearing later in the FITS file. Each such TABLE must *not* contain a `DATA` child; the table content is taken from the BINTABLE in the next unused table HDU. For instance the Primary HDU content annotating a single table might look like this:

```
<?xml version='1.0'?>
<VOTABLE version="1.3" xmlns="http://www.ivoa.net/xml/VOTable/v1.3">
<RESOURCE>
<TABLE nrows="1000">
  <FIELD datatype="double" name="RA" ucd="pos.eq.ra;meta.main"/>
  <FIELD datatype="double" name="Dec" ucd="pos.eq.dec;meta.main"/>
  <!-- Dummy VOTable - no DATA element here -->
</TABLE>
</RESOURCE>
</VOTABLE>
```

The first extension HDU would then contain the two-column BINTABLE corresponding to the given metadata.

The VOTable metadata MUST be compatible with the structure of the annotated BINTABLE(s) in terms of number and datatypes of columns.

Note: This arrangement bears some similarity to VOTable/FITS encoding, in which the output file is a VOTable which references an inline or external FITS file containing the bulk data. However, the VOTable/FITS format is inconvenient in that either (for in-line data) the FITS file is base64-encoded and so hard to read efficiently especially for random access, or (for referenced data) the table is split across two files.

4.1.3.2 Wide FITS

The FITS BINTABLE standard (FITS Standard v4.0, section 7.3) permits a maximum of 999 columns in a binary table extension. Up to version 3.2 of STIL, attempting to write a table with more than 999 columns using one of the supported FITS-based writers failed with an error. In later versions, a non-standard convention is used which can store wider tables in a FITS BINTABLE extension. The various STIL FITS-based readers can (in their default configurations) read these tables transparently, allowing round-tripping of arbitrarily wide tables to FITS files. Note however that other FITS-compliant software is not in general aware of this convention, and will see a 999-column table. The first 998 columns will appear as intended, but subsequent ones will effectively be hidden.

The rest of this section describes the convention that is used to store tables with more than 999 columns in FITS BINTABLE extensions.

The BINTABLE extension type requires table column metadata to be described using 8-character keywords of the form `TXXXXnnn`, where `TXXXX` represents one of an open set of mandatory, reserved or user-defined root keywords up to five characters in length, for instance `TFORM` (mandatory), `TUNIT` (reserved), `TUCD` (user-defined). The `nnn` part is an integer between 1 and 999 indicating the index of the column to which the keyword in question refers. Since the header syntax confines this indexed part of the keyword to three digits, there is an upper limit of 999 columns in BINTABLE extensions.

Note that the FITS/BINTABLE format does not entail any restriction on the storage of column *data* beyond the 999 column limit in the data part of the HDU, the problem is just that client software cannot be informed about the layout of this data using the header cards in the usual way.

The following convention is used by STIL FITS-based I/O handlers to accommodate wide tables in FITS files:

Definitions:

- *BINTABLE columns* are those columns defined using the FITS BINTABLE standard
- *Data columns* are the columns to be encoded
- `N_TOT` is the total number of data columns to be stored
- Data columns with (1-based) indexes from 999 to `N_TOT` inclusive are known as *extended* columns. Their data is stored within the *container* column.
- BINTABLE column 999 is known as the *container* column. It contains the byte data for all the *extended* columns.

Convention:

- All column data (for columns 1 to `N_TOT`) is laid out in the data part of the HDU in exactly the same way as if there were no 999-column limit.
- The `TFIELDS` header is declared with the value 999.
- The container column is declared in the header with some `TFORM999` value corresponding to the total field length required by all the extended columns ('`B`' is the obvious data type, but any legal `TFORM` value that gives the right width MAY be used). The byte count implied by `TFORM999` MUST be equal to the total byte count implied by all extended columns.
- Other `TXXXX999` headers MAY optionally be declared to describe the container column in accordance with the usual rules, e.g. `TTYPE999` to give it a name.
- The `NAXIS1` header is declared in the usual way to give the width of a table row in bytes. This is equal to the sum of all the BINTABLE column widths as usual. It is also equal to the sum of all the data column widths, which has the same value.
- Headers for Data columns 1-998 are declared as usual, corresponding to BINTABLE columns 1-998.
- Keyword `XT_ICOL` indicates the index of the container column. It MUST be present with the integer value 999 to indicate that this convention is in use.
- Keyword `XT_NCOL` indicates the total number of data columns encoded. It MUST be present with an integer value equal to `N_TOT`.
- Metadata for each extended column is encoded with keywords of the form '`HIERARCH XT TXXXXnnnnn`', where `TXXXX` are the same keyword roots as used for normal BINTABLE extensions, and `nnnnn` is a decimal number written as usual (no leading zeros, as many digits as are required). Thus the formats for data columns 999, 1000, 1001 etc are declared with the keywords `HIERARCH XT TFORM999`, `HIERARCH XT TFORM1000`, `HIERARCH XT TFORM1001`, etc. Note this uses the ESO HIERARCH convention described at https://fits.gsfc.nasa.gov/registry/hierarch_keyword.html. The *name space* token has been chosen as '`XT`' (extended table).
- This convention MUST NOT be used for `N_TOT`<=999.

The resulting HDU is a completely legal FITS BINTABLE extension. Readers aware of this convention may use it to extract column data and metadata beyond the 999-column limit. Readers unaware of this convention will see 998 columns in their intended form, and an additional (possibly large) column 999 which contains byte data but which cannot be easily interpreted.

An example header might look like this:

```
XTENSION= 'BINTABLE'           /  binary table extension
BITPIX  =                    8 /  8-bit bytes
NAXIS   =                    2 /  2-dimensional table
NAXIS1  =                 9229 /  width of table in bytes
NAXIS2  =                   26 /  number of rows in table
PCOUNT  =                    0 /  size of special data area
GCOUNT  =                    1 /  one data group
TFIELDS =                  999 /  number of columns
XT_ICOL =                  999 /  index of container column
XT_NCOL =                 1204 /  total columns including extended
TTYPE1  = 'posid_1 '           /  label for column 1
TFORM1  = 'J       '           /  format for column 1
TTYPE2  = 'instrument_1'       /  label for column 2
TFORM2  = '4A      '           /  format for column 2
TTYPE3  = 'edge_code_1'        /  label for column 3
TFORM3  = 'I       '           /  format for column 3
TUCD3   = 'meta.code.qual'
 ...
TTYPE998= 'var_min_s_2'        /  label for column 998
TFORM998= 'D       '           /  format for column 998
TUNIT998= 'counts/s'           /  units for column 998
TTYPE999= 'XT_MORECOLS'        /  label for column 999
TFORM999= '813I    '           /  format for column 999
HIERARCH XT TTYPE999         = 'var_min_u_2' / label for column 999
HIERARCH XT TFORM999         = 'D' / format for column 999
HIERARCH XT TUNIT999         = 'counts/s' / units for column 999
HIERARCH XT TTYPE1000        = 'var_prob_h_2' / label for column 1000
HIERARCH XT TFORM1000        = 'D' / format for column 1000
 ...
HIERARCH XT TTYPE1203        = 'var_prob_w_2' / label for column 1203
HIERARCH XT TFORM1203        = 'D' / format for column 1203
HIERARCH XT TTYPE1204        = 'var_sigma_w_2' / label for column 1204
HIERARCH XT TFORM1204        = 'D' / format for column 1204
HIERARCH XT TUNIT1204        = 'counts/s' / units for column 1204
END
```

This general approach was suggested by William Pence on the FITSBITS list in June 2012, and by François-Xavier Pineau (CDS) in private conversation in 2016. The details have been filled in by Mark Taylor (Bristol), and discussed in some detail on the FITSBITS list in July 2017.

4.1.4 Custom I/O Formats

It is in principle possible to configure TOPCAT to work with table file formats and schemes other than the ones listed in this section. It does not require any upgrade of TOPCAT itself, but you have to write or otherwise acquire an input/output/scheme handler for the table format in question.

The steps that you need to take are:

1. Write java classes which constitute your input/output/scheme handler. Such classes must have a no-arg constructor and implement the appropriate interface: `uk.ac.starlink.table.TableBuilder` for input handlers, `uk.ac.starlink.table.StarTableWriter` for output handlers, `uk.ac.starlink.table.TableScheme` for scheme handlers.
2. Ensure that these classes are available on your classpath while TOPCAT is running (see Section 10.2.1).
3. Set the relevant system property to the name of the handler class (see Section 10.2.3): `startable.readers` for input handlers, `startable.writers` for output handlers, `startable.schemes` for scheme handlers.

Explaining how to write such handlers is beyond the scope of this document - see the user document and javadocs for STIL (http://www.starlink.ac.uk/stil/).

4.2 Input Locations

In many cases loading tables will be done using GUI dialogues such as the Filestore Load Window, where you just need to click on a filename or directory to indicate the load location. However in some cases, for instance specifying tables on the command line (Section 10.1) or typing pathnames directly into the Load Window **Location** field, you may want give the location of an input table using only a single string.

Most of the time you will just want to type in a filename; either an absolute pathname or one relative to TOPCAT's starting directory can be used. However, TOPCAT also supports direct use of URLs, including ones using some specialised protocols. Here is the list of URL types allowed:

`http:`
Read from an HTTP resource.

`https:`
Read from an HTTPS resource.

`ftp:`
Read from an anonymous FTP resource.

`file:`
Read from a local file, using the syntax `file:///path/to/file`. This is similar to specifying the filename directly, but there is a difference: using this form forces reads to be sequential rather than random access, which may allow you to experience a different set of performance characteristics and bugs.

`jar:`
Specialised protocol for looking inside Java Archive files - see JarURLConnection documentation.

`myspace:`
(Obsolete?) Accesses files in the AstroGrid "MySpace" virtual file store. These URLs look something like "`myspace:/survey/iras_psc.xml`", and can access files in the myspace are that the user is currently logged into. These URLs can be used for both input and output of tables. To use them you must have an AstroGrid account and the AstroGrid WorkBench or similar must be running; if you're not currently logged in a dialogue will pop up to ask you for name and password.

`ivo:`
(Obsolete?) Understands ivo-type URLs which signify files in the AstroGrid "MySpace" virtual file store. These URLs look something like "`ivo://uk.ac.le.star/filemanager#node-2583`". These URLs can be used for both input and output of tables. To use them you must have an AstroGrid account and the AstroGrid WorkBench or similar must be running; if you're not currently logged in a dialogue will pop up to ask you for name and password.

`jdbc:`
JDBC URLs may be used, but they don't work in the same way as the others listed here, since they do not reference an input byte stream. See instead Section 4.3.3.

From the command line it is also possible to use the special value "`-`" to mean standard input; in this case the file format must be specified explicitly using the `-f` flag, and not all formats can be streamed from stdin. Finally, the form "`<syscmd`" or equivalently "`syscmd|`" may be used to read from the standard output of a shell pipeline (probably only works on Unix-like systems).

As with the GUI-based load dialogues, data compression in any of the supported formats (gzip, bzip2, Unix compress) is detected and dealt with automatically for input locations.

Note that tables can also be supplied by name from non-serialized sources, as described in Input Schemes.

4.3 Input Schemes

As well as being able to load tables from external data streams, TOPCAT offers a way to specify tables that do not correspond to a stream of bytes. These may be defined programmatically or interact with external services in some way that is not as straightforward as decoding a stream of bytes.

Such tables are defined with a *scheme specification* string, of the form:

```
   :<scheme-name>:<scheme-specific-part>
```

so that for instance "`:skysim:1e6`" specifies a million-row simulated star catalogue, as described by the skysim scheme documentation below.

At present, the only ways that scheme-specified tables can be loaded are:

- naming them on the command line at startup
- entering them in the **Location** field in the Load Window

Some more GUI-friendly way to load them may be introduced in future releases.

The following subsections describe all the schemes that are available by default. It is also possible to add new schemes at runtime by using the `startable.schemes` system property.

4.3.1 `skysim`

Usage: `:skysim:<nrow>`

Generates a simulated all-sky star catalogue with a specified number of rows. This is intended to provide crude test catalogues when no suitable real dataset of the required size is available. In the current implementation the row count, is the only parameter, so the specification "`:skysim:5e6`" would give a 5 million-row simulated catalogue. The row count may be given as a plain integer (`1000`), or with embedded underscores (`1_000`), or in exponential format (`1e3`).

The current implementation provides somewhat realistic position-dependent star densities and distributions of magnitudes and colours based on positionally averaged values from Gaia EDR3. The source positions do not correspond to actual stars. The columns and the statistics on which the output is based may change in future releases.

Example:

```
:skysim:6
+-----------+------------+------------+------------+-----------+-----------+------------+
| ra        | dec        | l          | b          | gmag      | rmag      | b_r        |
+-----------+------------+------------+------------+-----------+-----------+------------+
| 266.7702  | -32.689117 | -3.044958  | -2.217145  | 18.168278 | 16.03555  | 1.046624   |
| 276.83398 | 8.132022   | 37.491447  | 9.031909   | 18.331224 | 18.786451 | 1.4425725  |
| 92.04118  | 23.79857   | -173.02219 | 1.7854756  | 16.743847 | 15.623316 | 1.690048   |
| 271.08215 | -5.2012086 | 22.848532  | 8.022958   | 21.538874 | 17.782997 | 2.1645386  |
| 298.83368 | 31.401922  | 67.75605   | 1.6348709  | 20.145718 | 17.728764 | 1.1521724  |
| 204.9299  | -77.07571  | -54.29949  | -14.464215 | 19.044079 | 20.277771 | 0.92987275 |
+-----------+------------+------------+------------+-----------+-----------+------------+
```

4.3.2 `attractor`

Usage: `:attractor:<nrow>[,(clifford[,a,b,c,d]|rampe[,a,b,c,d,e,f]|henon[,a,b,c])]`

Generates tables listing points sampled from one of a specified family of strange attractors. These can provide tables with (X,Y) or (X,Y,Z) columns and arbitrarily many rows. They can be used, for instance, to make (beautiful) example large-scale scatter plots in 2-d or 3-d space.

The specification syntax is of the form `:attractor:<nrow>,<family-name>[,<args>]` where `<nrow>` is the number of rows required, `<family-name>` is the name of one of the supported families of attractors, and `<args>` is an optional comma-separated list of numeric arguments specifying the family-specific parameters of the required attractor. If the `<args>` part is omitted, an example attractor from the family is used. Note that picking `<args>` values at random will often result in rather boring (non-strange) attractors.

The `<nrow>` argument may be given as a plain integer (`1000`), or with embedded underscores (`1_000`), or in exponential format (`1e3`).

The following families are currently supported:

clifford

clifford attractors are 2-dimensional and have 4 parameters, with suggested values in the range +/-2.0.

The iteration is defined by the equations:

```
x' = sin(a*y) + c * cos(a*x)
y' = sin(b*x) + d * cos(b*y)
```

Examples:

- `:attractor:9999,clifford`
- `:attractor:1_000_000,clifford,1.32,-1.44,-1.7,-1.58`
- `:attractor:65536,clifford,1.27,-1.35,0.82,1.8`
- `:attractor:1e7,clifford,-1.9,1.18,-1.21,1.07`
- `:attractor:400,clifford,-1.27,-1.28,1.0,-1.26`
- `:attractor:4e6,clifford,1.8,0.9,-1.8,0.8`

rampe

rampe attractors are 3-dimensional and have 6 parameters, with suggested values in the range +/-2.0.

The iteration is defined by the equations:

```
x' = x * z * sin(a*x) - cos(b*y)
y' = y * x * sin(c*y) - cos(d*z)
z' = z * y * sin(e*z) - cos(f*x)
```

Examples:

- `:attractor:10e6,rampe`
- `:attractor:4,rampe,-1.81,1.35,-0.85,0.32,1.68,-1.62`
- `:attractor:5.5e5,rampe,0.23,-1.77,1.32,-1.44,-1.7,-1.58`
- `:attractor:9999,rampe,-0.3,1.78,-0.87,1.69,1.42,1.21`
- `:attractor:1_000_000,rampe,1.42,-1.98,0.39,1.32,1.79,-0.37`

henon

henon attractors are 2-dimensional and have 3 parameters, with suggested values in the range +/-2.0.

The iteration is defined by the equations:

```
x' = y + a + b*x*x
y' = c*x
```

Examples:

- `:attractor:65536,henon`
- `:attractor:1e7,henon,-0.68,1.64,0.36`
- `:attractor:400,henon,1.73,0.29,-0.99`
- `:attractor:4e6,henon,0.88,-0.9,-0.93`
- `:attractor:10e6,henon,1.4,-1.13,-0.01`

Example:

```
:attractor:6,rampe
+----------------------+---------------------+----------------------+
| x                    | y                   | z                    |
+----------------------+---------------------+----------------------+
| -0.5759098296568739  | 0.09844750286352466 | -0.6712534741282851  |
| -1.3295344852011892  | -0.9829776649068059 | -0.7814409891660122  |
| -1.1910376215054008  | 0.04335596646295736 | -1.0308958690758545  |
| -2.0144704755218514  | -0.9699626185329038 | -0.35169532148364757 |
| -0.16145296509226564 | 0.5245428249077974  | 0.17929370340580017  |
| -0.8409807675257591  | -0.9598486078341374 | -0.955769158222801   |
+----------------------+---------------------+----------------------+
```

4.3.3 `jdbc`

Usage: `:jdbc:<jdbc-part>`

Interacts with the JDBC system (JDBC sort-of stands for Java DataBase Connectivity) to execute an SQL query on a connected database. The `jdbc:...` specification is the JDBC URL. For historical compatibility reasons, specifications of this scheme may omit the leading colon character, so that the following are both legal, and are equivalent:

```
   jdbc:mysql://localhost/dbl#SELECT TOP 10 ra, dec FROM gsc
   :jdbc:mysql://localhost/dbl#SELECT TOP 10 ra, dec FROM gsc
```

In order for this to work, you must have access to a suitable database with a JDBC driver, and some standard JDBC configuration is required to set the driver up. The following steps are necessary:

1. the driver class must be available on the runtime classpath
2. the `jdbc.drivers` system property must be set to the driver classname

More detailed information about how to set up the JDBC system to connect with an available database, and of how to construct JDBC URLs, is provided elsewhere in the documentation.

4.3.4 `loop`

Usage: `:loop:<count>|<start>,<end>[,<step>]`

Generates a table whose single column increments over a given range.

The specification may either be a single value N giving the number of rows, which yields values in the range 0..N-1, or two or three comma-separated values giving the *start*, *end* and optionally *step* corresponding to the conventional specification of a loop variable.

The supplied numeric parameters are interpreted as floating point values, but the output column type will be 32- or 64-bit integer or 64-bit floating point, depending on the values that it has to take. Any embedded underscores will be ignored.

Examples:

- `:loop:5`: a 5-row table whose integer column has values 0, 1, 2, 3, 4
- `:loop:10,20`: a 10-row table whose integer column has values 10, 11, ... 19
- `:loop:1,2,0.25`: a 10-row table whose floating point column has values 1.00, 1.25, 1.50, 1.75
- `:loop:1e10`: a ten billion row table, with 64-bit integer values

Example:

```
:loop:6
+---+
| i |
+---+
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+---+
```

4.3.5 `test`

Usage: `:test:[<nrow>[,<opts-ibsfgvwmk*>]]`

Generates a table containing test data. The idea is to include columns of different data types, for instance to provide an example for testing I/O handler implementations. The columns will contain some variety of more or less meaningless values, but the content is reproducible between runs, so the same specification will produce the same output each time. Updates of the implementation might change the output however, so the output is not guaranteed to be the same for all time.

The table specification has two comma-separated parameters:

- `<nrow>`: row count
- `<opts>`: a string of letter options specifying what types of data will be included; options are:
  - **i**: an integer index column
  - **b**: a few basic columns
  - **s**: a selection of typed scalar columns
  - **f**: a selection of fixed-length 1-d array columns
  - **g**: a selection of fixed-length 1-d array columns excluding strings
  - **v**: a selection of variable-length 1-d array columns
  - **w**: a selection of variable-length 1-d array columns excluding strings
  - **m**: a selection of multi-dimensional array columns
  - **k**: almost a thousand columns
  - *****: equivalent to all of the above

If `<opts>` and/or `<nrow>` are omitted, some default values are used.

The `<nrow>` argument may be given as a plain integer (`1000`), or with embedded underscores (`1_000`), or in exponential format (`1e3`).

Example:

```
:test:10,is
+---------+--------+---------+-------+--------+---------+----------+----------------+---------
| i_index | s_byte | s_short | s_int | s_long | s_float | s_double | s_string       | s_boolea
+---------+--------+---------+-------+--------+---------+----------+----------------+---------
| 0       | 0      | 0       | 0     | 0      | 0.0     | 0.0      | zero           | false
| 1       |        | 1       | 1     | 1      | 1.0     | 1.0      | one            | true
| 2       | 2      |         | 2     | 2      | 2.0     | 2.0      | two            | false
| 3       | 3      | 3       |       | 3      | 3.0     | 3.0      | three          | true
| 4       | 4      | 4       | 4     |        | 4.0     | 4.0      | four           | false
| 5       | 5      | 5       | 5     | 5      |         | 5.0      | five           | true
| 6       | 6      | 6       | 6     | 6      | 6.0     |          | six            | false
| 7       | 7      | 7       | 7     | 7      | 7.0     | 7.0      |                | true
| 8       | 8      | 8       | 8     | 8      | 8.0     | 8.0      | ' "\""' ; '&<> |
| 9       | 9      | 9       | 9     | 9      |         |          |                | true
+---------+--------+---------+-------+--------+---------+----------+----------------+--------
```

4.3.6 `class`

Usage: `:class:<TableScheme-classname>:<scheme-spec>`

Uses an instance of a named class that implements the `uk.ac.starlink.table.TableScheme` interface and that has a no-arg constructor. Arguments to be passed to an instance of the named class are appended after a colon following the classname.

For example, the specification "`:class:uk.ac.starlink.table.LoopTableScheme:10`" would return a table constructed by the code `new uk.ac.starlink.table.LoopTableScheme().createTable("10")`.

Example:

```
:class:uk.ac.starlink.table.LoopTableScheme:5
+---+
| i |
+---+
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
+---+
```

4.3.7 `hapi`

Usage:
`:hapi:<server-url>;<dataset>;start=<start>;stop=<stop>[;maxChunk=<n>][;failOnLimit=<true|false`

Generates a table by interacting with a HAPI service. HAPI, the Heliophysics Data Application Programmer's Interface is a protocol for serving streamed time series data.

In most cases it is not essential to use this scheme, since pointing the HAPI table input handler at a URL with suitable parameters will be able to read the data, but this scheme provides some added value by negotiating with the server to make sure that the correct version-sensitive request parameter names and the most efficient data stream format are used, and can split the request into multiple chunks if the service rejects the whole query as too large.

The first token in the specification is the base URL of the HAPI service, the second is the dataset identifier, and others, as defined by the HAPI protocol, are supplied as `<name>=<value>` pairs, separated by a semicolon ("`;`") or an ampersand ("`&`"). The `start` and `stop` parameters, giving ISO-8601-like bounds for the interval requested, are required.

Additionally, some parameters may be supplied which affect load behaviour but are not transmitted to the HAPI service. These are:

maxChunk=<n>
divides the request up into at most `<n>` smaller chunks if the server refuses to supply the whole range at once.

failOnLimit=<true|false>
determines what happens if the service does refuse to serve the whole range (in chunks or otherwise); if true, the table load will fail, but if false as many rows as are available will be loaded.

Some variant syntax is permitted; an ampersand ("`&`") may be used instead of a semicolon to separate tokens, and the names "`time.min`" and "`time.max`" may be used in place of "`start`" and "`stop`".

Note that since semicolons and/or ampersands form part of the syntax, and these characters have special meaning in some contexts, it may be necessary to quote the scheme specification on the command line.

Example:

```
:hapi:https://vires.services/hapi;GRACE_A_MAG;start=2009-01-01T00:00:00;stop=2009-01-01T00:00:
+--------------------------+---------------+---------------+
| Timestamp                | Latitude      | Longitude     |
+--------------------------+---------------+---------------+
| 2009-01-01T00:00:03.607Z | -74.136357526 | -78.905620222 |
| 2009-01-01T00:00:05.607Z | -74.009378676 | -78.884853931 |
| 2009-01-01T00:00:06.607Z | -73.945887793 | -78.874590667 |
| 2009-01-01T00:00:07.607Z | -73.882397005 | -78.864406236 |
| 2009-01-01T00:00:08.607Z | -73.818903534 | -78.854396448 |
+--------------------------+---------------+---------------+
```

5 Joins and Matches

TOPCAT allows you to join two or more tables together to produce a new one in a variety of ways, and also to identify "similar" rows within a single table according to their cell contents. This section describes the facilities for performing these related operations.

There are two basic ways to join tables together: top-to-bottom and side-by-side. A top-to-bottom join (which here I call **concatenation**) is fairly straightforward in that it just requires you to decide which columns in one table correspond to which columns in the other. A side-by-side join is more complicated - it is rarely the case that row *i* in the first table should correspond to row *i* in the second one, so it is necessary to provide some criteria for deciding which (if any) row in the second table corresponds to a given row in the first. In other words, some sort of **matching** between rows in different tables needs to take place. This corresponds to what is called a *join* in database technology. Matching rows within a single table is a useful operation which involves many of the same issues, so that is described here too.

5.1 Concatenating Tables

Two tables can be concatenated using the Concatenation Window, which just requires you to specify the two tables to be joined, and for each column in the first ("Base") table, which column in the second ("Appended") table (if any) corresponds to it. The Apparent Table (Section 3) is used in each case. The resulting table, which is added to the list of known tables in the Control Window, has the same columns as the Base table, and a number of rows equal to the sum of the number of rows in the Base and Appended tables.

As a very simple example, concatenating these two tables:

```
   Messier   RA       Dec      Name
   -------   --       ---      ----
   97        168.63   55.03    Owl Nebula
   101       210.75   54.375   Pinwheel Galaxy
   64        194.13   21.700   Black Eye Galaxy
```

and

```
   RA2000    DEC2000   ID
   ------    -------   --
   185.6     58.08     M40
   186.3     18.20     M85
```

with the assignments RA->RA2000, Dec->DEC2000 and Messier->ID would give:

```
   Messier   RA       Dec      Name
   -------   --       ---      ----
   97        168.63   55.03    Owl Nebula
   101       210.75   54.375   Pinwheel Galaxy
   64        194.13   21.700   Black Eye Galaxy
   M40       185.6    58.08
   M85       183.6    18.20
```

Of course it is the user's responsibility to ensure that the correspondance of columns is sensible (that the two corresponding columns mean the same thing).

You can perform a concatenation using the Concatenation Window; obtain this using the **Joins|Concatenate Tables** ( ) menu option in the Control Window.

5.2 Matching Rows Between Tables

When joining two tables side-by-side you need to identify which row(s) in one correspond to which row(s) in the other. Conceptually, this is done by looking at each row in the first table, somehow identifying in the second table which row "refers to the same thing", and putting a new row in the joined table which consists of all the fields of the row in the first table, followed by all the fields of its matched row in the second table. The resulting table then has a number of columns equal to the sum of the number of columns in both input tables.

In practice, there are a number of complications. For one thing, each row in one table may be matched by zero, one or many rows in the the other. For another, defining what is meant by "referring to the same thing" may not be straightforward. There is also the problem of actually identifying these matches in a relatively efficient way (without explicitly comparing each row in one table with each row in the other, which would be far too slow for large tables).

A common example is the case of matching two object catalogues - suppose we have the following catalogues:

```
    Xpos        Ypos        Vmag
    ----        ----        ----
  1134.822     599.247      13.8
    659.68    1046.874      17.2
    909.613    543.293       9.3
```

and

```
    x            y          Bmag
    -            -          ----
  909.523     543.800       10.1
  1832.114    409.567       12.3
  1135.201    600.100       14.6
   702.622   1004.972       19.0
```

and we wish to combine them to create one new catalogue with a row for each object which appears in both tables. To do this, you have to specify what counts as a match - in this case let's say that a row in one table matches (refers to the same object as) a row in the other if the distance between the positions indicated by their X and Y coordinates matches to within one unit (sqrt((Xpos-x)^2 + (Ypos-y)^2)<=1)). Then the catalogue we will end up with is:

```
    Xpos        Ypos        Vmag    x           y          Bmag
    ----        ----        ----    -           -          ----
  1134.822     599.247      13.8  1135.201    600.100      14.6
    909.613    543.293       9.3   909.523    543.800      10.1
```

There are a number of variations on this however - your match criteria might involve sky coordinates instead of Cartesian ones (or not be physical coordinates at all), you might want to match more than two tables, you might want to identify groups of matching objects in a single table, you might want the output to include rows which don't match as well...

The Match Window allows you to specify

- Which tables are to be matched
- What the criteria are for matching rows
- What to do for rows that don't have exactly one match
- What rows to include in the output table

and to start the matching operation. Depending on the type of match chosen, some additional columns may be appended to the resulting table giving additional details on how the match went. Usually, the 'match score' is one of these; The exact value and meaning of this column depends on the match, but it typically gives the distance between the matched points in some sensible units; the smaller the value, the better the match. You can find out exactly what this score means by examining the column's description in the Columns Window. Columns in the resulting table retain their original names unless that would lead to ambiguity, in which case a disambiguating suffix "_1" or "_2" is added to the column name.

To match two tables, use the **Pair Match** ( ) button in the Control Window; to match more tables than two at once, use the other options on the Control Window's **Join** menu.

5.3 Matching Rows Within a Table

Although the effect is rather different, searching through a single table for rows which match each other (refer to the same object, as explained above) is a similar process and requires much of the same information to be specified, mainly, what counts as a match. You can do this using the Internal Match Window, obtained by using the **Internal Match** ( ) button in the Control Window's **Joins** menu.

5.4 Multi-Object Matches

The matching in TOPCAT is determined by specified match criteria, as described in Appendix A.8.1.1. These criteria give conditions for whether two items (table rows) count as matched with each other. In the case of a pair match, it is clear how this is to be interpreted.

However, some of the matching operations such as the Internal Match search for match groups which may have more than two members. This section explains how TOPCAT applies the pair-wise matching criteria it is given to identifying multi-object groups.

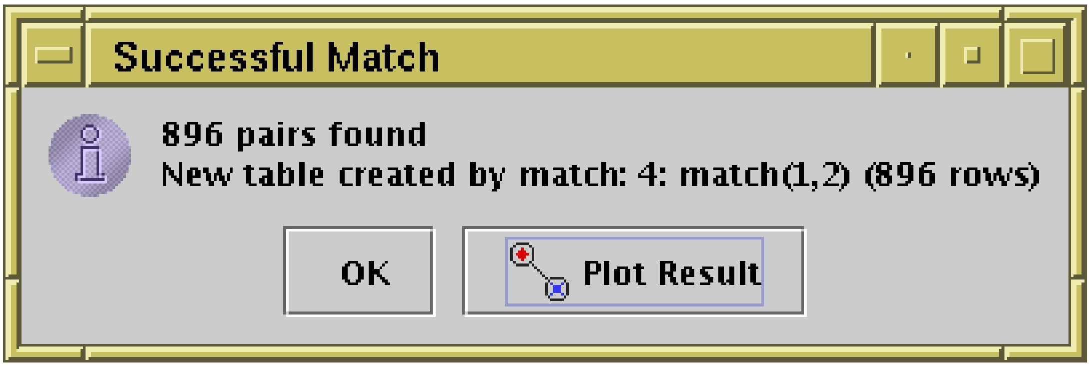
In a multi-object match context, the matcher identifies a matched group as the largest possible group of objects in which each is linked by a pair match to *any* other object in the group - it is a group of "friends of friends". Formally, the set of matched groups is a set of disjoint graphs whose nodes are input table rows and whose edges are successful pair matches, where no successful pair match exists between nodes in different elements of that set. Thus the set has a minimal number of elements, and each of its elements is a matched group of maximal size. The important point to note is that for any particular pair in a matched group, there is no guarantee that the two objects match each other, only that you can hop from one to the other via pairs which do match.

So in the case of a multi-object sky match on a field which is very crowded compared to the specified error radius, it is quite possible for *all* the objects in the input table(s) to end up as part of the same large matching group. Results at or near this percolation threshold are (a) probably not useful and (b) likely to take a long time to run. Some care should therefore be exercised when specifying match criteria in multi-object match contexts.

5.5 Plotting Match Results

Having performed a crossmatch, it is very often a good idea to look at the results graphically. If you can plot the input catalogues, and the output catalogue, with an indication of which rows in the inputs have been joined together, you can rapidly get a good idea of the result of the match, and whether it did what you expected. This is especially true in the presence of crowded fields, tentative match criteria, or other circumstances under which the match might not go as expected.

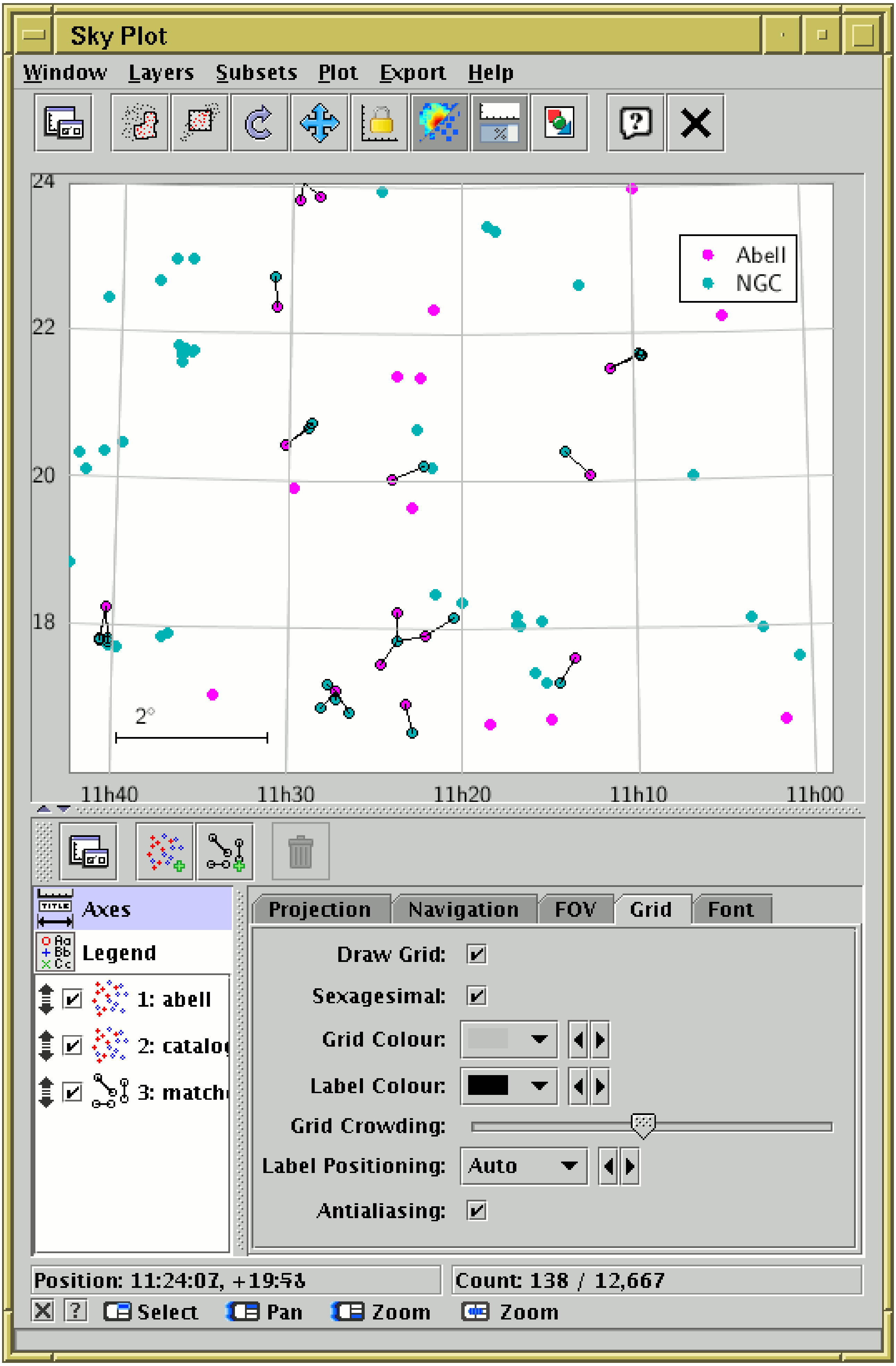
Since TOPCAT version 4.1, for most pair matches, you can do this automatically. When a suitable match completes, you may see a confirmation dialogue like this:



896 pairs found
New table created by match: 4: match(1,2) (896 rows)

Pair plot completion confirmation dialogue

If you click the **OK** button, it will simply dismiss the dialogue. However, if you click the **Plot Result** button, a new plot window will appear with all the points from the input catalogues and pair links (lines between the joined positions) for the joined rows. The result is something like this:



Draw Grid: ☑

Sexagesimal: ☑

Grid Colour:

Label Colour: ■

Grid Crowding:

Label Positioning: Auto

Antialiasing: ☑

Position: 11:24:07, +19:58

Count: 138 / 12,667

Output from crossmatch Plot Result

It shows clearly which points have been joined. You can fiddle with the plot controls in the usual way to adjust the output (see Appendix A.4). It is also possible to set up such plots by hand.

5.6 Notes on Matching

This section provides a bit more detail on the how the row matching of local tables (sections Section 5.2 and Section 5.3) is done. It is designed to give a rough idea to interested parties; it is *not* a tutorial description from first principles of how it all works.

The basic algorithm for matching is based on dividing up the space of possibly-matching rows into an (indeterminate) number of bins. These bins will typically correspond to disjoint cells of a physical or notional coordinate space, but need not do so. In the first step, each row of each table is assessed to determine which bins might contain matches to it - this will generally be the bin that it falls into and any "adjacent" bins within a distance corresponding to the matching tolerance. A reference to the row is associated with each such bin. In the second step, each bin is examined, and if two or more rows are associated with it every possible pair of rows in the associated set is assessed to see whether it does in fact constitute a matched pair. This will identify all and only those row pairs which are related according to the selected match criteria. During this process a number of optimisations may be applied depending on the details of the data and the requested match.

The matching algorithm described above is roughly an $O(N \log(N))$ process, where N is the total number of rows in all the tables participating in a match. This is good news, since the naive interpretation would be $O(N^2)$. This can break down however if the matching tolerance is such that the number of rows associated with some or most bins gets large, in which case an $O(M^2)$ component can come to dominate, where M is the number of rows per bin. The average number of rows per bin is reported in the logging while a match is proceeding, so you can keep an eye on this.

For more detail on the matching algorithms, see the javadocs for the `uk.ac.starlink.table.join` package, or contact the author.

5.7 Matching against a Remote Table

TOPCAT provides a number of facilities for positional crossmatches against tables which are exposed via Virtual Observatory web services (Cone Search, Simple Image Access and Simple Spectral Access) and general SQL-like matching (TAP). These work rather differently from the other functions described in this section, which operate on local tables, though conceptually the result is similar. See Section 6 for more details.

It also provides access to the X-Match service provided by the CDS, as described in Appendix A.11.1, which allows fast matching against any VizieR table or SIMBAD.

In many cases the CDS Upload X-Match window is the easiest, fastest and best way to match against remote tables. TAP is more flexible, but has a somewhat steeper learning curve, and may not be as fast or scalable for large local tables. The multi-SIA and multi-SSA windows may be useful for performing per-row searches of modest size against remote image or spectral archives. The multi-cone window is (since version 4.2) present mostly for historical reasons, and is not usually a good choice, since it is much less efficient than TAP or CDS X-Match.

6 Virtual Observatory Access

Several of the windows in TOPCAT allow you to interface with so-called *Data Access Layer* services provided within the Virtual Observatory (VO). The buzzwords are not important, but the basic idea is that this allows you to locate a service providing data which may be of interest to you, and then query that service to obtain the data. The VO is not a single monolithic entity, but a set of protocols which allow a general purpose tool such as TOPCAT to talk to services made available by many different participating data providers in a uniform way, without having to have prior knowledge of what services are out there or the details of how their data is arranged.

The basic operation is similar for all of TOPCAT's access to these services:

1. Select the *registry* to query for services (or use the default one)
2. Query that registry for services of interest
3. From the returned list of services, select one that you wish to query for data
4. Specify the query to send to the data service
5. Start the query and wait for the result

These ideas are explained in more detail in the following subsections. The windows from which this is done are documented in Appendix A.9.

Note: For information on SAMP or PLASTIC, which are protocols developed within the Virtual Observatory context, but are not necessarily concerned with remote data access, see Section 9.

6.1 The Registry

The **Registry** is fundamental to the way that the VO works. A registry is a list of all the services made available by different data providers. Each entry records some information about the type of service, who provided it, what kind of data it contains, and so on (registries may contain other types of entry as well, but we will not discuss these further here). Any data provider can add an entry to the registry to advertise that it has certain datasets available for access.

Several registries exist; they tend to be maintained by different regional VO organisations. At the time of writing, there are registries maintained for public access by GAVO, ESA, STScI and (remnants of) the UK's AstroGrid, amongst others. Particular projects may also maintain their own registries with limited holdings for internal use. The main public access registries talk to each other to synchronize their contents, so to a first approximation, they contain similar lists of entries to each other, and it shouldn't matter too much which one you use. In practice, there are some differences of format and content between them, so one may work better than another for you or may contain a record that you need. In most cases though, using the default registry (currently the GAVO one) will probably do what you want it to.

6.2 Data Access Services

A number of different service types are defined and listed in the registry; the ones that TOPCAT currently knows how to access are the following:

Cone Search:
retrieve entries in a certain region of the sky from a remote catalogue

Simple Image Access (SIA):
find image data (often in FITS format) in a certain region of the sky from a remote image archive

Simple Spectral Access (SSA):
find spectral data, usually in a certain region of the sky, from a remote archive of spectral

observations or theoretical models

Table Access Protocol (TAP):
make free-form queries to a remote database using an SQL-like query language

Detailed technical information about these protocols can be found at the IVOA web site (http://www.ivoa.net/) in the Cone Search, SIA, SSA and TAP documents, but these are by no means required reading for users of the services. These protocols (apart from Cone Search) are quite complex and have many specialised and optional features. The options offered for Cone Search and TAP are reasonably complete, but for SIA and SSA TOPCAT only provides a fairly basic level of interaction, and many features (for instance SSA queries by wavelength, or non-positional queries for theoretical data) are not accessible from it.

Cone Search, SIA and SSA are *positional* protocols meaning that they query a single region of the sky. TOPCAT provides access to these service types in two main ways:

Single positional query
If you enter by hand a sky position (RA, Dec) and radius, you can download a table containing the results of a search for a single (usually, small) region on the sky. See the sections on Cone, SIA, SSA load dialogues in Appendix A.9.

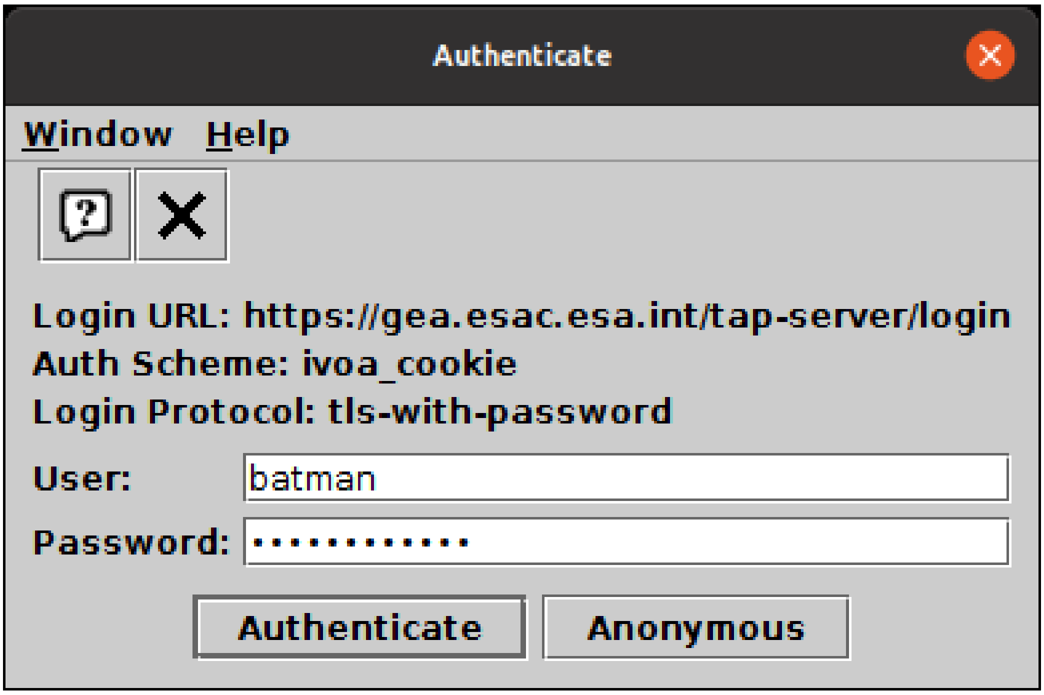
Multiple positional query
You can define how each row of an input table selects a region on the sky. This will usually correspond to selecting a column for RA and a column for Dec, and either a fixed radius or a column giving the radius. Then a positional query can be made for each row of the input table. The result is effectively a join between the input Apparent Table and the remote table of object data, images or spectra. See the sections on Cone, SIA, SSA multiple query windows in Appendix A.9.

TAP is a much more powerful protocol not restricted to positional queries, and has its own interface. See the TAP load dialogue section in Appendix A.9.

6.3 Authentication

Some services in the Virtual Observatory are authenticated, meaning that you have to log in to use them, or in some cases that you can optionally log in to gain additional rights, such as access to protected datasets or increased resource usage limits. Authentication for these situations mostly behaves as you would expect for an application working with external restricted services, but the details are given below.

If you attempt to access data which is restricted to authorised users, a window will pop up asking you to supply your username and password. Information about the URL you are trying to access will be displayed, so if you are authorized to use the service in question you will hopefully be able to supply the relevant login information. Additional information about how the authentication will be done may also be shown. If you log in successfully data access will proceed as expected, otherwise it will be blocked. Additionally, in the TAP Window you can log in to services for which authentication is optional by using the Log In/Out button.



Login URL: https://gea.esac.esa.int/tap-server/login
Auth Scheme: ivoa_cookie
Login Protocol: tls-with-password
User: batman
Password: •••••••••••••

Authentication popup dialogue

Following a successful login attempt, subsequent access to similarly restricted data from the same source should make use of the same authentication information, so it ought not to be necessary to log in to the same service more than once, though in some cases the session might expire after some time, in which case a new login would be required.

If you find yourself logged in to services that you don't want to be logged into any more (perhaps because you want to authenticate as a different user), you can log out of all currently authenticated services using the **Reset Authentication** () action in the main control window **File** menu.

Authentication status does not persist between TOPCAT sessions, so you will have to log in again to services every time you run the application.

Note: These authentication arrangements in TOPCAT are new at version 4.9, and rely on VO standards that are still under discussion. The behaviour and user interface may change in future releases, and at time of writing not all data services that provide authentication advertise it in a way that TOPCAT can work with. It is hoped that authentication interoperability will improve in future versions of TOPCAT and of server-side software.

7 Algebraic Expression Syntax

TOPCAT allows you to enter algebraic expressions in a number of contexts:

1. To define a new column in terms of existing columns in the Synthetic Column dialogue
2. To define a new Row Subset (Section 3.1) on the basis of table data in the Algebraic Subset dialogue
3. When faced with a column selector for plotting or other purposes, in most cases you can type in an expression rather than selecting or typing a simple column name.
4. To define certain custom Activation Action (Section 8)s (Execute code, Run system command, SAMP message) in the Activation window.

This is a powerful feature which permits you to manipulate and select table data in very flexible ways - you can think of it like a sort of column-oriented spreadsheet. The syntax for entering these expressions is explained in this section.

What you write are actually expressions in the Java language, which are compiled into Java bytecode before evaluation. However, this does not mean that you need to be a Java programmer to write them. The syntax is pretty similar to C, but even if you've never programmed in C most simple things, and some complicated ones, are quite intuitive.

The following explanation gives some guidance and examples for writing these expressions. Unfortunately a complete tutorial on writing Java expressions is beyond the scope of this document, but it should provide enough information for even a novice to write useful expressions.

The expressions that you can write are basically any function of all the column values and subset inclusion flags which apply to a given row; the function result can then define the per-row value of a new column, or the inclusion flag for a new subset, or the action to be performed when a row is activated by clicking on it. If the built-in operators and functions are not sufficient, or it's unwieldy to express your function in one line of code, you can add new functions by writing your own classes - see Section 7.10.

Note: if Java is running in an environment with certain security restrictions (a security manager which does not permit creation of custom class loaders) then algebraic expressions won't work at all, and the buttons which allow you to enter them will be disabled.

7.1 Referencing Cell Values

To create a useful expression for a cell in a column, you will have to refer to other cells in different columns of the same table row. You can do this in three ways:

By Name
The Name of the column may be used if it is unique (no other column in the table has the same name) and if it has a suitable form. This means that it must have the form of a Java variable - basically starting with a letter and continuing with letters or numbers. In particular it cannot have any spaces in it. The underscore and currency symbols count as letters for this purpose. Column names are treated case-insensitively.

By $ID
The "$ID" identifier of the column may always be used to refer to it; this is a useful fallback if the column name isn't suitable for some reason (for instance it contains spaces or is not unique). This is just a "$" sign followed by a unique integer assigned by the program to each column when it is first encountered. You can find out the $ID identifier by looking in the Columns Window.

By ucd$ specifier
If the column has a Unified Content Descriptor (this will usually only be the case for VOTable

or possibly FITS format tables) you can refer to it using an identifier of the form "`ucd$<ucd-spec>`". Depending on the version of UCD scheme used, UCDs can contain various punctuation marks such as underscores, semicolons and dots; for the purpose of this syntax these should all be represented as underscores ("`_`"). So to identify a column which has the UCD "`phot.mag;em.opt.R`", you should use the identifier "`ucd$phot_mag_em_opt_r`". Matching is not case-sensitive. Futhermore, a trailing underscore acts as a wildcard, so that the above column could also be referenced using the identifier "`ucd$phot_mag_`". If multiple columns have UCDs which match the given identifer, the first one will be used.

Note that the same syntax can be used for referencing table parameters (see Section 7.3); columns take preference so if a column and a parameter both match the requested UCD, the column value will be used.

By utype$ specifier

If the column has a **Utype** (this will usually only be the case for VOTable or possibly FITS format tables) you can refer to it using an identifier of the form "`utype$<utype-spec>`". Utypes can contain various punctuation marks such as colons and dots; for the purpose of this syntax these should all be represented as underscores ("`_`"). So to identify a column which has the Utype "`ssa:Access.Format`", you should use the identifier "`utype$ssa_Access_Format`". Matching is not case-sensitive. If multiple columns have Utypes which match the given identifier, the first one will be used.

Note that the same syntax can be used for referencing table parameters (see Section 7.3); columns take preference so if a column and a parameter both match the requested Utype, the column value will be used.

Using value*() functions

You can use the special functions `valueDouble`, `valueInt`, `valueLong`, `valueString` and `valueObject` to obtain the typed value of a column with a given name. The argument of the function is a string giving the exact (case-sensitive) column name, for instance `valueDouble("b_E(BP-RP)")` will yield the value of the column named "`b_E(BP-RP)`" as a double-precision floating point value. These functions are *not* the generally recommended way to get column values, since they are slower and provide less type-checking than the other options listed above, and can occasionally lead to some other esoteric problems. However, if you need to refer by name to strangely-named columns they are sometimes a convenient option.

With the Object$ prefix

If a column is referenced with the prefix "`Object$`" before its identifier (e.g. "`Object$BMAG`" for a column named `BMAG`) the result will be the column value as a java Object. Without that prefix, numeric columns are evaluated as java primitives. In most cases, you *don't want to do this*, since it means that you can't use the value in arithmetic expressions. However, if you need the value to be passed to a (possibly user-defined activation) method, and you need that method to be invoked even when the value is null, you have to do it like this. Null-valued primitives otherwise cause expression evaluation to abort.

The value of the variables so referenced will be a primitive (boolean, byte, short, char, int, long, float, double) if the column contains one of the corresponding types. Otherwise it will be an Object of the type held by the column, for instance a String. In practice this means: you can write the name of a column, and it will evaluate to the numeric (or string) value that that column contains in each row. You can then use this in normal algebraic expressions such as "`B_MAG-U_MAG`" as you'd expect.

7.2 Referencing Row Subset Flags

If you have any Row Subsets defined you can also access the value of the boolean (true/false) flag indicating whether the current row is in each subset. Again there are two ways of doing this:

By Name

The name assigned to the subset when it was created can be used if it is unique and if it has a suitable form. The same comments apply as to column names above.

By _ID

The "_ID" identifier of the subset may always be used to refer to it. Like the "$ID" identifier for columns above, this is a unique integer preceded by a special symbol, this time the underscore, "_".

Note: in early versions of TOPCAT the hash sign ("#") was used instead of the underscore for this purpose; the hash sign no longer has this meaning.

In either case, the value will be a boolean value; these can be useful in conjunction with the conditional "`? :`" operator or when combining existing subsets using logical operators to create a new subset.

7.3 Referencing Table Parameters

Some tables have constant values associated with them; these may represent such things as the epoch at which observations were taken, the name of the catalogue, an angular resolution associated with all observations, or any number of other things. Such constants are known as *table parameters* and can be viewed and modified in the Parameter Window. The values of such parameters can be referenced in algebraic expressions as follows:

param$*name*

If the parameter name has a suitable form (starting with a letter and continuing with letters or numbers) it can be referenced by prefixing that name with the string `param$`.

ucd$*ucd-spec*

If the parameter has a Unified Content Descriptor it can be referenced by prefixing the UCD specifier with the string `ucd$`. Any punctuation marks in the UCD should be replaced by underscores, and a trailing underscore is interpreted as a wildcard. See Section 7.1 for more discussion.

utype$*utype-spec*

If the parameter has a Utype, it can be referenced by prefixing the Utype specifier with the string `utype$`. Any punctuation marks in the Utype should be replaced by underscores. See Section 7.1 for more discussion.

Note that if a parameter has a name in an unsuitable form (e.g. containing spaces) and has no UCD then it cannot be referenced in an expression.

7.4 Special Tokens

There are a few pseudo-variables which have special functions in the expression language. The following specials are column-like, in that they have a different value for each row:

`$index` or `$0`

The current row number in the underlying table (the first row is 1). This is the value seen in the grey numbers at the left of the grid in the Data Window, and is not affected by the current Row Subset or Row Order. Note that this value is a `long` (8-byte integer); when using it in certain expressions you may find it necessary to convert it to an `int` (4-byte integer) using the `toInteger()` function. *The deprecated alias "`INDEX`" may also be used.*

`$index0` or `$00`

The current row number in the apparent table (the first row is 1). This value is sensitive to the current Row Subset and Row Order of the table. It has a null value for rows not in the current subset.

`$random` ***(Deprecated)***
A double-precision random number `0<=x<1`. **NOTE:** this token is deprecated since it can behave unpredictably (the same cell does not always yield the same result). Use instead the `random()` function in class Maths (Appendix B.1.15).

The following specials are parameter-like, in that their value is not sensitive to the row:

`$nrow`
The number of rows in the table. This figure refers to the underlying table, not the apparent table, so it is not affected by the value of the current subset. Note that this value is a `long` (8-byte integer); when using it in certain expressions you may find it necessary to convert it to an `int` (4-byte integer) using the `toInteger()` function.

`$ncol`
The number of columns in the table. This figure refers to the underlying table, not the apparent table, so it is not affected by hiding columns.

`$nrow0`
The number of rows in the apparent table. This value may be less than `$nrow` if a non-default current subset is selected.

`$ncol0`
The number of columns in the apparent table. This value will be less than `$ncol` if some columns are currently hidden.

7.5 Null Values

When no special steps are taken, if a null value (blank cell) is encountered in evaluating an expression (usually because one of the columns it relies on has a null value in the row in question) then the result of the expression is also null.

It is possible to exercise more control than this, but it requires a little bit of care, because the expressions work in terms of primitive values (numeric or boolean ones) which don't in general have a defined null value. The name "null" in expressions gives you the java `null` reference, but this cannot be matched against a primitive value or used as the return value of a primitive expression.

For most purposes, the following two tips should enable you to work with null values:

Testing for null
To test whether a column contains a null value, prepend the string "`NULL_`" (use upper case) to the column name or $ID. This will yield a boolean value which is true if the column contains a blank or a floating point NaN (not-a-number) value, and false otherwise. Note that if combined with other boolean expressions, this null test should come first, i.e. write "`NULL_i || i==999`" rather than "`i==999 || NULL_i`", though this is only essential for integer or boolean variables.

Returning null
To return a null value from a numeric expression, use the name "`NULL`" (upper case). To return a null value from a non-numeric expression (e.g. a String column) use the name "`null`" (lower case).

Null values are often used in conjunction with the conditional operator, "`? :`"; the expression

```
test ? tval : fval
```

returns the value `tval` if the boolean expression `test` evaluates true, or `fval` if `test` evaluates false.

So for instance the following expression:

```
Vmag == -99 ? NULL : Vmag
```

can be used to define a new column which has the same value as the Vmag column for most values, but if Vmag has the "magic" value -99 the new column will contain a blank. The opposite trick (substituting a blank value with a magic one) can be done like this:

```
NULL_Vmag ? -99 : Vmag
```

Some more examples are given in Section 7.9.

Note that for floating point data, TOPCAT treats `null` and NaN (Not-a-Number) values somewhat interchangeably. Blank values arising either from an input file format that can represent missing values, or from processing that fails to provide a definite value, are in most cases represented internally as `null` for integer-type values and NaN for floating point values. However in general users should not rely on distinguishing between `null` and NaN.

7.6 Operators

The operators are pretty much the same as in the C language. The common ones are:

Arithmetic

`+` (add)
`-` (subtract)
`*` (multiply)
`/` (divide)
`%` (modulus)

Logical

`!` (not)
`&&` (and)
`||` (or)
`^` (exclusive-or)
`==` (numeric identity)
`!=` (numeric non-identity)
`<` (less than)
`>` (greater than)
`<=` (less than or equal)
`>=` (greater than or equal)

Bitwise

`&` (and)
`|` (or)
`^` (exclusive-or)
`<<` (left shift)
`>>` (right shift)
`>>>` (logical right shift)

Numeric Typecasts

`(byte)` (numeric -> signed byte)
`(short)` (numeric -> 2-byte integer)
`(int)` (numeric -> 4-byte integer)
`(long)` (numeric -> 8-byte integer)
`(float)` (numeric -> 4-byte floating point)
`(double)` (numeric -> 8-byte floating point)

Note you may find the numeric conversion functions in the **Maths** class described in Appendix B.1 below more convenient for numeric conversions than these.

Other

`+` (string concatenation)
`[]` (array dereferencing - first element is zero)
`?:` (conditional switch)
`instanceof` (class membership)

7.7 Functions

Many functions are available for use within your expressions, covering standard mathematical and trigonometric functions, arithmetic utility functions, type conversions, and some more specialised astronomical ones. You can use them in just the way you'd expect, by using the function name (unlike column names, this is case-sensitive) followed by comma-separated arguments in brackets, so

```
max(IMAG,JMAG)
```

will give you the larger of the values in the columns IMAG and JMAG, and so on.

The functions are grouped into the following classes:

Arithmetic
Standard arithmetic functions including things like rounding, sign manipulation, and maximum/minimum functions.

Arrays
Functions which operate on array-valued cells.

Bits
Bit manipulation functions.

Conversions
Functions for converting between strings and numeric values.

CoordsDegrees
Functions for angle transformations and manipulations, with angles generally in degrees.

CoordsRadians
Functions for angle transformations and manipulations, based on radians rather than degrees.

Coverage
Functions related to coverage and footprints.

Distances
Functions for converting between different measures of cosmological distance.

Fluxes
Functions for conversion between flux and magnitude values.

Formats
Functions for formatting numeric values.

Gaia
Functions related to astrometry suitable for use with data from the Gaia astrometry mission.

Json
Functions for working with JSON strings.

KCorrections
Functions for calculating K-corrections.

Lists
Functions which operate on lists of values.

Maths
Standard mathematical and trigonometric functions.

Randoms
Functions concerned with random number generation.

Shapes
Functions useful for working with shapes in the (X, Y) plane.

Sky
Functions useful for working with shapes on a sphere.

Strings
String manipulation and query functions.

Tilings
Pixel tiling functions for the celestial sphere.

Times
Functions for conversion of time values between various forms.

TrigDegrees
Standard trigonometric functions with angles in degrees.

URLs
Functions that construct URLs for external services.

VO
Virtual Observatory-specific functions.

Full documentation of the functions in these classes is given in Appendix B.1, and is also available within TOPCAT from the Available Functions Window.

7.7.1 Technical Note

This note provides a bit more detail for Java programmers on what is going on here; only read on if you want to understand how the use of functions in TOPCAT algebraic expressions relates to normal Java code.

The expressions which you write are compiled to Java bytecode when you enter them (if there is a 'compilation error' it will be reported straight away). The functions listed in the previous subsections are all the `public static` methods of the classes which are made available by default. The classes listed are all in the packages `uk.ac.starlink.ttools.func` and `uk.ac.starlink.topcat.func` (`uk.ac.starlink.topcat.func.Strings` etc). However, the public static methods are all imported into an anonymous namespace for bytecode compilation, so that you write (`sqrt(x)` and not `Maths.sqrt(x)`. The same happens to other classes that are imported (which can be in any package or none) - their public static methods all go into the anonymous namespace. Thus, method name clashes are a possibility.

This cleverness is all made possible by the rather wonderful JEL (http://www.gnu.org/software/jel/).

7.8 Instance Methods

There is another category of functions which can be used apart from those listed in the previous section. These are called, in Java/object-oriented parlance, "instance methods" and represent functions that can be executed on an object.

It is possible to invoke any of its public instance methods on any object (though not on primitive values - numeric and boolean ones). The syntax is that you place a "." followed by the method invocation after the object you want to invoke the method on, hence `NAME.substring(3)` instead of `substring(NAME,3)`. If you know what you're doing, feel free to go ahead and do this. However, most of the instance methods you're likely to want to use have equivalents in the normal functions listed in the previous section, so unless you're a Java programmer or feeling adventurous, you are probably best off ignoring this feature.

7.9 Examples

Here are some general examples. They could be used to define synthetic columns or (where numeric) to define values for one of the axes in a plot.

Average

```
(first + second) * 0.5
```

Square root

```
sqrt(variance)
```

Angle conversion

```
radiansToDegrees(DEC_radians)
degreesToRadians(RA_degrees)
```

Conversion from string to number

```
parseInt($12)
parseDouble(ident)
```

Conversion from number to string

```
toString(index)
```

Conversion between numeric types

```
toShort(obs_type)
toDouble(range)
```

or

```
(short) obs_type
(double) range
```

Conversion from sexagesimal to degrees

```
hmsToDegrees(RA1950)
dmsToDegrees(decDeg,decMin,decSec)
```

Conversion from degrees to sexagesimal

```
degreesToDms($3)
degreesToHms(RA,2)
```

Outlier clipping

```
min(1000, max(value, 0))
```

Converting a magic value to null

```
jmag == 9999 ? NULL : jmag
```

Converting a null value to a magic one

```
NULL_jmag ? 9999 : jmag
```

Taking the third scalar element from an array-valued column

```
psfCounts[2]
```

Converting spectral type to numeric value (e.g. "L3.5" -> 23.5, "M7" -> 17)

```
"MLT".indexOf(spType.charAt(0)) * 10 + parseDouble(substring(spType,1)) + 10
```

Note this uses a couple of Java String instance methods (Section 7.8) which are not explicitly documented in this section.

Here are some examples of boolean expressions that could be used to define row subsets (or to create boolean synthetic columns):

Within a numeric range

```
RA > 100 && RA < 120 && Dec > 75 && Dec < 85
```

Within a circle

```
$2*$2 + $3*$3 < 1
skyDistanceDegrees(ra0,dec0,hmsToDegrees(RA),dmsToDegrees(DEC))<15./3600.
```

First 100 rows

```
index <= 100
```

Every tenth row

```
index % 10 == 0
```

String equality/matching

```
equals(SECTOR, "ZZ9 Plural Z Alpha")
equalsIgnoreCase(SECTOR, "zz9 plural z alpha")
startsWith(SECTOR, "ZZ")
contains(ph_qual, "U")
```

String regular expression matching

```
matches(SECTOR, "[XYZ] Alpha")
```

Subset intersection

```
_1 && _2
```

Subset union

```
_1 || _2
```

Other subset combinations

```
(in_cluster && has_photometry) || ! _6
```

Test for non-blank value

```
! NULL_ellipticity
```

7.10 Adding User-Defined Functions

The functions provided by default for use with algebraic expressions, while powerful, may not provide all the operations you need. For this reason, it is possible to write your own extensions to the expression language. In this way you can specify arbitrarily complicated functions. Note however that this will only allow you to define new columns or subsets where each cell is a function only of the other cells in the same row - there is no way to define a value in one row as a function of values in other rows.

In order to do this, you have to write and compile a (probably short) program in the Java language. A full discussion of how to go about this is beyond the scope of this document, so if you are new to Java and/or programming you may need to find a friendly local programmer to assist (or mail the author). The following explanation is aimed at Java programmers, but may not be incomprehensible to non-specialists.

The steps you need to follow are:

1. Write and compile a class containing one or more static public methods representing the function(s) required
2. Make this class available on the application's classpath at runtime as described in Section 10.2.1
3. Specify the class's name to the application, *either* as the value of the `jel.classes` or `jel.classes.activation` system properties (colon-separated if there are several) as described in Section 10.2.3 *or* during a run using the Available Function Window's **Add Class** (+) button

Any public static methods defined in the classes thus specified will be available for use in the Synthetic Column, Algebraic Subset or (in the case of activation functions only) Activation windows. They should be defined to take and return the relevant primitive or Object types for the function required (in the case of activation functions the return value should normally be a short log string).

For example, a class written as follows would define a three-value average:

```
    public class AuxFuncs {
        public static double average3(double x, double y, double z) {
            return (x + y + z) / 3.0;
        }
    }
```

and the expression "`average3($1,$2,$3)`" could then be used to define a new synthetic column, giving the average of the first three existing columns. Exactly how you would build this is dependent on your system, but it might involve doing something like the following:

1. Writing a file named "AuxFuncs.java" containing the above code
2. Compiling it using a command like "`javac AuxFuncs.java`"
3. Starting up TOPCAT with the flags: "`topcat -Djel.classes=AuxFuncs -classpath .`"

Note that (in versions later than TOPCAT 4.3-2) variable-length argument lists are supported where the final declared argument is an array, so for instance a method declared like:

```
       public static double sum(double[] values) {
           double total = 0;
           for (int i = 0; i < values.length; i++) {
               total += values[i];
           }
           return total;
       }
```

can be invoked like "`sum(UMAG,GMAG,RMAG,IMAG,ZMAG)`". The alternative form "`double... values`" can be used in the declaration with identical effect.

8 Activation Actions

Note that the activation action framework has changed considerably at TOPCAT v4.6. It is now much more flexible than in previous versions.

As well as seeing the overview of table data provided by a plot or statistics summary, it is often necessary to focus on a particular row of the table, which according to the nature of the table may represent an astronomical object, an event or some other entity. In the Data Window a table row is simply a row of the displayed JTable, and in a scatter plot it corresponds to one plotted point.

If you click on a plotted point in one of the graphics windows, or on a row in the Data Window (or in a few other circumstances - see below) the corresponding table row will be *activated*. When a row is activated, four things happen:

1. If that row is represented by a point in any open 2- or 3-dimensional scatter plot windows, a visible cursor marker will be drawn centred on that point.
2. If the Data Window is visible, the table will be scrolled to show the row and it will be highlighted
3. The contents of the **Activated** Row Subset will be updated to include (only) that row
4. If an *activation action* has been defined, it will be invoked

The first two of these mean that you can easily see which point in a plot corresponds to which row in the table and vice versa - just click on one and the other will be highlighted. You can also click on a point in one plot and easily see the corresponding point in any other plots of the same data. If you want to make the activation more visible in a plot, you can make sure the Activated entry is checked in the layer control Subsets tab, and configure it in some distinctive way.

The fourth item is much more flexible. By using the Activation Window, you can set up one or several configurable events to take place on row activation. Examples include things like viewing a cutout image near the activated row's sky position or sending the sky position to an external all-sky viewer so that it displays that region of the sky. So for instance having spotted an interesting point in a plot of a galaxy catalogue you can click on it, and immediately see an observed image to identify its morphological type. Other options include communicating with external applications using SAMP for each activated row, for instance asking an image viewer such as DS9 to load an image in a table ImageURL column. All the options, along with details of how to configure them, are listed in Appendix A.10.1. Since v4.6-1, all the defined Activation Actions are saved when you save the session (though not if you just save the table).

If none of these options fits your particular requirements, there are various ways to implement custom behaviour. One is to invoke some kind of external program such as a shell script, and pass parameters to it based on row values; this can be done using the Run system command option. You can also execute custom code in TOPCAT's expression language using the Execute code option using some *activation functions* specially provided to perform useful actions (e.g. image display) rather than just return results. Finally, advanced users can supply their own activation functions for use with the Execute Code option, or can implement their own activation actions and plug them in at runtime using the `topcat.activators` system property.

A row can be activated in the following circumstances:

- User clicks on the row in the Data Window
- User clicks on the corresponding point in one of the plot windows (or old-style plot windows)
- TOPCAT receives a `table.highlight.row` SAMP message
- A Column Search finds exactly one matching row

9 Tool Interoperability

TOPCAT is able to communicate with other tools using one or other of two messaging protocols:

- SAMP (Simple Applications Messaging Protocol)
- PLASTIC (PLatform for AStronomical InterConnection

An example of the kind of thing which can be done might be:

1. TOPCAT sends a catalogue to an image display tool
2. The image display tool shows the catalogue entries as markers placed appropriately on a displayed image
3. User actions which highlight one of the points in one tool can then automatically highlight the same point in the other

Examples of the kind of tool which TOPCAT can interoperate with in this way are image analysis tools (SAOImage DS9, GAIA, Aladin), table analysis tools (VisIVO, STILTS, other instances of TOPCAT itself), spectrum analysis tools (SPLAT, VOSpec), sky visualisation tools (Aladin, World Wide Telescope, VirGO), scripting languages (Astropy), and others.

SAMP and PLASTIC do much the same job as each other, and work in much the same way. SAMP is an evolution of PLASTIC with a number of technical improvements, and PLASTIC has been deprecated in favour of SAMP since around 2009. PLASTIC is therefore of mainly historical interest, though TOPCAT can still run in PLASTIC mode on request, if you need it to communicate with older tools that can only speak PLASTIC.

The communication architecture of the two protocols is basically the same: all tools communicate with a central "Hub" process, so a hub must be running in order for the messaging to operate. If a hub is running when TOPCAT starts, or if one starts up while TOPCAT is in operation, it will connect to it automatically. If no SAMP hub is running, TOPCAT will set one up during application startup.

TOPCAT can work in either SAMP or PLASTIC mode, but not both at once. It determines which mode to use at startup: if the `-samp` or `-plastic` flag is supplied on the command line the corresponding mode will be used; otherwise it will try to use SAMP. It is easy to tell which mode is being used by looking at the Control Window; in SAMP mode the SAMP panel displaying connection and message status is visible at the bottom of the right hand panel (there are a few other changes to the GUI as well).

This communication has two aspects to it: on the one hand TOPCAT can send messages to other applications which causes them to do things, and on the other hand TOPCAT can receive and act on such messages sent by other applications. The sent and received messages are described separately in the subsections below. There are also sections on the, somewhat different, ways to control and monitor messaging operatiion for the cases of SAMP and PLASTIC.

Note that the new activation action framework introduced in TOPCAT v4.6, unlike the activation window in previous versions, in most cases only works with SAMP and not PLASTIC.

9.1 SAMP control

When running in SAMP mode, the Control Window features several ways to view and control SAMP status.

SAMP Status button ( )

Pops up the SAMP Window which shows a detailed view of the status of SAMP communications, and allows you to register or unregister with the hub, and to start a hub if required.

SAMP Status panel
The bottom part of the right hand panel has an area labelled **SAMP**. This gives a summary view of registration status, other connected applications, and messages recently sent and received. It is described in more detail in Appendix A.2.4.

Table Broadcast () and Table Send () menu options
The **Interop** menu provides these options which allow you to send the currently selected table to one or all connected clients. Note they will only be enabled if the hub is running, and there is at least one connected client which knows how to receive a table. The **Broadcast** option is also available on the toolbar.

Stop Internal Hub () menu option
By default, when TOPCAT starts up, it will look for a SAMP hub, and if none appears to be running it will start one internally, which will normally run until TOPCAT exits. This is usually not problematic, but if you would prefer to run a hub external to TOPCAT, then you may need to shut down TOPCAT's before starting a new one. Using this option shuts down TOPCAT's internal hub.

A number of other windows feature an **Interop** menu with **Broadcast** () and **Send** () operations for data types appropriate to that window. Sometimes Broadcast appears in the toolbar as well. In some cases there are also **Accept** () toggle options in the Interop menu. These control whether TOPCAT will respond to appropriate messages sent by other SAMP applications.

9.2 PLASTIC control

Note: the PLASTIC protocol has been deprecated in favour of SAMP since about 2009, and this functionality, though still present, is of mostly historical interest.

You can view and control operations relating to the PLASTIC hub from the **Interop** menu in the Control Window. It contains the following options:

Register with PLASTIC
Attempts to locate a running PLASTIC hub and register with it.

Unregister with PLASTIC
If currently registered with a PLASTIC hub, unregisters with it.

Show registered applications
Pops up a small window which displays what applications are currently registered with any PLASTIC hub with which TOPCAT is registered. This will be kept up to date as applications register and unregister.

Start internal PLASTIC hub
Starts a PLASTIC hub as part of the process within which TOPCAT is running. This will only work if a hub is not already running. The hub will terminate when TOPCAT exits.

Start external PLASTIC hub
Starts a PLASTIC hub as a separate process. This will only work if a hub is not already running. The hub will continue running until it is explicitly terminated (e.g. using a `kill` command). Because this has some system-dependent features, it's not guaranteed to work, especially in non-Unix environments.

Help on interoperability

Opens an appropriate help section in the help browser.

Broadcast Table

Broadcasts the currently selected table to all listening applications using PLASTIC.

Send Table to ...

Sends the currently selected table to a selected listening application using PLASTIC. Select the desired recipient application from the submenu.

9.3 Messages Transmitted

This section describes the messages which TOPCAT can transmit to other tools which understand the SAMP or PLASTIC protocol, and how to cause them to be sent. Approximately the same behaviour results for either SAMP or PLASTIC, except as noted.

In most cases you can choose two ways to transmit a message from TOPCAT:

Broadcast

Broadcasts the message to all other applications currently registered with the hub which understand that message.

Send

Sends the message to a single application which you select. The suitable applications (ones which are registered with the hub and claim to understand that message) are listed and you can choose one.

Examining the list of applications in the **Send** menu gives you an indication of which ones a **Broadcast** would broadcast to. Note however that just because an application appears in this list doesn't necessarily mean it will do something substantial with the message, for instance some applications register with the hub just to monitor traffic. In general the **Broadcast** and **Send** actions will be disabled (greyed-out) if TOPCAT is not registered with a hub, or if there are no applications listening which claim to support the relevant message.

Below is a list of places you can cause TOPCAT to transmit messages. The SAMP *MTypes* and PLASTIC *message IDs* are listed along with the descriptions; unless you are a tool developer you can probably ignore these.

Transmit Table

The Control Window's **Interop** menu provides **Broadcast Table** and **Send Table** options which cause the currently selected table to be transmitted to other listening applications. They are invited to load the table in its current ("apparent") form. The **Broadcast** action is also available in the toolbar.

The Send VOTable activation action also sends this message (SAMP only).

SAMP MTypes: `table.load.votable` or `table.load.fits`

PLASTIC Message IDs: `ivo://votech.org/votable/load` or `ivo://votech.org/votable/loadFromURL`

Transmit Subset

The Subset Window (Appendix A.3.4)'s **Interop** menu contains **Broadcast Subset** and **Send Subset** options. These cause the selected subset to be sent to other listening applications (these actions are only available when one of the subsets is currently selected). Applications are invited to highlight the rows corresponding to that subset. Note this will only have an effect if the other application(s) are displaying the table that this subset relates to. This will be the case in one of two situations: (1) the table has been loaded from the same URL/filename by the other tool(s) or (2) the other tool(s) have acquired the table because it has already been broadcast using SAMP/PLASTIC.

Also, whenever a new subset is created, for instance by entering an algebraic expression or tracing out a region on a plot (see Section 3.1.1), you have the option of transmitting the subset directly to one or all listening applications as an alternative to adding the new subset to the table's subset list.

SAMP MType: `table.select.rowList`

PLASTIC Message ID: `ivo://votech.org/votable/showObjects`

Transmit Row

The Send Row Index activation action will send the row index of the activated row to other applications (SAMP only). As for Transmit Subset above, this will only have an effect if the other application(s) are displaying the same table.

SAMP MType: `table.highlight.row`

Transmit Coordinates

The Send Sky Coordinates activation action will send the sky position of the activated row to other applications (SAMP only). These other applications are invited to point out that sky position, for instance by placing a marker or centering the current window on it.

SAMP MType: `coord.pointAt.sky`

Transmit Image

The Send FITS Image and Send HiPS Cutout activation actions can send a URL representing a FITS image to a suitable image analysis application for loading or display.

The (old-style, somewhat deprecated) Density Plot (Appendix A.5.7) window also produces a 2-d histogram which is actually an image, and this can be sent to image applications from its Interop menu.

SAMP MType: `image.load.fits`

PLASTIC Message ID: `ivo://votech.org/fits/image/loadFromURL`

Transmit Spectrum

The Send Spectrum activation action can send the contents of a nominated URL column to a suitable spectrum analysis application for loading or display.

SAMP MType: `spectrum.load.ssa-generic`

Transmit Resource List

The Registry Query Panel present in most of the Virtual Observatory windows allows you to send lists of VO registry resource identifiers to other applications which can make use of them. Note this only works in SAMP mode, not for PLASTIC.

SAMP MTypes: `voresource.loadlist.cone`, `voresource.loadlist.siap`, `voresource.loadlist.ssap`

Custom messaging

The SAMP Message activation action can be used to send SAMP messages with arbitrary MTypes and parameter lists.

9.4 Messages Received

This section describes the messages which TOPCAT will respond to when it receives them from other applications via the SAMP/PLASTIC hub. The SAMP MTypes and PLASTIC message IDs are listed along with the descriptions; unless you are a tool developer you can probably ignore these.

Load Table

Loads a table sent by another application. It is added to the list of tables in the Control Window.

SAMP MType: `table.load.votable`, `table.load.fits`, `table.load.cdf`, `table.load.pds4`, `table.load.stil` or `coverage.load.moc.fits`.

PLASTIC Message ID: `ivo://votech.org/votable/load` or `ivo://votech.org/votable/loadFromURL`

Note the non-standard MType `table.load.stil` is supported for loading tables with SAMP. This is like the other `table.load.*` MTypes, but any table format supported by the application is permitted. This MType has an additional parameter "`format`", which must contain the table format name (not case sensitive).

New Subset

Loads a row selection sent from another application. If an external application is looking at the same table as one that TOPCAT has loaded, it can send a row selection. In this case, TOPCAT will add that selection as a new Row Subset for the table. The name of the received subset will be that of the sending application, for instance "Aladin". If a subset of that name already exists (which is probably because the same application has sent a row selection previously) then its content will be overwritten by the new selection. In either case, receiving the message causes plots displaying data from the table in question to show the points from the new subset.

Note: this behaviour differs from the behaviour in TOPCAT versions prior to v3.0. Different options for handling exactly how a received row selection is treated may be made available in future versions.

SAMP MType: `table.select.rowList`

PLASTIC Message ID: `ivo://votech.org/votable/showObjects`

Highlight Row

If TOPCAT already has loaded the table identified by the request, it will activate (Section 8) the requested row. This will result in the row being marked in any currently visible plots and in the Data Window if it is visible, as well as performing any additional actions you have configured in the Activation Window.

SAMP MType: `table.highlight.row`

PLASTIC Message ID: `ivo://votech.org/votable/highlightObject`

Load Resource Identifiers

Receives a list of VO registry resource identifiers from another application. Several of TOPCAT's Virtual Observatory access windows display a list of resources in their Registry Query Panel. Normally, the displayed resources are a default set or are those you have selected by entering keywords. However, another application can send a list of resources, and if they are of an appropriate type for the window in question, and if the window is currently visible, its currently displayed list will be replaced with the ones which have been sent.

SAMP MTypes: `voresource.loadlist`, `voresource.loadlist.cone`, `voresource.loadlist.siap`, `voresource.loadlist.ssap`

Get Table

Allows clients to retrieve tables currently loaded into TOPCAT. This MType has a mandatory parameter `format` giving the (case-insensitive) table format name required for the output table. It also takes an optional integer-like parameter `id`, giving the ID number (as shown in the Control Window table list) of the table required. If `id` is missing or non-positive, the current table is used. The returned response has an output parameter `url` giving the URL of the apparent table in question. This MType is experimental and may be modified or withdrawn in the future.

SAMP MType: `table.get.stil`.

System-level messages are also responded to. For SAMP these are:

- `samp.hub.disconnect`
- `samp.hub.event.shutdown`
- `samp.hub.event.register`
- `samp.hub.event.unregister`
- `samp.hub.event.metadata`
- `samp.hub.event.subscriptions`
- `samp.app.ping`

and for PLASTIC:

- `ivo://votech.org/test/echo`
- `ivo://votech.org/info/getName`
- `ivo://votech.org/info/getIconURL`
- `ivo://votech.org/hub/event/ApplicationRegistered`
- `ivo://votech.org/hub/event/ApplicationUnregistered`
- `ivo://votech.org/hub/event/HubStopping`

10 Invoking TOPCAT

Starting up TOPCAT may just be a case of typing "`topcat`" or clicking on an appropriate icon and watching the Control Window pop up. If that is the case, and it's running happily for you, you can probably ignore this section. What follows is a description of how to start the program up, and various command line arguments and configuration options which can't be changed from within the program. Some examples are given in Section 10.5. Actually obtaining the program is not covered here; please see the TOPCAT web page http://www.starlink.ac.uk/topcat/.

There are various ways of starting up TOPCAT depending on how (and whether) it has been installed on your system; some of these are described below.

There may be some sort of short-cut icon on your desktop which starts up the program - in this case just clicking on it will probably work. Failing that you may be able to locate the jar file (probably named `topcat.jar`, `topcat-full.jar` or `topcat-extra.jar`) and click on that. These files would be located in the `java/lib/topcat/` directory in a standard Starjava installation. Note that when you start by clicking on something you may not have the option of entering any of the command line options described below - to use these options, which you may need to do for serious use of the program, you will have to run the program from the command line.

Alternatively you will have to invoke the program from the command line. On Unix-like operating systems, you can use the `topcat` script. If you have the full starjava installation, this is probably in the *starjava*/java/bin directory. If you have one of the standalone jar files (topcat-*.jar), it should be in the same directory as it/them. If it's not there, you can unpack it from the jar file itself, using a command like `unzip topcat-full.jar topcat`. If that directory (and java) is on your path then you can write:

```
   topcat [java-args] [topcat-args]
```

In this case any arguments which start `-D` or `-X` are assumed to be arguments to the java command, any arguments which start `-J` are stripped of the `-J` and then passed as arguments to the java command, a `-classpath` *path* defines a class path to be used in addition to the TOPCAT classes, and any remaining arguments are used by TOPCAT.

If you're not running Unix then to start from the command line you will have to use the `java` command itself. The most straightforward way of doing this will look like:

```
   java [java-args] -jar path/to/topcat.jar [topcat-args]
```

(or the same for `topcat-full.jar` etc). However **NOTE**: using java's `-jar` flag ignores any other class path information, such as the CLASSPATH environment variable or java's `-classpath` flag - see Section 10.2.1.

The meaning of the optional `[topcat-args]` and `[java-args]` sequences are described in Section 10.1 and Section 10.2 below respectively.

10.1 TOPCAT Command-line Arguments

You can start TOPCAT from the command line with no arguments - in this case it will just pop up the command window from which you can load in tables. However you may specify flags and/or table locations and formats.

If you invoke the program with the "`-help`" flag you will see the following usage message:

```
Usage: topcat <flags> [[-f <format>] <table> ...]

    General flags:
        -help          print this message and exit
        -version       print component versions etc and exit
        -verbose       increase verbosity of reports to console
        -debug         add debugging information to console log messages
        -demo          start with demo data
        -running       use existing TOPCAT instance if one is running
        -memory        use memory storage for tables
        -disk          use disk backing store for large tables
        -samp          use SAMP for tool interoperability
        -plastic       use PLASTIC for tool interoperability
        -[no]hub       [don't] run internal SAMP/PLASTIC hub
        -exthub        run external SAMP/PLASTIC hub
        -noserv        don't run any services (PLASTIC or SAMP)
        -nocheckvers   don't check latest version
        -stilts <args> run STILTS not TOPCAT
        -jsamp <args>  run JSAMP not TOPCAT

    Useful Java flags:
        -classpath jar1:jar2..  specify additional classes
        -XmxnnnM                use nnn megabytes of memory
        -Dname=value            set system property

    Auto-detected formats:
        fits-plus, colfits-plus, colfits-basic, fits, votable, cdf, ecsv, pds4, mrt, parquet,

    All known formats:
        fits-plus, colfits-plus, colfits-basic, fits, votable, cdf, ecsv, pds4, mrt, parquet,

    Useful system properties (-Dname=value - lists are colon-separated):
        java.io.tmpdir          temporary filespace directory
        jdbc.drivers            JDBC driver classes
        jel.classes             custom algebraic function classes
        jel.classes.activation  custom action function classes
        star.connectors         custom remote filestore classes
        startable.load.dialogs  custom load dialogue classes
        startable.readers       custom table input handlers
        startable.writers       custom table output handlers
        startable.storage       storage policy
        mark.workaround         work around mark/reset bug
            (see topcat -jsamp -help for more)
```

The meaning of the flags is as follows:

-f <format>
Signifies that the file(s) named after it on the command line are in a particular file format. Some file formats (VOTable, FITS) can be detected automatically by TOPCAT, but others (including Comma-Separated Values) cannot - see Section 4.1.1. In this case you have to specify with the `-f` flag what format the named files are in. Any table file on the command line following a `-f <format>` sequence must be in the named format until the next `-f` flag. The names of both the auto-detected formats (ones which don't need a `-f`) and the non-auto-detected formats (ones which do) are given in the usage message you can see by giving the `-help` flag (this message is shown above). You may also use the classname of a class on the classpath which implements the `TableBuilder` interface - see SUN/252.

-help
If the `-help` (or `-h`) flag is given, TOPCAT will write a usage message and exit straight away.

-version
If the `-version` flag is given, TOPCAT will print a summary of its version and the versions and availability of some its components, and exit straight away.

-verbose
The `-verbose` (or `-v`) flag increases the level of verbosity of messages which TOPCAT writes to standard output (the console). It may be repeated to increase the verbosity further. The messages it controls are currently those written through java's standard logging system - see the description of the Log Window (Appendix A.11.13) for more information about this.

-debug
The `-debug` flag affects how logging messages appear (whether they appear is affected by the

`-verbose` flag). If present, these messages will provide more detail about where each message was generated from.

-demo

The `-demo` flag causes the program to start up with a few demonstration tables loaded in. You can use these to play around with its facilities. Note these demo tables are quite small to avoid taking up a lot of space in the installation, and don't contain particularly sensible data, they are just to give an idea.

-running

The `-running` flag can be used when specifying tables on the command line. If an existing instance of TOPCAT is already running, the tables are loaded into it. If no instance of TOPCAT is running (or at least one cannot be discovered), then the `-running` flag has no effect and a new instance is started up as usual.

-memory

If the `-memory` flag is given then the program will store table data in memory, rather than the default which is to store small tables in memory, and larger ones in temporary disk files.

-disk

If the `-disk` flag is given then the program will store table data in temporary disk files, rather than the default which is to store small tables in memory, and larger ones on disk. If you get out of memory messages, running with the `-disk` flag might help, though the default policy should work fairly well. The temporary data files are written in the default temporary directory (defined by the `java.io.tmpdir` system property - often `/tmp` - and deleted when the program exits, unless it exits in an unusual way.

-samp

Forces TOPCAT to use SAMP for tool interoperability (see Section 9). SAMP rather than PLASTIC is the default, so this flag has no effect.

-plastic

Forces TOPCAT to use PLASTIC instead of SAMP for tool interoperability (see Section 9).

-[no]hub

The `-hub` flag causes TOPCAT to run an internal SAMP or PLASTIC hub (SAMP by default), if no hub is already running, and the `-nohub` flag prevents this from happening. The default behaviour, when neither of these flags is given, is to start a hub automatically for SAMP but not for PLASTIC. The hub will terminate when TOPCAT does, or can be shut down manually with the **Interop|Stop Internal Hub** (⊘) menu item. See Section 9.

-exthub

Causes TOPCAT to attempt to run an external SAMP or PLASTIC hub, if no hub is already running. The hub will show up in its own window, and can be stopped by closing the window. The hub will continue to run after TOPCAT terminates. Invoking external processes from Java is inherently unreliable, so this is done on a best-efforts basis. See Section 9.

-noserv

Prevents TOPCAT from running any services. Currently the services which it may run are SAMP or PLASTIC (see above).

-checkvers

Prevents TOPCAT from checking an external URL so it can determine whether the latest version is being run.

-stilts <stilts-args>

This flag is a convenience which allows you to run the STILTS command-line tool instead of TOPCAT. This is possible because TOPCAT is built on top of STILTS and contains a superset of its code. See the STILTS manual (or `topcat -stilts -help`) for the form of the `<stilts-args>`.

-jsamp <jsamp-args>
This flag is a convenience which allows you to run the jsamp command-line tool, from the JSAMP package, instead of TOPCAT. This is possible because TOPCAT contains the JSAMP library. JSAMP provides various SAMP-related utilities, such as a freestanding hub and diagnostic tools. See the JSAMP documentation, or do `topcat -jsamp -help` for more information.

Other arguments on the command line are taken to be the locations of tables. Any tables so specified will be loaded into TOPCAT at startup. These locations are typically filenames, but could also be URLs or scheme specifications. In addition they may contain "fragment identifiers" (with a "#") to locate a table within a given resource, so that for instance the location

```
/my/data/cat1.fits#2
```

means the second extension in the multi-extension FITS file `/my/data/cat1.fits`. Section 4.2 describes in more detail the kinds of URLs which can be used here.

Note that options to Java itself may also be specified on the command-line, as described in the next section.

10.2 Java Options

As described above, depending on how you invoke TOPCAT you may be able to specify arguments to Java itself (the "Java Virtual Machine") which affect how it runs. These may be defined on the command line or in some other way. The following subsections describe how to control Java in ways which may be relevant to TOPCAT; they are however somewhat dependent on the environment you are running in, so you may experience OS-dependent variations.

10.2.1 Class Path

The classpath is the list of places that Java looks to find the bits of compiled code that it uses to run an application. When running TOPCAT this always has to include the TOPCAT classes themselves - this is part of the usual invocation and is described in Section 10. However, for certain things Java might need to find some other classes, in particular for:

- JDBC drivers (see Section 10.3)
- Providing extended algebraic functions (see Section 7.10)
- Installing I/O handlers for new table formats (see Section 4.1.4)

If you are going to use these facilities you will need to start the program with additional class path elements that point to the location of the classes required. How you do this depends on how you are invoking TOPCAT. If you are using tht `topcat` startup script, you can write:

```
topcat -classpath other-paths ...
```

(this adds the given paths to the standard ones required for TOPCAT itself). If you are invoking java directly, then you can either write on the command line:

```
java -classpath path/to/topcat.jar:other-paths
     uk.ac.starlink.topcat.Driver ...
```

or set the CLASSPATH environment variable something like this:

```
setenv CLASSPATH path/to/topcat.jar:other-paths
```

In any case, multiple (extra) paths should be separated by colons in the *other-paths* string.

Note that if you are running TOPCAT using java's `-jar` flag, any attempt you make to specify the classpath will be ignored! This is to do with Java's security model. If you need to specify a classpath which includes more than the TOPCAT classes themselves, you can't use `java -jar` (use `java -classpath topcat-full.jar:... uk.ac.starlink.topcat.Driver` instead).

10.2.2 Memory Size

If TOPCAT fails during operation with a message that says something about a `java.lang.OutOfMemoryError`, then your heap size is too small for what you are trying to do. You will have to run java with a bigger heap size using the `-Xmx` flag. Invoking TOPCAT from the `topcat` script you would write something like:

```
    topcat -Xmx256M ...
```

or using java directly:

```
    java -Xmx256M ...
```

which means use up to 256 megabytes of memory (don't forget the "M" for megabyte). JVMs often default to a heap size of 64M. You probably don't want to specify a heap size larger than the physical memory of the machine that you are running on.

There are other types of memory and tuning options controlled using options of the form `-X<something-or-other>`; if you're feeling adventurous you may be able to find out about these by typing "`java -X`".

10.2.3 System properties

System properties are a way of getting information into Java (they are the Java equivalent of environment variables). The following ones have special significance within TOPCAT:

`apple.laf.useScreenMenuBar`
On the Apple Macintosh platform only, this property controls whether menus appear at the top of the screen as usual for Mac, or at the top of individual windows as usual for Java. By default it is set to `true` for TOPCAT, so menus mostly appear at the top of the screen (though it's not true to say that TOPCAT obeys the Mac look and feel completely); if you prefer the more Java-like look and feel, set it to `false`.

`http.proxyHost`
If you are operating inside a firewall which prohibits direct HTTP connections, you can set this to the name of an HTTP proxy server in order to access remote servers as required for VO functionality etc. There are a number of related system properties which you may or may not need to use in this case: `http.proxyPort`, `https.proxyHost` etc. See the appropriate java documentation (e.g. by googling for "http.proxyHost") for details.

`java.io.tmpdir`
The directory in which TOPCAT will write any temporary files it needs. This defaults to the system temporary directory (e.g. `/tmp` on Unix), so if working with large unmapped (e.g. CSV) tables on a machine with limited space on the default disk, it may be necessary to change it.

`java.util.concurrent.ForkJoinPool.common.parallelism`
Controls the level of parallelisation done by certain processing, currently mainly visualisation. By default it is typically set to one less than the number of processing cores on the current machine. To inhibit parallelisation (e.g. if you suspect that the parallel output is giving different results to sequential processing) you can set this to 1.

`jdbc.drivers`
Can be set to a (colon-separated) list of JDBC driver classes using which SQL databases can

be accessed (see Section 10.3).

`jel.classes`
Can be set to a (colon-separated) list of classes containing static methods which define user-provided functions for synthetic columns or subsets. (see Section 7.10).

`jel.classes.activation`
Can be set to a (colon-separated) list of classes containing static methods which define user-provided functions for use in custom activation expressions. (see Section 7.10).

`jsamp.hub.profiles`
This property determines what profiles a SAMP hub will use if you run an internal or external hub from TOPCAT (either with the `-hub`/`-exthub` flag or from the GUI). The value is a comma-separated list of profile specifiers; options are "`std`" for Standard Profile, "`web`" for Web Profile or the name of a class implementing the `org.astrogrid.samp.hub.HubProfile` interface. The default setting runs just a Standard Profile hub, but, for instance, setting it to "`std,web`" would run a Web Profile as well. Note you should include `std` in the list, otherwise TOPCAT will not be able to talk to the hub. See the JSAMP documentation for more detail.

`jsamp.localhost`
Sets the hostname by which the local host is to be identified in URLs, for instance server endpoints. If unset, the default is currently the loopback address "`127.0.0.1`". However, if this property is set (presumably to the local host's fully- or partly-qualified domain name) its value will be used instead. Two special values may also be set: "`[hostname]`" for the host's fully qualified domain name, and "`[hostnumber]`" for the IP number.

`jsamp.server.port`
Gives a preferred port number on which to open TOPCAT's internal HTTP server, used for SAMP communications amongst other things. If this port is unavailable, some other port will be used instead.

`jsamp.xmlrpc.impl`
Indicates which pluggable XML-RPC implementation should be used for SAMP communications. By default an internal implementation is used. If it is set to "`xml-log`" or "`rpc-log`" then all XML-RPC communications will be logged in very or fairly verbose terms respectively to standard output. The classname of an `org.astrogrid.samp.xmlrpc.XmlRpcKit` implementation may be given instead to use a custom implementation.

`lut.files`
Can be set to a (colon-separated on *nix, semicolon-separated on Windows) list of files giving custom lookup tables for auxiliary axis and density map colour maps. Each file should be a text file containing a number of space-separated lines, each containing red, green, blue samples in the range 0-1. For instance the file

```
    1.0  1.0  0.0
    1.0  0.0  1.0
```

would give a colour map that fades from yellow to magenta. Any number of samples may be given; the scale is interpolated.

`mark.workaround`
If set to "true", this will work around a bug in the `mark()`/`reset()` methods of some java `InputStream` classes. These are rather common, including in Sun's J2SE system libraries. Use this if you are seeing errors that say something like "`Resetting to invalid mark`". Currently defaults to "false".

`myspace.cache`
If set to "true", this will prevent directories in the MySpace file browser from being read more than once. This is a workaround for MySpace performance problems at time of writing (April 2006); setting it true may lead to out of date file listings in MySpace, but it may be much

faster. This behaviour may be withdrawn in future versions if MySpace performance improves. Currently defaults to "false".

`service.maxparallel`
Raises the maximum number of concurrent queries that may be made during a multi-cone operation. You should only increase this value **with great care** since you risk overloading servers and becoming unpopular with data centres. As a rule, you should only increase this value if you have obtained permission from the data centres whose services on which you will be using the increased parallelism.

`auth.schemes`
Configures the list of authentication schemes that will be considered when connecting to services issuing a WWW-Authenticate challenge. A comma-separated list of scheme names or `uk.ac.starlink.auth.AuthScheme` implementation classnames may be provided.

`star.connectors`
Can be set to a (colon-separated) list of classes which provide access to remote filespace implementations. Thus-named classes should implement the `uk.ac.starlink.connect.Connector` interface which specifies how you can log on to such a service and provides a hierarchical view of the filespace it contains.

`startable.load.dialogs`
Can be set to a (colon-separated) list of custom table load dialogue classes. Briefly, you can install your own table import dialogues at runtime by providing classes which implement the `uk.ac.starlink.table.gui.TableLoadDialog` interface and naming them in this property. See STIL documentation for more detail.

`startable.readers`
Can be set to a (colon-separated) list of custom table format input handler classes (see Section 4.1.4).

`startable.schemes`
Can be set to a (colon-separated) list of custom table scheme handler classes. Each class must implement the `uk.ac.starlink.table.TableScheme` interface, and must have a no-arg constructor. The schemes thus named will be available alongside the standard ones listed in Section 4.3.

`startable.storage`
Can be set to determine the default storage policy. Setting it to "`disk`" has basically the same effect as supplying the "`-disk`" argument on the TOPCAT command line (see Section 10.1). Other possible values are "`adaptive`", "`memory`", "`sideways`" and "`discard`"; see SUN/252. The default is "`adaptive`", which means storing smaller tables in memory, and larger ones on disk.

`startable.unmap`
Determines whether and how unmapping of memory mapped buffers is done. Possible values are "`sun`" (the default), "`cleaner`", "`unsafe`" or "`none`". In most cases you are advised to leave this alone, but in the event of unmapping-related JVM crashes (not expected!), setting it to `none` may help.

`startable.writers`
Can be set to a (colon-separated) list of custom table format output handler classes (see Section 4.1.4).

`topcat.activators`
Can be set to a (colon-separated) list of custom Activation Actions. Each element must be the class path of a class implementing the `uk.ac.starlink.topcat.activate.ActivationType` interface, and which has a no-arg constructor.

`topcat.exttools`
Defines extension tools to be used by TOPCAT. The value is a colon-separated list of class

names, each of which must implement the `uk.ac.starlink.topcat.TopcatToolAction` interface and have a no-arg constructor. The actions corresponding to any such classes will be added to toolbar. This is an experimental extensibility feature, which may be modified or withdrawn in a future release.

`user.dir`
: Sets the current working directory. This determines the default from which the file browsers will start.

`votable.namespacing`
: Determines how namespacing is handled in input VOTable documents. Known values are "`none`" (no namespacing, xmlns declarations in VOTable document will probably confuse parser), "`lax`" (anything that looks like it is probably a VOTable element will be treated as a VOTable element) and "`strict`" (VOTable elements must be properly declared in one of the correct VOTable namespaces). May also be set to the classname of a `uk.ac.starlink.votable.Namespacing` implementation. The default is "`lax`".

`votable.strict`
: Controls the behaviour when encountering a VOTable `FIELD` or `PARAM` element with a `datatype` attribute of `char`/`unicodeChar`, and no `arraysize` attribute. The VOTable standard says this indicates a single character, but some VOTables omit arraysize specification by accident when they intend `arraysize="*"`. If `votable.strict` is set `true`, a missing arraysize will be interpreted as meaning a single character, and if `false`, it will be interpreted as a variable-length array of characters (a string). The default is `true`.

`votable.version`
: Selects the version of the VOTable standard which output VOTables will conform to by default. May take the values "`1.0`", "`1.1`", "`1.2`", "`1.3`" or "`1.4`". By default, version 1.4 VOTables are written.

To define these properties on the command line you use the `-D` flag, which has the form

```
-D<property-name>=<value>
```

If you're using the TOPCAT startup script, you can write something like:

```
topcat -Djdbc.drivers=org.postgresql.Driver ...
```

or if you're using the `java` command directly:

```
java -Djdbc.drivers=org.postgresql.Driver ...
```

Alternatively you may find it more convenient to write these definitions in a file named `.starjava.properties` in your home directory; the above command-line flag would be equivalent to inserting the line:

```
jdbc.drivers=org.postgresql.Driver
```

in your `.starjava.properties` file.

10.3 JDBC Configuration

This section describes additional configuration which must be done to allow TOPCAT to access SQL-compatible relational databases for reading (see Appendix A.6.4) or writing (see Appendix A.7.2.4) tables. If you don't need to talk to SQL-type databases, you can ignore the rest of this section. The steps described here are the standard ones for configuring JDBC (which sort-of stands for Java Database Connectivity); you can find more information on that on the web. The best place to look may be within the documentation of the RDBMS you are using.

To use TOPCAT with SQL-compatible databases you must:

- Have access to an SQL-compatible database locally or over the network
- Have a JDBC driver appropriate for that database
- Install this driver for use with TOPCAT
- Know the format the driver uses for URLs to access database tables
- Have appropriate privileges on the database to perform the desired operations

Installing the driver consists of two steps:

1. Ensure that the classpath you are using includes this driver class as described in Section 10.2.1
2. Set the `jdbc.drivers` system property to the name of the driver class as described in Section 10.2.3

Below are presented the results of some experiments with JDBC drivers. Note however that **this information may be be incomplete and out of date**. If you have updates, feel free to pass them on and they may be incorporated here.

To the author's knowledge, TOPCAT has so far successfully been used with the following RDBMSs and corresponding JDBC drivers:

MySQL
MySQL has been tested on Linux with the Connector/J driver and seems to work; tested versions are server 3.23.55 with driver 3.0.8 and server 4.1.20 with driver 5.0.4. Sometimes tables with very many (hundreds of) columns cannot be written owing to SQL statement length restrictions. Note there is known to be a column metadata bug in version 3.0.6 of the driver which can cause a ClassCastException error when tables are written. Check the driver's documentation for additional parameters, for instance "`useUnicode=true&characterEncoding=UTF8`" may be required to handle some non-ASCII characters.

PostgreSQL
PostgreSQL 7.4.1 apparently works with its own driver. Note the performance of this driver appears to be rather poor, at least for writing tables.

Oracle
You can use Oracle with the JDBC driver that comes as part of its Basic Instant Client Package. However, you *may not be able to* use the SQL load/SQL save dialogue boxes to do it. You have to specify a JDBC URL specifying the query to read/table to write as a string in the **Location** field of the normal table load/save dialogue boxes. The URL will look something like

```
jdbc:oracle:thin:@//hostname:1521/database#SELECT ...
```

for querying an existing database (loading) and

```
jdbc:oracle:thin:@//hostname:1521/database#new-table-name
```

for writing a new table (saving).

SQL Server
There is more than one JDBC driver known to work with SQL Server, including jTDS and its own JDBC driver. Some evidence suggests that jTDS may be the better choice, but your mileage may vary.

Sybase ASE
There has been a successful use of Sybase 12.5.2 and jConnect (jconn3.jar) using a JDBC URL like "`jdbc:sybase:Tds:hostname:port/dbname?user=XXX&password=XXX#SELECT...`". An earlier attempt using Sybase ASE 11.9.2 failed.

It is probably possible to use other RDBMSs and drivers, but you may have to do some homework.

Here are some example command lines to start up TOPCAT using databases that at least have worked at some point.

PostgreSQL

```
java -classpath topcat-full.jar:pg73jdbc3.jar \
     -Djdbc.drivers=org.postgresql.Driver \
     uk.ac.starlink.topcat.Driver
```

MySQL

```
java -classpath topcat-full.jar:mysql-connector-java-3.0.8-bin.jar \
     -Djdbc.drivers=com.mysql.jdbc.Driver \
     uk.ac.starlink.topcat.Driver
```

Oracle

```
java -classpath topcat-full.jar:ojdbc14.jar \
     -Djdbc.drivers=oracle.jdbc.driver.OracleDriver \
     uk.ac.starlink.topcat.Driver
```

SQL Server with jTDS

```
java -classpath topcat-full.jar:jtds-1.1.jar \
     -Djdbc.drivers=net.sourceforge.jtds.jdbc.Driver \
     uk.ac.starlink.topcat.Driver
```

10.4 Tips for Large Tables

Considerable effort has gone into making TOPCAT capable of dealing with large datasets. In particular it does not in general have to read entire files into memory in order to do its work, so it's not restricted to using files which fit into the java virtual machine's 'heap memory' or into the physical memory of the machine. As a rule of thumb, the program will work at reasonable speed with tables up to about 1-10 million rows, depending on the machine it's running on. It may well work work with hundreds of millions of rows, but performance may be more sluggish. The number of columns is less of an issue, though see below concerning performance.

However, the way you invoke the program affects how well it can cope with large tables; you may in some circumstances get a message that TOPCAT has run out of memory (either a popup or a terse "OutOfMemoryError" report on the console), and there are some things you can do about this:

Increase Java's heap memory
When a Java program runs, it has a fixed maximum amount of memory that it will use; Java does not really make use of virtual memory. The default amount depends on your machine and java implementation. You can increase this by using the `-Xmx` flag, followed by the maximum heap memory, for instance "`topcat -Xmx1000M`" or "`java -Xmx1000M -jar topcat-full.jar`". Don't forget the "`M`" to indicate megabytes or "`G`" for gigabytes. It's generally reasonable to increase this value up to nearly the amount of free physical memory in your machine if you need to (taking account of the needs of other processes running at the same time) but attempting any more will usually result in abysmal performance. See Section 10.2.2.

Use FITS files
Because of the way that FITS files are organised, TOPCAT is able to load tables which are stored as uncompressed FITS binary tables on disk almost instantly regardless of their size (in

this case loading just reads the metadata, and any data elements are only read if and when they are required). So if you have a large file stored in VOTable or ASCII-based form which you use often and takes a long time to load, it's a good idea to convert it to FITS format once for subsequent use. You can do this either by loading it into TOPCAT once and saving it as FITS, or using the separate command-line package STILTS. Note that the "fits-plus" variant which TOPCAT uses by default retains all the metadata that would be stored in a corresponding VOTable, so you won't normally lose information by doing this (see Section 4.1.2.1). As well as speeding things up, using FITS files will also reduce the need to use `-Xmx` flags as above. Feather format also has the same advantages.

Run in 64-bit mode

If you are working with a file or files whose total size approaches or exceeds about 2 Gbyte, you should use a 64-bit version of java. This means that you will need a 64-bit operating system, and also a 64-bit version of the Java Virtual Machine. Executing "`java -version`" (or "`topcat -version`") will probably say something about 64-bit-ness if it is 64-bit.

Use column-oriented storage

For really large tables storing them in the **colfits** variant of the FITS output format can significantly improve performance. This stores all the elements of a single column contiguously on disk, which means that scanning through all the values in one or a few columns can proceed with much less unnecessary I/O than in normal (row-oriented) FITS format. It will make most difference when the table is larger than the amount of physical memory available, and the table has many columns. Be aware however that operations which require all the cells in all the rows (for instance, calculating row statistics) may be somewhat slower using this format. Feather format is also column-oriented.

It is also possible to use column-oriented storage for non-FITS files by specifying the flag `-Dstartable.storage=sideways`. This is like using the `-disk` flag but uses column-oriented rather than row-oriented temporary files. However, using it for such large files means that the conversion is likely to be rather slow, so you may be better off converting the original file to `colfits` format in a separate step and using that.

Use the -disk flag

The way TOPCAT stores table data is configurable, and the details can be controlled by setting its *Storage Policy*. The default storage policy (since version 3.5), "`adaptive`", means that the data for relatively small tables are stored in memory, and for larger ones in temporary disk files. This usually works fairly well and you're not likely to need to change it. However, you can experiment if you like, and a small amount of memory may be saved if you encourage it to store all table data on disk, by specifying the `-disk` flag on the command line. You can achieve the same effect by adding the line `startable.storage=disk` in the `.starjava.properties` in your home directory. See Section 10.1, Section 10.2.3.

As far as performance goes, the memory size of the machine you're using does make a difference. If the size of the dataset you're dealing with (this is the size of the FITS HDU if it's in FITS format but may be greater or less than the file size for other formats) will fit into unused physical memory then general everything will run very quickly because the operating system can cache the data in memory; if it's larger than physical memory then the data has to keep being re-read from disk and most operations will be much slower, though use of column-oriented storage can help a lot in that case.

10.5 Examples

Here are some examples of invoking TOPCAT from the command line. In each case two forms are shown: one using the `topcat` script, and one using the jar file directly. In the latter case, the `java` command is assumed to be on the your path, and the jar file itself, assumed in directory `my/tcdir`, might be named `topcat.jar`, `topcat-full.jar`, or something else, but the form of the command is the same.

No arguments

```
topcat
java -jar topcat.jar
```

Output usage message

```
topcat -h
java -jar topcat.jar -h
```

Load a FITS file

```
topcat testcat.fits
java -jar my/tcdir/topcat.jar testcat.fits
```

Loading files of various formats

```
topcat t1.fits -f ascii t2.txt t3.txt -f votable t4.xml
java -jar my/tcdir/topcat.jar t1.fits -f ascii t2.txt t3.txt -f votable t4.xml
```

Use disk storage format and boosted heap memory

```
topcat -Xmx256M -disk
java -Xmx256M -jar my/tcdir/topcat.jar -disk
```

Make custom functions available

```
topcat -classpath my/funcdir/funcs.jar -Djel.classes=my.ExtraFuncs t1.fits
java -classpath my/tcdir/topcat.jar:my/funcdir/funcs.jar \
     -Djel.classes=func.ExtraFuncs \
     uk.ac.starlink.topcat.Driver t1.fits
```

Make PostgreSQL database connectivity available

```
topcat -classpath my/jdbcdir/pg73jdbc3.jar -Djdbc.drivers=org.postgresql.Driver
java -classpath my/tcdir/topcat.jar:my/jdbcdir/pg73jdbc3.jar \
     -Djdbc.drivers=org.postgresql.Driver uk.ac.starlink.topcat.Driver
```

Use custom I/O handlers

```
topcat -classpath my/driverdir/drivers.jar \
       -Dstartable.readers=my.MyTableBuilder \
       -Dstartable.writers=my.MyTableWriter \
java -classpath my/tcdir/topcat.jar:my/driverdir/drivers.jar \
     -Dstartable.readers=my.MyTableBuilder \
     -Dstartable.writers=my.MyTableWriter \
     uk.ac.starlink.topcat.Driver
```

The `-Dx=y` definitions can be avoided by putting equivalent `x=y` lines into the `.starjava.properties` in your home directory.

A TOPCAT Windows

This appendix gives a tour of all the windows that form the TOPCAT application, explaining the anatomy of the windows and the various tools, menus and other controls. Attributes common to many or all windows are described in Appendix A.1, and the subsequent sections describe each of the windows individually.

When the application is running, the **Help For Window** (?) toolbar button will display the appropriate description for the window on which it is invoked.

A.1 Common Window Features

This section describes some features which are common to many or all of the windows used in the TOPCAT program.

A.1.1 Toolbar

Each window has a toolbar at the top containing various buttons representing actions that can be invoked from the window. Most of them contain the following buttons:

✕ **Close**

Closes the window. This convenience button has the same effect as closing the window in whatever way your graphics platform normally allows. In most cases, closing the window does not lose state associated with it (such as fields filled in); if you recover the window later it will look the same as when you closed it.

? **Help**

Pops up a Help browser window, or makes sure it is visible if it has already been opened. The window will display help information relevant to the window in which you pushed this button.

Buttons in the toolbar often appear in menus of the same window as well; you can identify them because they have the same icon. This is a convenience; invoking the action from the toolbar or from the menu will have the same effect.

Often an action will only be possible in certain circumstances, for instance if some rows in the associated JTable have been selected. If the action is not possible (i.e. it would make no sense to invoke it) then the button in the toolbar and the menu option will be greyed out, indicating that it cannot be invoked in the current state.

A.1.2 Menus

Most windows have a menu bar at the top containing one or more menus. These menus will usually provide the actions available from the toolbar (identifiable because they have the same icons), and may provide some other less-commonly-required actions too.

Here are some of the menus common to several windows:

Window menu

Nearly all windows have this menu. At least the following options are available:

▤ **Control Window**

Ensures that the Control Window is visible on the screen, deiconifying and raising it if necessary. This can be useful if you 'lose' the window behind a proliferation of other ones.

Scrollable

If selected, this causes the entire content of the window to be wrapped in scrollbars. It is not generally recommended to use this option, since in general the windows are arranged so that resizing them will resize sensible parts of them, but it may be useful if using some of the larger windows on an unusually small screen.

Close

Closes the window. This convenience button has the same effect as closing the window in whatever way your graphics platform normally allows. In most cases, closing the window does not lose state associated with it (such as fields filled in); if you recover the window later it will look the same as when you closed it.

Exit

Exits TOPCAT. You will be prompted to confirm this action if tables are loaded, since it might result in loss of data.

Help menu

Nearly all windows have this menu. The following options are available:

Help

Pops up the Help Window.

Help for Window

Pops up the Help Window; the window will display help information relevant to the window in which the menu appears.

Help in Browser

Attempts to show the application help in a web browser. This is somewhat system dependent and is not guaranteed to work.

Help in Browser (single page)

Attempts to show the application help in a web browser as a single (long page). This can be useful if you want to search for a given word or string in the text. This is somewhat system dependent and is not guaranteed to work.

Help for Window in Browser

Attempts to show the help page relevant to the window in which the menu appears in a web browser. This is somewhat system dependent and is not guaranteed to work.

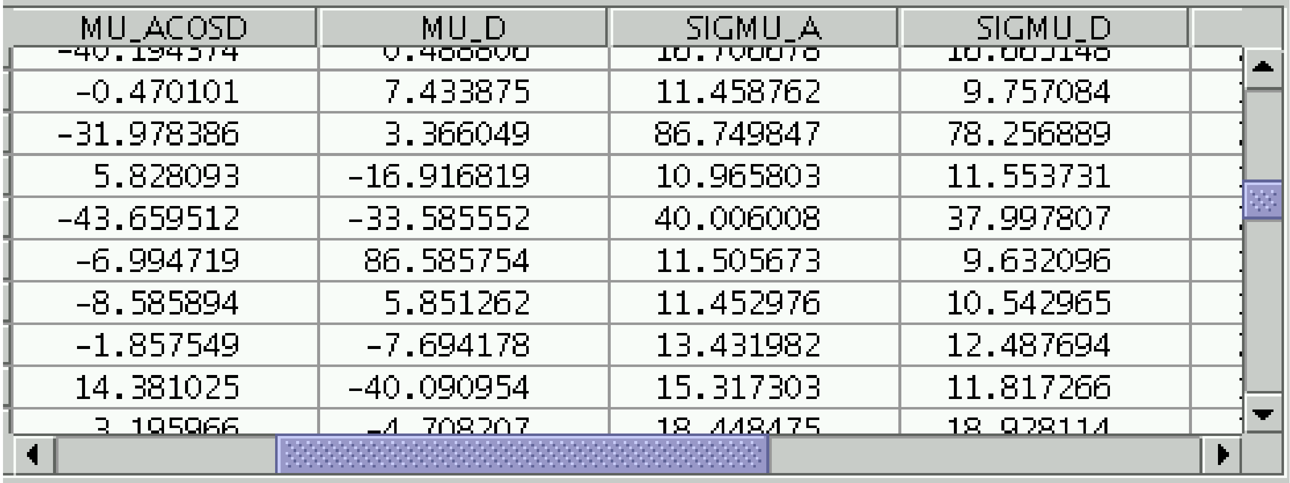
About TOPCAT

Pops up a little window giving information on the version and authorship of the program. It also reports on availability of some optional components.

Display menu

This menu is available for most windows which display their data using a JTable component. If present, it contains a list of the columns in the JTable with tickboxes next to them - clicking on a column name in this menu toggles whether the column is visible in the window.

A.1.3 JTables



MU_ACOSD	MU_D	SIGMU_A	SIGMU_D
-40.194374	[illegible]	[illegible]	[illegible]
-0.470101	7.433875	11.458762	9.757084
-31.978386	3.366049	86.749847	78.256889
5.828093	-16.916819	10.965803	11.553731
-43.659512	-33.585552	40.006008	37.997807
-6.994719	86.585754	11.505673	9.632096
-8.585894	5.851262	11.452976	10.542965
-1.857549	-7.694178	13.431982	12.487694
14.381025	-40.090954	15.317303	11.817266
3.195966	-4.708207	18.448475	18.928114

An example JTable

Many of the windows, including the Data Window, display their data in a visual component called a **JTable**. This displays a grid of values, with headings for each column, in a window which you can scroll around. Although JTables are used for a number of different things (for instance, showing the table data themselves in the Data Window and showing the column metadata in the Columns Window), the fact that the same widget is used provides a common look and feel.

Here are some of the things you can do with a JTable:

Scroll around
Using the scrollbars which may appear to the right and below the table you can scroll around it to see parts which are not initially visible. You can grab the sliders and drag them around by holding the mouse button down while you move it, or click in the slider "trough" one side or other of the current slider position to move a screenful. Under some circumstances the cursor arrow keys and PageUp/PageDown keys may move the table too. If the JTable is small enough to fit within the window the scrollbars may not appear.

Move columns
By clicking on the header (grey title bit at the top) of a column and dragging it left or right, you can change the order of columns as displayed. In some cases (the Data Window) this actually has the effect of changing the order of the columns in the table; in other cases it is just cosmetic.

Resize columns
By dragging on the line between row headers you can change the width of the columns in the table.

Edit cells
In some cases, cells are editable. If they are, then double-clicking in the cell will begin an edit session for that cell, and pressing Return will confirm that the edit has been made.

Select rows
Sometimes rows can be highlighted; you can select one row by clicking on it or a number of contiguous rows by clicking and dragging from the first to the last. To add further rows to a set already selected without deselecting the first lot, hold the "Control" key down while you do it.

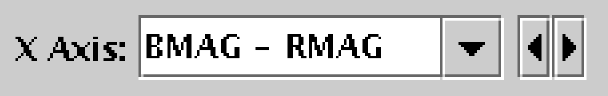
Sort rows
In some cases you can sort the entries in a JTable by clicking on the header of the column you want to sort by. A sorted column displays a little up or down arrow in the header. Clicking once sorts up, clicking again on the same header sorts down, and clicking a third time restores

the natural ordering. This feature is available for most, but not all displayed JTables. Note also that some columns do not define a sort order; clicking on the header of such a row has no effect.

Popup column menu
In some cases right-clicking will show a popup menu that can perform actions on the column currently under the mouse pointer. This may offer actions like **Sorting** by that column or Searching the content of the column for chosen text.

In some cases where a JTable is displayed, there will be a menu on the menu bar named **Display**. This permits you to select which columns are visible and which are hidden. Clicking on the menu will give you a list of all the available columns in the table, each with a checkbox by it; click the box to toggle whether the column is displayed or not.

A.1.4 Column Selector



X Axis: BMAG - RMAG

Editable Column Selector

Several windows in TOPCAT invite you to select a column value from one of the loaded tables by presenting a selection box like the one in the figure. Examples are the Plot and Match windows.

In the simplest case you can simply select the value from the list of columns by clicking on the down-arrow at the right and then selecting one from the drop-down list of columns which is offered. Note that only appropriate columns will be offered - for instance if a numeric value is required, string-valued columns will not be included.

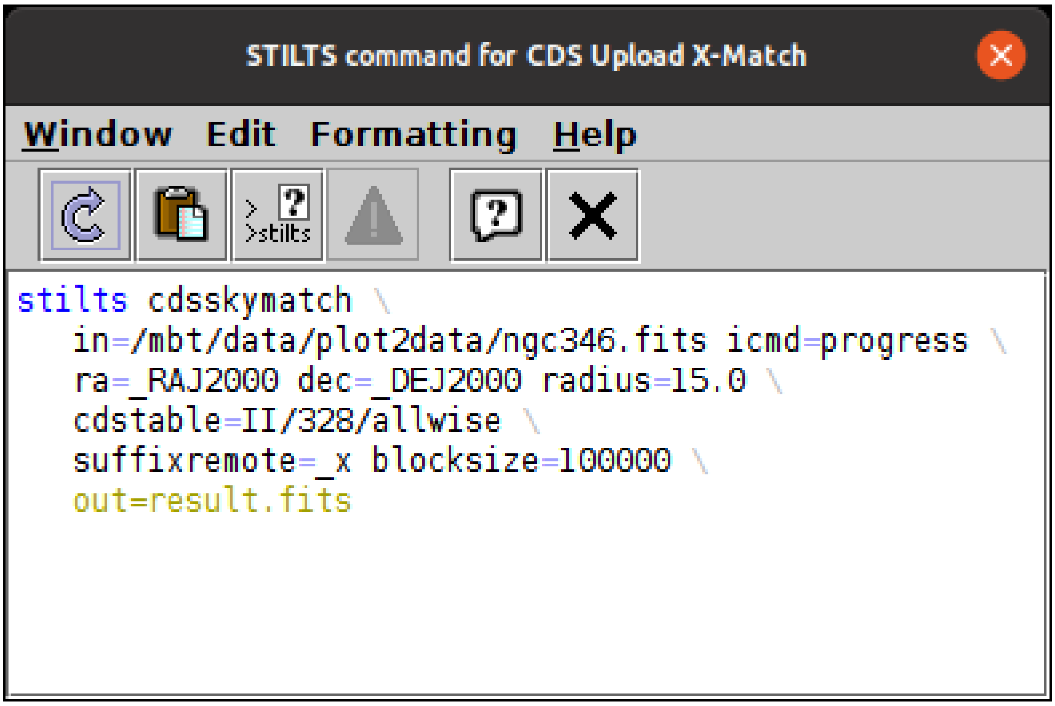
Another possibility is to use the little left/right arrows on the far right to cycle through all the known columns. This can be useful in plots, for instance to see a sequence of all the available plots against a given abcissa using only one click at a time.

Finally, you can type in an expression giving the required value. This can either be the name of a column just as if you had selected it from the drop-down list, or an expression based on column names, or even a constant value. The syntax of the expression language is explained in Section 7. Typing the column name rather than selecting it may be more convenient if the selection list is very long; typing an expression obviously gives you a lot more possibilities.

Note that depending on your platform the selection box may not look exactly like the figure above. However, if you can type into it (probably by clicking on it) then you should be able to use expressions as described above. Some selectors however only take column names; if you can't type a value into it, you have to choose one of the options given.

A.1.5 STILTS Window

The **STILTS** button (>stilts) available in several windows opens a window displaying the command you would have to issue to the STILTS package to replicate the action of the current window. Its text is continually updated to reflect the current state of the window.


STILTS command for CDS Upload X-Match
Window Edit Formatting Help
>stilts

```
stilts cdsskymatch \
   in=/mbt/data/plot2data/ngc346.fits icmd=progress \
   ra=_RAJ2000 dec=_DEJ2000 radius=15.0 \
   cdstable=II/328/allwise \
   suffixremote=_x blocksize=100000 \
   out=result.fits
```

STILTS window for the CDS Upload X-Match window

STILTS is a command-line interface to much of the same functionality that TOPCAT provides from a Graphical User Interface. It is fully documented in its manual, but if you don't want to read that, you can set up the action you want in a TOPCAT window, hit the **STILTS** button, and copy/paste the displayed text into the shell or a script. Depending on configuration, the command may end with a parameter setting like "`out=result.fits`", which writes the result to the named file. You can change the filename, remove that parameter setting to display the results directly to standard output, or replace it with settings to postprocess the results using the extensive pipeline processing offered by STILTS.

TOPCAT itself contains the STILTS application, so you don't need to install any additional software to use it. You can run it by adding the "`-stilts`" flag to a topcat command, or on a Unix-like OS use the `stilts` script. If you don't already have it, you can download `stilts` to the directory containing your TOPCAT jar file; see also SUN/256. The **Invocation** formatting menu options described below can also help.

The STILTS window button is currently available from the following windows: Match, TAP, CDS Upload X-Match, and single cone, SIA, SSA and multiple cone, SIA, SSA windows. All the Plot windows have similar functionality from their STILTS Control component.

NOTE: The STILTS command displayed in this panel is **not guaranteed** to work from the command line. There are a few reasons for this. The STILTS command tries to name the tables you have loaded into TOPCAT. If they have straightforward filenames this will probably work, but if a plotted table is for example the result of a match operation carried out in the current session, it will not exist on disk yet so it can't be named on the command line. Similarly, using the names of columns or non-algebraic Row Subsets created during the current session may result in command lines that won't work as written, since those values don't exist in the input files. Constructions that may cause trouble are flagged with text in blue, and ones that almost certainly won't work are written in red. In this case you can prepare a table corresponding to the current TOPCAT state, save that, and edit the STILTS command to use the name of that file for input (or you can reload the file and adjust the window to use that one). Subsets as such cannot be saved in this way, but you can achieve much the same thing by storing subset information in a boolean column using the **To Column** action (▦) from the Subsets Window. Note STILTS does not understand TOPCAT's saved session format. There may also be bugs in the (rather complex) command-generation code that cause the command to fail. Hopefully there are not too many of these, but if you find one please report it.

The following actions are available in the toolbar:

Update

Updates the displayed text to match the current state of the window. It is normally not necessary to use this action since the text is generally updated whenever something in the window changes, but in case they get out of sync this will ensure that the text is up to date.

Copy

Copy the displayed text to the system clipboard so it can be pasted elsewhere. You can alternatively copy the text using the mouse or whatever is the normal way for your platform.

Help for Command

Opens documentation for the particular STILTS command displayed in a web browser. Note this displays documentation for the most recent public release, so it's not guaranteed to match parameter settings for the version of STILTS that you have installed.

Show Error

If a problem is detected in the displayed command, this button will be enabled, and clicking on it will show more detail about why the command is expected to fail. If the command appears to be in good order, this button is disabled.

The **Formatting** menu controls details of how the command is laid out. Some of these options are cosmetic, and some may need to be changed according to your shell syntax (the default roughly corresponds to `bash`).

Invocation

Determines how the `stilts` command itself is introduced. Options are:

- **stilts**: Just the word "`stilts`", which will work if a command with that name is on the execution path.
- **topcat**: The expression "`topcat -stilts`", which will work if the `topcat` command is on the path.
- **Class-path**: A longer expression based on the location of the java class files that the currently running TOPCAT application is using. This should work as along as a `java` executable is on the path.

Table Names

Determines how tables in TOPCAT are referenced in the generated command line. Options are:

- **Pathname**: Full pathname of the input table file, where available. Should be fairly robust for tables loaded from files or, in most cases, URLs.
- **Filename**: Tail of filename only (no directory name) of the input table file, where available. More compact than Pathname, and should work if the STILTS command is executed in the directory in which all files exist.
- **Label**: User-assigned label for the table, as shown in the Control Window.
- **TNum**: "`T`" followed by the table identifier. The identifier is the small integer shown in the table list in the Control Window.

Line Endings

STILTS commands can be quite long and are usually displayed over several screen lines. This control configures how lines breaks are displayed. Options are:

- **plain**: just a new line character terminates lines
- **backslash**: a backslash ("\") is added at the end of each line; suitable for `bash` and other Unix-like shells.
- **caret**: a caret ("^") is added at the end of each line; suitable for Windows CMD/.bat scripts.
- **backtick**: a backtick ("`") is added at the end of each line; suitable for Windows PowerShell
- **none**: the command is displayed as one long line

Indent
Configures the amount of whitespace by which groups of related lines are indented.

Include Defaults
Parameters in STILTS commands usually have default values, and if the required values are equal to the defaults, those can be omitted from the command line. By default, parameters which take their default values are not displayed in this window. But if you set this checkbox true, even parameters taking their default values are displayed as well. Default-valued parameters are shown in a fainter colour than non-default ones. This may give you a better idea of the various options which are available for the current command.

Suggest Output File
If true, then where applicable a parameter something like "`out=results.fits`" is added to the invocation. This isn't necessary (without it the table will be written to standard output), but it may make it more obvious how to use the command.

A.2 Control Window

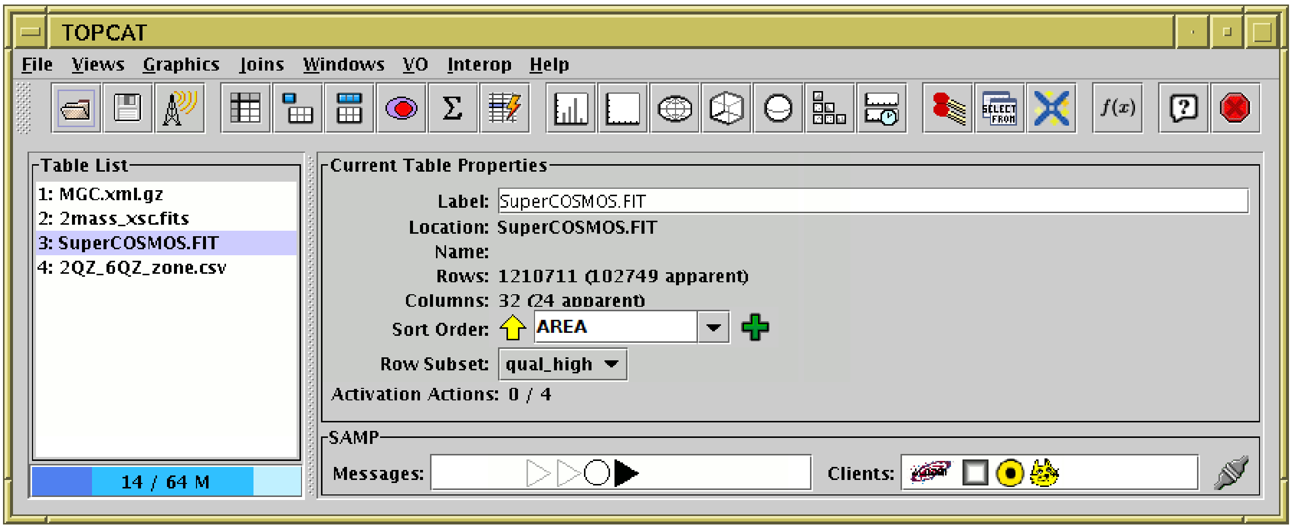


Table List
1: MGC.xml.gz
2: 2mass_xsc.fits
3: SuperCOSMOS.FIT
4: 2QZ_6QZ_zone.csv

14 / 64 M

Current Table Properties
Label: SuperCOSMOS.FIT
Location: SuperCOSMOS.FIT
Name:
Rows: 1210711 (102749 apparent)
Columns: 32 (24 apparent)
Sort Order: AREA
Row Subset: qual_high
Activation Actions: 0 / 4

SAMP
Messages:
Clients:

The Control Window

The Control Window is the main window from which all of TOPCAT's activities are controlled. It lists the known tables, summarises their characteristics, and allows you to open other windows for more specialised tasks. When TOPCAT starts up you will see this window - it may or may not have some tables loaded into it according to how you invoked the program.

The window consists of two main parts: the **Table List** panel on the left, and the **Current Table Properties** panel on the right. Tables loaded into TOPCAT are shown in the Table List, each identified by an index number which never changes for a given table, and a label which is initially set from its location, but can be changed for convenience.

One of the tables in the list is highlighted, which means it is the currently selected table; you can change the selection by clicking on an item in the list. Information about the selected table is shown in the properties panel on the right. This shows such things as the number of rows and columns, current sort order, current row subset selection and so on. It contains some controls which allow you to change these properties. Additionally, many of the buttons in the toolbar relate to the currently selected table.

Additionally there is a toolbar and some menus at the top which display windows and perform other actions, there is a memory monitor at the bottom left, and there may, depending on how TOPCAT was invoked, be a panel labelled **SAMP** at the bottom of the right hand panel.

The Table List, Current Table Properties panel, memory monitor, SAMP panel, and actions available from the Control Window's toolbar and menus are described in the following subsections.

A.2.1 Table List

The Table List panel on the left of the Control Window is pretty straightforward - it lists all the tables currently known to TOPCAT. If a new table is introduced by loading it from the Load Window or as a result of some action such as table joining then its name and number will appear in this list. The currently selected table is highlighted - clicking on a different table name (or using the arrow keys if the list has keyboard focus) will change the selection. The properties of the selected table are displayed in the Current Table Properties panel to its right, and a number of the toolbar buttons and menu items refer to it.

If you double-click on a table in the list, or press Return while it is selected, that table's Data Window will appear.

Certain other applications (Treeview or even another instance of TOPCAT) can interoperate with TOPCAT using drag-and-drop, and for these the table list is a good place to drag/drop tables. For instance you can drag a table node off of the main panel of Treeview and drop it onto the table list of TOPCAT, and you will be able to use that table as if it had been loaded from disk. You can also paste the filename or URL of a table onto the table list, and it will be loaded.

Sometimes while a table is being loaded a greyed-out entry will appear in this list with text like "*Loading SAMP table*" or similar. Such entries cannot be selected, but they let you know that something is happening.

A.2.2 Current Table Properties panel

The **Current Table Properties** panel on the right hand side of the Control Window contains a number of controls which relate to the currently selected table and its Apparent properties (Section 3); they will be blank if no table is selected. Here is what the individual items mean:

Label
: The short name associated with the selected table. It is used in the Table List panel and in labelling view windows so you can see which table they refer to. It usually set initially according to where the table came from, but you can change it by typing into the text field.

Location
: The original source of the selected table. This is typically a filename or URL (perhaps abbreviated), but may be something else depending on where the table came from.

Name
: A name associated with the selected table. This may be derived from the table's filename if it had one or from some naming string stored within the table metadata.

Rows

The number of rows in the selected table. If the current Row Subset does not include all the rows, then an indication of how many are visible within that subset will be given too. In certain circumstances, some other information may be presented here: the table metadata may contain a flag indicating that an error or overflow occurred when generating the result of a (VO) query. In particular, if the text "**OVERFLOW**" appears, it means that you may not have got all the rows that you asked for because of size limits applied by the service. (In technical terms: the content of non-`OK` `INFO/@name="QUERY_STATUS"` VOTable nodes is summarised here, in accordance with the DALI standard).

Columns

The number of columns in the selected table. If some are currently hidden (not included in the current Column Set), an indication of how many are visible will be given too.

Sort Order

The columns or expressions (if any) which determine the current Row Order. Selectors show the values (if any) on which the rows of the Apparent Table are sorted. The little arrow beside it indicates whether the sort is ascending (⇧) or descending (⇩). The **Add** button (✚) can be used to add more selectors; ones further right are used only if several rows have the same value for expressions to the left. The **Remove** button if present can be used to remove the currently least-significant selector.

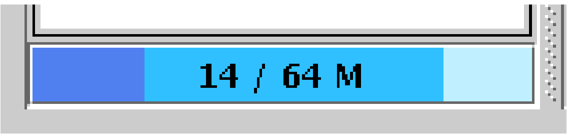
Row Subset

The name of the current Row Subset. A selector shows the name of the subset which determines which rows are part of the Apparent Table and allows you to choose another one. "All" indicates that all rows are included. If you select a new value using this selector, then other windows which display subset-sensitive information for the current table will change their displayed subset to the newly-selected one. However the reverse does not happen - you can change the visible subset in the statistics window for instance or one of the graphics windows and it will not affect the value displayed here.

Activation Actions

Summarises the currently available and active Activation Actions. A value of the form *active-count* / *available-count* is displayed, where *active-count* is the number of actions that currently take place whenever a row is activated, and *available-count* is the number of actions that could be made active without any additional configuration (a number of other actions could probably be used too, but require some manual setup). You can view and configure the activation actions by using the **Activation Actions** toolbar button to open the Activation Window.

A.2.3 Memory Monitor

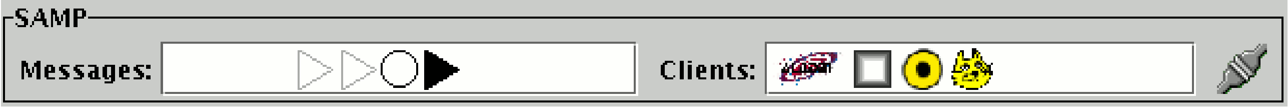


Control Window Memory Monitor

The memory monitor is a small widget at the bottom left of the Control Window which gives an indication of TOPCAT's memory usage. The numbers indicate the currently used and maximum heap size that Java will use (e.g. "64 M" for 64 megabytes), and the two darker colours filled in from the left indicate the current total and used proportions of this. It's not necessary to understand in detail what these mean, but if the darkest (used) colour comes to fill up the whole area, the application will slow down and then signal an Out Of Memory Error. See Section 10.4 for some tips on what to do if this happens.

If you click on the memory monitor, you will request that the Java Virtual Machine performs a garbage collection round, which may have the effect of reclaiming some memory and modifying the current usage. This is not really useful (Java will garbage collect at sensible times without prompting), but you can do it if you're bored.

A.2.4 SAMP Panel



Control Window SAMP Panel

If TOPCAT is running in SAMP mode, the SAMP panel at the bottom of the Control Window gives a quick view of the current status of SAMP communications. For a discussion of the whats and whys of SAMP, see Section 9. **Note** that if not running in SAMP mode (e.g. if in PLASTIC or no-server mode) this panel will not appear. SAMP mode is the default under normal circumstances.

The panel is made up of the following main parts:

Message View
This shows a graphical representation of any messages which have been recently sent to or received from other applications by TOPCAT. Triangles to the left of the central circle represent incoming messages and triangles to the right represent outgoing ones. A filled triangle represents a message which is still waiting for an answer, and an open one represents a message which either is not expecting an answer or has already received one. Colour coding is used to indicate success or failure. The triangles disappear from the display shortly after they are no longer waiting for a reply.

Client View
An icon is shown in this panel for each application currently registered with the SAMP hub, including TOPCAT itself. If no icon is available for a registered application, a generic grey square is used.

Connection Indicator
An icon at the right may be visible to indicate whether or not TOPCAT is currently registered with the hub.

When TOPCAT is not connected to a SAMP hub (most likely because none is running) these panels will be greyed out.

More detail and control for the information presented in this panel is available in the SAMP Window.

A.2.5 Toolbar Buttons

There are a number of buttons on the Control Window's toolbar; these are the most convenient way of invoking most of TOPCAT's functions. They are grouped as follows:

Import and export

Load Table
Pops up the Load Table dialogue which allows you to load a table into TOPCAT. If a table is loaded it becomes the new current table.

Save Table(s)/Session
Pops up the Save Table dialogue which allows you to write out tables in various ways.

You can either write out the current Apparent Table, or multiple tables, or save (much of) the current session state.

Broadcast Table

Broadcasts the current **Apparent Table** to any applications registered using the SAMP or PLASTIC protocol. See Section 9.

Table views (see Appendix A.3)

Data Window

Displays the table rows and columns in a scrollable viewer so you can see the cell contents themselves.

Parameters Window

Displays table "parameters", that is metadata which applies to the whole table.

Columns Window

Displays metadata about each column such as data type, units, description, UCDs etc.

Subsets Window

Displays the currently defined row subsets (Section 3.1) and enables new ones to be defined.

Statistics Window

Displays a window for calculating statistical quantities for the values in each column of the table.

Activation Window

Displays a window for configuring activation actions.

Graphics windows

Histogram Plot

Displays a window for plotting 1D histograms and kernel density estimates.

Plane Plot

Displays a window for making various types of plot on 2D Cartesian axes.

Sky Plot

Displays a window for making various types of plot on the celestial sphere.

Cube Plot

Displays a window for making various types of plot on 3D Cartesian axes.

Sphere Plot

Displays a window for making various types of plot on 3D axes using spherical polar coordinates.

Corner Plot

Displays a window for showing a grid of scatter plots using all pair combinations of a larger list of coordinates.

Time Plot

Displays a window with special features for plotting data which has a time coordinate.

Matching etc (see Section 5)

Pair Match Window

Displays a dialog for joining tables side-by-side by locating rows which match between them.

TAP Query

Displays a dialogue window for querying remote databases using the Table Access Protocol (TAP). This is a very powerful way to access remote data by writing SQL-like queries, and can be used to do joins with remote tables as well as simply downloading data.

CDS Upload X-Match

Displays a dialog window which uses the X-Match provided by the Centre de Données astronomiques de Strasbourg to allow fast matching local tables with any tables in the remote VizieR or SIMBAD databases.

Note that other options for joining tables, including matches involving a single table or more than two tables, and joining tables top-to-bottom, are available from the **Joins** menu.

Miscellaneous

Available Functions

Displays a window containing all the functions which can be used for writing algebraic expressions (see Section 7).

Help

Displays the help browser, open at the entry on the Control Window.

Exit

Queries for confirmation, then exits the application.

A.2.6 Menu Items

This section describes actions available from the Control Window menus additional to those also available from the toolbar (described in the previous section) and those common to other windows (described in Appendix A.1.2).

The **File** menu contains the following additional actions:

Duplicate Table

Adds a new copy of the current Apparent Table to the list of known tables. This is like loading in the current table again, except that its apparent characteristics become the basic characteristics of the copied one, so for instance whatever is the current row order becomes the natural order of the new one.

Discard Table(s)

Removes the current table from the list and closes and forgets any view windows associated with it. A discarded table cannot be reinstated. You will be prompted to confirm this action. Discarding a table in this way *may* free up memory, for other operations, but often will not; whether it does or not depends on the details of where the table comes from. This action can also be triggered by hitting the **Delete** key on the keyboard when the table list has keyboard focus. If multiple tables are selected at the same time, you will be prompted to remove them all.

Move Table Up

Moves the currently selected table up one slot in the tables list. This can be convenient for viewing, and it also influences the order in which tables are saved when doing a Multiple Table or Session save. This action can also be triggered by hitting **ALT-up-arrow** key on the keyboard when the table list has keyboard focus.

Move Table Down

Moves the currently selected table down one slot in the tables list. This can be convenient for viewing, and it also influences the order in which tables are saved when doing a Multiple Table or Session save. This action can also be triggered by hitting **ALT-down-arrow** key on the keyboard when the table list has keyboard focus.

Send Table to ...

Sends the currently selected table to a selected listening application using SAMP or PLASTIC. Select the desired recipient application from the submenu.

View Log

Pops up the Log Window to display logging messages generated by the application. Intended mainly for debugging.

Reset Authentication

Logs out of any external services that you have logged in to. If you try to access these services again, you will either access them anonymously or (in cases where authentication is mandatory) be prompted to log in again.

The **Views** menu contains actions for launching the windows which give certain views of the table metadata. Most of these are provided as toolbar buttons as well, but one is rather special-interest, so appears only in this menu:

Datalink Window

Displays a window for display of tables in the {links}-response format defined by the DataLink standard.

The **Graphics** menu contains actions for launching the windows which give various plotting and visualisation options. The most important ones are provided as toolbar buttons as well, but this menu contains some actions not available elsewhere. The ones marked "(old)" are the plotting windows that were available before TOPCAT version 4; they have a different user interface, are generally less powerful, and are somewhat deprecated, but still work.

Histogram Plot (old)

Displays the old-style Histogram window for plotting 1-d histograms.

2D Plot (old)

Displays the old-style 2D plot window for plotting 2-d scatter plots.

3D Plot (old)

Displays the old-style 3D plot window for plotting 3-d scatter plots on Cartesian axes.

Sphere Plot (old)

Displays the old-style sphere window for plotting 3-d scatter plots on spherical polar axes, with or without a radial coordinate.

Stacked Line Plot (old)

Displays the old-style Stacked Line Plot window for plotting multiple vertically stacked line plots against a single X coordinate.

Density Map (old)

Displays the old-style Density Map window for plotting 2-d density maps (image-like histograms on a 2-d grid of bins).

The **Joins** menu contains actions for various types of table join. The items provided additional to those on the toolbar are:

Concatenation Window

Displays the Contatenate Tables window, which allows you to join two tables top-to-bottom.

Multiple Cone Search

Displays the Multiple Cone Search Window which allows you to crossmatch a local table with a remote catalogue exposed via the Cone Search protocol. Note this is somewhat deprecated in favour of TAP or CDS X-Match Upload.

Multiple SIA

Displays the Multiple SIA Window which allows you to crossmatch with a remote image service.

Multiple SSA

Displays the Multiple SSA Window which allows you to crossmatch with a remote spectrum service.

Internal Match

Displays the Internal Match Window for finding internal matches between rows in the same table.

N-Table Match

Displays a dialog for matching more than two tables together.

The **Windows** menu contains actions for controlling which table view windows (Appendix A.3) are currently visible on the screen. If you have lots of tables and are using various different views of several of them, the number of windows on the screen can get out of hand and it's easy to lose track of what window is where. The actions on this menu do some combination of either hiding or revealing all the various view windows associated with either the selected table or all the other ones. Windows hidden are removed from the screen but if reactivated (e.g. by using the appropriate toolbar button) will come back in the same place and the same state. Revealing all the windows associated with a given table means showing all the view windows which have been opened before (it won't display windows which have never explicitly been opened).

Show Selected Views Only
Reveal all view windows associated with the currently selected table and hide all others.

Show Selected Views
Reveal all view windows which are associated with the currently selected table.

Show All Views
Reveal all view windows associated with all tables.

Hide Unselected Views
Hide all view windows associated with tables other than the currently selected one.

Hide Selected Views
Hide all view windows associated with the currently selected table.

Hide All Views
Hide all the view windows. If you get really confused, this is a good one to select to clear up your screen prior to reinstating the ones that you actually want to look at.

Note that the **Control Window** item () on menus on all other windows is also useful for window management - it brings back the control window if it's been hidden.

The **VO** menu groups together those actions which access remote data sources. All of these options can also be found in either the Load Window toolbar or the **Joins** menu.

The **Interop** menu contains options relevant to SAMP (or possibly PLASTIC) tool interoperability; see Section 9:

SAMP Window
Pops up the SAMP Window which displays detail about the current status, and allows configuration, of SAMP messaging. Note this option will not be available if TOPCAT is running in PLASTIC mode.

Stop Internal Hub
By default, when TOPCAT starts up, it will look for a SAMP hub, and if none appears to be running it will start one internally, which will normally run until TOPCAT exits. This is usually not problematic, but if you would prefer to run a hub external to TOPCAT, then you may need to shut down TOPCAT's before starting a new one. Using this option shuts down TOPCAT's internal hub.

Broadcast Table
Broadcasts the currently selected table to all listening applications using SAMP.

Send Table to ...
Sends the currently selected table to a selected listening application using SAMP. Select the

desired recipient application from the submenu.

A.3 Table View Windows

Many of the windows you will see within TOPCAT display information about a single table. There are several of these, each displaying a different aspect of the table data - cell contents, statistics, column metadata etc. There is one of each type for each of the tables currently loaded, though they won't necessarily all be displayed at once. The title bar of these windows will say something like **TOPCAT(3): Table Columns**, which indicates that it is displaying information about the column metadata for the table labelled "3:" in the Control Window.

To open any of these windows, select the table of interest in the Control Window and click the appropriate toolbar button (or the equivalent item in the **Table Views** menu). This will either open up a new window of the sort you have requested, or if you have opened it before, will make sure it's visible.

If you have lots of tables and are using various different views of several of them, the number of windows on the screen can get out of hand and it's easy to lose track of what window is where. In this case the Control Window's **Windows** menu (described in Appendix A.2.6), or the **Window|Control Window** menu item in any of the view windows can be handy to keep them under control.

The following sections describe each of these table view windows in turn.

A.3.1 Data Window

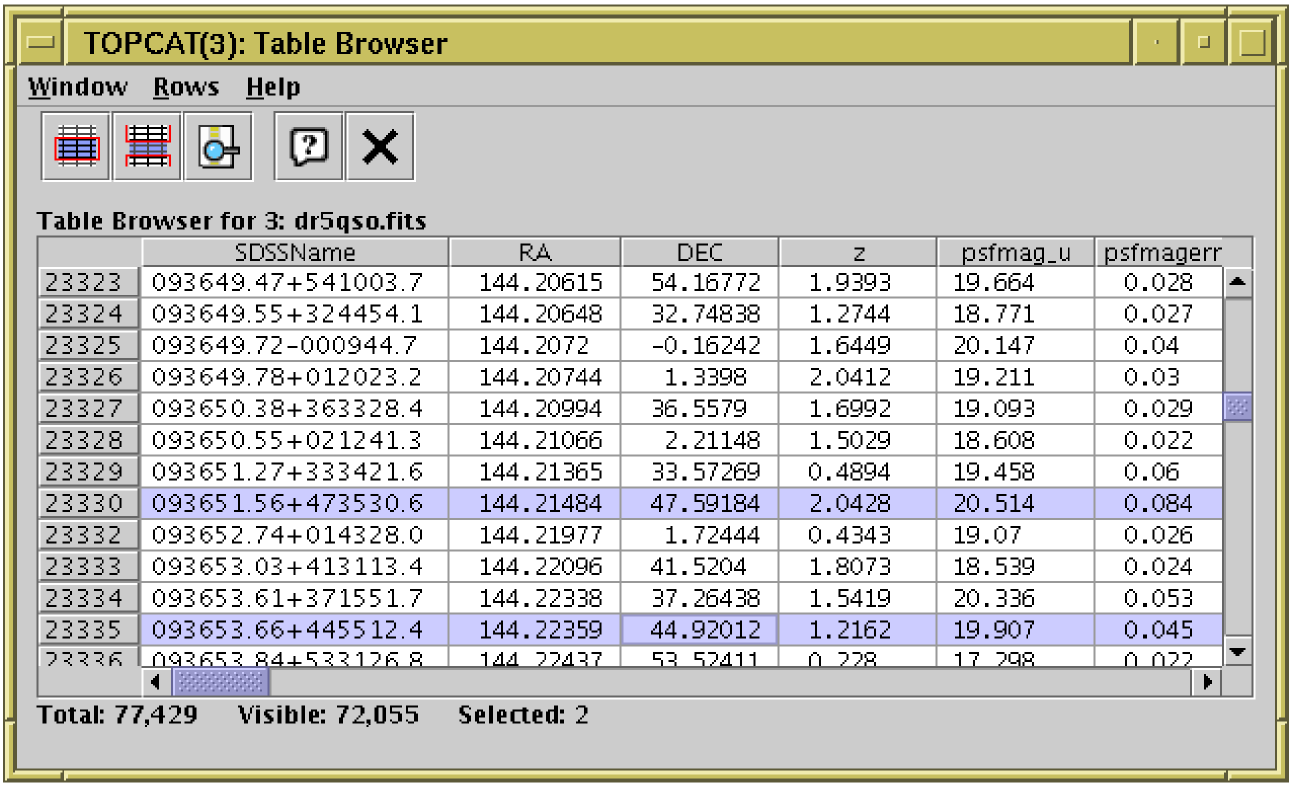


Table Browser for 3: dr5qso.fits

	SDSSName	RA	DEC	z	psfmag_u	psfmagerr
23323	093649.47+541003.7	144.20615	54.16772	1.9393	19.664	0.028
23324	093649.55+324454.1	144.20648	32.74838	1.2744	18.771	0.027
23325	093649.72-000944.7	144.2072	-0.16242	1.6449	20.147	0.04
23326	093649.78+012023.2	144.20744	1.3398	2.0412	19.211	0.03
23327	093650.38+363328.4	144.20994	36.5579	1.6992	19.093	0.029
23328	093650.55+021241.3	144.21066	2.21148	1.5029	18.608	0.022
23329	093651.27+333421.6	144.21365	33.57269	0.4894	19.458	0.06
23330	093651.56+473530.6	144.21484	47.59184	2.0428	20.514	0.084
23332	093652.74+014328.0	144.21977	1.72444	0.4343	19.07	0.026
23333	093653.03+413113.4	144.22096	41.5204	1.8073	18.539	0.024
23334	093653.61+371551.7	144.22338	37.26438	1.5419	20.336	0.053
23335	093653.66+445512.4	144.22359	44.92012	1.2162	19.907	0.045
23336	093653.84+533126.8	144.22437	53.52411	0.228	17.298	0.022

Total: 77,429 Visible: 72,055 Selected: 2

Data Window

The Data Window presents a JTable containing the actual cells of the Apparent Table (Section 3). You can display it using the **Table Data** (▦) button when the chosen table is selected in the Control Window's Table List.

You can scroll around the table in the usual way. In most cases you can edit cells by double-clicking in them, though some cells (e.g. ones containing arrays rather than scalars) cannot currently be edited. If it looks like an edit has taken place, it has.

There is a grey column of numbers on the left of the JTable which gives the row index of each row. This is the value of the special Index column, which numbers each row of the original (not apparent) table starting at 1. If the table has been sorted these numbers may not be in order.

The status line at the bottom displays three row counts:

- **Total:** the number of rows in the table; this does not change
- **Visible:** the number of rows displayed in this window; this is the row count of the current subset
- **Selected:** the number of rows selected in the displayed table; rows appear highlighted when they are selected, and can be selected/unselected by clicking on them, and in some cases by other actions

Note that reordering the columns by dragging their headings around will change the order of columns in the table's Column Set and hence the Apparent Table.

If you have a table with very many columns it can be difficult to scroll the display sideways so that a column you are interested in is in view. In this case, you can go to the Columns Window and click on the description of the required column in the display there. This has the effect of scrolling the Data Window sideways so that your selected column is visible in the centre of the display here. To make it easier to find the required column in the Columns Window you can sort the items by column Name, UCD, Units etc first by clicking on the relevant header.

The following buttons are available in the toolbar:

Subset From Selected Rows

Defines a new Row Subset consisting of those rows which are currently highlighted. You can highlight a contiguous group of rows by dragging the mouse over them; further contiguous groups can be added by holding the Control key down while dragging. This action is only available when some rows have been selected.

Subset From Unselected Rows

Defines a new Row Subset consisting of those rows which are visible but currently not highlighted. You can highlight a contiguous group of rows by dragging the mouse over them; further contiguous groups can be added by holding the Control key down while dragging. This action is only available when some rows have been selected.

Search Column

Pops up the Column Search Window that allows you to locate a given text string in the cells of a chosen column. When invoked like this, you have to select the column by hand. The column popup menu described below can fill in the column automatically.

The **Rows** menu additionally contains the following item:

Highlight Subset

This is a submenu in which all the currently defined row subsets are listed. Choosing one of them highlights the rows corresponding to that subset as if they have been selected.

As well as the normal menu, right-clicking over one of the columns in the displayed table will present a **Column Popup Menu**, which provides a convenient way to do some things with the column in question:

Replace Column

Pops up a Synthetic Column dialogue to replace this column with a new synthetic one. The dialogue is initialised with the same name, units etc as the selected column, and with an expression that evaluates to its value. You can alter any of these, and the new column will replace the old one, which will be hidden and renamed by appending a suffix like "_old" to its name.

New Synthetic Column

Pops up a Synthetic Column dialogue to insert a new synthetic column just after this one.

Sort up

Sorts the table rows according to ascending value of the contents of the column. Only available if some kind of order (e.g. numeric or alphabetic) can sensibly be applied to the column. This will overwrite the current Sort Order.

Sort down

Sorts the table rows according to descending value of the contents of the column. Only available if some kind of order (e.g. numeric or alphabetic) can sensibly be applied to the column. This will overwrite the current Sort Order.

Hide

Hides the column. It can be reinstated from the Columns window.

Search Column

Pops up the Column Search Window that allows you to locate a given text string in the cells of a chosen column. When invoked like this, the column on which you clicked to get the menu is filled in automatically.

Explode Array Column

For columns which have an array value with a fixed number of elements, selecting this option will hide the original column and replace it by a set of scalar columns, one for each of its elements. For instance if a column PMAG contains a 5-element vector of type `float[]` representing magnitudes in 5 different bands, selecting this option will hide PMAG and insert 5 new `Float`-type columns PMAG_1...PMAG_5 in its place, each containing one of the magnitudes.

A.3.2 Parameters Window

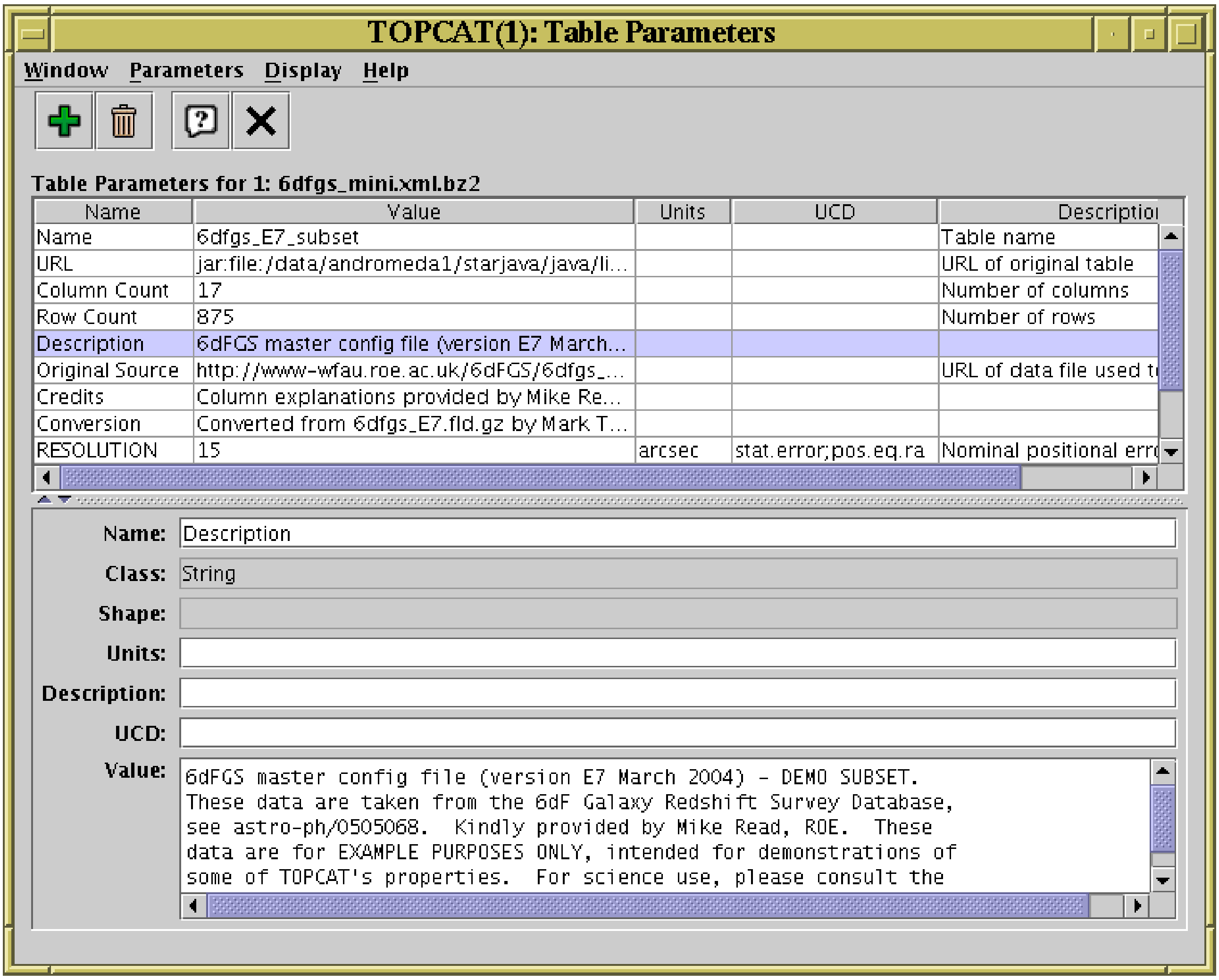


Table Parameters for 1: 6dfgs_mini.xml.bz2

Name	Value	Units	UCD	Descriptio
Name	6dfgs_E7_subset			Table name
URL	jar:file:/data/andromeda1/starjava/java/li...			URL of original table
Column Count	17			Number of columns
Row Count	875			Number of rows
Description	6dFGS master config file (version E7 March...			
Original Source	http://www-wfau.roe.ac.uk/6dFGS/6dfgs_...			URL of data file used t
Credits	Column explanations provided by Mike Re...			
Conversion	Converted from 6dfgs_E7.fld.gz by Mark T...			
RESOLUTION	15	arcsec	stat.error;pos.eq.ra	Nominal positional err

Name: Description

Class: String

Shape:

Units:

Description:

UCD:

Value:

```
6dFGS master config file (version E7 March 2004) - DEMO SUBSET.
These data are taken from the 6dF Galaxy Redshift Survey Database,
see astro-ph/0505068.  Kindly provided by Mike Read, ROE.  These
data are for EXAMPLE PURPOSES ONLY, intended for demonstrations of
some of TOPCAT's properties.  For science use, please consult the
```

Parameters Window

The Parameters Window displays metadata which applies to the whole table (rather than that for each column). You can display it using the **Table Parameters** (⊞) button when the chosen table is selected in the Control Window's Table List.

In table/database parlance, an item of per-table metadata is often known as a "parameter" of the table. At least the number of rows and columns will be listed. Some table file formats (for instance VOTable and FITS) have provision for storing other table parameters, while others (for instance CSV) do not. In the latter case there may not much much of interest displayed in this window.

The top part of the display is a JTable with one row for each parameter. It indicates the parameter's name, its value, the type of item it is (integer, string etc) and other items of interest such as units, dimensionality or UCD if they are defined. If a column of the table has no entries (for instance, the Units column might be empty because none of the parameters has had units defined for it) then that column may be absent from the display - in this case the **Display** menu can be used to reveal it.

You can interact with this JTable in the usual ways, for instance dragging columns sideways, changing their widths, and sorting the entries by clicking on the headings.

You can edit some parameter values and descriptions by double-clicking on them as usual.

The bottom part of the display gives an expanded view of a selected parameter (click on a row in the top part to select one). This is especially useful if the parameter value is too long to show fully in the table display. In most cases you can edit the fields here to change the value and other characteristics of a parameter.

The following items are available in the toolbar:

Add Parameter

Pops up a New Parameter Window to allow you to add a new parameter to the table.

Remove Parameter

If one or more of the parameters displayed in the JTable in this window have been selected by clicking on the relevant rows, then clicking this button will remove them. Some parameters such as Row Count cannot be removed.

A.3.3 Columns Window

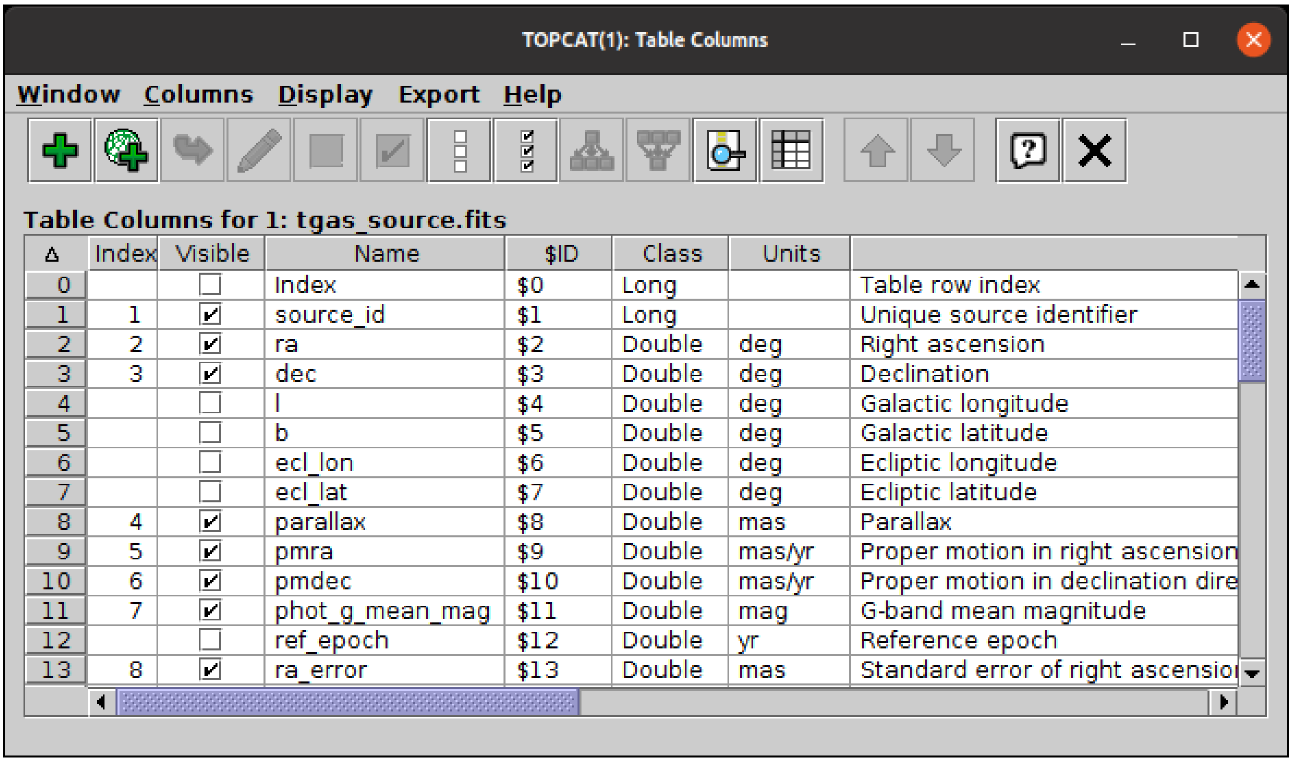


Table Columns for 1: tgas_source.fits

Δ	Index	Visible	Name	$ID	Class	Units	
0		☐	Index	$0	Long		Table row index
1	1	☑	source_id	$1	Long		Unique source identifier
2	2	☑	ra	$2	Double	deg	Right ascension
3	3	☑	dec	$3	Double	deg	Declination
4		☐	l	$4	Double	deg	Galactic longitude
5		☐	b	$5	Double	deg	Galactic latitude
6		☐	ecl_lon	$6	Double	deg	Ecliptic longitude
7		☐	ecl_lat	$7	Double	deg	Ecliptic latitude
8	4	☑	parallax	$8	Double	mas	Parallax
9	5	☑	pmra	$9	Double	mas/yr	Proper motion in right ascension
10	6	☑	pmdec	$10	Double	mas/yr	Proper motion in declination dire
11	7	☑	phot_g_mean_mag	$11	Double	mag	G-band mean magnitude
12		☐	ref_epoch	$12	Double	yr	Reference epoch
13	8	☑	ra_error	$13	Double	mas	Standard error of right ascensio

Columns Window

The Columns Window displays a JTable giving all the information (metadata) known about each column in the table. You can display it using the **Column Info** ( ) button when the chosen table is selected in the Control Window's Table List.

The display may take a little bit of getting used to, since each *column* in the main data table is represented by a *row* in the JTable displayed here. The order and widths of the columns of the JTable widget can be changed in the same way as those for the Data Window JTable, but this has no effect on the data.

The column on the left labelled **Visible** contains a checkbox in each row (one for each column of the data table). Initially, these are all ticked. By clicking on those boxes, you can toggle them between ticked and unticked. When unticked, the column in question will become hidden. The row can still be seen in this window, but the corresponding data column is no longer a part of the Apparent Table, so will not be seen in the Data Window or appear in exported versions of the table.

You can tick/untick multiple columns at once by highlighting a set of rows by dragging the mouse over them and then using the **Hide Selected** (□) or **Reveal Selected** (☑) toolbar buttons or menu items. If you want to hide or reveal all the columns in the table, use the **Hide All** (□□□) or **Reveal All** (☑☑☑) buttons.

If you **select** one of the JTable rows by clicking on it, the table view in the Data Window will be scrolled sideways so that the corresponding data column is visible in (approximately) the middle of the screen. This can be a boon if you are dealing with a table that contains a large number of columns.

Each column in the displayed JTable corresponds to one piece of information for each of the columns in the data table - column name, description, UCD etc. Tables of different types (e.g. ones read from different input formats) can have different categories of metadata. By default a metadata category is displayed in this JTable if at least one table column has a non-blank value for that metadata category, so for instance if no table columns have a defined UCD then the UCD column will not appear. Categories can be made to appear and disappear however by using the **Display** menu. The metadata items are as follows:

Index
The index of the column in the current Column Set (Section 3.3). The column with value "1" here will be the leftmost one in the Data window etc. This value is blank for hidden columns. Sorting on this column (by clicking its header) will show all the visible table columns in order at the top of the display.

Visible
Indicates whether the column is part of the Apparent Table. If this box is not filled in, then for most purposes the column will be hidden from display. You can toggle visibility by clicking on this column.

Name
The name of the column.

$ID
A unique and unchanging ID value for each column. These are useful in defining algebraic expressions (see Section 7) since they are guaranteed unique for each column. Although the column Name can be used as well, the Name may not be unique and may not have the correct form for use in an algebraic expression.

Class
The Java class of the items in that column. You don't have to know very much Java to understand these; they are Float or Double for floating point numbers; Byte, Short, Integer or Long for integer numbers, Boolean for a logical (true/false) flag, or String for a string of ASCII or Unicode characters. There are other possibilities, but these will cover most. The characters '[]' after the name of the class indicates that each cell in the column holds an array of the indicated type.

Shape
Cells of a table can contain arrays as well as scalars. If the column contains an array type, this indicates the shape that it should be interpreted as. It gives the dimensions in column-major order. The last element may be a '*' to indicate that the size of the array may be variable. For scalar columns, this item will be blank.

Element Size
Gives the size of a single element from a scalar or array column. It usually denotes the fixed length, if any, of a String value.

Units

The units in which quantities in this column are expressed.

Domain
Indicates whether TOPCAT recognises this column as representing a particular kind of value such as a timestamp.

Expression
The algebraic expression defining the values in this column. This will only be filled in if the column in question is a synthetic column which you have added, rather than one present in the data in their original loaded form.

Description
A textual description of the function of this column.

UCD
The UCD associated with this column, if one is specified. UCDs are Uniform Content Descriptors, and indicate the semantics of the values in this column.

UCD Description
If the string in the UCD column is the identifier of a known UCD, the standard description associated with that UCD is shown here.

XType
The "extended data type" associated with this column. The standard values for xtype are described in the IVOA DALI standard.

There may be other items in the list specific to the table in question.

You can edit most of these items, e.g. to rename a column or change the expression defining a synthetic column, by double-clicking on them as usual.

By default, the order in which the rows are displayed is determined by the table's current Column Set (Section 3.3). However, you can change the display order in this window by clicking on the column headers (in the same way as for some other JTables). The little up arrow at the top left of the scrolled JTable display indicates that the display is in its "natural" (Column Set) order, but by clicking on headers you can sort by column name, units UCD etc. Clicking sorts up, clicking again sorts down, and a third time (or clicking on the top left) restores natural order.

You can change the order of the columns in the **Column Set** by dragging the grey number cell at the left of the corresponding row up or down. Note however this is only possible for non-hidden columns, and it **only** works if this JTable is currently displayed either in its *natural* order or sorted by the **Index** column (see above) - dragging rows wouldn't have any effect if some other sort order was active. An alternative way to change Column Set order is to drag the column headers left or right in the Data Window.

A good way to find a column in the Data window if your table is too wide to do it by browsing is to sort the table in this window on some suitable item (e.g. Name, Units, UCD), scroll to the column of interest, and then click on it; that causes the view in the **Data Window** to scroll sideways so that the selected column is visible.

The following buttons are available in the toolbar:

New Synthetic Column
This pops up a Synthetic Column Window which allows you to define a new column in terms of the existing ones by writing an algebraic expression. The new column will be added by default after the last selected column, or at the end if none is selected.

Add Sky Coordinate Columns

This pops up a Sky Coordinates Window (Appendix A.11.8) which allows you to define a pair of new sky coordinate columns based on an existing pair of sky coordinate columns.

Replace Column With Synthetic

If a single column is selected, then clicking this button will pop up a Synthetic Column dialogue (Appendix A.11.7) to replace the selected column with a new synthetic one. The dialogue is initialised with the same name, units etc as the selected column, and with an expression that evaluates to its value. You can alter any of these, and the new column will replace the old one, which will be hidden and renamed by appending a suffix like "_old" to its name.

Edit Column Definition

If a single column is selected, then clicking this button will pop up a Synthetic Column dialogue (Appendix A.11.7) that lets you edit its metadata, and Expression if it has one, in place.

Hide Selected Column(s)

If any of the columns are selected, then clicking this button will hide them, that is, remove them from the current Column Set. This has the same effect as deselecting all the checkboxes corresponding to these columns in the **Visible** column.

Reveal Selected Column(s)

If any of the columns are selected, then clicking this button will make sure they are visible, that is, that they appear in the current Column Set. This has the same effect as selecting all the checkboxes corresponding to these columns in the **Visible** column.

Hide All Columns

Clicking this button will hide all the columns in the table; the table will have no columns visible in it following this. If you just want to see a few columns, it may be convenient to use this button and then select a few visible ones individually to reveal.

Reveal All Columns

Clicking this button will ensure that all the table's columns are visible (none are hidden).

Explode Array Column

If a column is selected which has an array type and a fixed number of elements, clicking this button will replace it with scalar-valued columns containing each of its elements. For instance if a column PMAG contains a 5-element vector of type `float[]` representing magnitudes in 5 different bands, then selecting it and hitting this button will hide PMAG and insert 5 new `Float`-type columns PMAG_1...PMAG_5 in its place each containing one of the magnitudes. If the column does not have a fixed number of elements listed in the **Shape** column of this window, this button is disabled. In that case, if you know how many columns you want to explode it into, you can enter that value into the Shape field by double-clicking on it. This will only work for columns that are actually arrays.

Collapse Columns to Array

If multiple (*N*) numeric columns are selected, clicking this button will prompt for the name of a new column containing *N*-element array values, collected from all the selected columns. Currently, the output type will always be `double[]`, and blank values in the input columns will show up as blank array elements (NaNs). Currently, the elements in the output column will appear in the order of the input columns' appearance in the table, regardless of the current ordering of the rows in this window; the new column's Description text lists the input columns in order, if there are not too many. The effect is more or less the opposite of the Explode option above.

Search Column

Pops up the Column Search Window that allows you to find rows in the display with a given field matching text that you enter. It can be useful if you have a very wide data table.

Import as Table

The table of column metadata as displayed by this window (rows corresponding to table columns and columns corresponding to items of metadata) is itself a table. This action loads it into TOPCAT as a new table so it can be manipulated in all the usual ways.

Sort Selected Up

If a single column is selected then the table's current Sort Order will be set to sort ascending on that column. Otherwise this action is not available.

Sort Selected Down

If a single column is selected then the table's current Sort Order will be set to sort descending on that column. Otherwise this action is not available.

Several of these actions operate on the currently selected column or columns. You can select columns by clicking on the corresponding row in the displayed JTable as usual.

As well as the normal menu, right-clicking over one of the columns in the displayed table will present a **Column Popup Menu**, which provides a convenient way to do some things with the column under the mouse cursor:

Search

Pops up the Column Search Window that allows you to find rows in the display for which the chosen column value matches text you enter. It can be useful if you have a very wide data table.

Sort up

Sorts the table rows according to ascending value of the contents of the column. Only available if some kind of order (e.g. numeric or alphabetic) can sensibly be applied to the column.

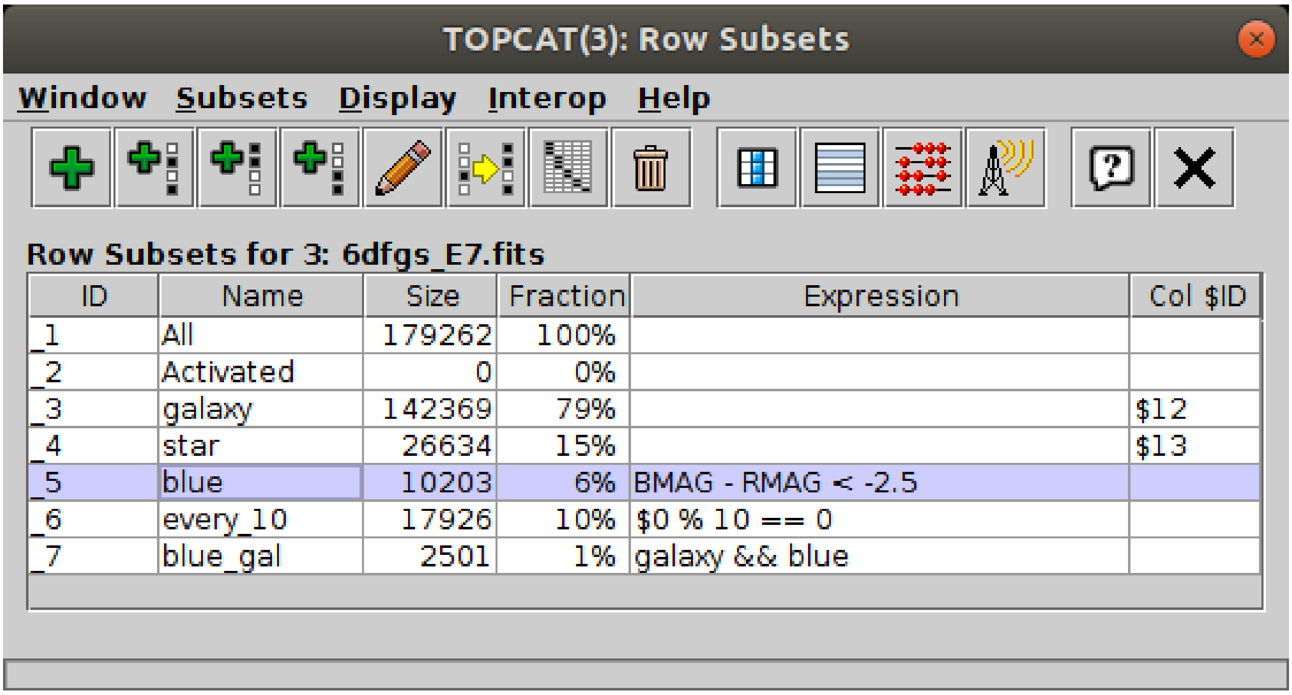
Sort down

Sorts the table rows according to descending value of the contents of the column. Only available if some kind of order (e.g. numeric or alphabetic) can sensibly be applied to the column.

Hide

Hides the column. It can be reinstated from the **Display** menu.

A.3.4 Subsets Window



Row Subsets for 3: 6dfgs_E7.fits

ID	Name	Size	Fraction	Expression	Col $ID
_1	All	179262	100%		
_2	Activated	0	0%		
_3	galaxy	142369	79%		$12
_4	star	26634	15%		$13
_5	blue	10203	6%	BMAG - RMAG < -2.5	
_6	every_10	17926	10%	$0 % 10 == 0	
_7	blue_gal	2501	1%	galaxy && blue	

Subsets Window

The Subsets Window displays the Row Subsets (Section 3.1) which have been defined. You can display it using the **Row Subsets** ( ) button when the chosen table is selected in the Control Window's Table List.

Two special subsets are always present: **All** includes the whole table, and **Activated** contains a single row if one has been activated by clicking on a row or point that corresponds to it.

The subsets are displayed in a JTable widget with a row for each subset. You can interact with this JTable in the usual ways, for instance dragging columns sideways, changing their widths, and sorting the entries by clicking on the headings.

The columns of the JTable are as follows:

_ID
: A unique and unchanging identifier for the subset, which consists of a "_" character (underscore) followed by an integer. This can be used to refer to it in expressions (Section 7) for synthetic columns or other subsets.

  Note: in previous versions of TOPCAT the hash sign ("#") was used instead of the underscore for this purpose; the hash sign no longer has this meaning.

Name
: A name used to identify the subset. It is ideally, but not necessarily, unique. It can be edited (double-click on the cell) to change the name.

Size
: The number of rows in this subset. This column is usually filled in when the subset is defined, but it is not guaranteed to remain correct if the table data change, since counting may be an expensive process so it is not automatically done with every change. If values in this column are blank or suspected incorrect, a recount can be forced by using the **Count Subsets** ( ) button described below.

Fraction
: Shows the same information as the preceding **Size** column, but as a percentage of the total

number of rows in the table.

Expression
If the subset has been defined by an algebraic expression, this will be here. It can be edited (double-click on the cell) to change the expression.

Column $ID
If the subset has been defined by equivalence with a boolean-valued column, this will show the $ID of the column that it came from (see Appendix A.3.3).

Entries in the **Name** and **Expression** columns can be edited by double-clicking on them in the normal way.

The following toolbar buttons are available in this window:

New Subset
Pops up the Algebraic Subset Window to allow you to define a new subset algebraically.

Add Sample Subset
Pops up a dialogue window to allow you to define a new subset consisting of every *N*'th row of the table.

Add Head Subset
Pops up a dialogue window to allow you to define a new subset consisting of the first *N* rows of the table.

Add Tail Subset
Pops up a dialogue window to allow you to define a new subset consisting of the last *N* rows of the table.

Edit Subset
Pops up the Algebraic Subset Window that lets you edit the subset's Name, and Expression if it has one, in place.

Invert Subset
Creates a new subset which is the complement of the selected one. The new one will include all the rows which are excluded by the selected one (and vice versa). To use this action, first select a subset by clicking on its row in the JTable.

Classify By Column
Pops up the Column Classification Window, which can add a set of mutually exclusive subsets based on the contents of a given column or expression.

Remove Subset
Deletes one of the subsets in the list. Once deleted, a subset cannot be recovered. Note that attempts to use its name or _ID in algebraic expressions which you add or modify in the future will fail, though its use in existing expressions will continue to work.

To Column
If one of the rows in the JTable is selected, this will turn that subset into a new column. It will pop up the Synthetic Column Window, filled in appropriately to add a new boolean column to

the table based on the selected subset. You can either accept it as is, or modify some of the fields. To use this action, first select a subset by clicking on its row in the JTable.

Highlight Subset

Highlights the contents of this subset by marking the rows visibly in the data window and also ensuring that the rows are visible in any plot windows. This replicates what happens when the subset is first created.

Count Rows

Counts how many rows are in each subset and displays this in the **Size** column. This forces a count or recount to fill in or update these values.

Broadcast Subset

Causes the selected subset to be broadcast using SAMP or PLASTIC to other listening applications. Any other listening application which is displaying the same table is then invited to highlight the selection of rows corresonding to the selected subset. This option and the corresponding **Send Subset to ...** option are also available from the **Interop** menu. See Section 9 for more information about tool interoperability.

The following additional menu items are available:

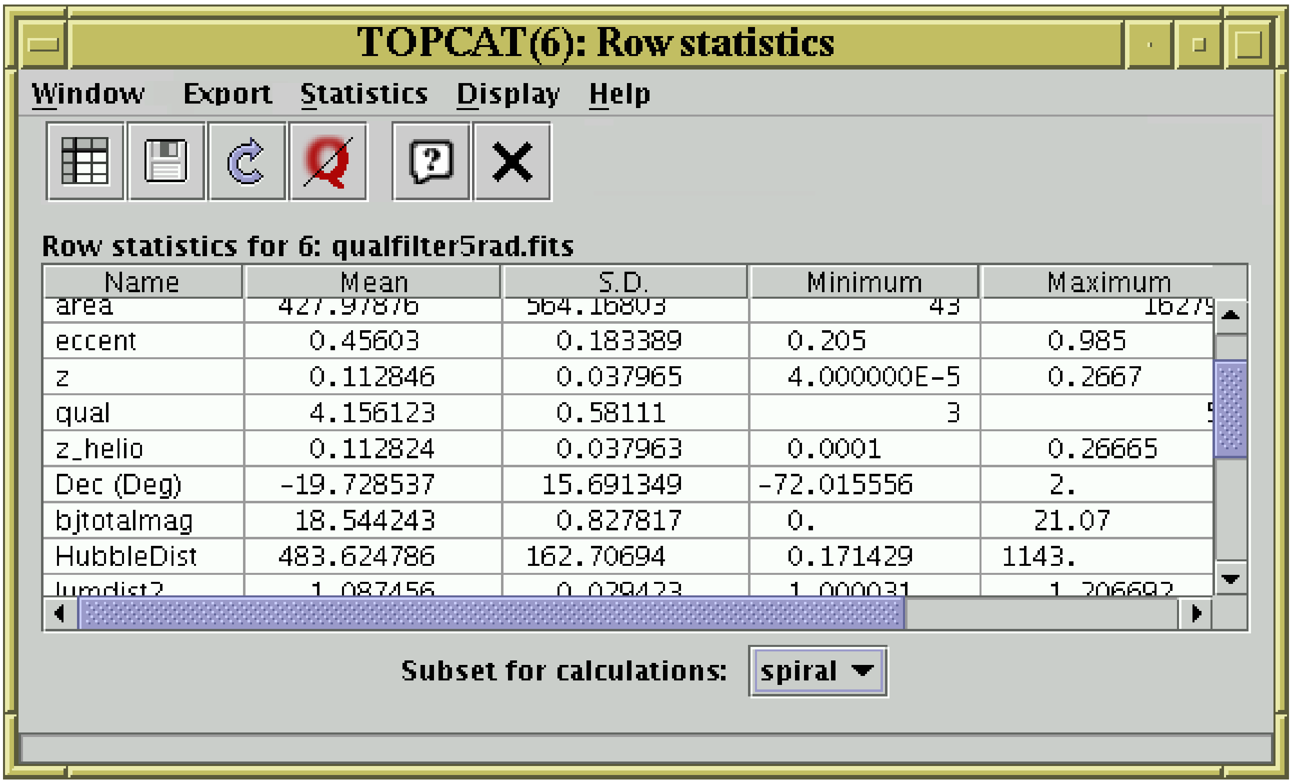
Send subset to ...

As for the **Broadcast Subset** item above, but sends the subset to only a single selected application. A submenu will give a list of all the currently registered applications which can make sense of a subset. If there are none, this item will be disabled.

Autocount rows

Normally, the **Size** and **Fraction** columns of the displayed table (see above) are filled in as soon as a new subset is defined. However for very large tables this could be slow - if you want to prevent this autocounting you can deselect this menu item. In this case the Size and Fraction columns will only be filled in when the **Count** () button is hit or if TOPCAT finds out the size as the result of some other operation (such as plotting).

A.3.5 Statistics Window



Row statistics for 6: qualfilter5rad.fits

Name	Mean	S.D.	Minimum	Maximum
area	427.97876	564.16803	43	16279
eccent	0.45603	0.183389	0.205	0.985
z	0.112846	0.037965	4.000000E-5	0.2667
qual	4.156123	0.58111	3	5
z_helio	0.112824	0.037963	0.0001	0.26665
Dec (Deg)	-19.728537	15.691349	-72.015556	2.
bjtotalmag	18.544243	0.827817	0.	21.07
HubbleDist	483.624786	162.70694	0.171429	1143.
lumdist2	1.087456	0.029423	1.000031	1.206692

Subset for calculations: spiral

Statistics Window

The Statistics Window shows statistics for the values in each of the table's columns. You can display it using the **Column Statistics** (Σ) button when the chosen table is selected in the Control Window's Table List.

The calculated values are displayed in a JTable widget with a row for each column in the main table, and a column for each of a number of statistical quantities calculated on some or all of the values in the data table column corresponding to that grid row.

You can interact with this JTable in the usual ways, for instance dragging columns sideways, changing their widths, and sorting the entries by clicking on the headings.

The following columns are shown by default:

Name
The name of the column in the main table represented by this grid row.

Mean
The mean value of the good cells. For boolean columns, this is the proportion of good cells which are True.

SD
The population standard deviation of the good cells.

Minimum
The minimum value. For numeric columns the meaning of this is quite obvious. For other columns, if an ordering can be reasonably defined on them, the 'smallest' value may be shown. For instance string values will show the entry which would be first alphabetically.

Maximum
As minimum, but shows the largest values.

nGood
The number of non-blank cells.

Several additional items of statistical information are also calculated, but the columns displaying these are hidden by default to avoid clutter. You can reveal these by using the **Display** menu:

Index
The index of the column in the table, i.e. the order in which it is displayed.

$ID
The unique identifier label for the column in the main table.

Sum
The sum of all the values in the column. For boolean columns this is a count of the number of True values in the column.

Variance
The population variance of the good cells.

Sample SD
The sample standard deviation of the good cells.

Sample Variance
The sample variance of the good cells.

Median Absolute Deviation
The median of absolute deviations from the median: `median(abs(x-median(x))`. This is a robust measure of statistical dispersion.

Scaled Median Absolute Deviation
The **Median Absolute Deviation** (see above) multiplied by 1.4826. This is supposed to be a consistent estimator for the standard deviation, on the assumption of a normal distribution.

Skew
Gamma 1 measure of skewness of the value distribution.

Kurtosis
Gamma 2 measure of peakedness of the value distribution.

Row of min
The index of the row in the main table at which the minimum value occurred.

Row of max
The index of the row in the main table at which the maximum value occurred.

nBad
The number of blank cells; the sum of this value and the Good cells value will be the same for each column.

Cardinality
If the column contains a small number of distinct values then that number, the column's *cardinality* will be shown here. Cardinality is the number of distinct values which appear in that column. If the number of values represented is large (currently >50) or a large proportion of the non-bad values (currently >75%) then no value is shown.

Some of these quantities are suitable only for array-valued columns, and calculate per-element array statistics that are arrays of the same length as the input values (the input arrays must all be the same length):

Array nGoods
Per-element count of the number of non-blank values in the input arrays.

Array Sums
Per-element sum of the values in the input arrays.

Array Means

Per-element mean of the values in the input arrays.

Array SDs
Per-element population standard deviation of the values in the input arrays.

In addition, some **quantile** values can calculated on demand (by selecting their values in the **Display** menu, as for the previous list). The available values are:

Q001:
value below which 0.1% of rows fall

Q01:
value below which 1% of rows fall (1st percentile)

Quartile1:
value below which 25% of rows fall (first quartile)

Median:
value below which 50% of rows fall (median)

Quartile3:
value below which 75% of rows fall (third quartile)

Q99:
value below which 99% of rows fall (99th percentile)

Q999:
value below which 99.9% of rows fall

These are considerably more expensive to calculate than the other statistical quantities, and so they are not provided by default (the same applies to the MAD). If you attempt to calculate them for large tables, you may get a message saying that there is insufficient memory. In this case you can use an approximate quantile calculation method which is not memory limited: see the description below of the **Approximate Quantile Calculation** ( ) option.

The quantities displayed in this window are not necessarily those for the entire table; they are those for a particular Row Subset (Section 3.1). At the bottom of the window is the **Subset For Calculations** selector, which allows you to choose which subset you want the calculations to be done for. By clicking on this you can calculate the statistics for different subsets. When the window is first opened, or when it is invoked from a menu or the toolbar in the Control Window, the subset will correspond to the current row subset.

The toolbar contains the following extra buttons:

Save as Table
Clicking this button will save the quantities displayed in this window to a table on disk. It can be saved in any of the tabular formats which TOPCAT understands.

Import as Table
The table of statistical quantities displayed by this window (rows corresponding to input table columns and columns corresponding to statistical quantities) is itself a table. By clicking this button it can be loaded into TOPCAT as a new table and manipulated in all the usual ways. This has the same effect as saving the statistics to file (see previous button) and then reloading that file.

Recalculate
Once statistics have been calculated for a given subset they are cached and not normally

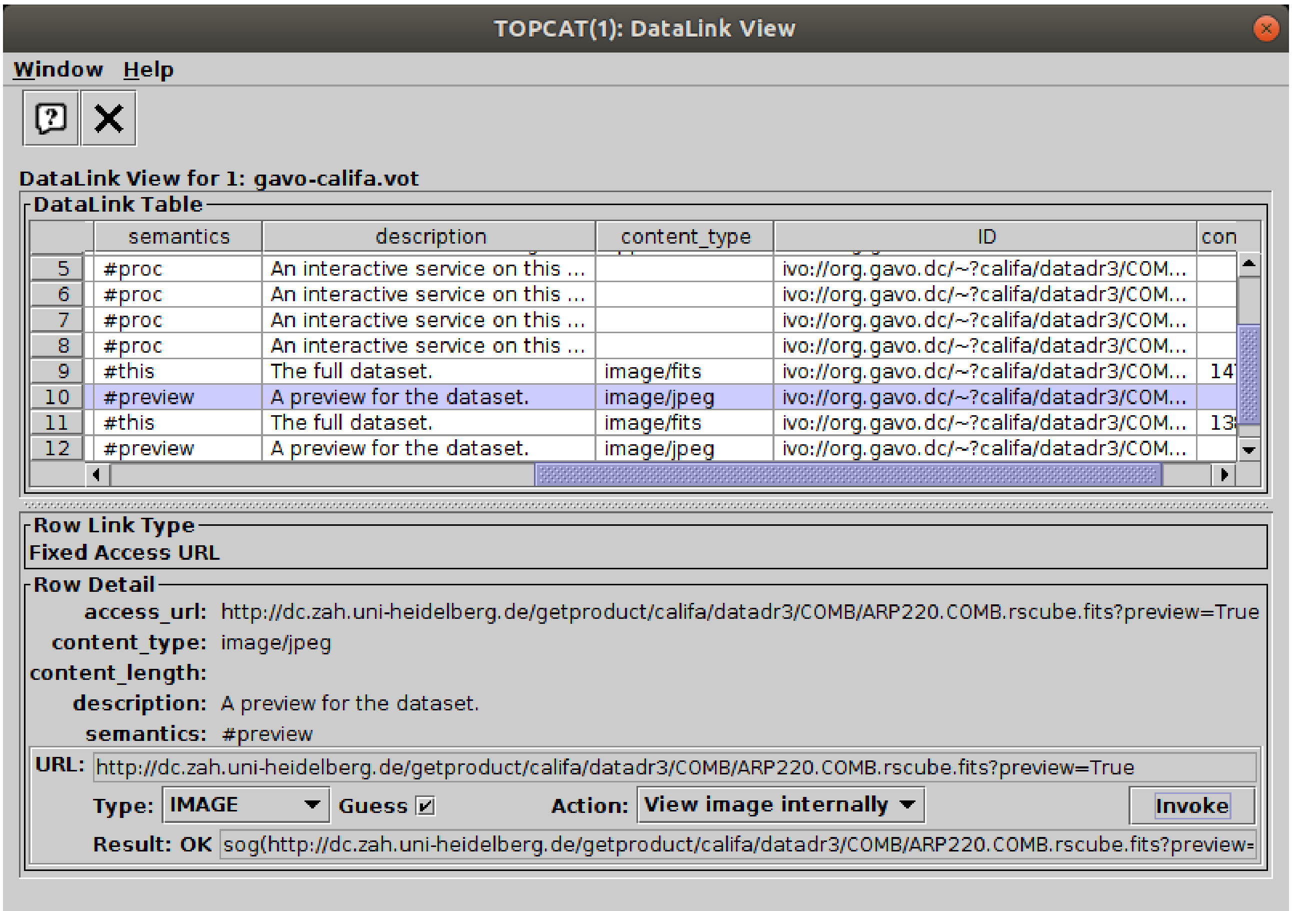
recalculated again. Use this button if you want to force a recalculation because the data may have changed.

Approximate Quantile Calculation

If selected this button will cause the quantiles to be calculated using a method which is both approximate and slower than the default (exact) method, for which reason it's usually not preferred. However, the approximate method executes in constant memory, while the exact method can fail by running out of memory for very large row counts.

For a large table the calculations may take a little while. While they are being performed you can interact with the window as normal, but a progress bar is shown at the bottom of the window. If you initiate a new calculation (by pushing the Recalculate button or selecting a new subset) or close the window during a calculation, the superceded calculation will be stopped.

A.3.6 DataLink Window



DataLink View for 1: gavo-califa.vot

DataLink Table

	semantics	description	content_type	ID	con
5	#proc	An interactive service on this ...		ivo://org.gavo.dc/~?califa/datadr3/COM...	
6	#proc	An interactive service on this ...		ivo://org.gavo.dc/~?califa/datadr3/COM...	
7	#proc	An interactive service on this ...		ivo://org.gavo.dc/~?califa/datadr3/COM...	
8	#proc	An interactive service on this ...		ivo://org.gavo.dc/~?califa/datadr3/COM...	
9	#this	The full dataset.	image/fits	ivo://org.gavo.dc/~?califa/datadr3/COM...	14
10	#preview	A preview for the dataset.	image/jpeg	ivo://org.gavo.dc/~?califa/datadr3/COM...	
11	#this	The full dataset.	image/fits	ivo://org.gavo.dc/~?califa/datadr3/COM...	13
12	#preview	A preview for the dataset.	image/jpeg	ivo://org.gavo.dc/~?califa/datadr3/COM...	

Row Link Type
Fixed Access URL

Row Detail

access_url: http://dc.zah.uni-heidelberg.de/getproduct/califa/datadr3/COMB/ARP220.COMB.rscube.fits?preview=True
content_type: image/jpeg
content_length:
description: A preview for the dataset.
semantics: #preview

URL: http://dc.zah.uni-heidelberg.de/getproduct/califa/datadr3/COMB/ARP220.COMB.rscube.fits?preview=True

Type: IMAGE Guess ☑ Action: View image internally Invoke

Result: OK sog(http://dc.zah.uni-heidelberg.de/getproduct/califa/datadr3/COMB/ARP220.COMB.rscube.fits?preview=

DataLink Window

The DataLink Window, unlike the other Table View Windows, is rather specialist, and only applies to a particular type of table, namely the *{links}-response* table format defined by the DataLink standard. This format is sometimes loosely known as the DataLink format, and is used to represent links of various kinds to external data resources.

Because of its specialist nature, this window does not appear in the Control Window toolbar like the other view windows, so to open it you have to select the **DataLink View** item from the **Views** menu in the Control Window. This menu item will only be enabled if the loaded table looks like it has the appropriate form.

The upper panel of this window displays the table content just like the Data Window, but the lower panel contains link-specific information about the row that is currently selected in the upper one.

The **Row Link Type** sub-panel reports what type of link the given row represents. Its value depends on which table columns are filled in, and may be one of:

- **Fixed Access URL:** The link is to a URL given in the `access_url` column.
- **Service Invocation:** The link is to a parameterised URL defined by a service descriptor referenced in the `service_def` column, with parameters supplied by other columns and/or user interaction.
- **Link Error:** The row reports an error in the `error_message` column indicating why a requested link cannot be supplied.

The **Row Detail** panel provides additional information on the row. For the rows that actually represent a link, information such as link semantics and resource content type are listed where available, along with a URL sub-panel as below.

The **URL** sub-panel, if present, shows the actual link URL and provides options to invoke it, i.e. to do something with the listed resource. The parts of this panel are:

URL
Displays the URL that can be invoked, with per-row parameters substituted in if applicable. You can cut and paste from this field to copy the URL if required.

Type
Displays TOPCAT's best guess, given the information available in the row, about what kind of resource the link represents. It doesn't always get it right. If it's wrong, you can select one of the options manually from this selector instead. The value in this selector is by default updated every time you select a new row in the table. However, if you want the selected type to be "sticky", i.e. not to be automatically updated, uncheck the **Guess** checkbox on the right of it. Then the selected value will stay the same until it's changed manually. The purpose of this selector is to give a hint to the Action selection.

Action
Provides a list of possible actions to perform on the identified URL. The action depends on the value of the **Type** field; when the Type changes, the Action changes to the value it had last time that Type was selected. Each Type has a default Action, but you can change it by choosing from the selector. If you don't want the action to get updated automatically, uncheck the checkbox on the left as explained above. Most of the action options have behaviour that is similar to a corresponding activation action, as noted below. The options are:

- **Report URL:** displays the URL in the Result field
- **View image internally:** displays the linked resource (an image in a format like PNG, GIF, JPEG, FITS) in an internal image viewer (like the Display Image action)
- **View FITS image internally:** displays the linked resource, a FITS image, in an internal image viewer (like the Display Image action)
- **Load Table:** loads the first table from the linked resource (a FITS or VOTable table) into this TOPCAT application (like the Load Table action)
- **Load Tables:** loads all tables from the linked resource (a FITS or VOTable table) into this TOPCAT application (like the Load Table action with Multiple Tables option selected)
- **Plane Plot Table:** pops up a plane plot window that allows to perform plots using the table referenced by the linked resource, without actually loading the table into the TOPCAT application (like the Plot Table action with Plot Type = Plane)
- **Corner Plot Table:** pops up a corner plot window that allows to perform plots using the table referenced by the linked resource, without actually loading the table into the TOPCAT application (like the Plot Table action with Plot Type = Corner)

- **View DataLink Table:** displays the linked resource (a DataLink table) in a new window like this one (like the View Datalink Table action)
- **Send FITS Image:** sends the linked resource (a FITS image) using SAMP to external image viewers (like the Send FITS Image action)
- **Send Spectrum:** sends the linked resource (a spectrum) using SAMP to external spectrum viewers (like the Send Spectrum action)
- **Send Table:** sends the linked resource (a VOTable) using SAMP to external table viewers (like the Send VOTable action)
- **Download URL:** downloads the linked resource to a local file, chosen using a popup filesystem browser
- **Show Web Page:** opens the linked resource in the system web browser (like the View in Web Browser action)

Note that there currently is not much opportunity to customise the behaviour of the various Action options supplied by this window; these actions are less configurable than their corresponding Activation Actions.

Invoke
Hitting this button actually invokes the selected Action on the current row.

Result
Displays the outcome of the most recently invoked URL. An OK or FAIL indicator is displayed, along with a text message that may provide some detail.

In the case of a Service Invocation link type, there may be user-supplied parameters for the link. If so, a **Parameters** sub-panel appears at the bottom with a list of the available parameters (defaults may or may not be present) to allow you to enter values. Note that the user interface for supplying these parameters is currently very basic, and may be improved in future releases.

A.4 Plot Windows

This section describes the plotting windows introduced at TOPCAT version 4. These provide most of the same functionality as the old-style plot windows (Appendix A.5), but additionally much more flexibility, configurability, extensibility and better performance and capabilities for making visual sense of very large data sets.

TOPCAT has a number of windows for performing data visualisation of various kinds. These share various characteristics which are described in the first subsection below; the specific windows themselves are described in the later subsections.

Seven plot windows are currently available:

- Histogram Plot
- Plane Plot
- Sky Plot
- Cube Plot
- Sphere Plot
- Corner Plot
- Time Plot

More may be introduced in future releases.

The rest of this section describes the plotting functions in detail.

A.4.1 Differences From Old-Style Plot Windows

A brief summary of improvements these windows offer over the old-style plot windows is:

New Sky Coordinate Plot

- Choice of projection: Sin (rotatable), Aitoff, Plate Carrée
- Data and view sky coordinate systems selected separately: options are equatorial, galactic, supergalactic, ecliptic
- Sky coordinate grid labelled, configurable and visible at all zooms

New data plot options

- Vectors
- Ellipses (with position angle)
- Pair data point links
- Contours
- Variable size markers
- Kernel density estimates
- ... and many more

Improved interactive response

- Various mouse wheel and drag actions pan and zoom the plot instantly
- In 3d plots right mouse button recentres cube on selected point
- In 3d plots zooming zooms data in the cube rather than enlarging the cube wireframe itself
- Many controls are sliders which update the plot as you slide

Better support for large datasets

Several features have been introduced to provide more meaningful visualisation of large datasets. Improved density-like plots and contours give you better ways to understand plots containing many more points than there are pixels to plot them on. There is separately some improvement in scalability: you can typically get reasonable interactive performance up to about 10 million points depending on available memory etc and what you're doing (though there are reports of it working with several 10^8 points). The intention is to improve this limit further in future.

New plot shading modes

Density colour coding for all plot types, with colour map either absolute or modifying dataset base colour. The result is something that looks like a scatter plot at low densities and a density plot for high densities, and generally means that you can easily make quantitative sense of overcrowded regions of a scatter plot. Flat, transparent and aux colour coding still available as before.

Improved axis labelling

- Choice of font size and style
- Option of LaTeX input for non-ASCII characters etc
- Log axes labelled better
- Minor tick option

Analytic function plotting in 2D

Plot functions of X or Y coordinate using algebraic expression language.

Configurability

Many more configuration options including legend placement, grid display and colour, antialiasing, text label crowding limits etc etc.

STILTS export
The GUI can report the text for the STILTS command that would reproduce the currently visible plot.

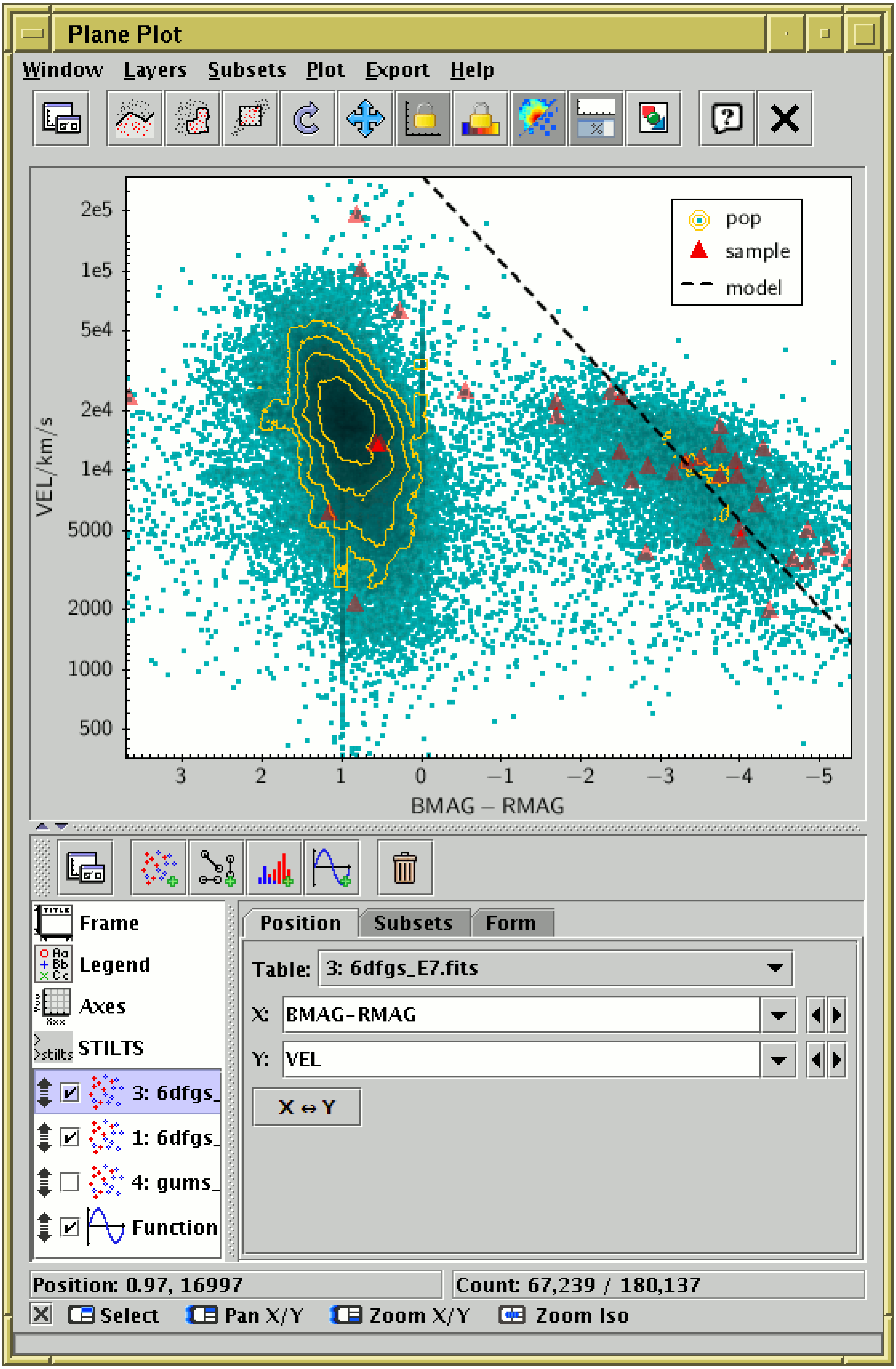
The new windows allow you to assemble a stack of layers representing different plot types of different data sets on the same axes. The user interface for controlling this is quite a bit different than for the old-style plot windows, and is described in the subsequent sections. However, making a simple plot is still simple: select a table, select the columns, and you're off.

Some features from the old-style plots that are *not* currently available in the new-style plots are:

- true RGB density maps (though density shading modes will often do as good or a better job)
- fogging in 3D plots

These may be introduced in a future release, possibly informed by user demand.

A.4.2 Plot Window Overview



☑ 3: 6dfgs_

☑ 1: 6dfgs_

□ 4: gums_

☑ Function

Table: 3: 6dfgs_E7.fits

X: BMAG-RMAG

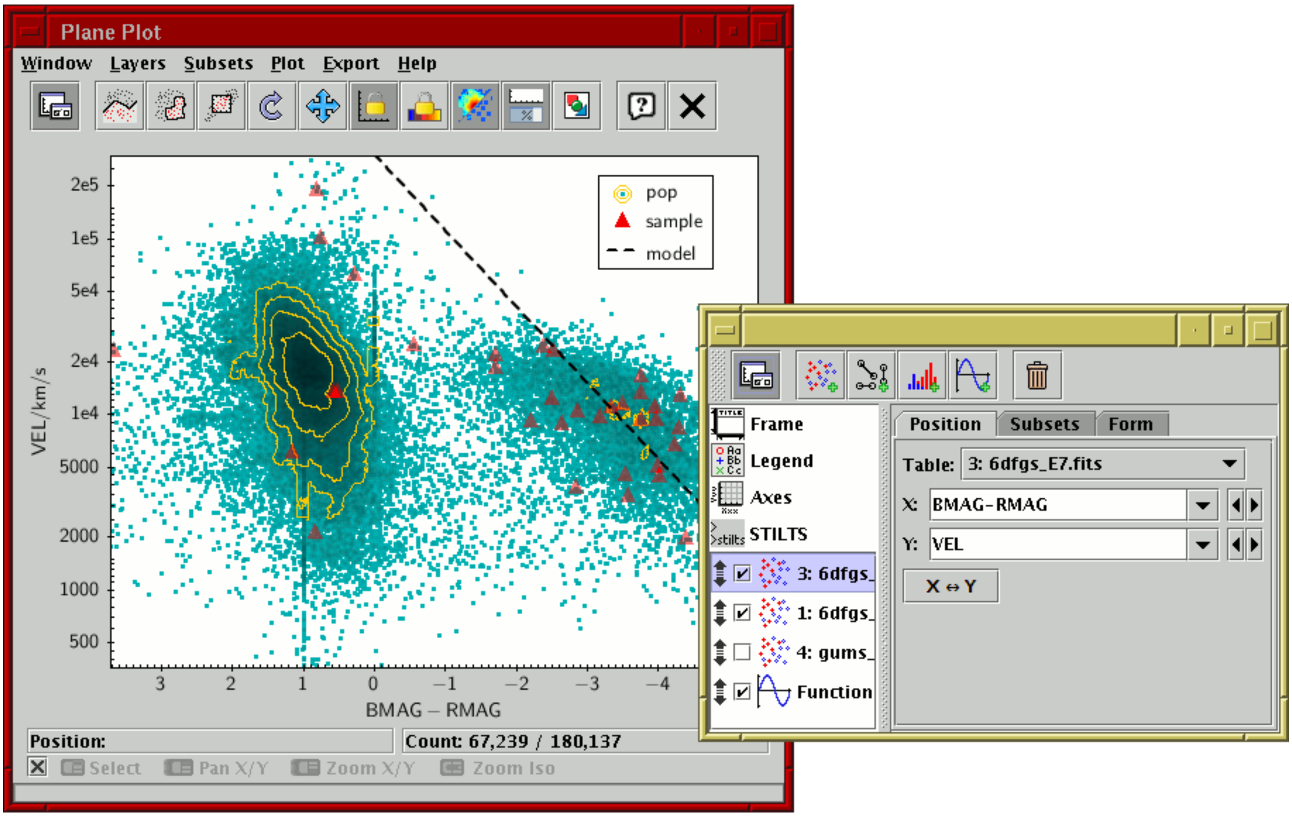
Y: VEL

Position: 0.97, 16997

Count: 67,239 / 180,137

Plot window

Plot windows consist of two main parts: the **Plot Panel** containing the actual plotted graphics (by default, at the top), and the **Control Panel** (by default, at the bottom). The Control Panel is where you configure what will be plotted. For a simple scatter plot it may just be a case of selecting what columns to plot against each other, but it can get quite detailed. If you want more screen space to play with, it can be helpful to float the control panel into a separate window using the **Float Controls** ( ) toolbar button; you can find this button in both the main and the control panel toolbars. To unfloat the control panel, either just close the control panel window, or click the Float Controls toolbar button again. With floating controls, the window looks like the following figure.



Position:

Count: 67,239 / 180,137

Table: 3: 6dfgs_E7.fits

X: BMAG-RMAG

Y: VEL

Plot window with floated control panel

The **control panel** itself has two main parts: a **control stack** on the left containing currently active controls (represented by names and icons), and a **detail panel** on the right which shows information about the currently selected control. Click on one of the control entries on the left to see its details on the right. Different controls have different detail panels, but in general each one will have multiple tabs for configuring different things. You can select these by clicking on the tab names. A good way to learn about the options is to click on the different controls and their tabs to see what's available and experiment with the various options to see what happens to the plot, but all the panels and tabs are also explained in this manual. The control panel also has a toolbar at the top, used for adding and removing controls from the stack.

The **control list** has two types of entry:

Fixed controls:

These control the overall plot appearance. In the above figure, the fixed controls **Frame** ( ), **Legend** ( ), **Axes** ( ) and **STILTS** ( ) are visible.

The different fixed controls are described in Appendix A.4.3.

Layer controls:

These determine the actual data that will be plotted and what graphical form it takes; each control contributes a layer or layers to the plot. To add a layer control, use the toolbar just above; each button with a little green "+" adds a control of the corresponding type. To remove a layer control, select it and use the **Remove Current Layer** (🗑) button in the toolbar. You can also use the **Layers** menu at the top of the main window to add and remove layers.

The checkbox beside each control determines whether it is currently active; if unchecked, any plotting instructions is contains will be ignored. You can set them active or inactive by clicking on the checkboxes.

You can also drag each control up or down by dragging with the grab handle (⇕). Layers lower down the list are plotted later (perhaps obscuring earlier one), so you can drag them up and down until you have the layers you want on top.

The different layer controls are described in Appendix A.4.4.

In the foot of the window, there are also four other small panels:

Position Panel

When the cursor is positioned over the plot itself, this reports its position in data coordinates. In some cases, such as 3-d plots, this may not be possible, in which case it's blank.

Count Panel

Displays the number of points currently plotted and the total number of points represented by the plot. The total (the second figure) is the number of positions in all the plotted data sets, and the current number (the first figure) excludes those in subsets not currently plotted and those outside the bounds of the visible plot.

Navigation Help

Shows some reminders for what different mouse gestures do. Little icons are supposed to represent clicking and dragging different mouse buttons and the mouse wheel. Note the content changes according to where the mouse is on the plot, since that affects navigation behaviour. For more information see Appendix A.4.2.1. Clicking on the ✖ button will make this panel go away.

Progress Bar

If something slow is happening you may see a progress bar at the bottom of the screen while you wait. Unless you are plotting several million points, you may not see this at all. For slow data loads, this will always be displayed. For actions like actually drawing the plot and turning a blob into a subset selection you can choose whether progress is shown using the **Show Plot Progress** ( ) button in the toolbar. It's nice to know that something's going on, but it can be distracting; displaying progress also slows the plot down a bit.

The **main toolbar** at the top of the window contains the following actions (repeated in the menus):

Float Controls

Puts the Control Panel into a floating window rather than at the bottom of the plot window, as described above. Once floating, the control panel can be joined back to the main window by clicking this button again.

Draw Subset Blob

Allows you to draw a region on the screen defining a new Row Subset. When you have finished drawing it, click this button again to indicate you're done. See Appendix A.4.2.3.3 for more details.

Subset From Visible

Defines a new **Row Subset** consisting of only the points which are currently visible on the plotting surface. See Appendix A.4.2.3.1 for more explanation.

Replot

Redraws the current plot. It is usually not necessary to use this button, since if you change any of the plot characteristics with the controls in this window the plot will be redrawn automatically. However if you have changed the data, e.g. by editing cells in the Data Window, or by redefining a subset, the plot is not automatically redrawn. Clicking this button redraws the plot taking account of any changes to the table data.

Rescale

Rescales the axes of the current plot so that it contains all the data points in the currently selected subsets. By default the plot will be initially scaled like this, but it it may have changed because of changes in the subset selection or from zooming in or out.

Lock Axes

Usually, when the data plotted has changed significantly, the axes are automatically rescaled so that all the points are visible. The application makes a guess about when it's a good idea to do this automatic rescaling. If you don't want it to auto-rescale, set this toggle button, and it won't rescale unless it really has to. This is not available for the **Sky Plot**.

Lock Aux Range

This controls when the Aux data range, sometimes used to colour data points (see the Aux Axis Control), is updated to match the currently visible data. By default, the range is updated dynamically as the plot changes, for instance when you pan or zoom it, so that the data range covered by the aux colour ramp matches the range of the currently visible data. But if this checkbox is checked, then the range is frozen to the current value. Data-sensitive updates to the aux range will then not be performed until it's unchecked again.

It also affects some other dynamic ranging calculations such as **Auto-Scale** sizes for plot forms Size, SizeXY, Vector, SkyVector, Ellipse, SkyEllipse, XYCorr and SkyCorr.

Sketch Frames

If selected causes intermediate "sketch" frames to be drawn when navigating around very large plots. For plots that take a long time (at least a non-negligable fraction of a second) to draw, if this option is selected then when navigating around it will paint intermediate frames based on a subsample of the data rather than painting the whole plot at every step. This can result in a somewhat flickering appearance, but it means that frame updates happen more frequently, so it's a bit more responsive.

Show Plot Progress

If selected a progress bar at the bottom of the window is active when large (slow) plots are in progress. This can be useful if you are navigating round a very large plot so that you can see something is happening rather than the application apparently just doing nothing. On the other hand a flickering progress bar can be distracting. Updating the progress bar may also slow the plot down a little.

Export Plot

Allows you to save the plot in a variety of graphics formats using the Plot Export window.

The window **menus** offer an alternative way to perform the actions available from the toolbar described above. They also provide some additional options:

Layers menu

This menu repeats the options available in the toolbar from the Control Panel at the bottom of the window; each one adds a new Layer Control of one of the available types to the stack, or **Removes** (🗑) the currently selected control.

Export menu

This menu provides some options for exporting graphics and data from the plot to external contexts:

Export Plot

Saves the visible plot in an image format; see Appendix A.4.2.5.

STILTS Command Window

Displays a command which can be executed from outside TOPCAT to reproduce the currently visible plot; see Appendix A.4.3.4.

Layer Data Import

Layer Data Save

These two sub-menus may or may not contain options. Some plotted layer types (for instance 1d or 2d histograms) generate table data as part of their calculations that can be exported separately. If one of these layers is currently plotted, then options may appear in these menus. The **Import** options retrieve the table and add it to the list of tables currently loaded in TOPCAT; the **Save** options can write the data directly to disk using one of the supported table formats.

The following subsections explain some other features common to all the plotting windows.

A.4.2.1 Navigation

All the plot windows are interactive, and you can navigate around them using the mouse buttons. For most plot types the left and right buttons and the mouse wheel do something; in 3d plots the middle button is used as well.

The details of what mouse actions do what depend on which plot window you are using, and they can also be configured to some extent using the **Navigation** tab in the **Axes** fixed control in the control stack. However, as a rule, the actions when used on the body of the plot are these:

Left drag

Drags the plot around. Where possible, the same plot position stays under the cursor as you drag. In 2d this pans the plot left/right/up/down, and in 3d it rotates it.

Right drag

Stretch zoom. Dragging up/down stretches/squashes the plot vertically, and dragging left/right stretches/squashes it horizontally. The zoom is centered on the position where you start the drag from, so that data position stays in the same place on the screen. In 3d, the zoom is along the two plot directions most closely aligned with the plane of the screen.

Middle drag

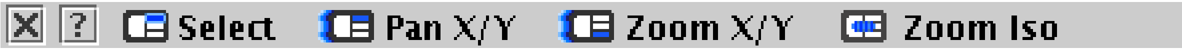
Frame zoom. Dragging right-and-down or right-and-up drags out a frame; when the button is released, the plot will be zoomed in to cover area enclosed by the frame. Dragging left (and up or down) does something like the opposite, you can zoom out using a similar (though not quite the same) mechanism.

Mouse wheel
Spinning the mouse wheel forwards/backwards will zoom in/out. In 2d the zoom is around the current position of the mouse, and in 3d it is (usually) around the center of the view cube.

Left click
Identifies a point. If there is a plotted point near the cursor, it will plot a marker on it and activate (Section 8) it. If you click on an empty bit of the plot, any existing activated point will be removed, otherwise nothing will happen.

In the 2d plot types, you can pan or zoom in just one direction by using the same actions outside of the plot itself. If you do a pan/zoom action to the left of the Y axis, it will pan/zoom only vertically, and if you do it below the X axis, it will do it only horizontally.

Most of these actions give you some visual feedback on the screen, showing a rectangle or some arrows to give you a clue what you're doing. If you find that distracting, you can turn it off using the **Plot|Show Navigation Graphics** ( ) menu item.



Navigation help panel at the bottom of plot windows

In each plot window there is a row at the bottom of the window giving hints on the currently available navigation actions. The little icons are each supposed to represent either a *click* ( ) or a *drag* ( ) with one of the three buttons pressed, or a wheel spin ( ), and are followed by a short description of what it will achieve. Note these hints change according to where the mouse is currently positioned on the screen. Moving the button over this panel will give you some help on the help. Clicking the ✕ button will make the line disappear, and you can bring it back with the **Window|Show Navigation Help** ( ) menu item. The ? button will bring up a help window specific to the navigation in this window.

If you are using a mouse with fewer than three buttons or no wheel, the following **subsitute gestures** can usually be used:

Left button
If you only have one button, this is it.

Right button
If you only have one button, you can use it with the CTRL key.

Middle button
If you only have one button, you can use it with the SHIFT key. If you have two buttons, sometimes pressing them both at once will work.

Wheel
Often a wheel action can be simulated on a trackpad by moving two fingers together up or down.

A.4.2.2 Table Data Layer Controls

Most, but not all, of the available layer controls (Appendix A.4.4) are concerned with plotting table data in some way or other. In general, these allow you to set up a way to generate a plot layer (some generated graphics) from some selected columns of a given table with a chosen style.

However, if you have more than one Row Subset for a given table, a single control can be used to generate several layers, one for each subset. In most cases it makes sense to have the graphics generated by the different subsets configured similarly, but still visually distinguishable.

These table layer controls generally have three tabs: **Position**, **Subsets** and **Form**. The **Position** tab is for specifying coordinates, and depends on the specific type of the plot and the layer control. The **Subsets** and **Form** tabs control which row subsets are plotted and how they are displayed, and a description of how they are arranged follows below.

Position Tab

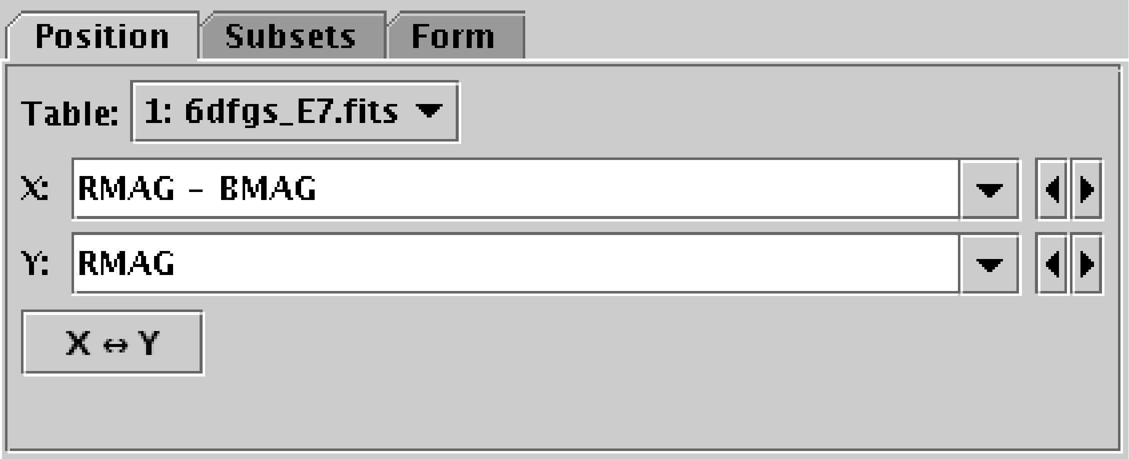
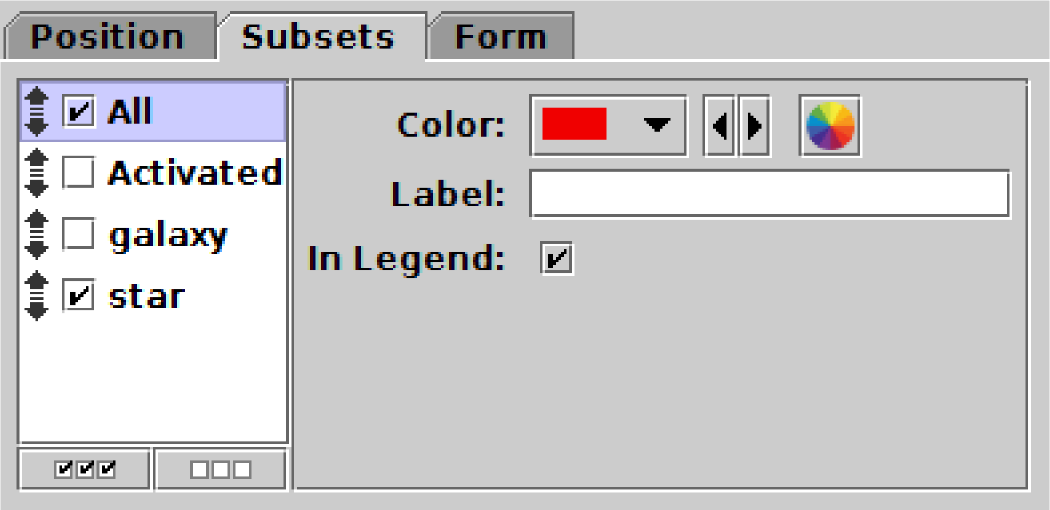


Table: 1: 6dfgs_E7.fits
X: RMAG - BMAG
Y: RMAG

Position tab of a table data layer control

The **Position** tab lets you specify a table and a basic set of coordinates for the positions to plot. The details of how the data point at each of those positions is represented, including in some cases more coordinate values (e.g. for error bar sizes etc), are specified in the other tabs (usually **Form**).

Subsets Tab



☑ All
☐ Activated
☐ galaxy
☑ star

Color: 🟥
Label:
In Legend: ☑

Subsets tab of a table data layer control

The **Subsets** tab lists the defined Row Subsets for the table you have selected in the **Position** tab (see the Subsets window). The subsets **All** and **Activated** are always present; others may or may not be depending on whether any have been defined. Note that actions you take in the plot (for instance selecting a new subset by region) can result in new entries being added to this list.

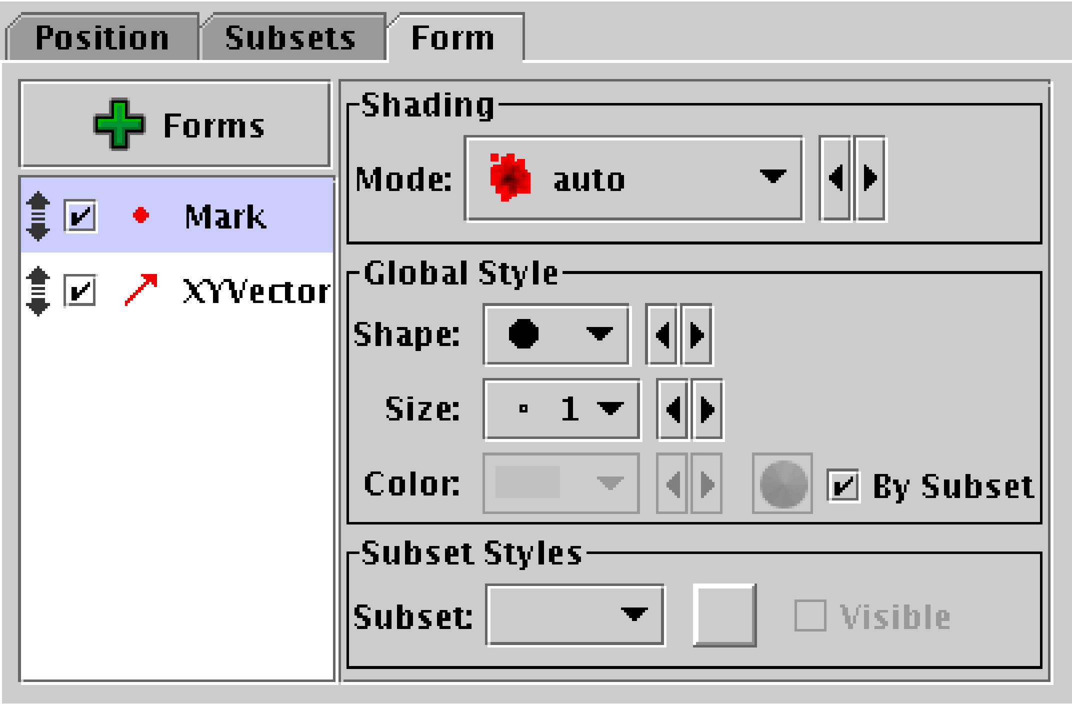
The list on the left names the subsets with an activation checkbox and a grab handle; the panel on the right gives the detail for the currently selected subset. Select a subset to see/change its detail by clicking on it.

For each subset you can select:

- Whether it is plotted (using the activation checkbox)
- The plotting sequence (by dragging it up and down using its grab handle)
- The default plotting colour
- The label with which it is annotated in the legend (if visible)
- Whether it appears in the legend at all

Although you can select plotting colours in the **Form** tab as well, it's generally better to do it here since this changes the colour of all the forms plotted for a given subset, rather than one form at a time. The little **Show/Hide All** (☑☑☑/□□□) buttons at the bottom of the list can be used as a convenience to make all the subsets visible or invisible at once (which might be useful if you have lots of subsets).

Form Tab



☑ Mark
☑ XYVector

Mode: auto

Shape: ●

Size: 1

Color:

☑ By Subset

Subset:

□ Visible

Form tab of a table data layer control

The **Form** tab lets you control how the data coordinates specified in the **Position** tab will be used to generate graphics on the screen. On the left is a stack of different Form types that will each be plotted; a default item **Mark** is usually included to start with, but you can add more from the ✚ **Forms** menu button. Each form in this stack can be turned on or off individually by clicking its activation checkbox, dragged up and down using its grab handle, or deleted using the 🗑 **Remove Selected Form** item from the Forms menu.

When you select a form by clicking on it in the stack, a configuration sub-panel will appear on the right hand side of this panel. This allows you to select style information to determine the details of how the plot will look for each layer, and sometimes also reports information calculated by the plot. The details differ greatly between forms, but they generally contain two sub-panels for defining the style details, **Global Style** and **Subset Styles**:

Global Style
Controls the style details for the chosen form, for instance marker shape and size. Options here affect all subsets, though by default the colour is taken from the **Subsets** tab. If you want to set the colour the same for all subsets, uncheck the **By Subset** checkbox which will activate the **Colour** selector.

Subset Styles
If you want to have different subsets represented with different styles, for instance different shapes for different subsets, you can select a subset here and alter style details for that subset only, overriding the Global settings above. The **Visible** checkbox indicates and controls whether the subset selected for specific configuration is currently visible in the plot.

A.4.2.3 Defining Subsets Graphically

The plot windows can be used to define new Row Subsets by indicating a screen region containing the points of interest. Depending on the type of plot, one or more of the following options will be available as toolbar/menu items:

- Subset From Visible
- Algebraic Subset From Visible
- Draw Blob Subset
- Draw Algebraic Subset

In all these cases, when you have graphically specified a region, a dialogue box will pop up to ask you the name of the corresponding subset(s). As described in Section 3.1.1, it's a good idea to use a name which is just composed of letters, numbers and underscores. You can optionally select a subset name which has been used before from the list, which will overwrite the former contents of that subset. When you enter a name and hit the **OK** button, the new subset will be created and added to the list shown in the Subset Window, and the points in it will be shown straight away on the plot using a new symbol. As usual, you can toggle whether the points in this subset are displayed using the **Subsets** tab in the table layer control.

The different options for making these graphical selections are described in the following subsections.

A.4.2.3.1 Subset from Visible

The **Subset From Visible** ( ) toolbar button creates a new subset containing all of the points that are visible on the current plot. When you click the button, you will be asked for a name for the new subset.

This option can be useful if you have panned, zoomed, set the axes manually, or otherwise navigated the plot so that only the points currently visible are the ones you are interested in. However, it is only suitable if the group you are interested in corresponds to a rectangular region in the plotting space.

A.4.2.3.2 Algebraic Subset From Visible

The **Algebraic Subset From Visible** () action (available in the **Subsets** menu of the Plane and Cube plot windows) creates subsets containing all of the points that are visible on the current plot. The subset content of this is the same as for the Subset From Visible action, but this action defines the subset as an algebraic expression based on the axis limits rather than just marking the points that can currently be seen. This can be more useful, for instance if you want to use the subset definition in some other context. However, it is only available for some plot types (Plane and Cube), since in other cases the rectangular visibility region does not correspond to a straightforward algebraic expression.

When you hit this button, the Multi Algebraic Subset Window will be displayed, showing the algebraic function(s) corresponding to the currently visible region, and offering to create a new subset (or, if there are multiple datasets plotted, several new subsets) from it.

A.4.2.3.3 Draw Blob Subset

The **Draw Blob Subset** () button allows you to draw a freehand region or regions on the plot containing the points you are interested in.

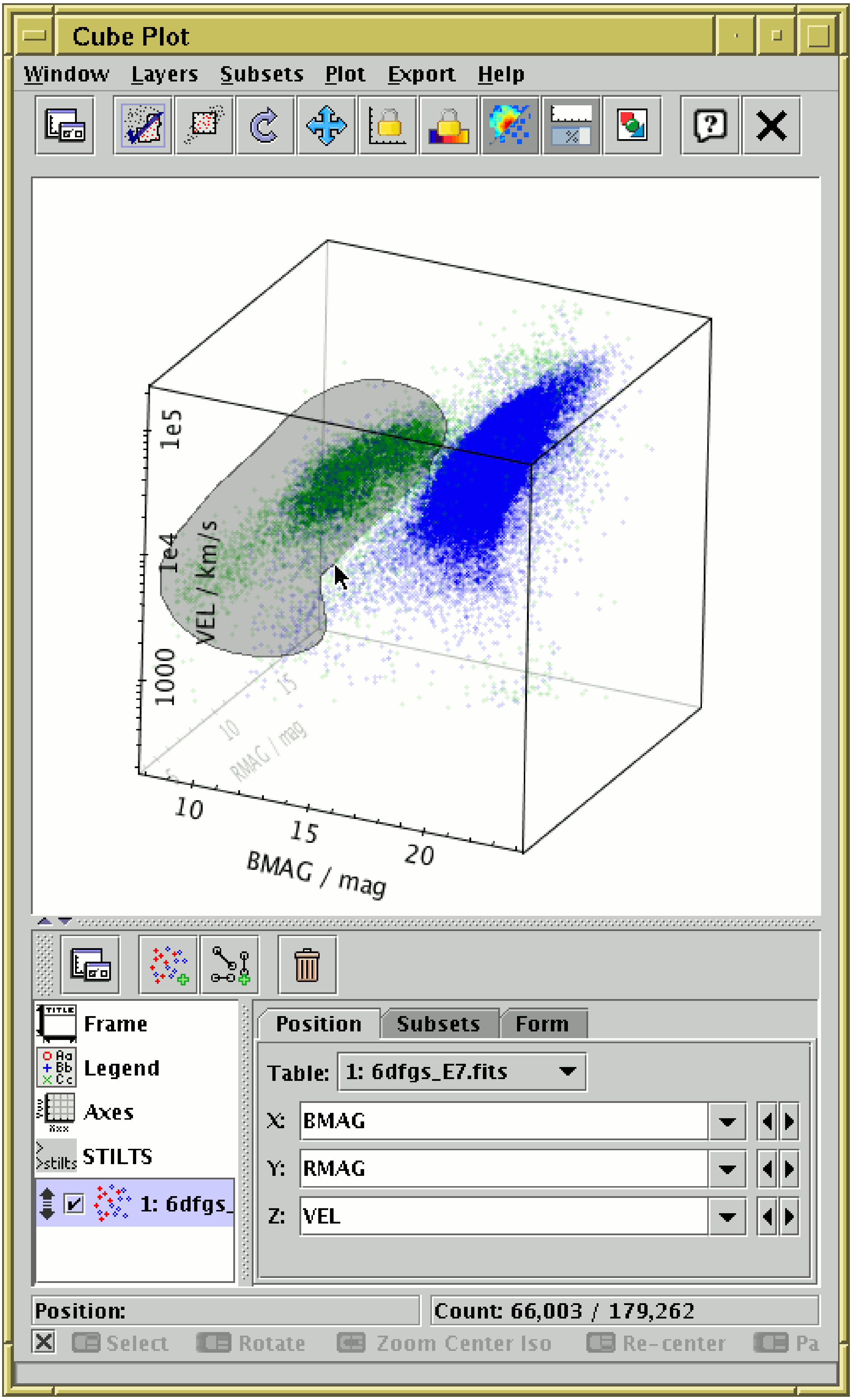


Table: 1: 6dfgs_E7.fits

X: BMAG

Y: RMAG

Z: VEL

Position:

Count: 66,003 / 179,262

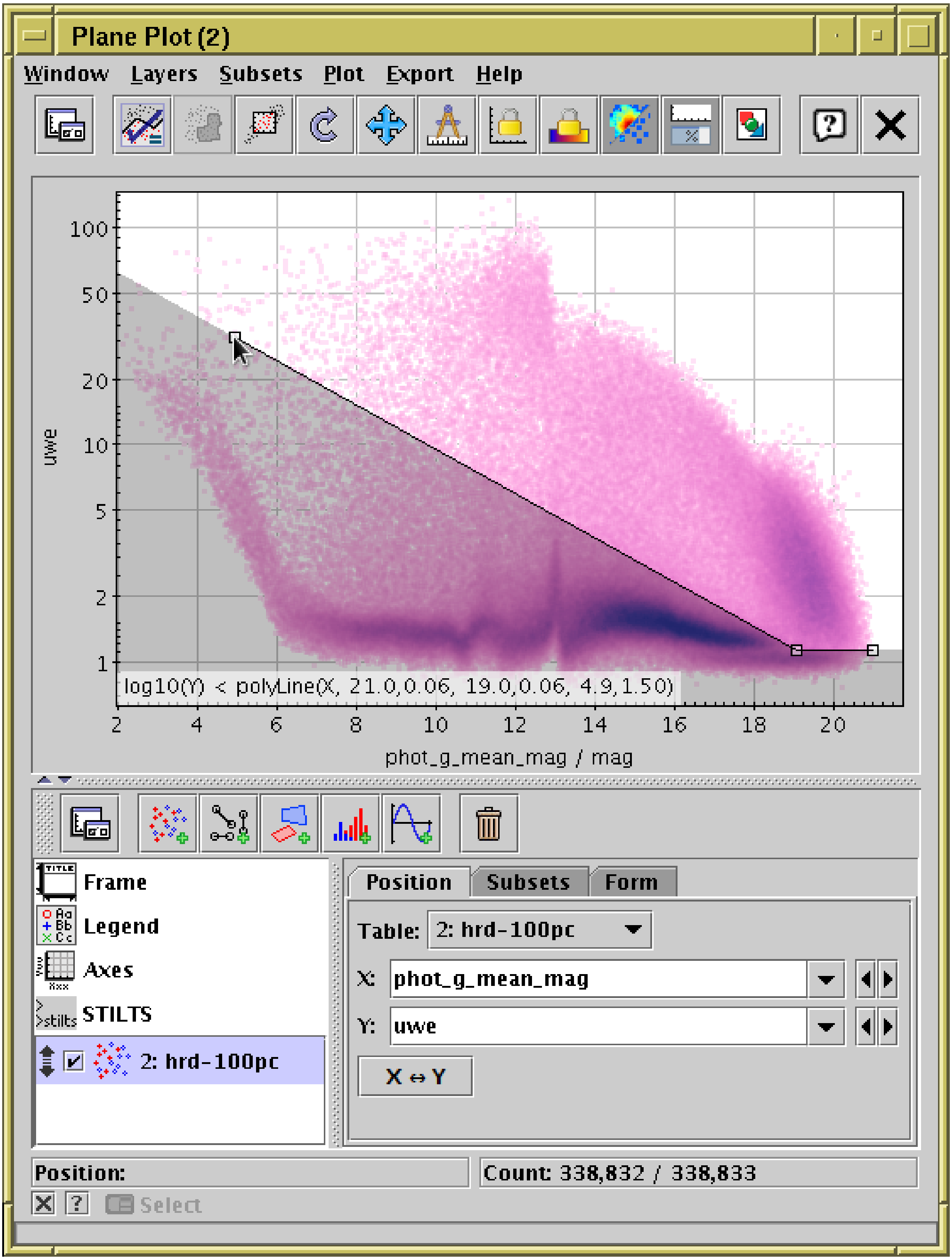
Defining a subset by blob drawing

When you click the toolbar button it will appear with a checkmark over it () to indicate that you are in blob-drawing mode. You can then drag the mouse around on the plot (keep the left mouse button down while you move) to enclose the points that you're interested in. As you do so, a translucent grey blob will be left behind - anything inside the blob will end up in the subset. You can draw one or many blobs, which may be overlapping or not. If you make a mistake while drawing a sequence of blobs, you can right-click (or Ctrl-click) and the most recently added blob will disappear. When you've finished drawing your blob(s) click the button again, and you will be asked for a name for the subset.

A.4.2.3.4 Draw Algebraic Subset

The **Draw Algebraic Subset** () action allows you to select points in a region of the plot by clicking on points of your choice to mark out a shape. Different shapes such as polygons and circles are available, depending on the plot type. When complete, subsets will be defined with an algebraic expression which you can see and edit. This can be particularly useful (and a better option than the blob) if you want to refer to the subset outside of the context of the current session, for instance in a STILTS command or a published paper.

This action is currently only available in the Plane, Sky and Corner plot windows.



log10(Y) < polyLine(X, 21.0,0.06, 19.0,0.06, 4.9,1.50)

☑ 2: hrd-100pc

Table: 2: hrd-100pc

X: phot_g_mean_mag

Y: uwe

Position:

Count: 338,832 / 338,833

Defining a subset by algebraic drawing. This shows use of mode *Below* in the Plane plot.

When you use this action to define a plot region, it operates in one of a number of *inclusion modes*, depending on the plot type. In all cases, you click on one or more points to define the boundaries of the region. The available modes are described at the end of this section.

Operation is as follows:

1. To start marking out a shape, hit the button in the toolbar, and a popup window will first ask you which inclusion mode you want to use. Alternatively, you can use one of the mode-specific sub-menu items in the **Subsets|Draw Algebraic Subset** menu to choose a mode without the extra popup.
2. Once in drawing mode, the toolbar button will appear with a checkmark over it (), and a little square marker will appear near the mouse pointer as long as it's over a suitable part of the plot. You can then click on the plotting area to mark the points, and the area thus defined (according to the mode you have chosen) will be shaded in grey. Each point you have clicked on to define the area is marked with the little square marker. The algebraic form of the expression for the points entered so far will be displayed at the bottom of the screen. A right-click (or Ctrl-click) will remove the most recently-added point.
3. When you've finished adding points, click on the button again. This will pop up the Multi Algebraic Subset Window, which displays the algebraic function corresponding to the region you have outlined, and offering to create a new subset (or, if there are multiple datasets plotted, several new subsets) from it.

The generated expression tries to be as compact and comprehensible as possible. Precision of the indicated points is determined from the pixel resolution of the plot, so literal numbers are not more unwieldy than they have to be.

The available inclusion modes depend on the plot type, as follows:

Plane Plot

- **Polygon**: Select at least three points to define a closed polygon. The selected region is inside it. The expression generated uses the `isInside` function.
- **Box**: Defines a rectangle aligned with the axes from two opposite corners. The expression is just of the form `x>x0 && x<x1 && y>y0 && y<y1`.
- **Circle**: Defines a circle in data space. The first point selected defines the circle center, and the next point is on the circumference. Note that in case of one or both axes logarithmic, the shape in graphics space may not be circular. The expression is of the form `hypot(x-x0, y-y0)<r`, using the `hypot` function (`hypot(x,y) = sqrt(x*x+y*y)`).
- **Aligned Ellipse**: Defines an ellipse in graphics space aligned with the plot axes. The first point selected defines the ellipse center, and the next point is on the boundary. If later points are selected, they reset the radii. The expression generated is of the form `square((x-x0)/a) + square((y-y0)/b)<1`, using the `square` function (`square(x)=x*x`).
- **Rotated Ellipse**: Defines an ellipse in graphics coordinates with arbitrary alignment. The first point selected defines the ellipse center, the second point is the end of the primary radius, and the third point defines the extent of the secondary radius. If later points are selected they reset the secondary radius. The expression generated is of the form `square(a(x-x0)+b(y-y0))+square(c(x-x0)+d(y-y0))<1`, using the `square` function (`square(x)=x*x`).
- **Below**, **Above**, **Left**, **Right**: The points define a jagged line representing a continuous function of *x* (or *y*) which has, in general, a discontinuous derivative. If just one point is selected, all the points on the corresponding side of that single point are included. Line segments at either end of the line are considered to continue to infinity - it's easy to understand how this works by trying it out (or see the figure above). If only one point is selected, the expression generated is of the form `y<y0`, and for two points it has the form `y<m*x+c`. If there are more points, the special `polyLine` function is used.

Sky Plot

- **Circle**: The first point selected defines the center of a small circle on the sphere (cone), and the next point is on its radius. If later points are selected, they reset the radius. The expression generated uses the `skyDistance` function.
- **Ellipse**: Defines an ellipse with arbitrary alignment. The first point selected defines the ellipse center, the second point is the end of the primary radius, and the third point defines the extent of the secondary radius. If later points are selected they reset the secondary radius. The expression generated uses the `inSkyEllipse` function.
- **Polygon**: Select at least three points to define a closed polygon. The edges of the polygon are the minor arcs of great circles (geodesics on the sphere). The expression generated uses the `inSkyPolygon` function.

A.4.2.4 Distance Measurement

The **Measure Distance** (📐) action is available from some plot windows (Plane, Histogram, Sky, Corner and Time plots). This allows you to use the mouse to measure the distance between two points on the plot. To use it, first click the **Measure Distance** toolbar button or menu item, and then press the mouse button on a point in the plot from which you wish to measure, and drag the mouse, holding down the button, to the point you wish to measure to. As you drag the mouse, the distance(s) between the two points are displayed and updated interactively. Once you release the mouse button, measurement mode is exited, and you need to invoke the action again to make another measurement.

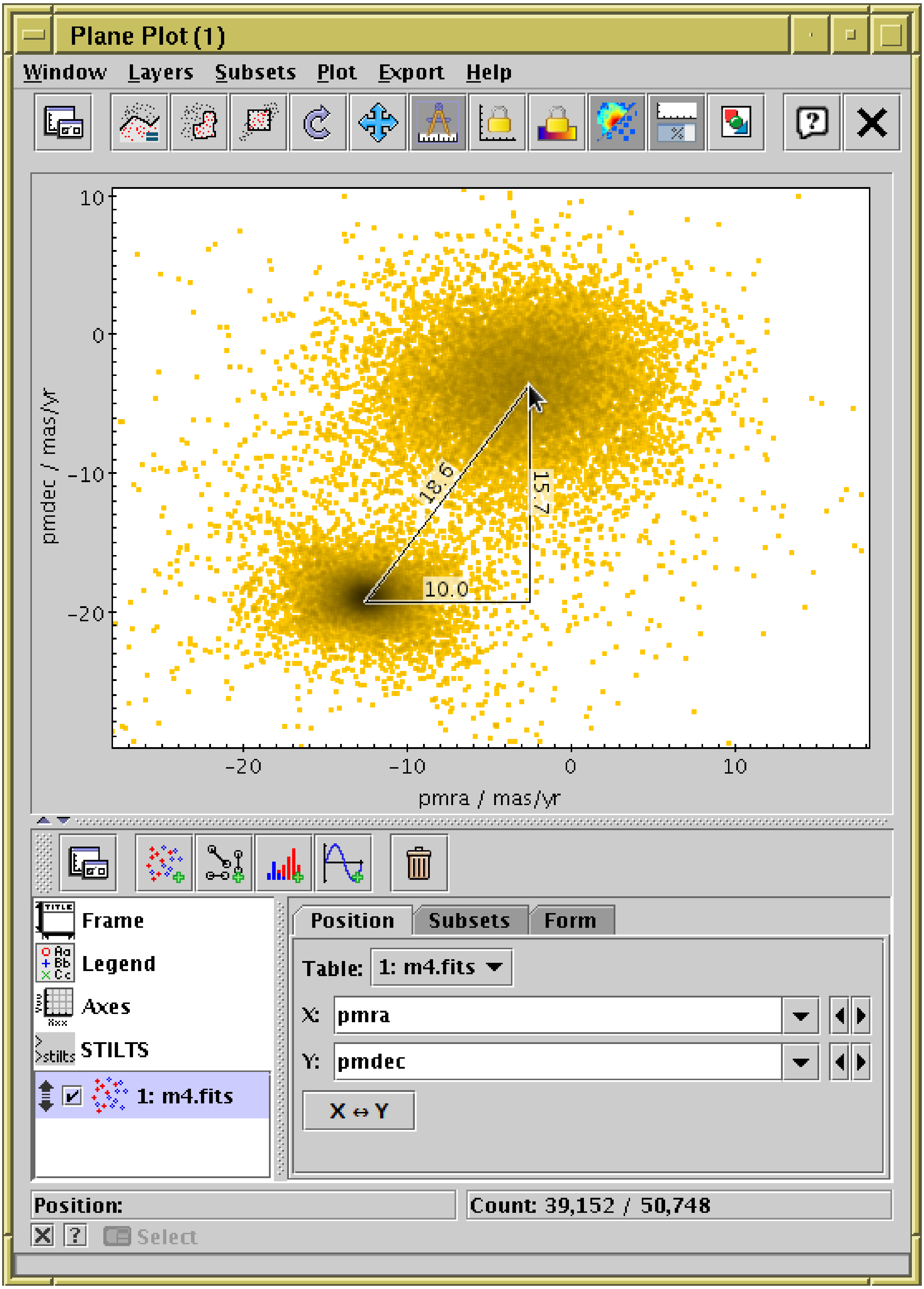


Table: 1: m4.fits

X: pmra

Y: pmdec

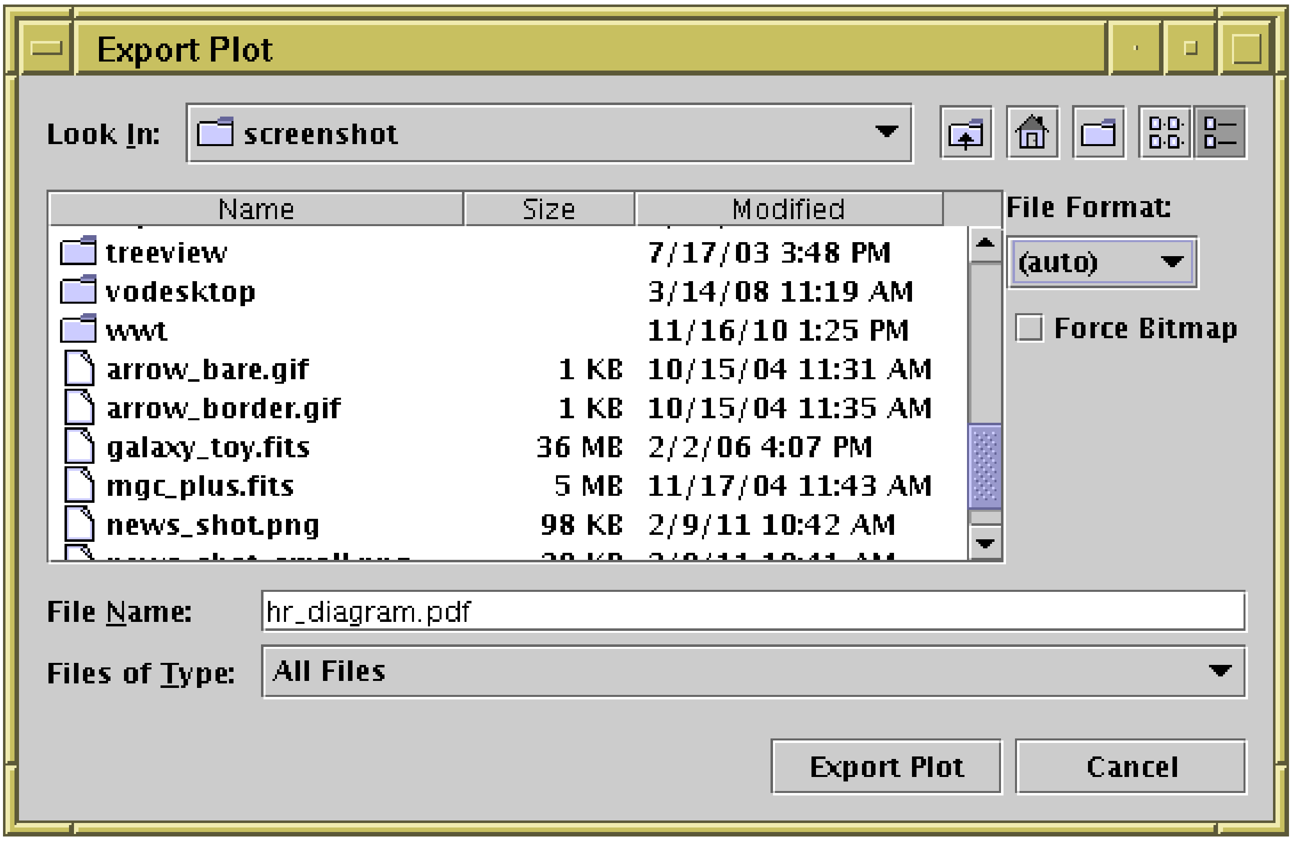
Position:

Count: 39,152 / 50,748

Measuring distance in the Plane Plot

The details of what distances are displayed depends on the plot window in question, according to what metrics are defined on the plot space. For the Sky plot, the shorter great circle arc is drawn on the plot between the two points, and the distance is labelled in degrees, minutes or seconds. For the Histogram and Time plots, vertical and horizontal components are drawn and labelled in terms of the distances along the corresponding axis. For the Plane plot, both components and the direct distance are measured.

A.4.2.5 Plot Export Window



Look In: screenshot

Name	Size	Modified
treeview		7/17/03 3:48 PM
vodesktop		3/14/08 11:19 AM
wwt		11/16/10 1:25 PM
arrow_bare.gif	1 KB	10/15/04 11:31 AM
arrow_border.gif	1 KB	10/15/04 11:35 AM
galaxy_toy.fits	36 MB	2/2/06 4:07 PM
mgc_plus.fits	5 MB	11/17/04 11:43 AM
news_shot.png	98 KB	2/9/11 10:42 AM

File Format: (auto)

□ Force Bitmap

File Name: hr_diagram.pdf

Files of Type: All Files

Plot export window

The **Plot Export Window** can be reached with the **Export plot to file** ( ) toolbar button in any of the plot windows.

You can select a file in the usual way, and save the plot in one of the following graphics formats:

png
PNG bitmap. The background is opaque.

png-transp
PNG bitmap with a transparent background. Background pixels that fall outside the plot surface itself (for instance outside the axes for a Plane plot or outside the celestial sphere for a Sky plot) are transparent.

gif
GIF bitmap; note the number of colours is limited to 256.

jpeg
JPEG bitmap; note that this is a lossy format, better suited to photographs than plots, and colours will be blurred.

pdf
Portable Document Format; in most cases this vector format gives pretty good output, in particular text will be rendered properly.

svg
Scalable Vector Graphics; this XML-based vector format mostly works quite well, but can result in OutOfMemoryErrors for large output files.

eps
Encapsulated PostScript; PostScript cannot handle transparency, which means that in some cases the output will come out wrong. PostScript files can also be very large if there are many data points.

eps-gzip
Just like **eps**, but the output is gzipped before output.

There are two additional controls on the right hand side of this window:

File Format
Selects the output file format as above. The default setting is **(auto)**, which guesses what format you want to use from the filename, and which usually does the right thing.

Force Bitmap
This option only has an effect for vector graphics formats (PDF, SVG and PostScript). If selected, it draws the data contents of the plot as a pixel map and embeds that into the output file rather than plotting each point in the output. This may make the output less beautiful (round markers will no longer be perfectly round), but it may result in a much smaller file if there are very many data points. Plot annotations such as axis labels will not be affected - they are still drawn as vector text. Note that in some cases (e.g. use of the auto, density or weighted shading modes) this kind of pixellisation will happen in any case.

Exporting to the pixel-based formats (GIF, JPEG, PNG) is fairly straightforward: each pixel on the screen appears as one pixel in the output file. PNG is generally recommended. GIF works well in most cases, but if there are more than 255 colours some of the colour resolution will be lost. JPEG can preserve a wide range of colours, but does not support transparency and is lossy, so close inspection of image features will reveal blurring.

When exporting to Portable Document Format (PDF), Scalable Vector Graphics (SVG) or Encapsulated PostScript (EPS), which are vector graphics formats, there are a few things to consider:

Positional Quantisation
Some of the shading modes (Density, Weighted, Auto) are inherently pixellated, and others (Flat, Aux) are not. In the former case you will see pixel boundaries for the plotted points rather than nice rounded edges at high magnifications (though text and axes will always be plotted nicely). In both cases, at present the *positional* resolution is the same as it would be on the screen, so if you have a 400-pixel high plot for instance, there are only 400 possible Y coordinates at which a marker can be plotted, which in general is not obvious by looking at the output plot. In future versions the positional resolution of non-pixellated modes may be improved. In either case, increasing the size of the plot on the screen by resizing the window before performing an export to PDF, SVG or EPS will reduce the effect of the positional quantisation. Note it will also have the effect of making the text labels proportionally smaller to the graphics, so you may want to increase the font size too.

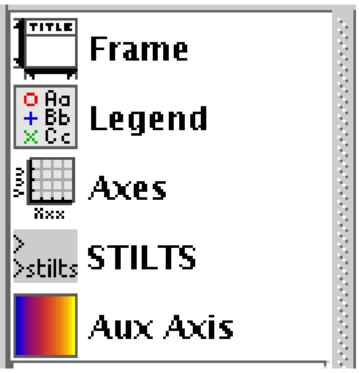
Transparency
For technical reasons transparent markers cannot easily be rendered when a plot is exported to PostScript. In some cases the plot is done using a bitmap in the PostScript output to permit transparency and in some cases the points are just plotted opaque. PDF does a bit better, but the compositing of transparent shapes is sometimes a bit different on the screen and rendered to a PDF. It's a good idea to check the output of screen exports by looking at the produced file - if it doesn't look like it should do, setting the **Force Bitmap** option will probably make sure it

does, though this will also pixellate the plotted symbols. There is more discussion of this point in the subsections for the various shading modes.

File Size

In some cases (2D and 3D scatter plots with many thousands of points or more) output EPS files can get extremely large; the size scales with the number of points drawn, currently with a factor of a few hundred bytes per point. In some cases you can work round this by plotting some points as transparent so that the plot is rendered as a bitmap (see the discussion of transparency above) which scales as the number of pixels rather than the number of points. The Gzipped EPS format helps somewhat (though can be slow); PDF output is better still. Even PDF files may be unmanageably large for very many points however.

A.4.3 Fixed Controls



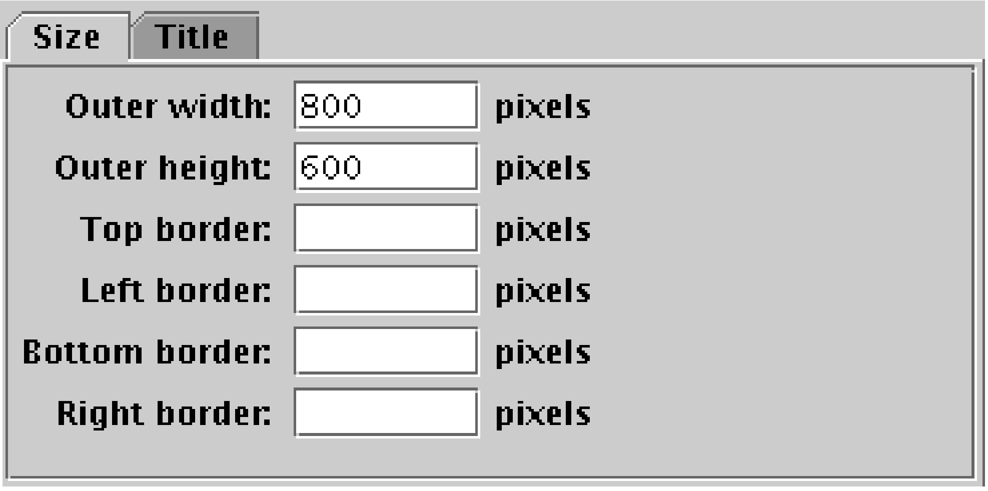
Fixed controls in control panel stack

Fixed controls are those controls that appear at the top of the stack in the control panel, and do not have a checkbox or a drag handle. They are used to configure or monitor the overall appearance of the plot, independent of any particular plot layer or data set. Those common to all plot types are described in the following subsections, though some plot types may also have their own.

A.4.3.1 Frame Control

The **Frame** control () controls the graphical area on which the plot graphic is drawn. It contains two tabs, **Size** and **Title**.

Size Tab



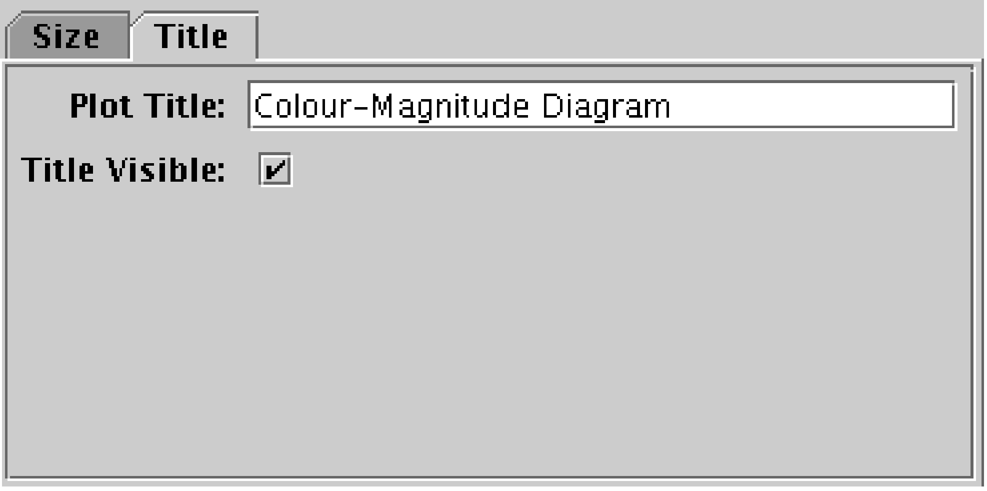
Outer width:	800	pixels
Outer height:	600	pixels
Top border:		pixels
Left border:		pixels
Bottom border:		pixels
Right border:		pixels

Frame control Size tab

The **Size** tab allows you to set the geometry of the generated plot in pixels. By default, the plot is simply resized to fit the current size and shape of the window. However, if you fill in the **Outer width** and/or **Outer height** fields, the size will instead be fixed to the given dimension in pixels. Note if you pick a large value, you may need to manually resize the window to see all of the plot. Similarly, the **border** fields fix the number of pixels surrounding the data region of the plot; if these are left blank, as by default, these borders are sized to accommodate the things that have to fit in them (such as axis annotations and the Aux axis colour ramp).

The main use for this tab is to fix the size and alignment of the generated images if you want to export a series of similar plots.

Title Tab



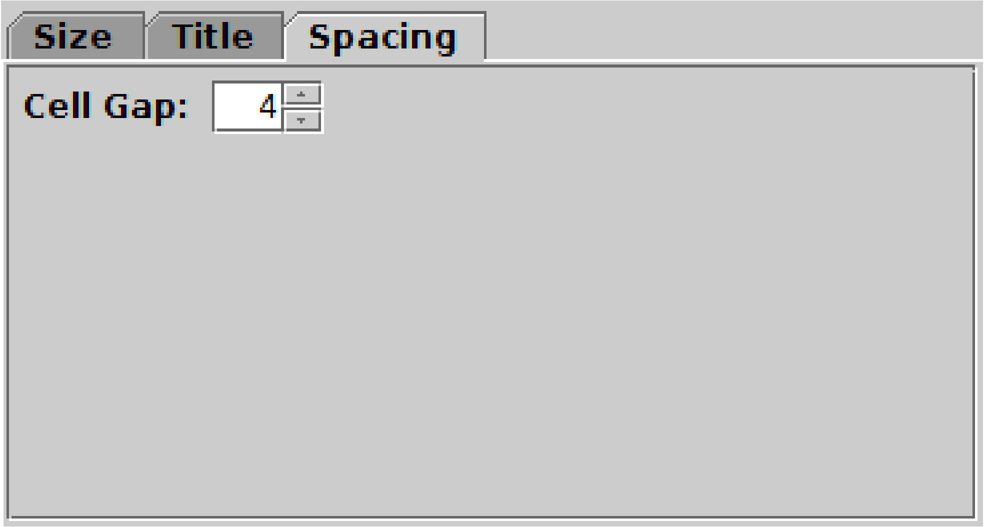
Plot Title: Colour-Magnitude Diagram

Title Visible: ☑

Frame control Title tab

The **Title** tab allows you to give a title for the plot. If text is filled in the **Plot Title** field, and if the **Title Visible** checkbox is checked, then the given text will appear at the top of the plot.

Spacing Tab



Cell Gap: 4

Frame control Spacing tab

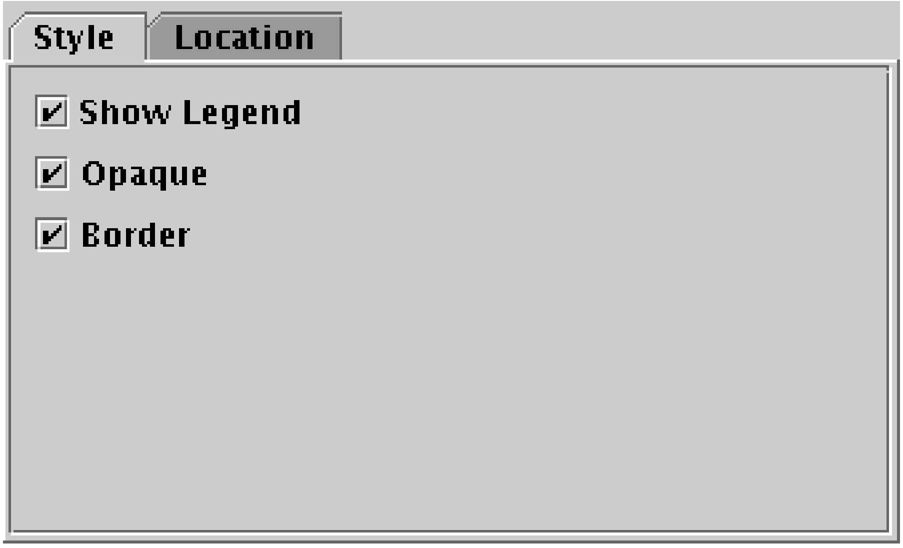
For some plot types only, this tab provides options to control the spacing between elements of the plot. For the Time Plot it contains the **Cell Gap** option which controls the number of blank pixels between individual panels in the stack of plots. For other plot types this configuration is not relevant, or is available from the **Axes** control.

A.4.3.2 Legend Control

The **Legend** control (○Aa +Bb ×Cc) is always visible in the plot Control Panel. It allows you to configure whether and how the plot legend appears.

This control has two tabs, **Style** and **Location**.

Style Tab



☑ Show Legend

☑ Opaque

☑ Border

Legend control Style tab

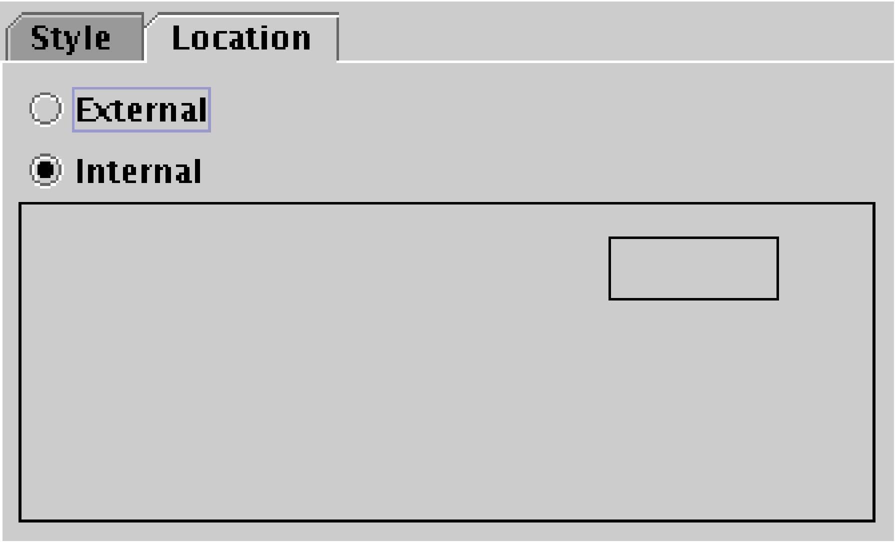
The **Style** tab configures the appearance of the legend panel, and has the folowing options:

Show Legend
Whether the legend appears in the plot. This is set automatically: if only one data set is present no legend is shown, but once multiple datasets are present the legend is visible. But the option can be overridden manually using this control.

Opaque
If true, the legend has an opaque white background. If false, it is transparent, and any plot data underneath it is visible.

Border
If true, a black line border is shown around the legend. If false, there is no border.

Location Tab



○ External

● Internal

Legend control Location tab

The **Location** tab configures where on the plot the legend is drawn (if present). There are two options, **External** and **Internal**. If Internal is chosen, then a control is activated showing a small box inside a large box. Drag the small box around with the mouse to change the position of the legend inside the plot bounds.

Note the font used in the legend is currently controlled by the font from the **Axes** selector.

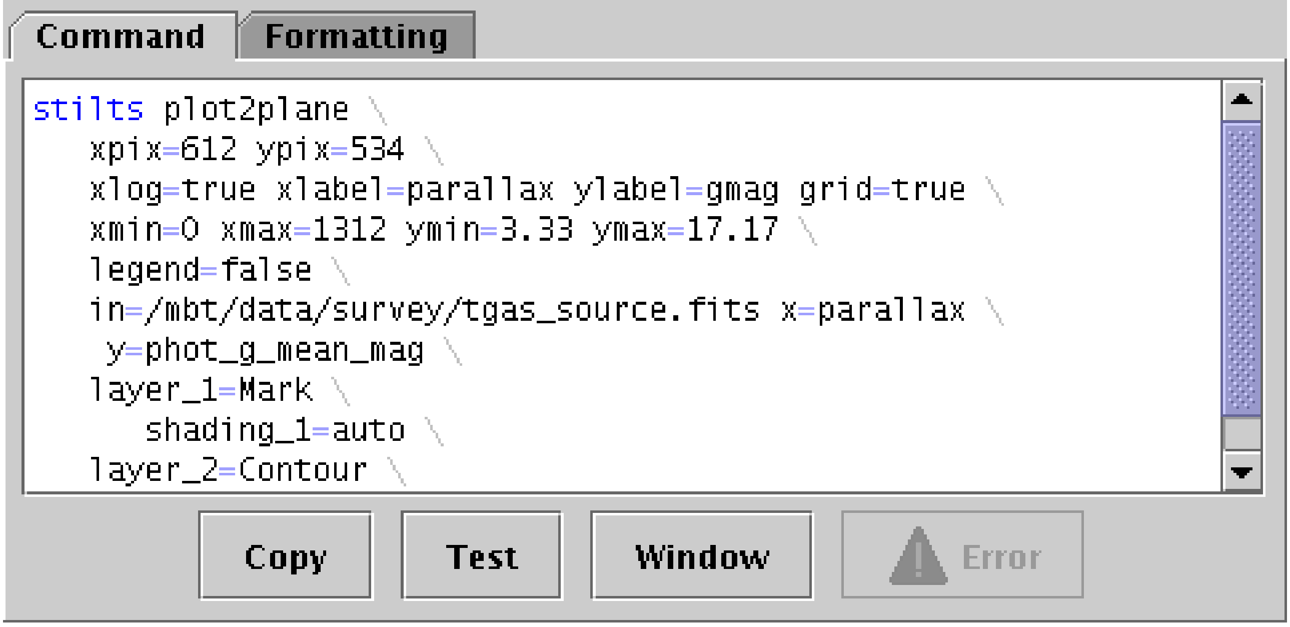
A.4.3.3 Axes Control

The **Axes** control () has a number of tabs specific to the particular plot type. These tabs allow you to configure things like the axis ranges, linear/log axis scaling, axis labelling, grid drawing, details of the navigation controls and so on. The different plot types have axis controls that differ significantly from each other, so the specific axis controls are described separately along with each plot type:

- Histogram axes control (Appendix A.4.8.2)
- Plane plot axes control (Appendix A.4.9.2)
- Sky plot axes control (Appendix A.4.10.2)
- Cube plot axes control (Appendix A.4.11.2)
- Sphere plot axes control (Appendix A.4.12.2)
- Corner plot axes control (Appendix A.4.13.2)
- Time plot axes control (Appendix A.4.14.3)

A.4.3.4 Stilts Control

The **STILTS** control (>stilts) displays the command you would have to issue to the STILTS package to reproduce the currently visible plot. The text is continuously updated to match the currently selected options, layers and navigation status.



```
stilts plot2plane \
   xpix=612 ypix=534 \
   xlog=true xlabel=parallax ylabel=gmag grid=true \
   xmin=0 xmax=1312 ymin=3.33 ymax=17.17 \
   legend=false \
   in=/mbt/data/survey/tgas_source.fits x=parallax \
    y=phot_g_mean_mag \
   layer_1=Mark \
      shading_1=auto \
   layer_2=Contour \
```

STILTS control Command tab

Unlike the other fixed controls, this one does not provide any options to change the current plot, and if you just want to use TOPCAT interactively you don't need to use it at all. But if you want to be able to regenerate the current plot from the command line or a script without setting it up again from a GUI, for instance to generate publishable figures in a reproducible way, you may find it useful.

STILTS is a command-line interface to all the functionality that TOPCAT provides from a Graphical User Interface. Any plot that can be displayed in TOPCAT can also be generated by providing the right parameters to one of the STILTS plotting commands. The STILTS user document provides a full tutorial introduction for these commands, as well as detailed documentation for each of the plotting commands (`plot2plane`, `plot2sky`, ...) and many examples. However, there is a large number of parameters to configure, and the command line to reproduce a complex plot can be quite lengthy, so this control helps you to set up such plots by constructing the command line corresponding to what you can currently see. The text displayed in this panel can either be copied directly to reproduce a plot you have set up interactively, or it can be used as a basis for later adjustments to some of the parameters.

TOPCAT itself contains the STILTS application, so you don't need to install any additional software to use it. You can run it by adding the "`-stilts`" flag to a topcat command, or on a Unix-like OS use the `stilts` script. If you don't already have it, you can download `stilts` to the directory containing your TOPCAT jar file; see also SUN/256. The `Invocation` formatting options described below can also help.

You can use this facility with very little understanding of the details of STILTS. You just need to copy the text (using the **Copy** button or however that's usually done on your OS) and paste it onto a system command line or into a script. Executing the resulting command should then pop up the current plot in a new window outside of TOPCAT. If it doesn't work, changing some of the **Formatting** options described below (e.g. setting *Invocation* to *Class-path*) may help. To write the result to a graphics file instead of displaying it in a window, just add a parameter like `out=plot.png` to the end of the line. The available graphics output formats are listed in Appendix A.4.2.5.

To understand the generated command and get a better idea of how to tweak the parameters to adjust the plot, you can consult the STILTS user document mentioned above. Just watching how the displayed command changes as you interact with the plot is another way to learn what does what.

This display is available both as a Fixed Control and as a separate window. Both do the same thing, but the window may be convenient if you want to see the way the command text changes as you interact with the GUI in other parts of the fixed or layer controls. To pop up or hide the separate window display, you can either use the **Window** button at the bottom of the fixed control, or the **Export|STILTS Command Window** menu item.

Some actions are available from buttons or menu items:

Copy
Copies the whole command into the system clipboard, for convenience if you want to paste it into an editor or at a shell prompt. You can achieve the same thing using the usual OS-specific gestures if you prefer, e.g. highlighting all the text with the mouse.

Test
Attempts to execute the displayed command. If successful the reproduced plot will pop up in a new window, and should look the same as the one currently visible in the plot window. This can help to detect some potential problems with the displayed command line, but not all (see more discussion below).

Window
Displays a separate window containing the information in this control (or hides it if already displayed). This can be convenient to see how the generated command line changes as you interact with the GUI controls.

Error
If TOPCAT detects some problem with the syntax of the displayed command, this button is enabled and clicking it will show an error message. This can happen if you are trying to refer to a non-algebraic subset by name, or for other reasons including bugs in the code.

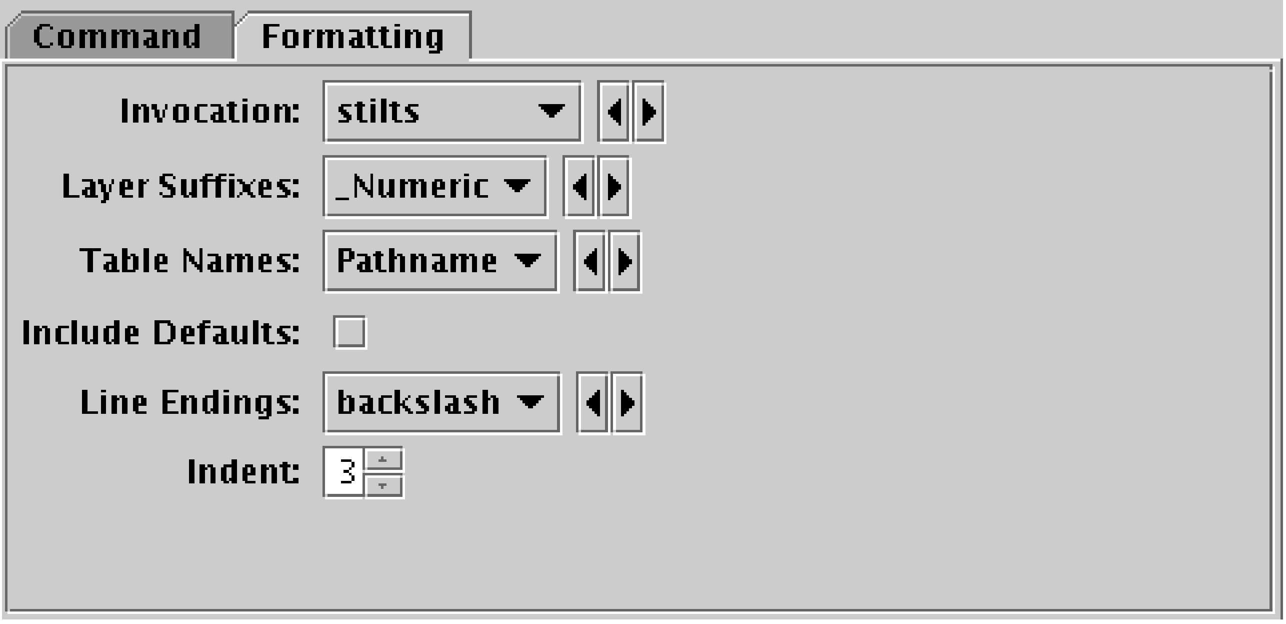
NOTE: The STILTS command displayed in this panel is **not guaranteed** to work from the command line. There are a few reasons for this. The STILTS command tries to name the tables you have loaded into TOPCAT. If they have straightforward filenames this will probably work, but if a plotted table is for example the result of a match operation carried out in the current session, it will not exist on disk yet so it can't be named on the command line. Similarly, using the names of columns or non-algebraic Row Subsets created during the current session may result in command lines that won't work as written, since those values don't exist in the input files. In this case you should prepare a table corresponding to the current TOPCAT state, save that, and edit the STILTS command to use the name of that file for input (or you can reload the file and do plots using that). Subsets as such cannot be saved in this way, but you can achieve much the same thing by storing subset information in a boolean column using the **To Column** action (▦) from the Subsets Window. Note STILTS does not understand TOPCAT's saved session format. There may also be bugs in the (rather complex) command-generation code that cause the command to fail or to generate a slightly different plot. Hopefully there are not too many of these, but if you find one please report it.

Another thing that can cause trouble is quoted values in algebraic expressions, since STILTS quoting syntax does not always work well on the shell command line. If there are quoted expressions within a quoted argument it is sometimes helpful to escape the inner quotes, e.g. converting `cmd_1='select "gmag > rmag"'` to `cmd_1='select \"gmag > rmag\"'`. Another approach is to avoid unnecessary spaces, which may allow inner quotes to be omitted.

An attempt is made to flag problems in the displayed command line. Constructions that are suspected or expected to cause trouble when executing it are highlighted in a different colour. If the command line itself seems to violate the plotting command syntax, the **Error** (⚠) button will become enabled; clicking on it will display some indication of what's wrong.

However, the best way to test whether a displayed command will work is to copy and paste it onto an actual command line and try to run it. If it works, the plot will show up in a new window. To export this into a graphic file, simply add or modify the `out` parameter (e.g. `out=tc-plot.pdf`).

Formatting Tab



Invocation: stilts

Layer Suffixes: _Numeric

Table Names: Pathname

Include Defaults: □

Line Endings: backslash

Indent: 3

STILTS control Formatting tab

The **Formatting** tab gives you various options to adjust how the generated STILTS command is displayed in the Command tab. In the popup window version of this display, these options are available as items in the **Formatting** *menu* instead.

You may want to adjust these options for personal taste, or so the output works better with the command-line environment you're using. The basic format is intended to work with a Unix-like shell such as `bash`; in most cases the `name=value` settings should not be too sensitive to shell-specific syntax, but it may be useful for instance to change line ending characters.

The available formatting options are:

Invocation
Determines how the `stilts` command itself is introduced. Options are:

- **stilts**: Just the word "`stilts`", which will work if a command with that name is on the execution path.
- **topcat**: The expression "`topcat -stilts`", which will work if the `topcat` command is on the path.
- **Class-path**: A longer expression based on the location of the java class files that the currently running TOPCAT application is using. This should work as along as a `java` executable is on the path.

Layer Suffixes
Determines how the parameters associated with different plot layers are labelled. STILTS groups the parameters corresponding to a given plot layer by using a common suffix, introduced by a parameter with the form "`layer<suffix>=<layer-type>`" (see SUN/252). You can choose any form for these suffixes that does not interfere with the non-suffix parts of the other parameters, and this option allows you to choose how they are generated. Options are:

- **_Numeric**: `_1`, `_2`, ...

- **_Alpha**: `_A`, `_B`, ...
- **_alpha**: `_a`, `_b`, ...
- **Numeric**: `1`, `2`, ...
- **Alpha**: `A`, `B`, ...

Zone Suffixes
Determines how suffixes assigned to different plot zones are labelled. Currently, plot zones are an experimental feature used only in the Time Plot Window, so this option does not appear in most plot types. The options are as described for the Layer Suffix selector above.

Table Names
Determines how tables in TOPCAT are referenced in the generated command line. Options are:

- **Pathname**: Full pathname of the input table file, where available. Should be fairly robust for tables loaded from files.
- **Filename**: Tail of filename only (no directory name) of the input table file, where available. More compact than Pathname, and should work if the STILTS command is executed in the directory in which all files exist.
- **Label**: User-assigned label for the table, as shown in the Control Window.
- **TNum**: "`T`" followed by the table identifier. The identifier is the small integer shown in the table list in the Control Window.

Include Defaults
Parameters in STILTS commands usually have default values, and if the required values are equal to the defaults, those can be omitted from the command line. By default, parameters which take their default values are not displayed in this window. But if you set this checkbox true, even parameters taking their default values are displayed as well. Default-valued parameters are shown in a fainter colour than non-default ones. This may give you a better idea of the various options which are available for the current plot.

Line Endings
STILTS commands can be quite long and are usually displayed over several screen lines. This control configures how lines breaks are displayed. Options are:

- **plain**: just a new line character terminates lines
- **backslash**: a backslash ("`\`") is added at the end of each line; suitable for `bash` and other Unix-like shells.
- **caret**: a caret ("`^`") is added at the end of each line; suitable for Windows CMD/.bat scripts.
- **backtick**: a backtick ("`` ` ``") is added at the end of each line; suitable for Windows PowerShell
- **none**: the command is displayed as one long line

Indent
Configures the amount of whitespace by which groups of related lines are indented.

A.4.3.5 Aux Axis Control

The **Aux Axis** control ( ) is only visible when at least one layer is using the shared colour map.

The following plot layers do this:

- Aux colour mode (Appendix A.4.6.6)
- Weighted colour mode (Appendix A.4.6.7)
- Grid form (Appendix A.4.5.18)

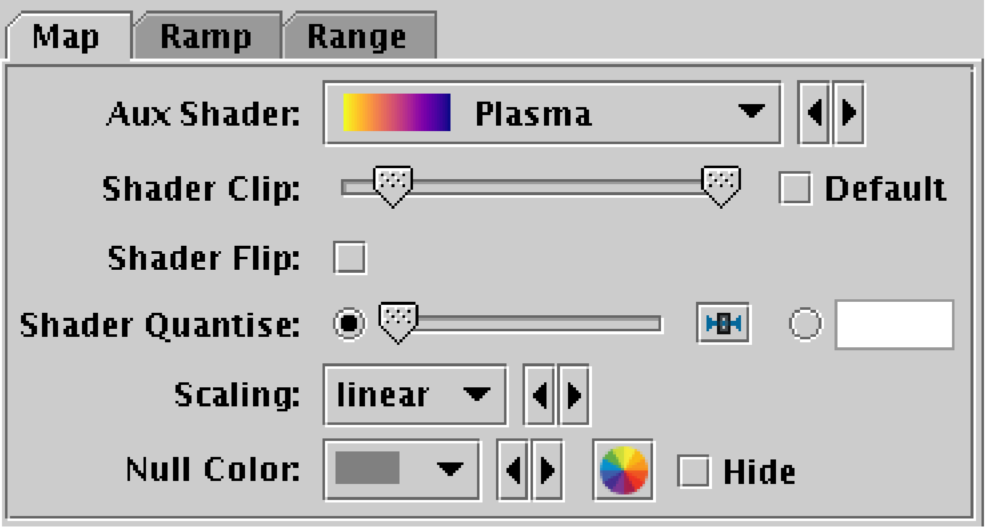
- Sky Density form (Appendix A.4.5.19)
- Healpix control (Appendix A.4.4.5)
- Line form (Appendix A.4.5.12)
- Line3d form (Appendix A.4.5.13)
- Spectrogram control (Appendix A.4.4.8)

These all colour parts of the plot in a way that is quantitatively significant, and the Aux Axis control gives you a chance to control the details of the value-colour mapping, and to display the mapping in a colour ramp displayed beside the plot. Some other layer types (e.g. Density mode) also shade the plot according to numeric values, but use their own colour map, and don't display the colour ramp.

If no layer is using the shared colour map, then this control will not appear in the stack.

When present, this control has three tabs, **Map**, **Ramp** and **Range**, described below.

Map Tab



Aux Shader: Plasma
Shader Clip: □ Default
Shader Flip: □
Shader Quantise: ● ○
Scaling: linear
Null Color: □ Hide

Aux axis control Map tab

The **Map** tab controls the aux axis colour map. It has the following options:

Aux Shader
: Select the colour map from a list of options. The available colour maps are listed in Appendix A.4.7.

Shader Clip
: Select a sub-range of the full colour map above. If the **Default** checkbox is checked, then all or most of the colour ramp from the **Shader** control is used. If you want to configure the range of colours from the map yourself, uncheck the Default checkbox, and slide the handles in from the end of the slider to choose exactly the range you want.

  The default range is clipped at one end for colour maps that fade to white, so that all the plotted colours will be distinguishable against a white background. That is generally a good idea for scatter-type plots, but may not be so good for plots where the whole plotting surface is coloured in, like density maps or spectrograms, so in that case you might want to uncheck Default and leave the handles at the extreme ends of the slider.

Shader Flip
: Whether the aux scale should map forwards or backwards into the colour map.

Shader Quantise
: Allows the colour map to be quantised. By default, the colour map is effectively continuous. If you slide the slider to the right, or enter a value in the text field, the map will be split into a decreasing number of discrete colours. This can be used to generate a contour-like effect, and

may make it easier to trace the boundaries of regions of interest by eye.

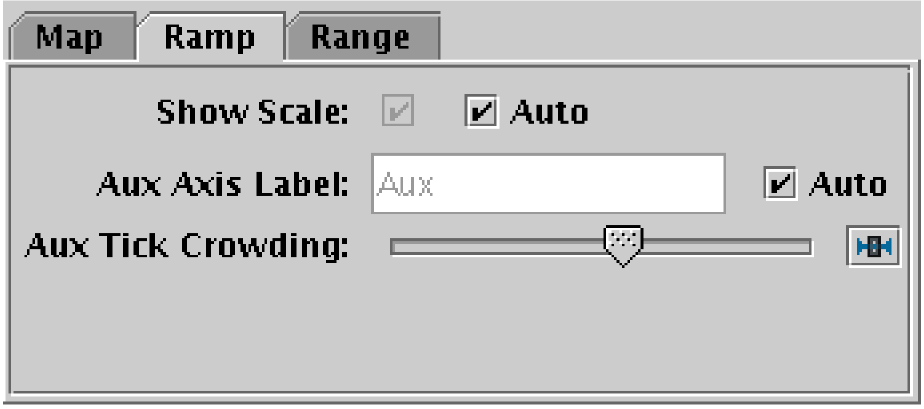
Scaling
Determines the function used to map the range of aux data values onto the colour map. The options are:

- `linear`
- `log`
- `histogram`
- `histolog`
- `square`
- `sqrt`
- `acos`
- `cos`

In all these cases, the full range of data values is displayed on the colour bar (though it can be restricted by using the **Aux Subrange** control in the **Range** tab, described below). The `linear`, `log`, `square`, `sqrt`, `acos` and `cos` options just apply the named function to the full data range. The histogram options on the other hand use a scaling function that corresponds to the actual distribution of the data, so that there are about the same number of points (or pixels, or whatever is being scaled) of each colour. The histogram options are somewhat more expensive, but can be a good choice if you are exploring data whose distribution is unknown or not well-behaved over its min-max range. The `histogram` and `histolog` options both assign the colours in the same way, but they display the colour ramp with linear or logarithmic annotation respectively; the `histolog` option also ignores non-positive values.

Null Colour
What colour should be used to represent points with a null value for the aux data coordinate. If the associated **Hide** option is selected, then those points will not appear in the plot at all.

<u>**Ramp Tab**</u>



Show Scale: ☑ ☑ Auto

Aux Axis Label: Aux ☑ Auto

Aux axis control Ramp tab

The **Ramp** tab controls the display and annotation of the colour ramp that displays the colour map on the plot. It has the following options:

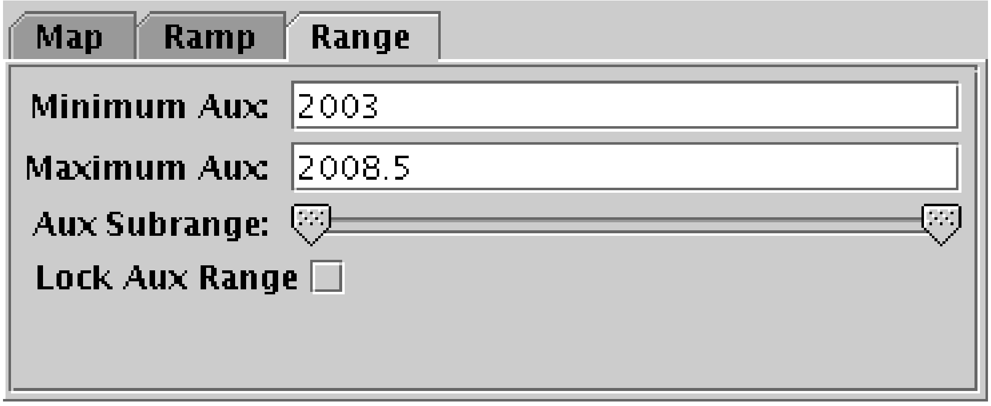
Show Scale
Whether the aux scale ramp is visible; if so an appropriately labelled colour ramp appears at the right of the plot. The associated **Auto** option makes this decision automatically: if any aux data is plotted, the scale will appear, otherwise it won't. Deselect Auto if you want to determine visibility by hand.

Aux Axis Label
Selects the axis label to be displayed near the aux colour ramp if it is visible. The associated

Auto option, if selected, uses the name of one of the coordinates supplying aux data; deselect Auto if you want to enter a label by hand.

Aux Tick Crowding
The slider influences how many tick marks are drawn on the colour ramp.

Range Tab



Minimum Aux: 2003
Maximum Aux: 2008.5
Lock Aux Range □

Aux axis control Range tab

The **Range** tab lets you enter lower and upper values for the aux data range by hand, and provides a double slider to restrict the range within these limits. If either the lower or upper range is left blank, it will be determined from the data.

The **Lock Aux Range** checkbox controls whether the aux range is automatically updated as the plot is adjusted (for instance as you navigate around it with the mouse). If locked, the range will stay the same, but otherwise (the default) it is dynamically updated for the current view of the plot. This checkbox is a duplicate of the toggle button on the plot window toolbar described in Appendix A.4.2.

Note the font used for labelling the aux axis is currently controlled by the font from the **Axes** selector.

A.4.4 Layer Controls

The layer controls are controls in the Control Stack that can be added, removed and moved around to determine what layers go to make up the contents of a plot. You can have zero, one or several of each. Which ones are available is dependent on which plot type you are using (for instance the Spectrogram control is only available for the Time plot). Add instances of each control to the control stack by using the appropriate button from the control panel toolbar (the one in the lower half of the window) or the corresponding item in the **Layers** menu.

The available layer controls are described in the following subsections. More may be added in future releases.

A.4.4.1 Position Layer Control

The **Position** layer control ( ) is available for all the plot types (except the Corner Plot, which uses the similar Matrix control). Most plots start off with one of these in the stack by default, and you can add a new instance by using the **Add Position Control** ( ) button in the control panel toolbar, or the corresponding item in the **Layers** menu.

This is the control which is used for most of the data plotting in the plotting windows. Each instance of this control in the stack does plotting for a particular set of positions from a single table. The set of positions is defined in the **Positions** tab as a column name or expression for each plot coordinate (e.g. for X and Y in a plane plot). However, the control can generate multiple layers from these positions; the **Subsets** tab controls which subsets are plotted and how each one is identified, and the **Form** tab provides many options for what graphics will be plotted based on the positions.

This control is a Table Data control as described in Appendix A.4.2.2. That section explains the **Subsets** tab; the **Position** and **Form** tabs are described in more detail below.

Position Tab

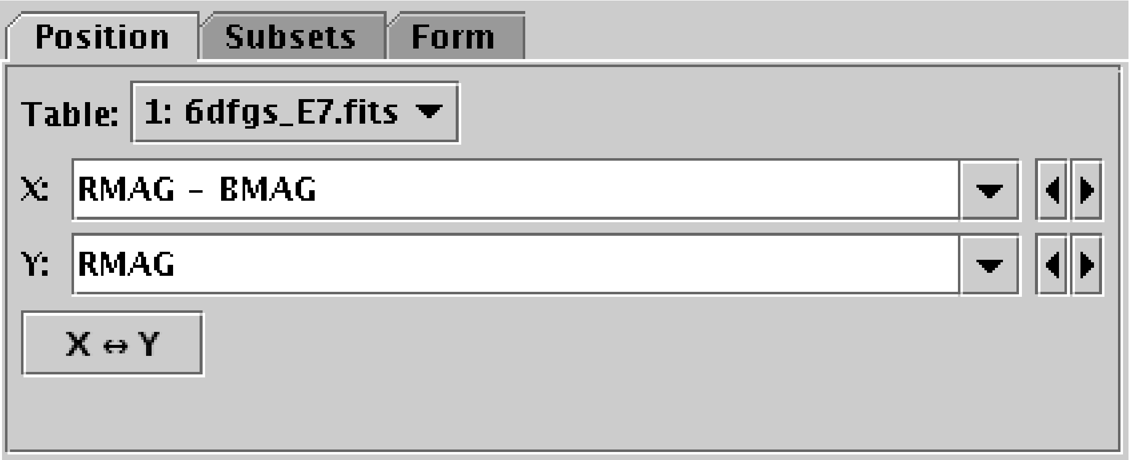


Table: 1: 6dfgs_E7.fits

X: RMAG - BMAG

Y: RMAG

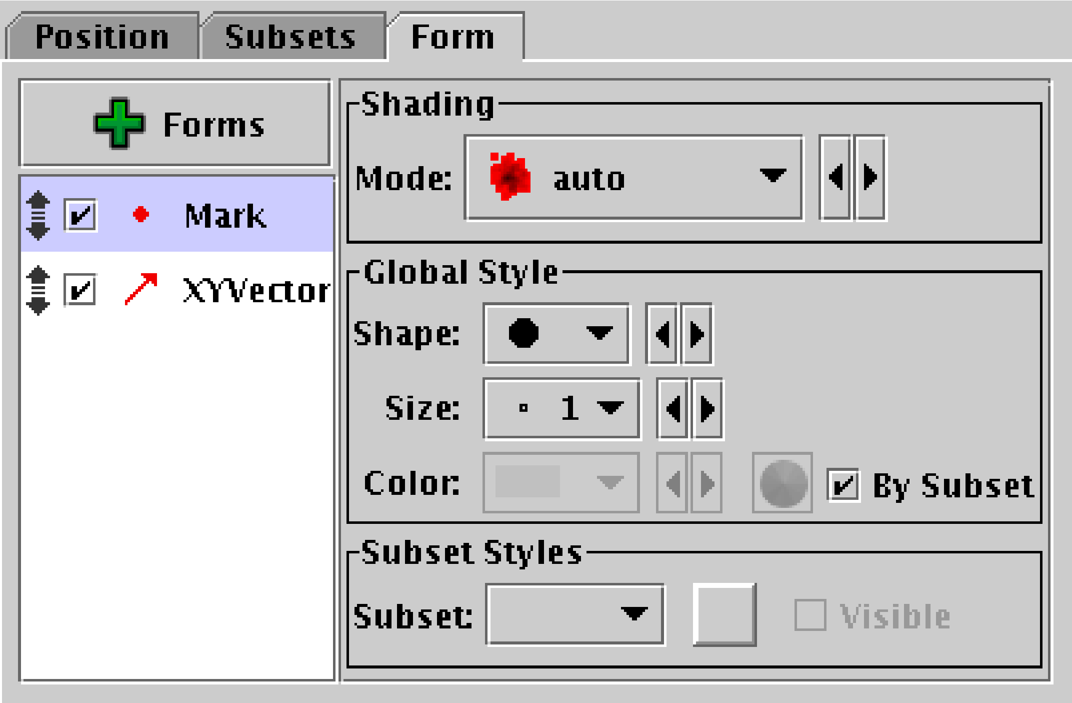
Position tab of Position layer control, for Plane plot

In the **Position** tab you enter the base position coordinates for each plotted point. This generally means selecting a table and providing a value (table column or expression) for each positional coordinate. When you first open a plot window, TOPCAT gives you a table layer control by default, and attempts to fill in the positional coordinates with some reasonable values from the table (for instance the first few numeric columns).

Note the details of the **Position** tab will be different for different plot types, for instance the Cube plot has a Z coordinate field alongside X and Y.

For the Plane plot only, there is an **X<->Y** button that lets you swap the contents of the **X** and **Y** fields for convenience. Note that swapping those contents is the only thing it does, it does not for instance swap the log flags for the axes, so the result may not be exactly the same as reflecting the plot about the X=Y line.

Form Tab



☑ Mark
☑ XYVector

Mode: auto

Shape: ●

Size: 1

Color:

☑ By Subset

Subset:

□ Visible

Form tab of Position layer control

The **Form** tab lets you define how the specified data set is plotted. The stack on the left gives a list of forms currently being plotted, and the panel on the right shows the detailed configuration for the currently selected form.

When first added, the stack contains a single entry, **Mark**, which plots a marker of a given fixed shape and size. The colour is by default determined by the setting in the Subsets tab. For a simple scatter plot, this is all that you need. However, there are a number of other forms that you can plot as well or instead of the simple markers - vectors, error bars, ellipses, contours, text labels etc. You add a new form to the stack by clicking on the ✚ **Forms** button, which gives you a menu of all the available forms for the current layer control. You can remove a form by selecting it and selecting the **Remove** (🗑) button in the same menu. You can also activate/deactivate the entries in the stack with the checkbox and move them up and down with the drag handle as usual. The list of forms that are avaiable depends on the plot type; the full list is in Appendix A.4.5.

The detail panel of each form depends on the form itself. It is divided into the following panels, though not all forms have all the panels.

Shading
The shading mode controls how points are shaded based on their chosen colour. The various options are described in Appendix A.4.6. Depending on the mode there may be more settings to fill in here.

Coordinates
If additional coordinates are required for this form, for instance the size of error bars, you need to enter the column or expression here. Each coordinate effectively adds another dimension to the plot. Many forms, like Mark, do not require any additional coordinates.

Global/Subset Styles
Controls the style details for the chosen form; see Appendix A.4.2.2.

A.4.4.2 Pair Position Layer Control

The **Pair Position** layer control ( ) allows you to plot symbols linking two positions in the plot space from the same table. You can add one of these controls to the stack by using the **Add Pair Control** ( ) button in the control panel toolbar, or the corresponding item in the **Layers** menu.

This control is particularly useful for visualising the results of a crossmatch, and it is used automatically by the **Plot Result** ( ) option offered by some of the Match Window results. If you want to plot pairs of points from different input tables, you first have to create a joined table by using one of the crossmatch functions.

It works in a similar way to the Single Position Layer Control, but the **Position** tab has fields for not one but two sets of coordinates to fill in.

This control is a Table Data control as described in Appendix A.4.2.2. That section explains the **Subsets** tab; the **Position** and **Form** tabs are described in more detail below.

Position Tab

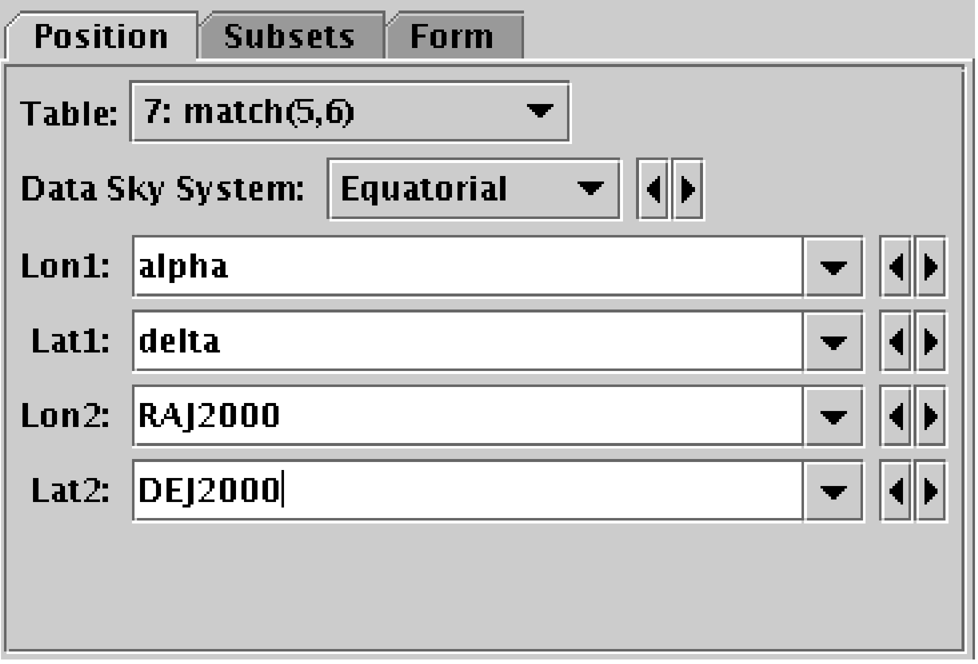


Table: 7: match(5,6)

Data Sky System: Equatorial

Lon1: alpha

Lat1: delta

Lon2: RAJ2000

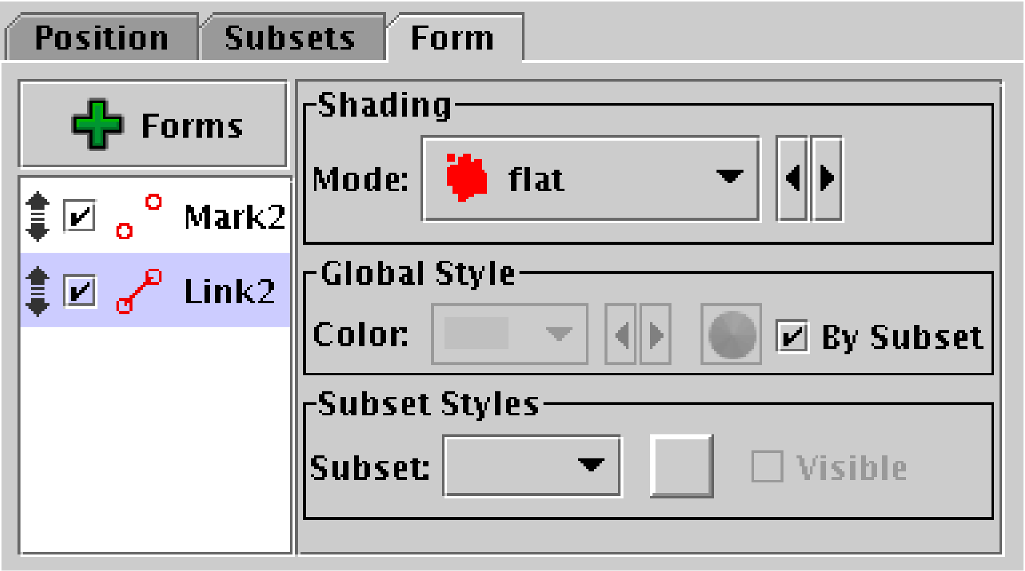
Lat2: DEJ2000

Position tab of Pair Position layer control, for Sky plot

In the **Position** tab you enter the pair of position coordinates for each plotted pair. This generally means selecting a table and providing a value (table column or expression) for the first and second sets of positional coordinates.

Note the details of the **Position** tab will be different for different plot types, for instance the Sky plot has **Lon1**, **Lat1**, **Lon2** and **Lat2** fields for the first and second longitude/latitude pairs, while a Plane plot has **X1**, **Y1**, **X2** and **Y2**.

Form Tab



Mode: flat

Color: ☑ By Subset

Subset: □ Visible

Form tab of Pair Position layer control

The **Form** tab lets you define how the specified data set is plotted. The list on the left gives a list of forms currently being plotted, and the panel on the right shows the detailed configuration for the currently selected form.

The available forms for a pair plot (select them using the ✚ **Forms** button) are currently Mark2 ( ) which draws the points at both ends of the pair, and Link2 ( ) which draws a line linking them. You can configure these, and select them on or off, separately. The detail panel for these forms are dependent on the form itself and are described in more detail in Appendix A.4.5, but the detail panels are divided into these parts:

Shading
The shading mode controls how points are shaded based on their chosen colour. The various options are described in Appendix A.4.6. Depending on the mode there may be more settings to fill in here.

Global/Subset Styles
Controls the style details for the chosen form; see Appendix A.4.2.2.

A.4.4.3 Quad Position Layer Control

The **Quad Position** layer control ( ) allows you to plot symbols defined by four positions in the plot space from each row of a table - typically some kind of quadrilateral. You can add one of these controls to the stack by using the **Add Quad Control** ( ) button in the control panel toolbar, or the corresponding item in the **Layers** menu.

It works like the Single Position and Pair Position Layer Controls, but the **Position** tab has fields for four sets of coordinates to fill in.

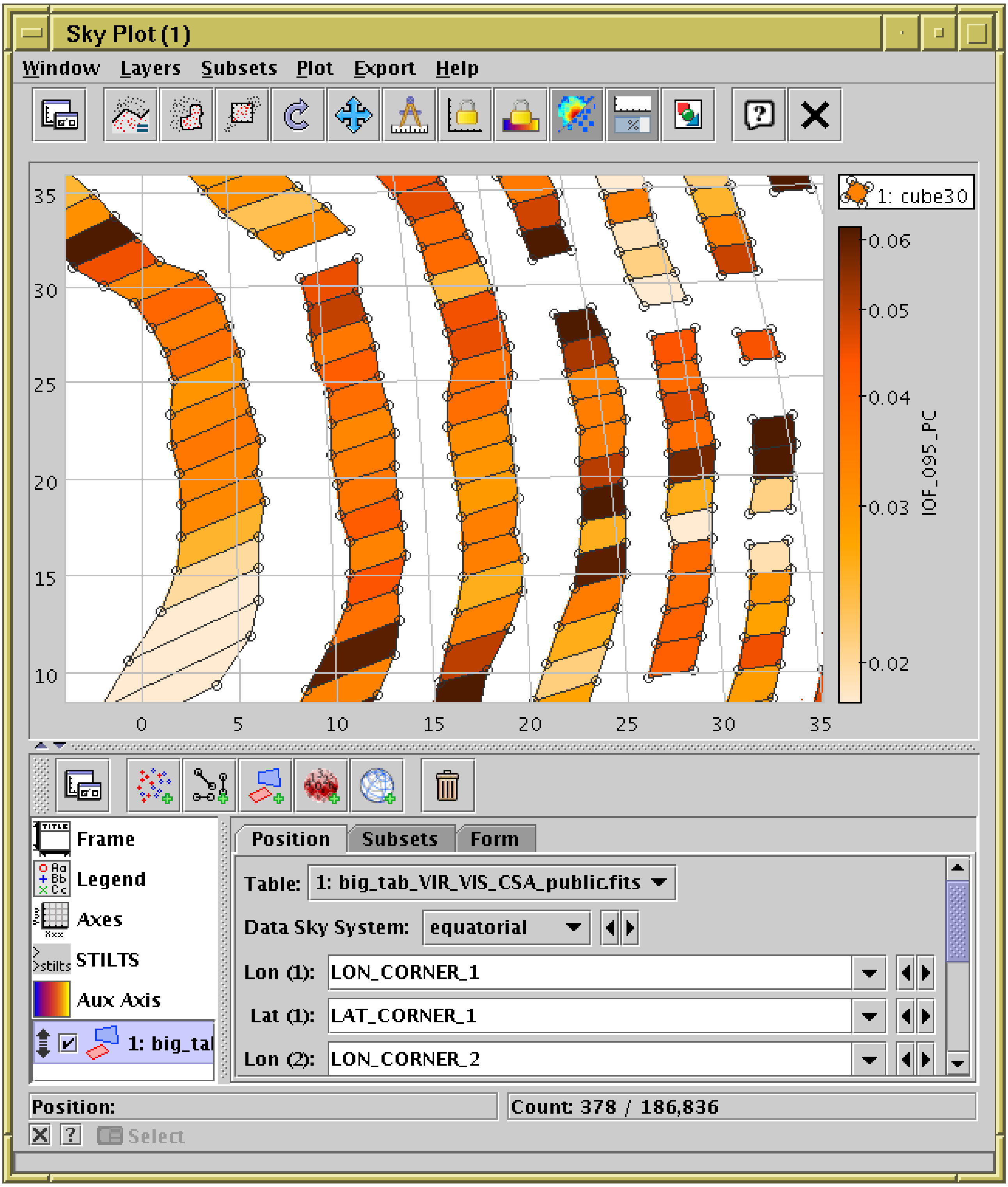


Table: 1: big_tab_VIR_VIS_CSA_public.fits

Data Sky System: equatorial

Lon (1): LON_CORNER_1

Lat (1): LAT_CORNER_1

Lon (2): LON_CORNER_2

Position:

Count: 378 / 186,836

Sky Plot Window with a Quad control

This control is a Table Data control as described in Appendix A.4.2.2. That section explains the **Subsets** tab; the **Position** and **Form** tabs are described in more detail below.

Position Tab

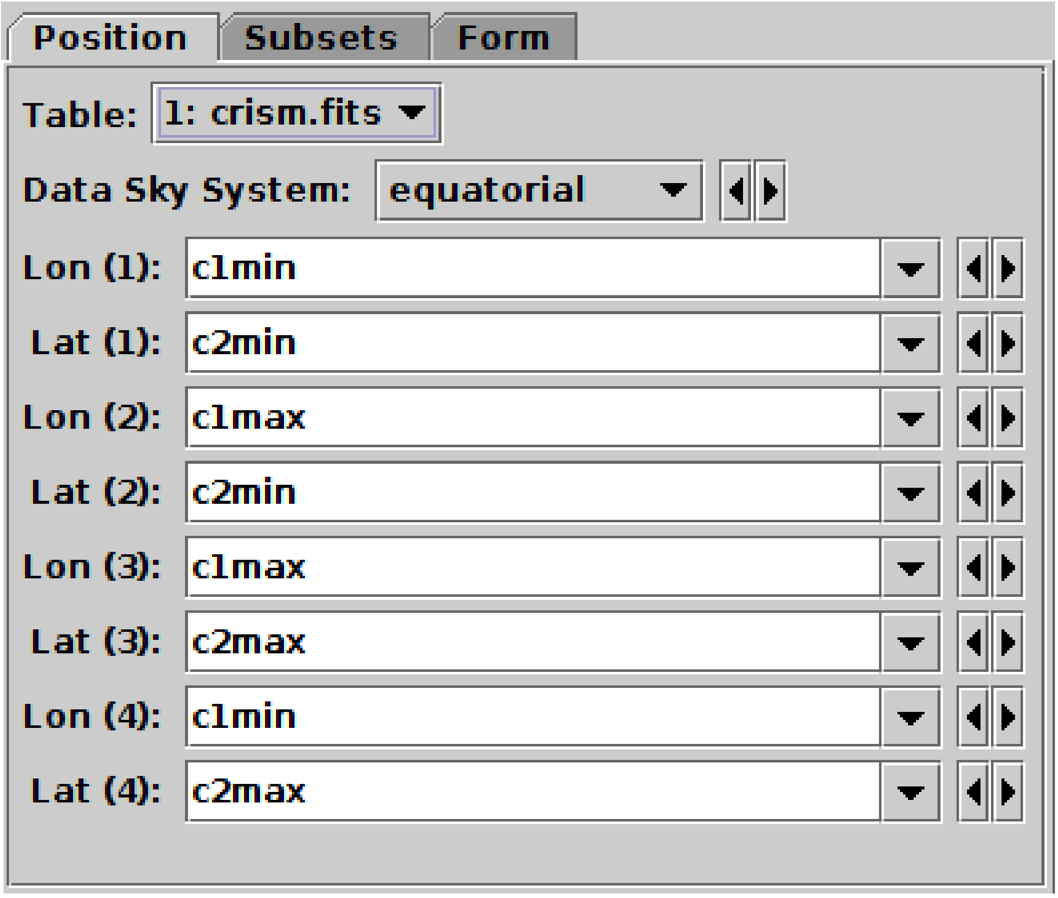


Table: 1: crism.fits

Data Sky System: equatorial

Lon (1): c1min

Lat (1): c2min

Lon (2): c1max

Lat (2): c2min

Lon (3): c1max

Lat (3): c2max

Lon (4): c1min

Lat (4): c2max

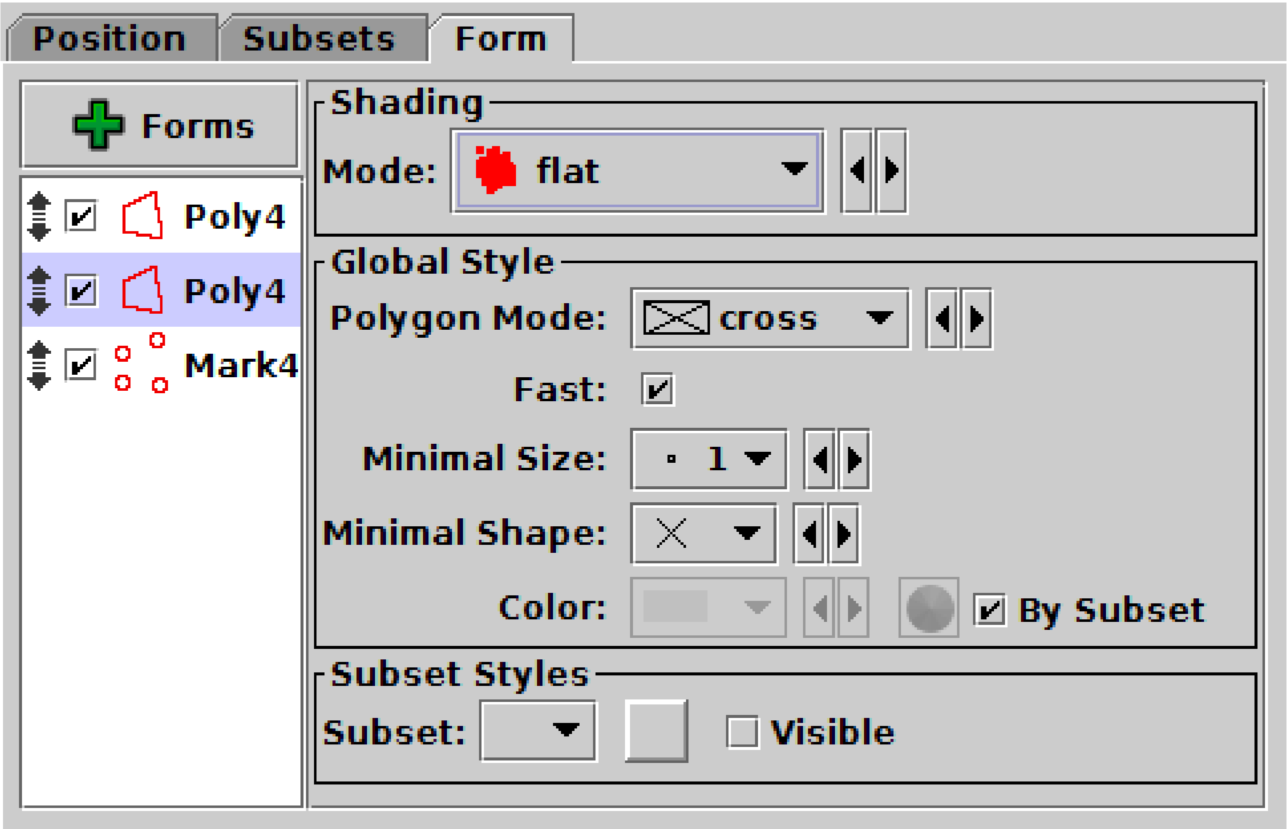
Position tab of Quad Position layer control, for Plane plot

In the **Position** tab you select a table and enter four sets of position coordinates (using a column name or expression for each one) to define the plotted quadrilaterals.

Note the details of the **Position** tab will be different for different plot types, for instance the Sky plot has **Lon (1)**, **Lat (2)** etc for the (longitude,latitude) pairs, while the Cube plot has **X (1)**, **Y (1)**, **Z (1)** etc. for the 3-dimensional Cartesian triples.

In some cases, for instance EPN-TAP tables with bounding boxes in sky plots, positions will be filled in automatically. You can of course change these default selections.

Form Tab



☑ Poly4
☑ Poly4
☑ Mark4

Shading
Mode: flat

Global Style
Polygon Mode: cross
Fast: ☑
Minimal Size: 1
Minimal Shape: ×
Color:
☑ By Subset

Subset Styles
Subset:
□ Visible

Form tab of Quad Position layer control

The **Form** tab lets you define how the specified data set is plotted. The list on the left gives a list of forms currently being plotted, and the panel on the right shows the detailed configuration for the currently selected form.

The available forms for a quad plot (select them using the **Forms** button) are currently

- Poly4: draws an outlined or filled quadrilateral
- Mark4: plots a marker on each point

You can have zero or more of each form and configure them separately. The detail panel for these forms is dependent on the form itself and are described in more detail in Appendix A.4.5, but the detail panels are divided into these parts:

Shading
The shading mode controls how points are shaded based on their chosen colour. The various options are described in Appendix A.4.6. Depending on the mode there may be more settings to fill in here.

Global/Subset Styles
Controls the style details for the chosen form; see Appendix A.4.2.2.

If you want to draw triangles instead of quadrilaterals, you can just supply the same coordinates for two of the positions. If you want a polygon with more vertices, try the Polygon form from the single position layer control.

A.4.4.4 Matrix Layer Control

The **Matrix** layer control () is available only for the Corner Plot. The Corner Plot window starts with one of these in the stack by default, and you can add a new instance by using the **Add Position Control** () button in the control panel toolbar, or the corresponding item in the **Layers** menu.

This is the control that supplies all the plots for the Corner plot. Each instance of this control in the stack does plotting for a set of multidimensional points from a single table. The set of points is defined in the **Position** tab as a column name or expression for each plot coordinate, while the **Fill** tab provides some controls that can help to fill in in the **Position** tab. The control can generate multiple different layers from these positions; the **Form** tab provides many options for what graphics will be plotted based on the positions, and the **Subsets** tab controls which subsets are plotted and how each one is identified.

This control is a Table Data control as described in Appendix A.4.2.2. That section explains the **Subsets** tab; the **Position**, **Fill** and **Form** tabs are described in more detail below.

Position Tab

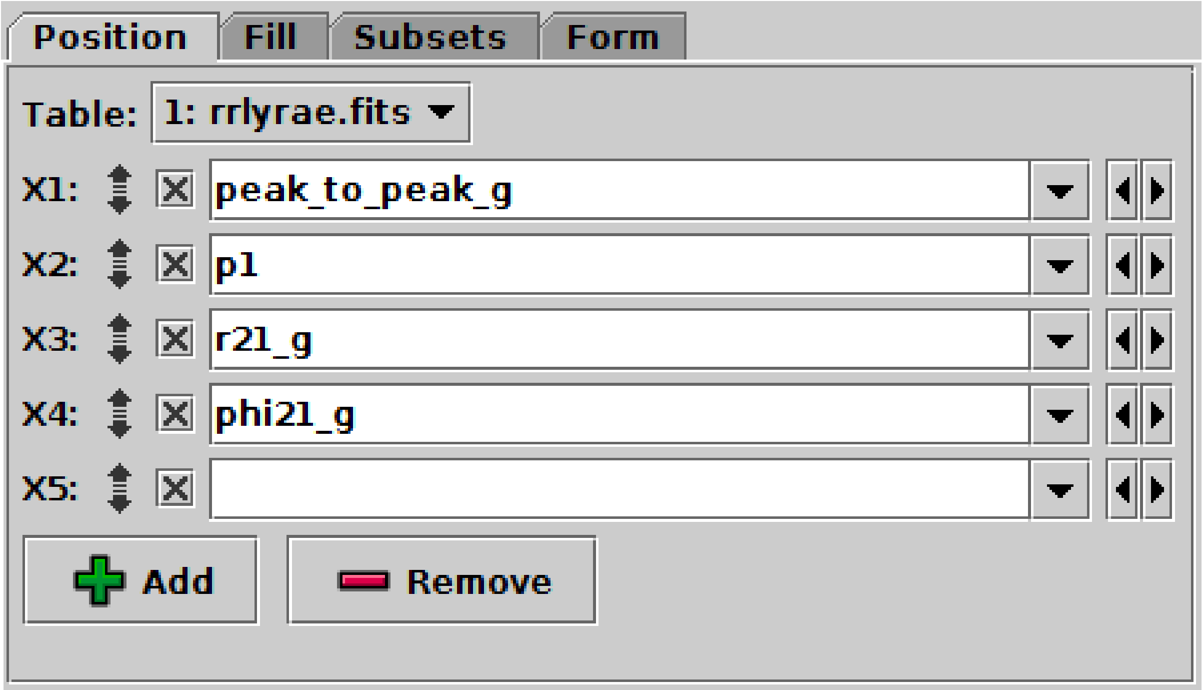


Table: 1: rrlyrae.fits

X1: ☒ peak_to_peak_g

X2: ☒ p1

X3: ☒ r21_g

X4: ☒ phi21_g

X5: ☒

Position Tab of Matrix layer control

To select the quantities that will be plotted against each other, in the **Position** tab select column names or enter algebraic expressions into the fields marked **X1**, **X2**, **X3** etc. Grid cells will be plotted for each distinct pair (scatter-plot-like) and single (histogram-like) combination of the specified coordinates.

This position entry panel has a few features not found in the other plot types:

- The **Add** (✚) and **Remove** (▬) buttons can be used to increase/decrease the number of fields visible for coordinate entry.
- There is a **Grab Handle** (↕) by each coordinate; you can drag this up or down using the mouse to re-order the coordinate values you have entered.
- There is a small **Delete** button (☒) by each coordinate; you can use this to remove the adjacent

value from the plot (all the ones below will move up one).

Fill Tab

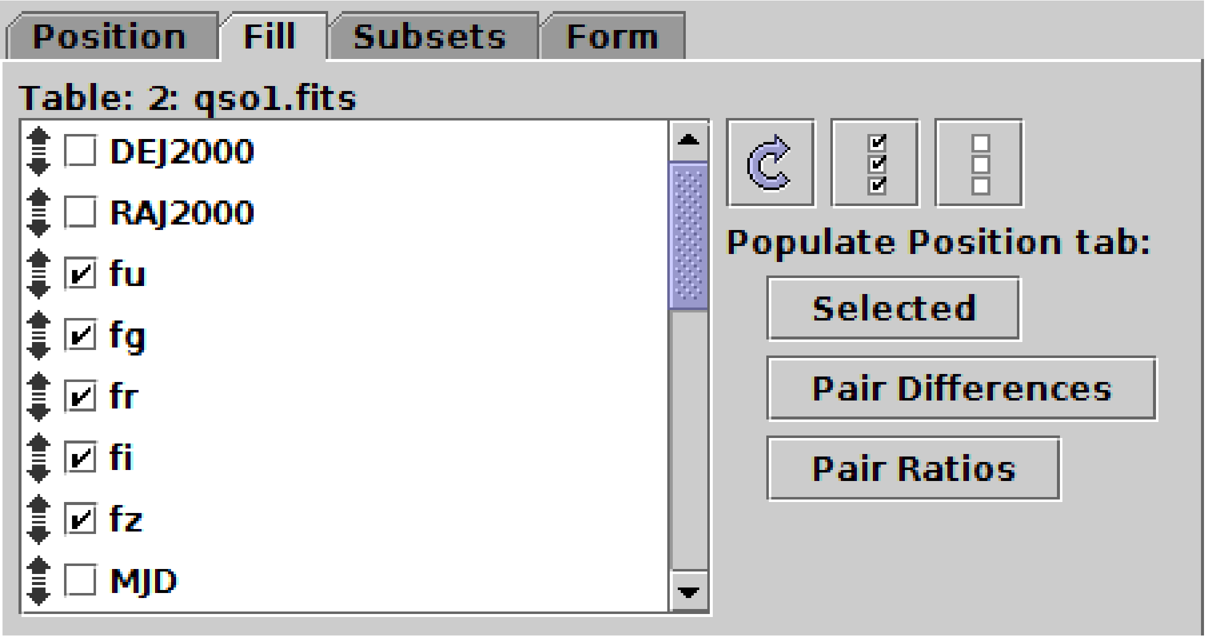


Table: 2: qsol.fits

⬍ □ DEJ2000
⬍ □ RAJ2000
⬍ ☑ fu
⬍ ☑ fg
⬍ ☑ fr
⬍ ☑ fi
⬍ ☑ fz
⬍ □ MJD

Fill Tab of Matrix layer control

The **Fill** tab lets you choose columns from the table you are plotting, and use them to populate the selectors in the **Position** tab. Moving and selecting the columns in this tab does nothing on its own, but if you hit one of the buttons under the **Populate Position tab** heading on the right, the **Position** tab, and hence the plot, is updated accordingly. This doesn't do anything that can't be done by filling in fields in the Position tab directly, but it can be a quicker and more convenient way to do it.

The left hand panel shows all the columns in the input table (as selected in the Position tab). They can be reordered using the mouse by dragging them with their **Grab Handle** (⬍) and the checkbox by each one can be selected. The three buttons at the top of the right hand panel also affect this list:

- ↻ : Reloads the column list from the table; this ensures that the order of the columns in the list is the same as the current column order in the table it comes from
- ☑☑☑ : Selects all the columns in the list
- □□□ : Unselects all the columns in the list

When you have selected the columns you're interested in, you can hit one of the buttons below the **Populate Position Tab** heading.

Selected
All the columns selected in this window will be used to fill in the coordinate selectors in the Position tab.

Pair Differences
All pair differences between the columns selected in this window will be used to fill in the coordinate selectors in the Position tab. So for instance if the four columns `A`, `B`, `C` and `D` are selected in the list here, then clicking this button will fill in the selectors in the Position tab with X1=`A-B`, X2=`A-C`, X3=`A-D`, X4=`B-C`, X5=`B-D`, X6=`C-D`. The number of replacement

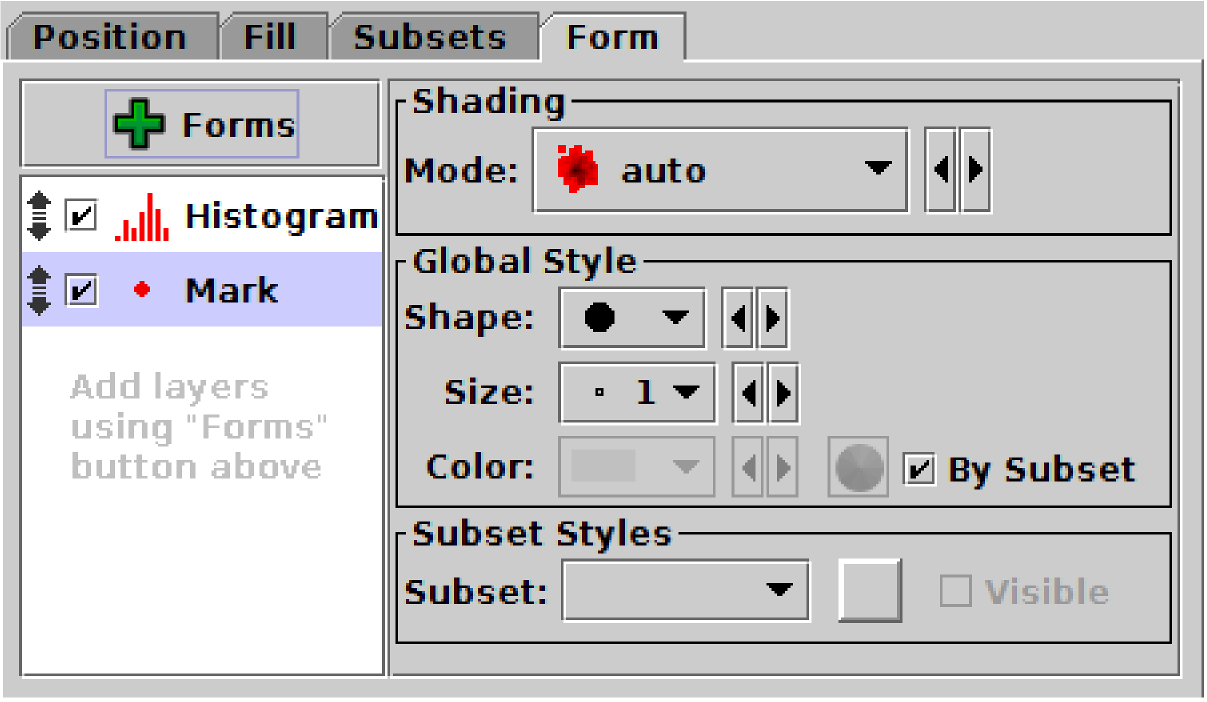
coordinates (hence the *linear* size of the grid) will be *N(N-1)/2* so it can end up generating a lot of plots. This is a rather special-interest option, but it can be handy for instance when working with sets of correlated parameters such as source magnitudes.

Pair Ratios

Like the **Pair Differences** above, but instead of differences, ratios are used, so the selectors would be filled in with X1=`A/B`, X2=`A/C`, X3=`A/D`, X4=`B/C`, X5=`B/D`, X6=`C/D`. This could be useful when working with sets of fluxes.

Note these buttons will only be enabled if they would result in a reasonable number of position coordinates according to the current column selection. If any of them would result in fewer than two, or more than some reasonable maximum (64 at time of writing), it is disabled.

Form Tab



Form Tab of Matrix layer control

The **Form** tab lets you define how the specified data set is plotted. The stack on the left gives a list of forms currently being plotted, and the panel on the right shows the detailed configuration for the currently selected form.

When first added, the stack contains two entries, **Mark** and **Histogram**. This gives you a scatter plot on all the off-diagonal grid positions, and a histogram on all the diagonal ones. However, there are a number of other forms that you can plot as well or instead of the simple markers; two-coordinate plots like contours and linear fits on the off-diagonal positions, and one-coordinate plots like KDEs on the diagonal positions. You add a new form to the stack by clicking on the ✚ **Forms** button, which gives you a menu of all the available forms. You can remove a form by selecting it and clicking the **Remove** (🗑) button in the same menu. You can also activate/deactivate the entries in the stack with the checkbox and move them up and down with the drag handle as usual.

The detail panel of each form depends on the form itself. It is divided into the following panels, though not all forms have all the panels.

Shading
The shading mode controls how points are shaded based on their chosen colour. The various options are described in Appendix A.4.6. Depending on the mode there may be more settings to fill in here.

Coordinates
If additional coordinates are required for this form, for instance the size of error bars, you need to enter the column or expression here. Each coordinate effectively adds another dimension to the plot. Many forms, like Mark, do not require any additional coordinates.

Global/Subset Styles
Controls the style details for the chosen form; see Appendix A.4.2.2.

A.4.4.5 Healpix Layer Control

The **Healpix** layer control () is available from the Sky Plot Window for plotting tables that represent HEALPix maps on the celestial sphere. You can add one of these controls to the stack by using the **Add HEALPix Control** () button in the control panel toolbar, or the corresponding item in the **Layers** menu.

Each row in the table must represent a single healpix tile, and a value from that row is used to colour the corresponding region of the sky plot. The resolution (healpix order) of the input table is supplied or may be guessed in order to do the plot, but the plot may be drawn at a degraded order (bigger pixels) if required.

Note this is different from the SkyDensity form, which takes a table containing sky positions, and represents that as a density map on the healpix grid by gathering the rows up into bins. The control described here works on a table for which that binning has already been done, for instance as a prepared sky map data product, by exporting from a SkyDensity layer, or by executing a suitable (GROUP BY healpix_index) database query.

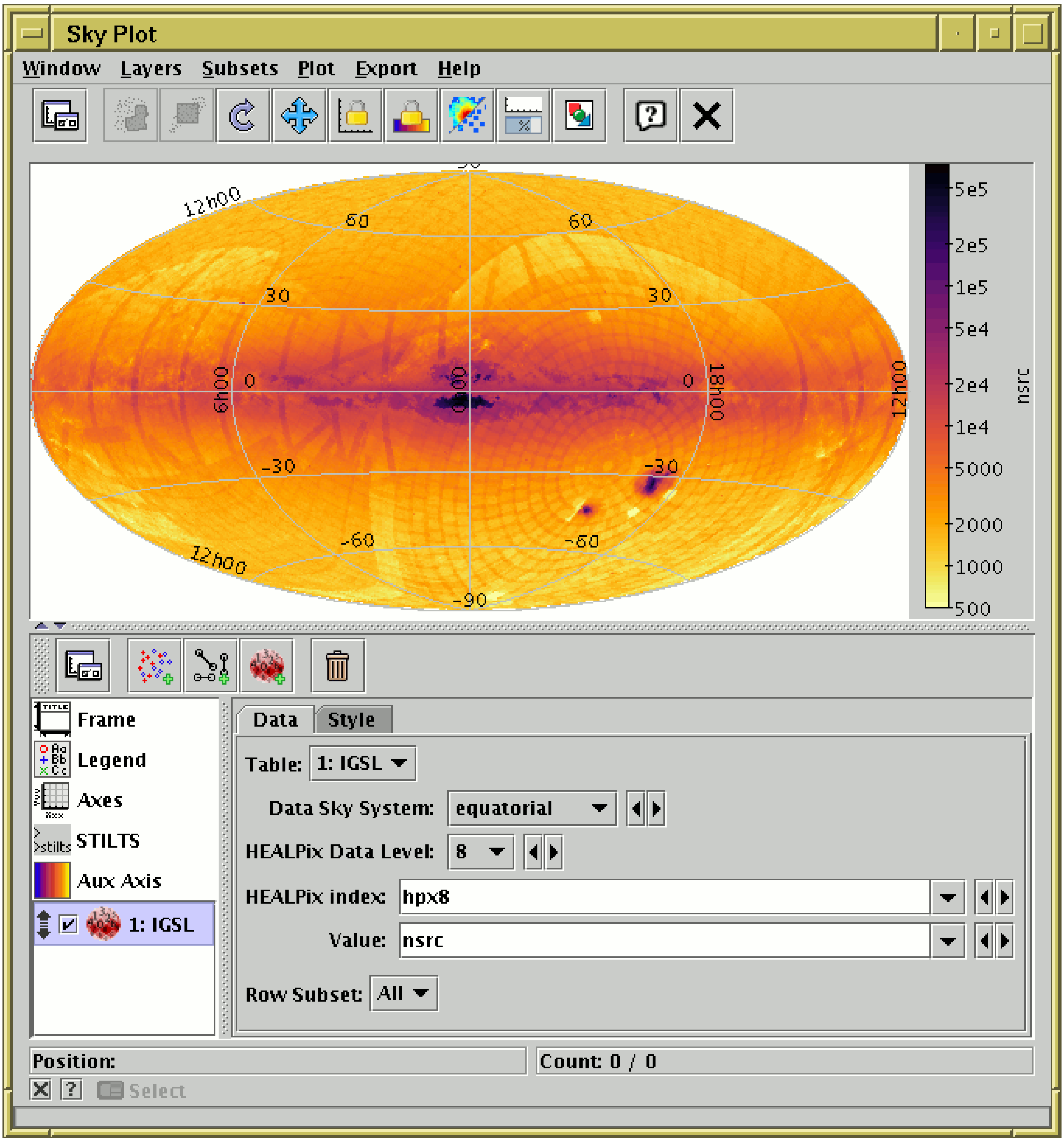


Table: 1: IGSL

Data Sky System: equatorial

HEALPix Data Level: 8

HEALPix index: hpx8

Value: nsrc

Row Subset: All

Position:

Count: 0 / 0

Sky Plot Window with a Healpix layer

This control has two tabs, **Data** and **Style**.

Data Tab

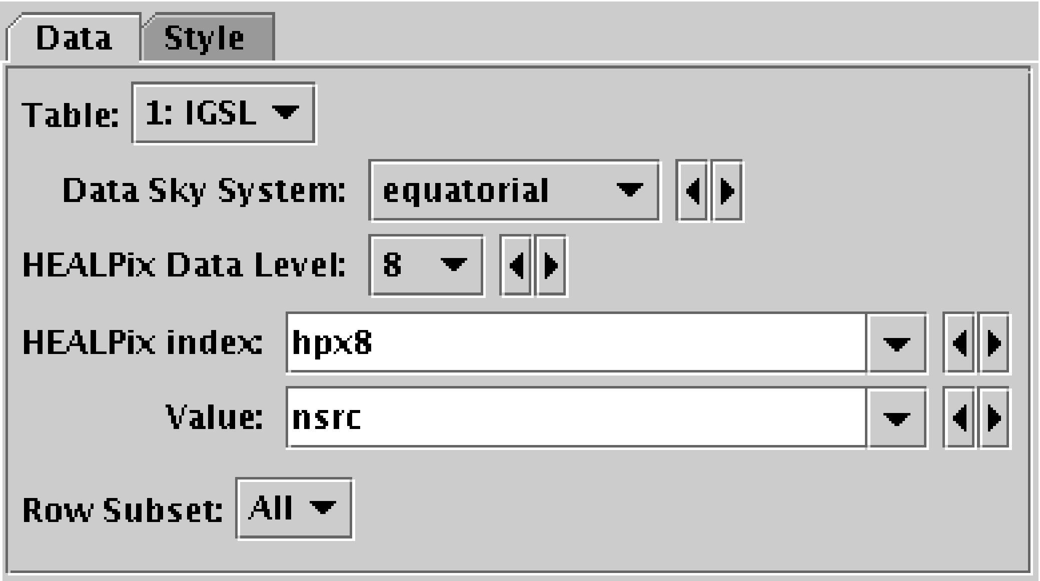


Table: 1: IGSL

Data Sky System: equatorial

HEALPix Data Level: 8

HEALPix index: hpx8

Value: nsrc

Row Subset: All

Healpix control Data tab

The **Data** tab lets you specify the table and columns containing the healpix tile data. It has the following fields:

Table
: The table supplying the data.

Data Sky System
: The sky coordinate system of the grid on which the pixels in the input table are laid out.

HEALPix Data Level
: HEALPix level of the (implicit or explicit) tile indices. Values up to 20 are currently supported. This must be the value assumed by the data in the input table. If **-1** is selected, an attempt is made to determine the correct level from the data; this may or may not be successful. Using the wrong value here will result in a nonsensical plot, or no plot at all.

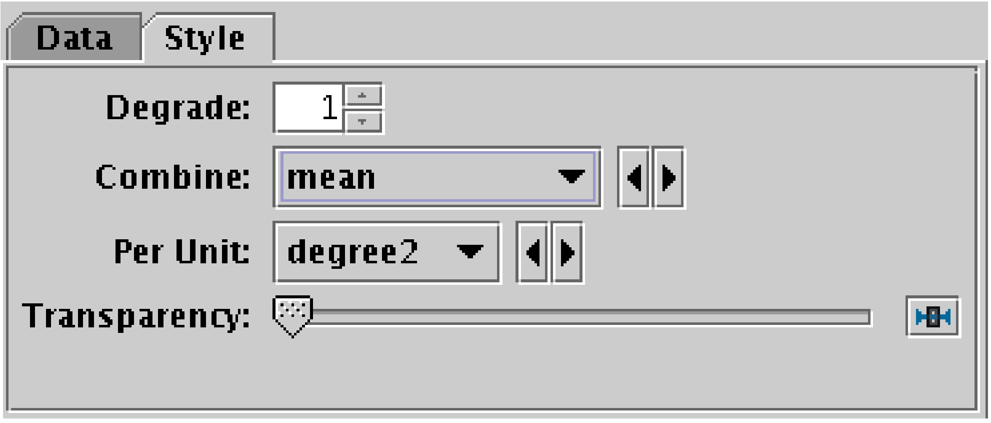
HEALPix index
: Column in the input table giving HEALPix index value. If this is left blank, then the row index is used, i.e. pixel #0 is assumed to be in the first row etc. This only works if the input table has full sky coverage and the rows are in the correct sequence. Note this value is zero-based, unlike the row index yielded by the special `$0` or `$index` token, you would have to write e.g. `$index-1`.

Value
: The column, or other expression, giving the colour to plot for the pixel at each row.

Row Subset
: May be used to restrict the plot to one of the defined row subsets.

To control the colour map used to colour the sky tiles, use the Aux fixed control.

Style Tab



Degrade: 1

Combine: mean

Per Unit: degree2

Transparency:

Healpix control Style tab

The **Style** tab lets you configure additional details of the plot's appearance. It has the following fields:

Degrade
Controls the resolution at which the pixel grid is actually plotted. If the value is zero, then the grid plotted is the same as that of the input data, but if a positive value is supplied then the healpix order will be reduced by that many. Each increment means that 4 pixels from the previous order will be combined into one. The way that combination is done is controlled by the **Combine** option.

Combine
Defines how values degraded to a lower HEALPix level are combined together to produce the value assigned to the larger tile, and hence its colour. This is mostly useful in the case that `degrade`>0.

For density-like values (`count-per-unit`, `sum-per-unit`) the scaling is additionally influenced by the **Per Unit** option.

The available options are:

- `sum`: the sum of all the combined values per bin
- `sum-per-unit`: the sum of all the combined values per unit of bin size
- `count`: the number of non-blank values per bin (weight is ignored)
- `count-per-unit`: the number of non-blank values per unit of bin size (weight is ignored)
- `mean`: the mean of the combined values
- `median`: the median of the combined values (may be slow)
- `q1`: the first quartile of the combined values (may be slow)
- `q3`: the third quartile of the combined values (may be slow)
- `min`: the minimum of all the combined values
- `max`: the maximum of all the combined values
- `stdev`: the sample standard deviation of the combined values
- `hit`: 1 if any values present, NaN otherwise (weight is ignored)

Per Unit
Defines the unit of sky area used for scaling density-like **Combine** values (e.g. `count-per-unit` or `sum-per-unit`). If the Combination mode is calculating values per unit area, this configures the area scale in question. For non-density-like combination modes (e.g. `sum` or `mean`) it has no effect.

The available options are:

- `steradian`: steradian
- `degree2`: square degree
- `arcmin2`: square arcminute

- `arcsec2`: square arcsecond
- `mas2`: square milliarcsec
- `uas2`: square microarcsec

Transparency
Adjusts the transparency of the filled area.

A.4.4.6 Histogram Layer Control

The **Histogram** layer control ( ), allows you to make a 1-dimensional histogram-like plots (weighted and unweighted histograms, various kinds of Kernel Density Estimates). It is the main control in the Histogram plot window, but is also available in the Plane and Time plot windows, so you can overplot a histogram on scatter or time plots, if you can think of some reason to do that. The Histogram window starts off with one of these controls in the stack by default, but as usual you can add more by using the **Add Histogram Control** ( ) button in the control panel toolbar, or the corresponding item in the **Layers** menu.

Each instance of this control in the stack can make histogram-like plots of a particular quantity from a particular table. The value to be histogrammed, and optionally an associated weighting term, are defined in the **Positions** tab using column names or expressions. However, the control can generate multiple layers from these coordinates; the **Subsets** tab controls which subsets are plotted and how each one is identified, and the **Form** tab provides many options for what graphics will be plotted based on the sample values specified in the Position tab.

This control is a Table Data control as described in Appendix A.4.2.2. That section explains the **Subsets** tab; the **Position** and **Form** tabs are described in more detail below.

Position Tab

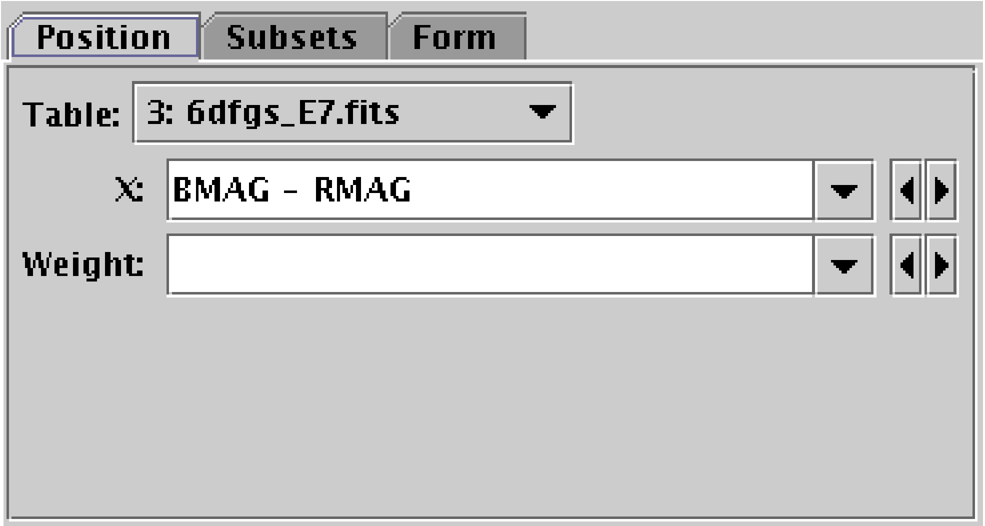


Table: 3: 6dfgs_E7.fits

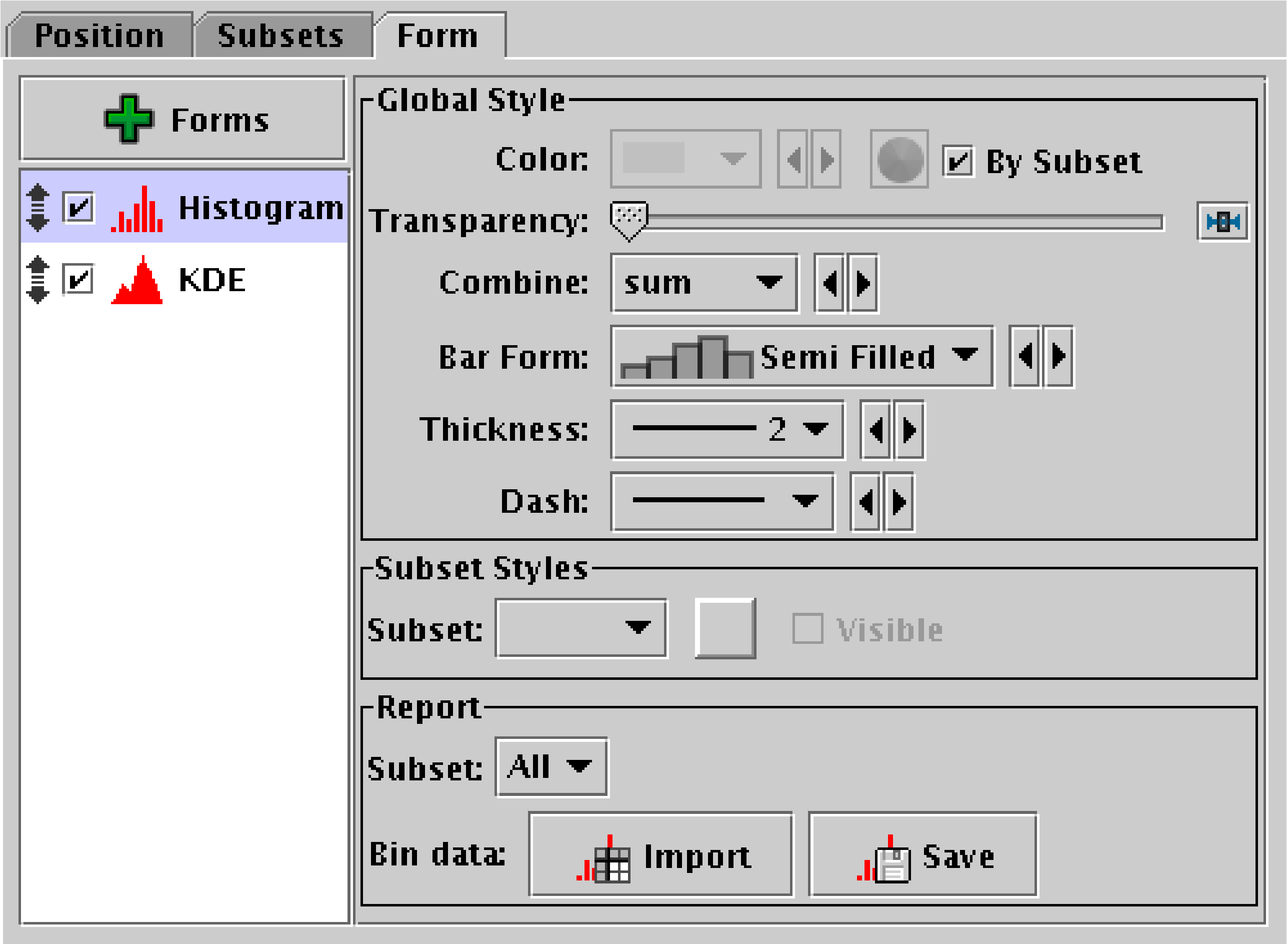
X: BMAG - RMAG

Weight:

Position tab of Histogram layer control

In the **Position** tab you enter the value of the **X** value, the one whose values along the X axis will be used to generate the plots. This may be a table column name or an expression. You may also optionally fill in a column name or expression in the **Weight** field. This weights the values as they are counted; each table row contributes a height value of the weighting coordinate to the bin into which its X coordinate falls. If Weight is not filled in, a value of 1 is assumed.

Form Tab



☑ Histogram
☑ KDE

Global Style
Color: ☑ By Subset
Transparency:
Combine: sum
Bar Form: Semi Filled
Thickness: 2
Dash:

Subset Styles
Subset: □ Visible

Report
Subset: All
Bin data:

Form tab of Histogram layer control

The **Form** tab lets you define how the specified data set is plotted. The stack on the left gives a list of forms currently included in the plot, and the panel on the right shows the detailed configuration for the currently selected form.

When first added, the stack contains a single entry, **Histogram**, which plots a simple histogram. The colour is by default determined by the setting in the **Subsets** tab. This may be all you need, but if you want to use other histogram-like plots such as Kernel Density Estimates, you can add new forms to the stack using items in the popup menu from the ➕ **Forms** button above the stack. You can remove a form by selecting it and using the **Remove** (🗑) item in the same menu. You can also activate/deactivate entries in the stack with the checkbox and move them up and down with the drag handle as usual. The available forms are currently Histogram; some variants on the idea of a Kernel Density Estimate: KDE, KNN, Densogram; and Gaussian, which effectively plots a Gaussian best fit to a histogram.

The **Global Style** and **Subset Style** sub-panels control shared and per-subset configuration specific to the selected form as described in Appendix A.4.2.2.

Histograms have some additional configuration items, as described in the Bins (📊) fixed control. When a histogram layer control is used in the Histogram window, those configuration items are found in the **Bins** fixed control, where they control the settings for all the histogram layers in the plot. However, if you use a histogram layer in a Plane plot, those items will appear as additional items in the **Form** tab and can be controlled separately for each histogram.

A.4.4.7 Area Layer Control

The **Area** layer control () allows you to plot a two-dimensional shape from each row of a table.

This might typically be something like an instrument coverage footprint corresponding to an observation. The shape may be specified in various different ways: as an STC-S region specification string (as found for instance in the `s_region` column of ObsCore or EPN-TAP query results), as an array specification of polygons or circles, or as an ASCII-encoded MOC. You can add one of these controls to the stack by using the **Add Area Control** () button in the control panel toolbar or the corresponding item in the **Layers** menu for suitable plot types (Plane, Sky or Sphere).

Areas specified in this way are generally intended for displaying relatively small shapes such as instrument footprints. Larger areas may also be specified, but there may be issues with use, for instance auto-determination of the initial plot region may not work so well, and the rendering of shapes that are large relative to the sky may be inaccurate. These issues may be addressed in future releases.

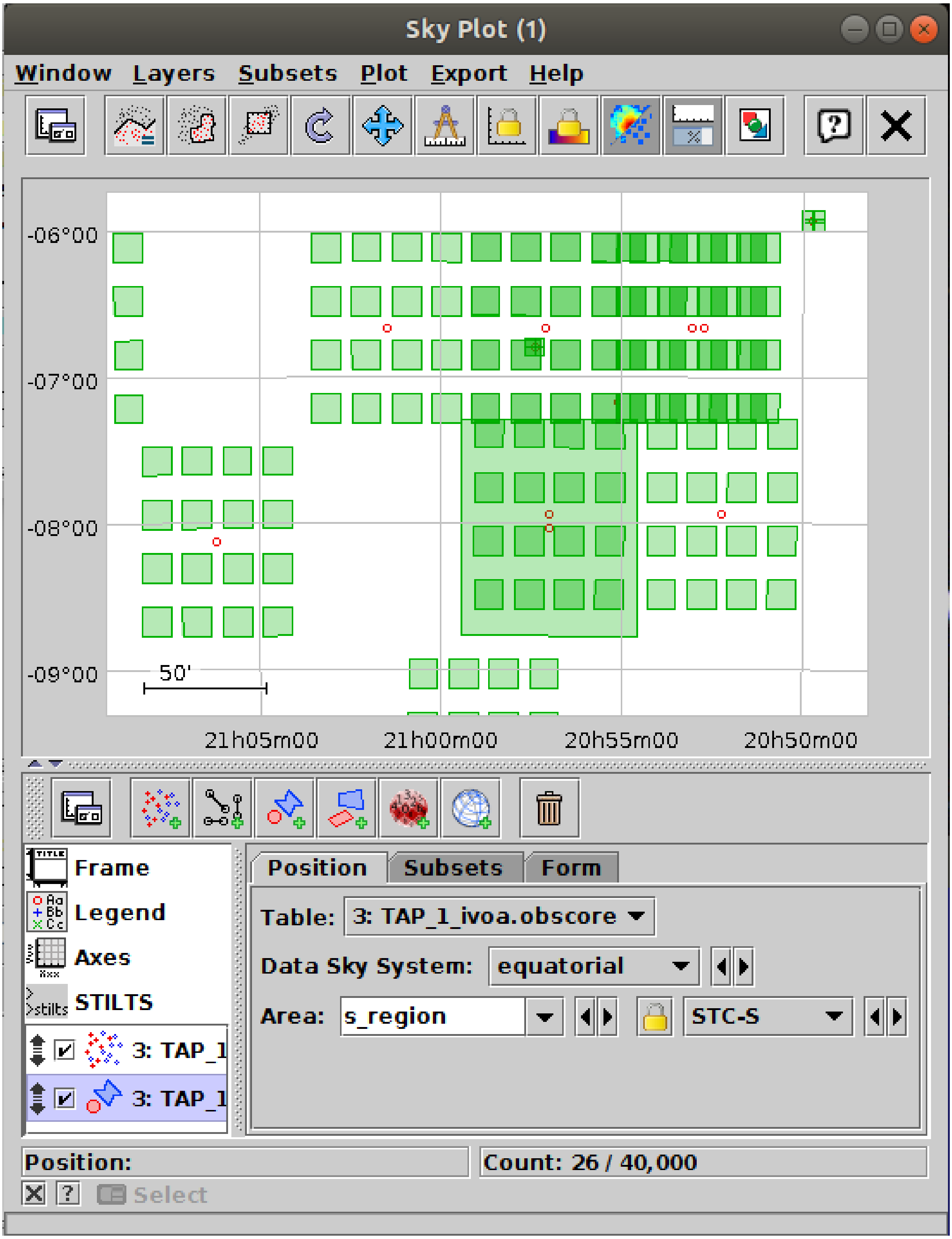


Table: 3: TAP_1_ivoa.obscore

Data Sky System: equatorial

Area: s_region STC-S

Position:

Count: 26 / 40,000

Sky Plot Window with an Area control

This control is a Table Data control as described in Appendix A.4.2.2. That section explains the **Subsets** tab; the **Position** and **Form** tabs are described in more detail below.

Position Tab

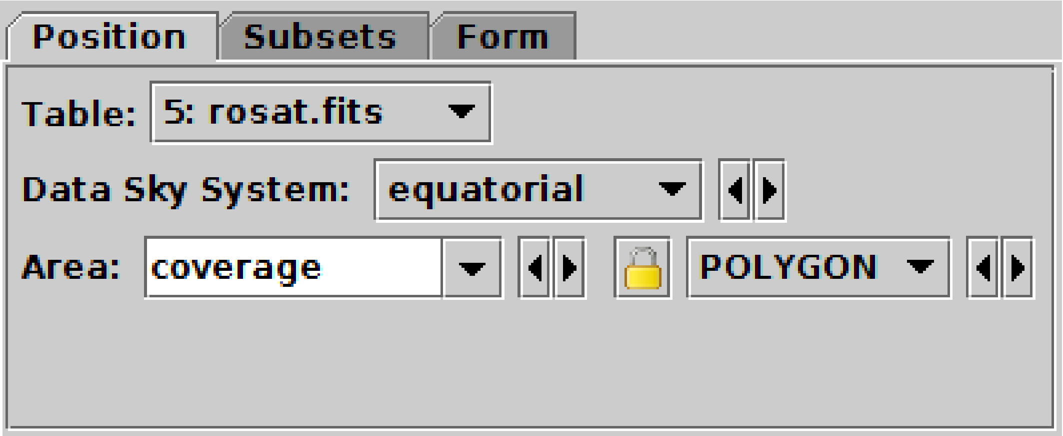


Table: 5: rosat.fits

Data Sky System: equatorial

Area: coverage POLYGON

Position tab of Area layer control, for Sky Plot

In the **Position** tab, you just need to select the input table using the **Table** selector, and then supply the expression for the region specification in the **Area** selector. This area value will typically be a table column containing the relevant information, for instance the **s_region** column in tables resulting from an ObsCore or EPN-TAP query. However, it's also possible to enter a suitable expression, for instance "`array(RA,DEC,RADIUS)`" if you want to specify a CIRCLE-format shape given position and radius columns in the table.

The area may be specified in various formats, currently:

STC-S
Region description using STC-S syntax; see TAP 1.0, section 6. Note there are currently some restrictions: `<frame>`, `<refpos>` and `<flavor>` metadata are ignored, polygon winding direction is ignored (small polygons are assumed) and the `INTERSECTION` and `NOT` constructions are not supported. The non-standard `MOC` construction is supported.

POLYGON
2*N*-element array (x1,y1, x2,y2, ..., x*N*,y*N*); a NaN,NaN pair can be used to delimit distinct polygons.

CIRCLE
3-element array (x,y,r).

POINT
2-element array (x,y)

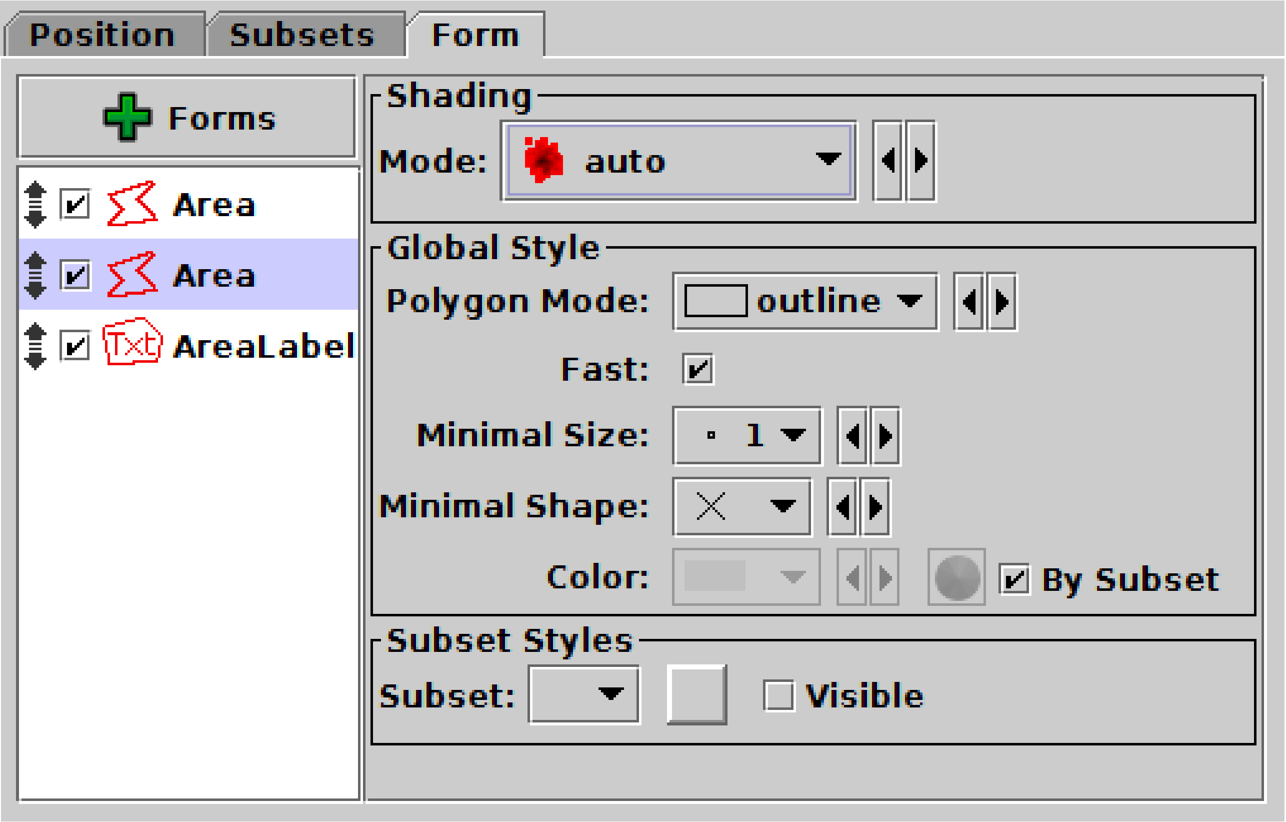
MOC-ASCII
Region description using ASCII MOC syntax; see MOC 1.1 section 2.3.2. Note there are currently a few issues with MOC plotting, especially for large tiles.

UNIQ
Region description representing a single HEALPix cell as defined by a UNIQ value; see MOC 1.1 sec 2.3.1.

TFCAT
Time-Frequency region defined by the TFCat standard. Support is currently incomplete; holes in Polygons and MultiPolygons are not displayed correctly, single Points may not be displayed, and Coordinate Reference System information is ignored.

The selector to the right of the **Area** selector defines which of these formats the selected value or expression will be understood as. TOPCAT tries to guess the right value when the Area selector is filled in; if it guesses wrong, you should select the correct option manually. If the Area value is changed, it will update the guess accordingly; if you don't want it to do that, use the **Lock** (🔒) button to stop it happening.

Form Tab



☑ Area
☑ Area
☑ AreaLabel

Shading
Mode: auto

Global Style
Polygon Mode: outline
Fast: ☑
Minimal Size: 1
Minimal Shape: ×
Color: ☑ By Subset

Subset Styles
Subset: ☐ Visible

Form tab of Area layer control

The **Form** tab lets you define how the specified region data will be plotted. The list on the left gives a list of forms currently being plotted, and the panel on the right shows the detailed configuration for the currently selected form.

The following forms are currently available for the Area plot:

- Area: draws an outlined or filled area
- Central: plots the central point of an area
- AreaLabel: adds a text label near the center of an area

It can be useful to have multiple instances of the **Area** form, for instance to paint the outline and fill the interior separately. The **Central** and **AreaLabel** forms just plot points/text in the same way as Mark/Label, but they work with an Area coordinate rather than normal positional ones.

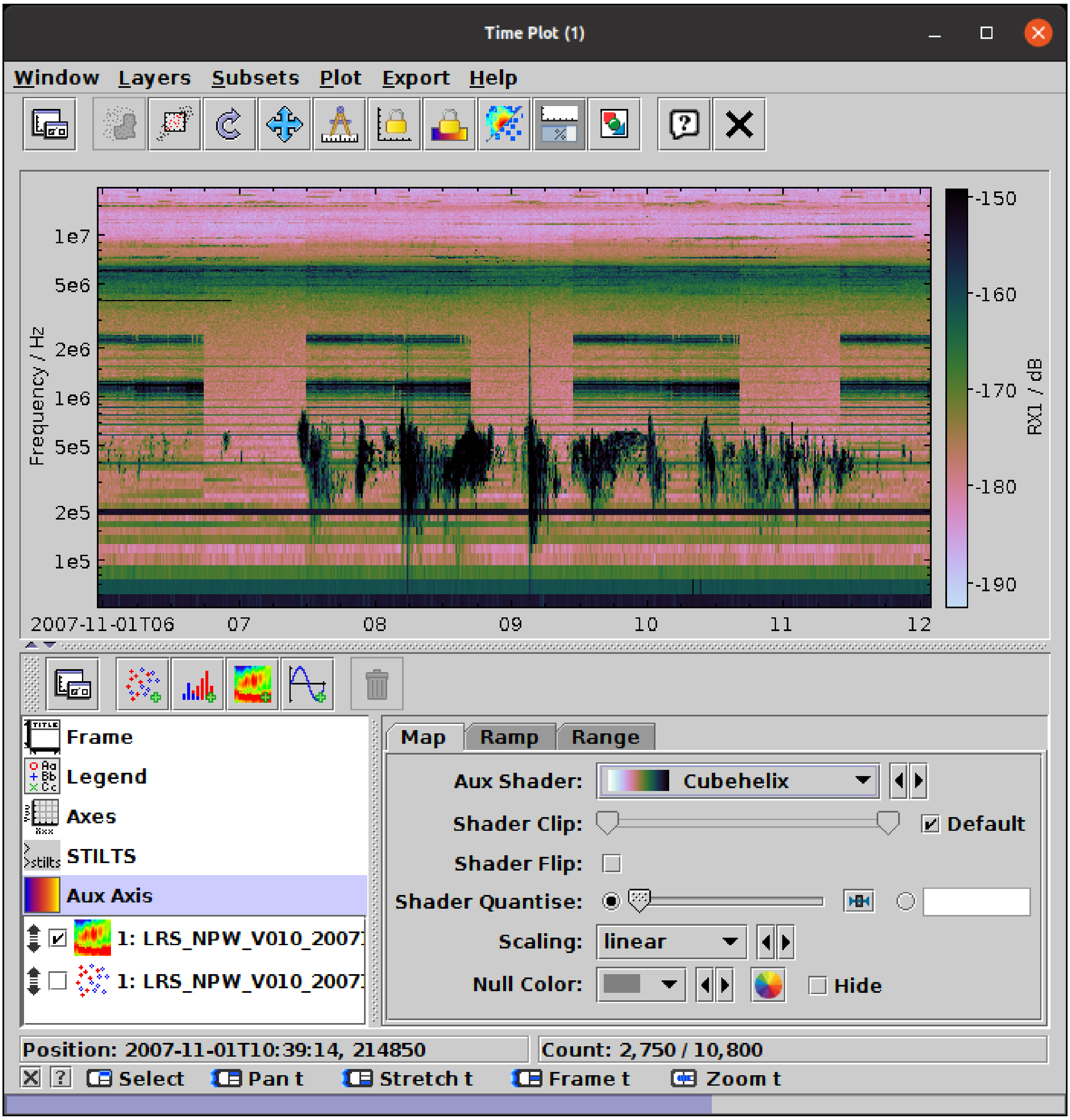
You can add a new form using the **Forms** button. Each such form can be configured separately using a panel divided into two parts:

Shading
The shading mode controls how points are shaded based on their chosen colour. The various options are described in Appendix A.4.6. Depending on the mode there may be more settings to fill in here.

Global/Subset Styles
Controls the style details for the chosen form; see Appendix A.4.2.2.

A.4.4.8 Spectrogram Layer Control

The **Spectrogram Layer Control** ( ) plots a spectrum at successive (usually, but not necessarily, regularly-spaced) points in a time series. It is only available for the Time Plot Window; you can add one of these controls to the stack by using the **Add Spectrogram Control** ( ) button in the control panel toolbar, or the corresponding item in the **Layers** menu.



Aux Shader: Cubehelix

Shader Clip: ☑ Default

Shader Flip: □

Shader Quantise: ● ○

Scaling: linear

Null Color: □ Hide

Position: 2007-11-01T10:39:14, 214850

Count: 2,750 / 10,800

Time Plot window with a Spectrogram layer

This control has layer-specific tabs **Data** and **Style**, described below. The **Zone** tab is described in Appendix A.4.14.1.

To control the colour map used to represent the spectral values, use the Aux fixed control.

Data Tab

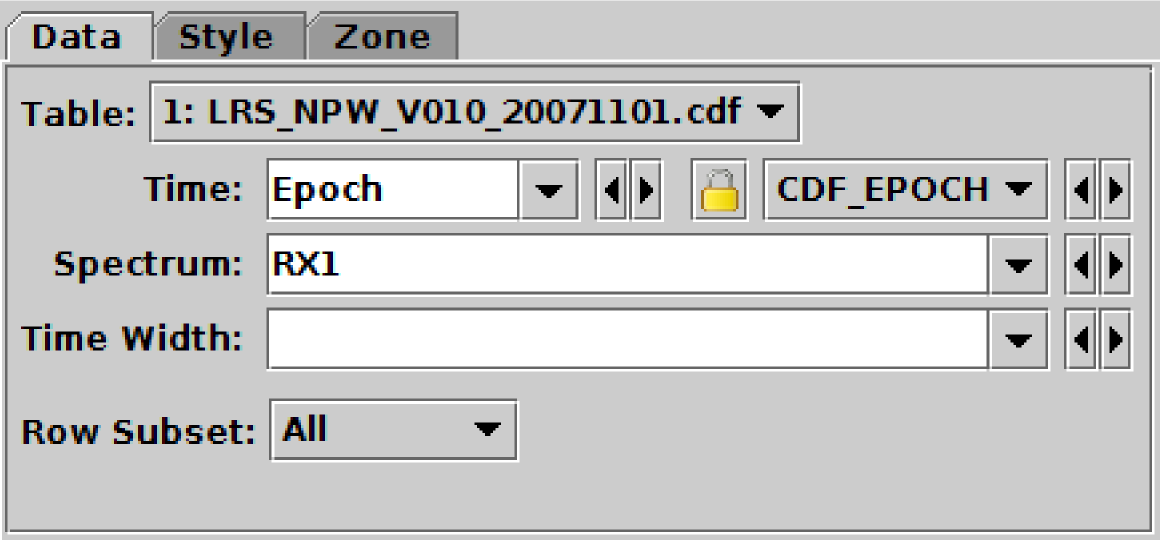


Table: 1: LRS_NPW_V010_20071101.cdf

Time: Epoch CDF_EPOCH

Spectrum: RX1

Time Width:

Row Subset: All

Spectrogram control Data tab

The **Data** tab allows you to specify which values from a table will generate a spectrogram. It has the following fields:

Table
The table supplying the data.

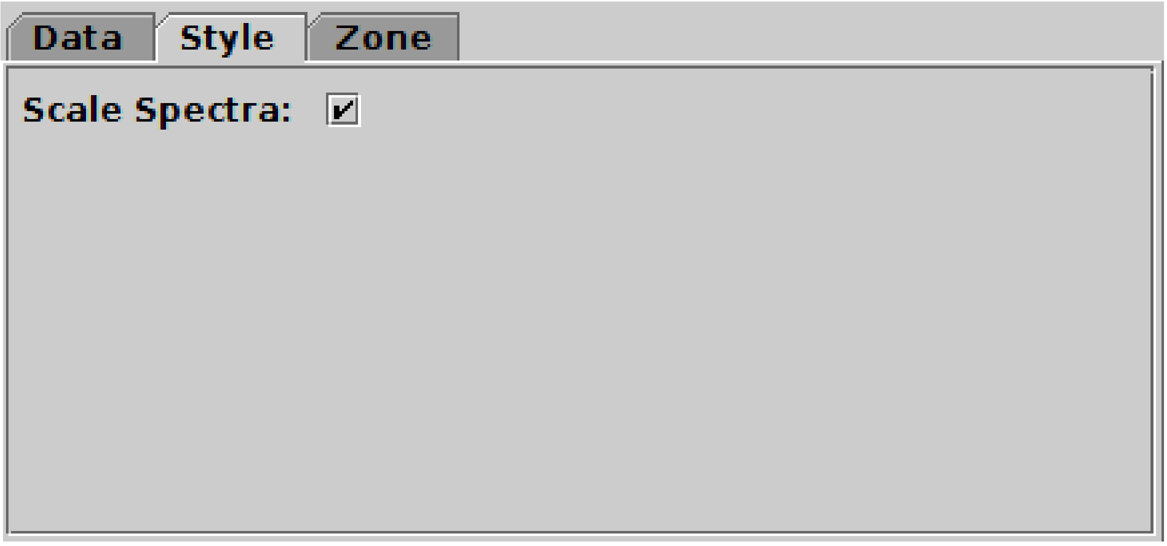
Time
A table column or expression giving the epoch coordinate at which spectra are located. This should normally be a time-typed column; if it is simply of numeric type it will be interpreted as seconds since 1 Jan 1970.

Spectrum
An array-valued table column giving the spectral data.

TimeWidth
A table column or expression (variable or constant) giving the temporal coverage of a plotted spectrum. If not filled in, it is assumed to be the most common (median) difference between time points.

Row Subset
The subset for which the spectrum should be plotted. To plot multiple subsets (not usually useful with this kind of plot) you would need multiple spectrogram layer controls in the stack.

Style Tab



Scale Spectra: ☑

Spectrogram control Style tab

The **Style** tab configures the plotting style. Options are:

Scale Spectra
: If this option is checked (the default), an attempt will be made to plot the spectra on a vertical axis that represents their physical values. This is only possible if the column or table metadata contains a suitable array that gives bin extents or central wavelengths or similar. An ad hoc search is made of column and table metadata to find an array that looks like it is intended for this purpose.

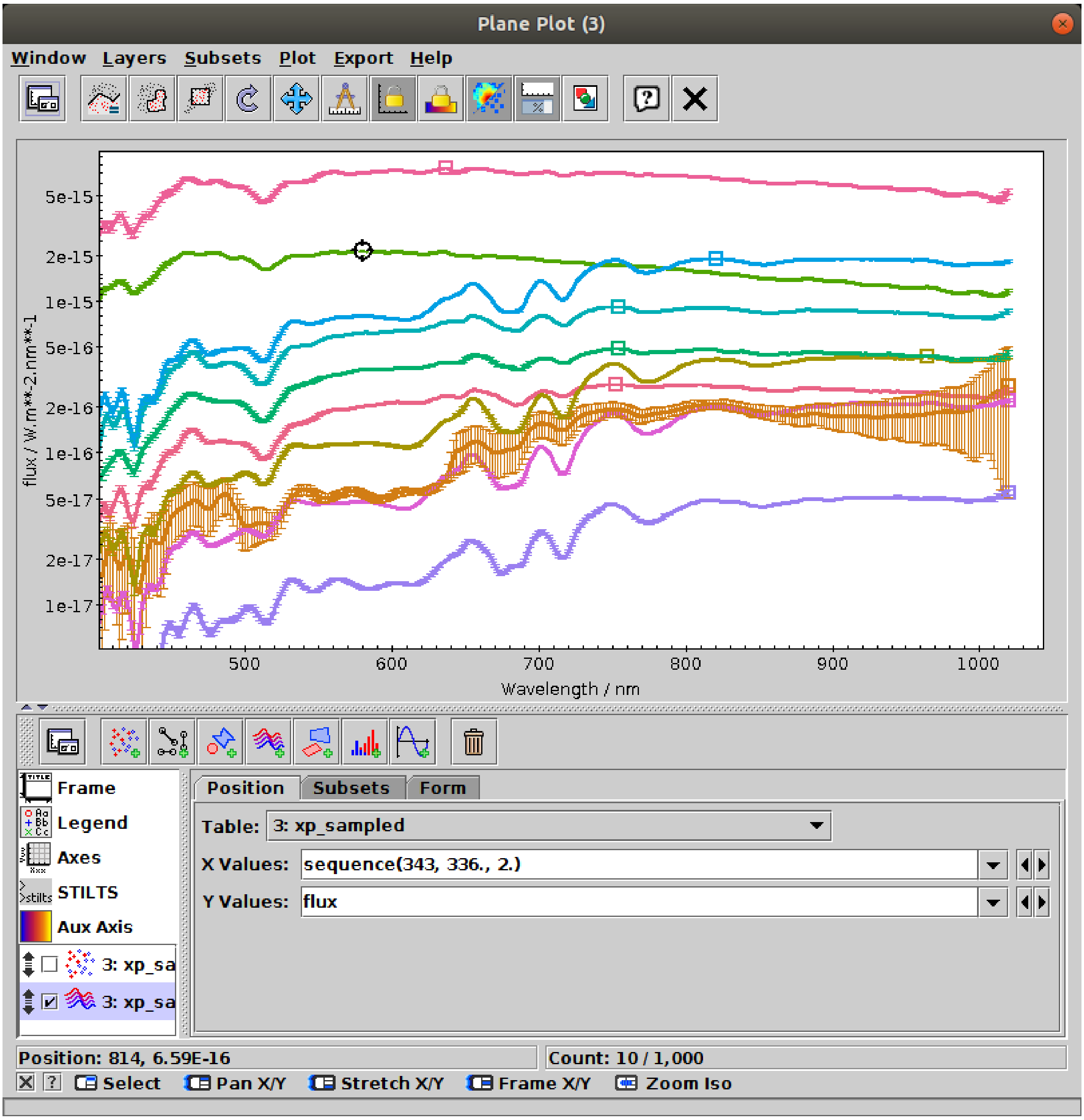
  If this option is unchecked, or if no suitable array can be found, the vertical axis just represents channel indices and so is labelled from 0 to the number of channels per spectrum.

  This configuration item is somewhat experimental; the details of how the spectral axis is configured may change in future releases.

A.4.4.9 XYArray Layer Control

The **XYArray** layer control () can be used to plot points, lines and error bars, but unlike the Position Layer Control it works with table cells whose values are arrays giving multiple coordinates, rather than single values. It is typically used where the value stored in each cell of a column is an array representing a whole spectrum or time series, or some related quantity (such as errors, frequencies, epochs etc). Each row of the table therefore corresponds to a full spectrum, timeseries, or other array of coordinate pairs.

You can add one of these controls to the Plane plot stack by using the **Add XYArray Control** () button in the control panel toolbar, or the corresponding item in the **Layers** menu. You may need to deactivate the default Position Layer Control stack entry () as well to avoid confusion and make sure the ranging works correctly. This control is only available for the Plane plot.



☐ 3: xp_sa

☑ 3: xp_sa

Table: 3: xp_sampled

X Values: sequence(343, 336., 2.)

Y Values: flux

Position: 814, 6.59E-16

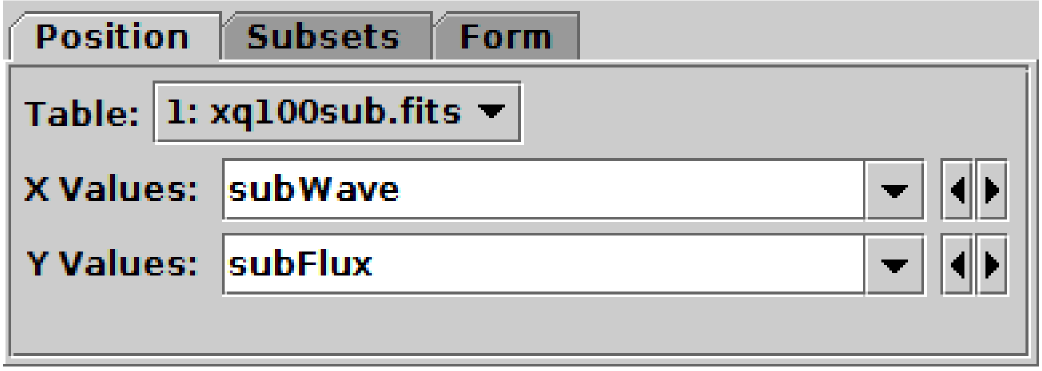
Count: 10 / 1,000

Plane Plot window with an XYArray layer

Note that unlike for instance plots of simple X,Y positions, there is no obvious single position associated with each plotted row (X/Y array pair, or wiggly line). That means that by default, you can't click on a plotted line to activate it, or see it highlighted when you activate the corresponding row by clicking on it in the Data Window. To do that, you can use the Handles Form ( ), which plots a "handle" marker at some configurable reference position near each line. That then serves as the effective position on the plot for each row for the purposes of activation, highlighting, graphical subset definition, etc. For convenience, a disabled Handles entry is added by default to the stack in the **Forms** tab, so you can enable it just by clicking on its checkbox. The handle positions and appearance can be adjusted using the configuration panel.

This control is a Table Data control as described in Appendix A.4.2.2. That section explains the **Subsets** tab; the **Position** and **Form** tabs are described in more detail below.

Position Tab



Position | Subsets | Form

Table: 1: xq100sub.fits

X Values: subWave

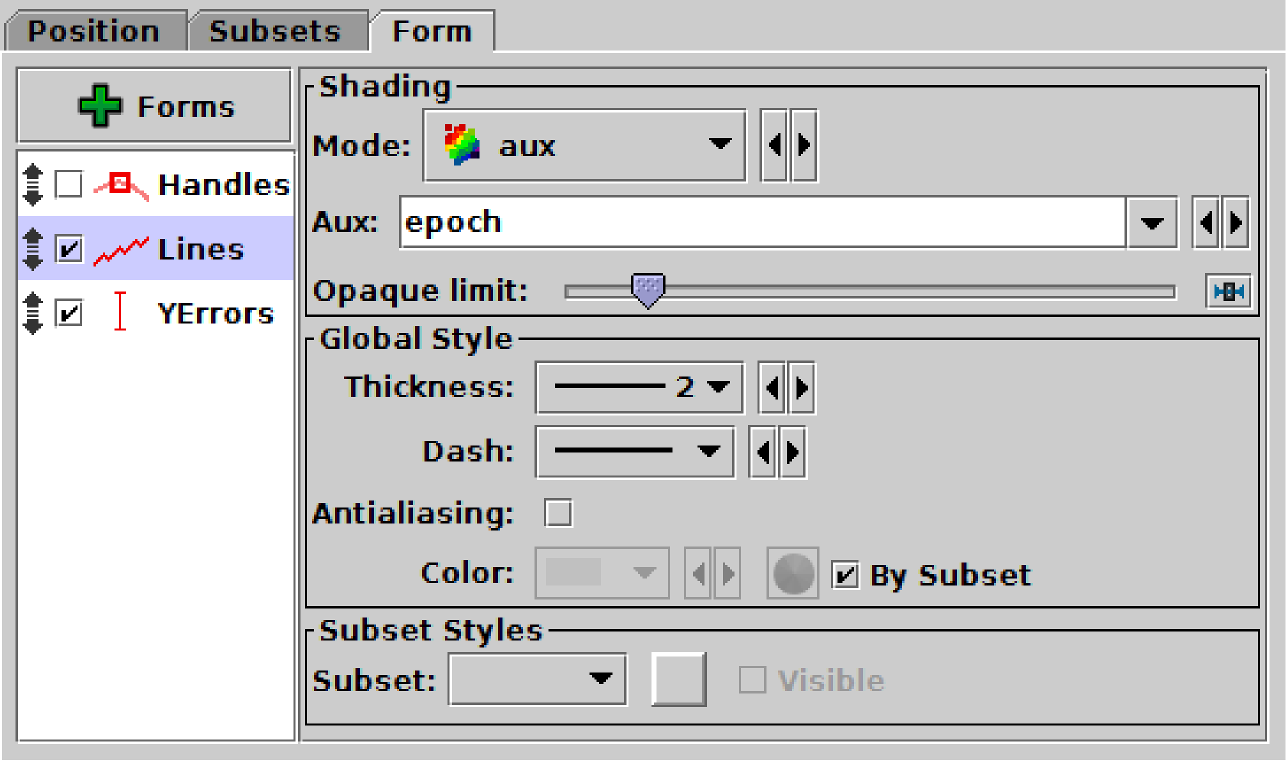
Y Values: subFlux

Position tab of XYArray layer control

In the **Position** tab you have to fill in array values for the **X Values** and **Y Values** plot coordinates. The contents can be array-valued column names or expressions. For each row, the X and Y coordinates must be arrays of the same length (if they are not, nothing will be plotted for that row), though different rows may have different array lengths. To manipulate array values, some of the expression language functions in the Arrays class may be useful; for instance the expression "`multiply(flux,1./mean(flux))`" can be used to normalise an input `flux` array. For most of the available plots, if either the X or Y array value is missing, it will be interpreted as a sequence (0,1,2,...) of the same length as the array that is present.

TOPCAT will try to fill in the X and Y coordinates with suitable array values from the table, if it can find some. As well as array-valued columns, array-valued table parameters can be used, and are often suitable for the X axis; these can be referred to using the syntax "`param$<name>`", as explained in Section 7.3. It helps if columns are declared with fixed array sizes, where applicable.

Form Tab



Position | Subsets | Form

Forms

□ Handles

☑ Lines

☑ YErrors

Shading

Mode: aux

Aux: epoch

Opaque limit:

Global Style

Thickness: 2

Dash:

Antialiasing: □

Color: ☑ By Subset

Subset Styles

Subset:

□ Visible

Form tab of XYArray layer control

The **Form** tab lets you define how the specified data set is plotted. The list on the left gives a list of forms currently being plotted, and the panel on the right shows the detailed configuration for the currently selected form.

The available forms for an XYArray plot (select them using the **Forms** button) are currently

- Lines: draws a line between the X,Y positions
- Marks: draws a marker for each X,Y position
- Handles: draws a single marker at a reference position for the whole array of X,Y pairs
- YErrors: draws a vertical error bar at each X,Y position
- XYErrors: draws horizontal and/or vertical error bars at each X,Y position
- StatLine: draws a single line giving per-element combinations (e.g. mean) of the input array values
- StatMark: draws a marker for each per-element combination (e.g. mean) of the input array values
- ArrayQuantile: draws quantile ranges for samples from the input array values

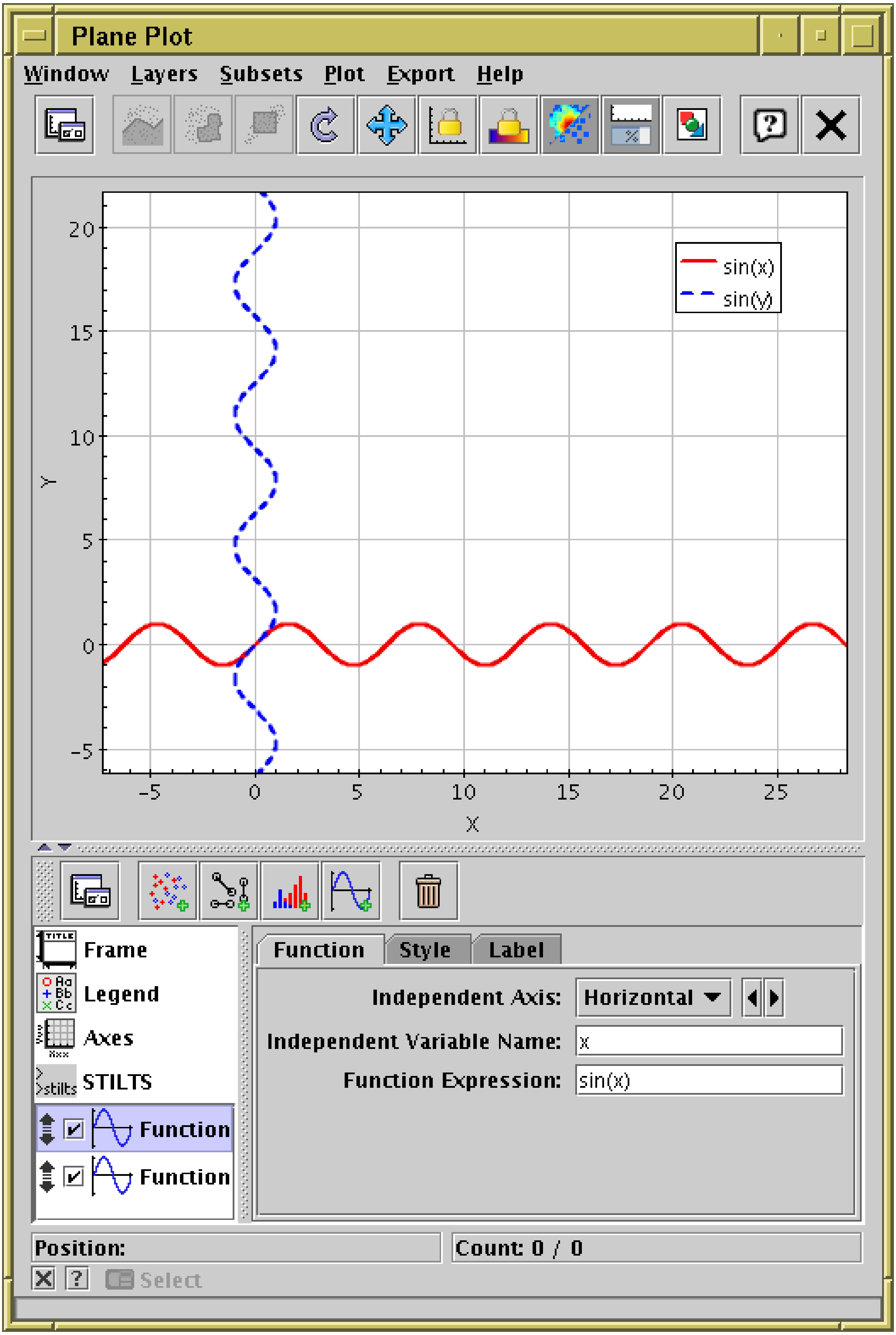
You can have zero or more of each form and configure them separately. The detail panel for these forms are dependent on the form itself and is described in more detail in Appendix A.4.5, but the detail panels are divided into these parts:

Shading
The shading mode controls how points are shaded based on their chosen colour. The various options are described in Appendix A.4.6. Depending on the mode there may be more settings to fill in here.

Global/Subset Styles
Controls the style details for the chosen form; see Appendix A.4.2.2.

A.4.4.10 Function Layer Control

The **Function** layer control ( ) is only available for the Plane, Histogram and Time plots. You can add one of these controls to the stack by using the **Add Function Control** ( ) button in the control panel toolbar, or the corresponding item in the **Layers** menu.



Independent Axis: Horizontal

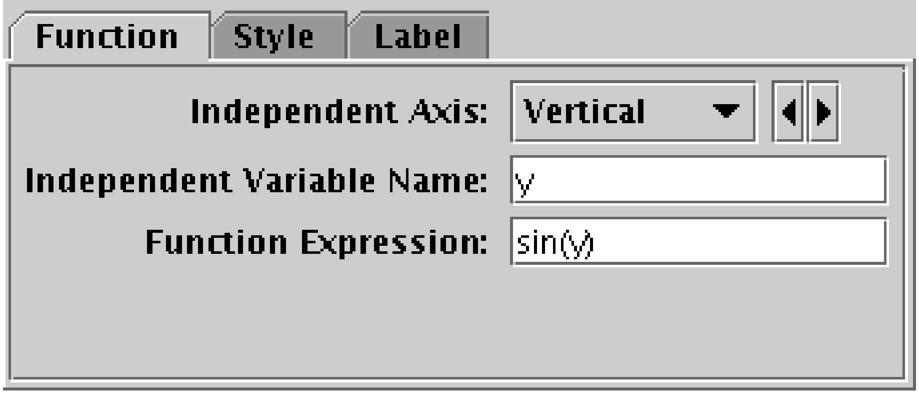
Independent Variable Name: x

Function Expression: sin(x)

Plane plot with two function layers plotted, one as a function of Horizontal axis value, and one as a function of Vertical axis value

This control has three tabs, **Function**, **Style** and **Label**, described below.

Function Tab



Independent Axis: Vertical

Independent Variable Name: y

Function Expression: sin(y)

Function control Function tab

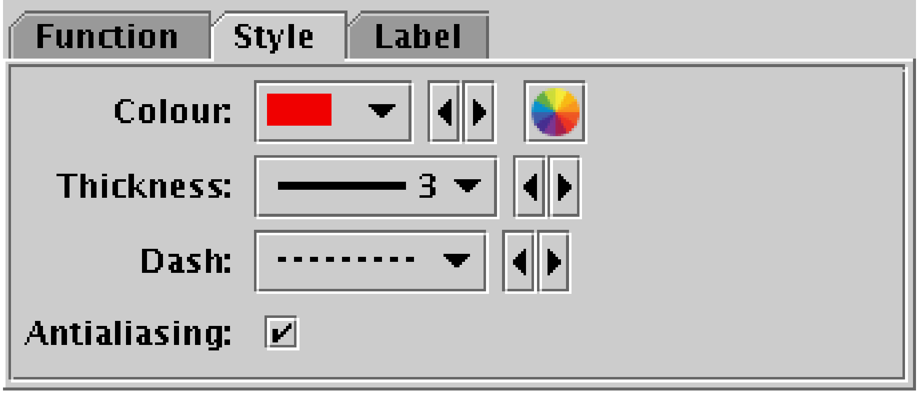
The **Function** tab defines the function to be plotted and has the following fields:

Independent Axis
: Which axis the independent variable varies along; options are currently **Horizontal** and **Vertical**.

Independent Variable Name
: Name of the independent variable. This is typically `x` for a horizontal independent variable and `y` for a vertical independent variable, but any string that is a legal expression language identifier (starts with a letter, continues with letters, numbers, underscores) can be used.

Function Expression
: An expression using TOPCAT's expression language in terms of the independent variable that defines the function. This expression must be standalone - it cannot reference any tables.

Style Tab



Colour: 🟥

Thickness: 3

Dash: ---------

Antialiasing: ☑

Function control Style tab

The **Style** tab configures the plotting style. Options are:

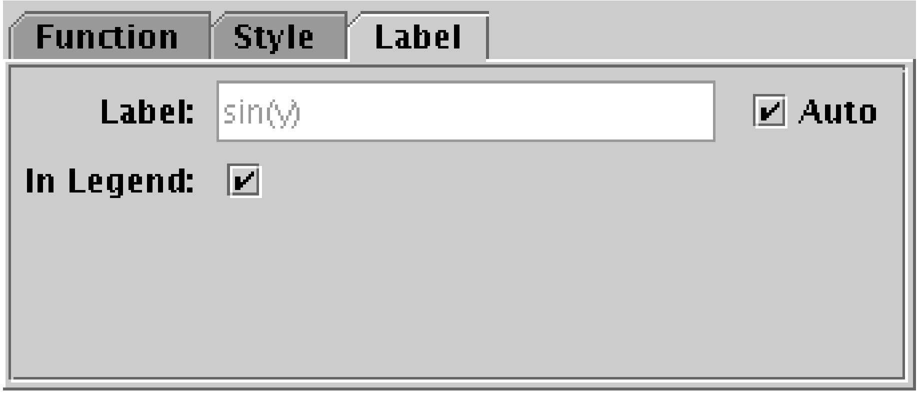
Colour
: Colour of the line.

Thickness
: Thickness of the line in pixels.

Dash
Dash pattern of the line - solid is the default, but various options are available.

Antialiasing
If true, lines are antialiased, which makes them look smoother on the screen or bitmapped export images. Has no effect on vector export images (PDF, SVG, EPS).

Label Tab



Label: sin(y) ☑ Auto

In Legend: ☑

Function control Label tab

The **Label** tab allows you to choose the text that appears in the legend. Options are:

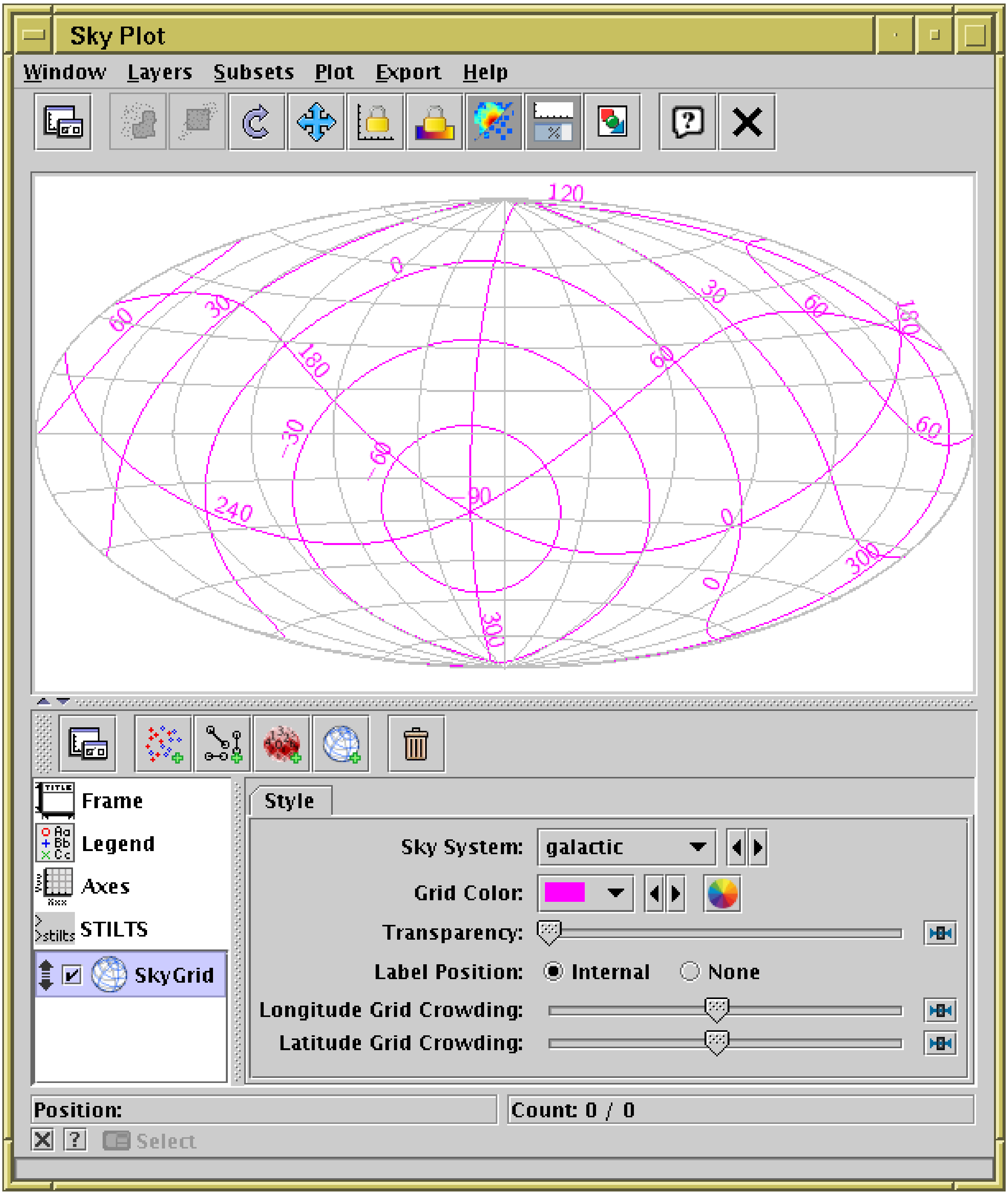
Label
Gives the label that will appear in the legend. By default the function expression is used, but if you want to override this you can deselect the associated **Auto** checkbox and enter your own value.

In Legend
If true, an entry for this function appears in the legend, if false it does not. Note the setting of this value does not affect whether the legend itself appears, which is controlled by the Legend control.

A.4.4.11 SkyGrid Layer Control

The **SkyGrid** layer control (🌐) is available from the Sky Plot Window, and plots an additional axis grid on the celestial sphere. This can be overlaid on the default sky axis grid so that axes for multiple sky coordinate systems are simultaneously visible. You can add one of these controls to the SkyPlot stack by using the **Add SkyGrid Control** (🌐) button in the control panel toolbar, or the corresponding item in the **Layers** menu.

Note that some of the configuration items for this plot layer, such as grid line antialiasing and the decimal/sexagesimal flag, are inherited from the values set for the main sky plot grid in the Sky Axes Control.



Sky System: galactic

Grid Color:

Transparency:

Label Position: ● Internal ○ None

Longitude Grid Crowding:

Latitude Grid Crowding:

Position:

Count: 0 / 0

Sky plot with no data, but an additional Galactic sky axis grid plotted over the default equatorial grid.

This control has only one tab, **Style**, with the following fields:

Sky System
: Selects the sky system (Equatorial, Galactic, Supergalactic or Ecliptic) whose coordinates the plotted grid lines should represent. Note this is assessed relative to the **View Sky System** selected in the Sky Axes Control. If the value here is the same as the View Sky System (by default Equatorial) the grid used by this layer will be the same as the default axis grid.

Grid Color

Selects the colour with which grid lines will be drawn.

Transparency
Determines how transparent the grid lines will be.

Label Position
Determines whether and how the numeric labels will be placed along the grid lines. Currently only two options, **Internal** and **None**.

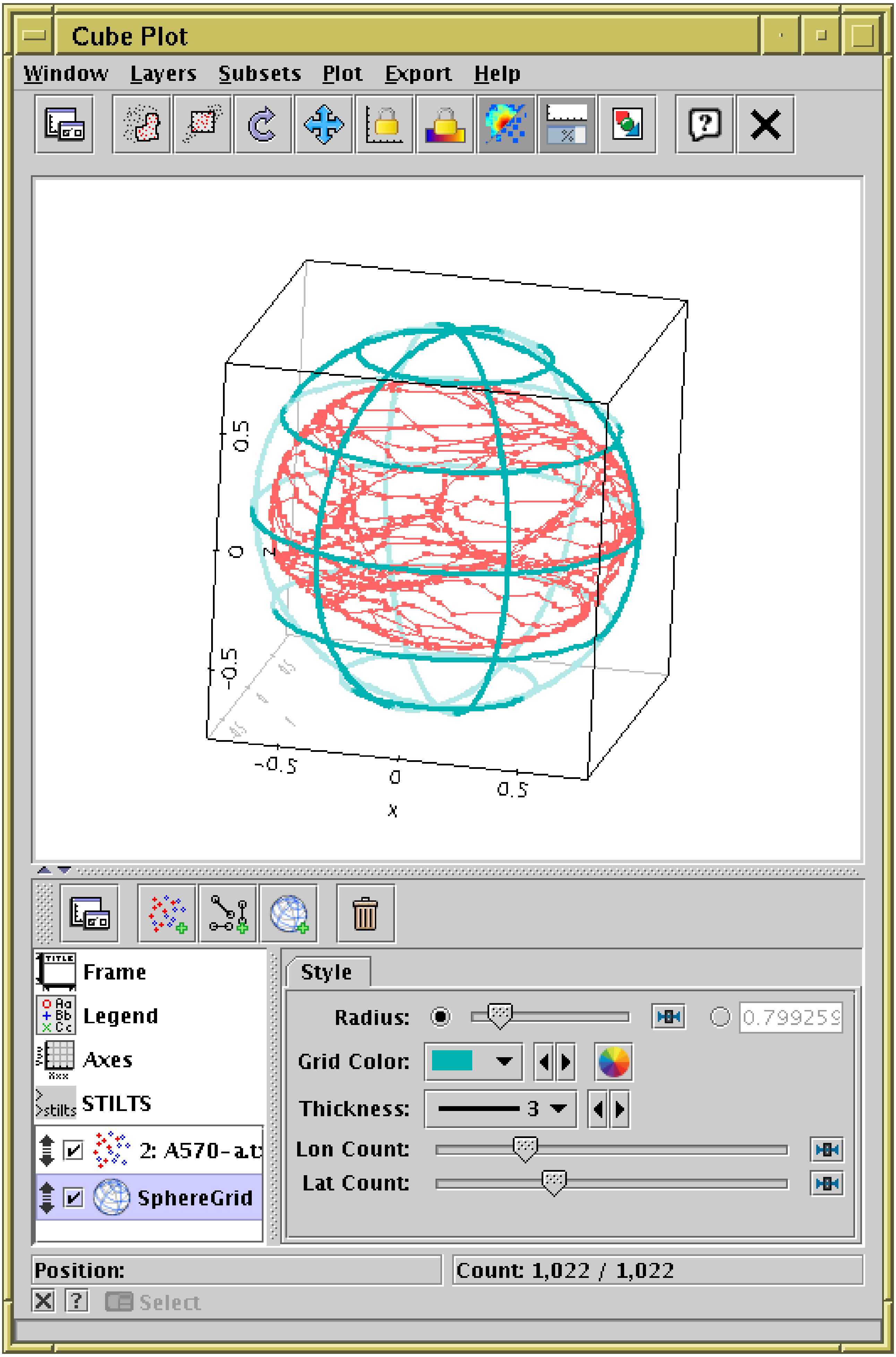
Longitude Grid Crowding
Use the slider to control how closely longitudinal grid lines (meridians) are packed.

Latitude Grid Crowding
Use the slider to control how closely latitudinal grid lines (parallels) are packed.

A.4.4.12 SphereGrid Layer Control

The **SphereGrid** layer control (🌐) plots a spherical net centered on the coordinate origin. It is available from the Sphere and Cube plot windows, and you can add one of these controls to the stack by using the **Add SphereGrid Control** (🌐) button in the control panel toolbar, or the corresponding item in the **Layers** menu.

The plotted grid is currently not labelled, but it can provide orientation to make clear for instance the position of a bounding unit sphere. Its radius can be controlled by a slider or by entering a fixed value. Since the sphere is always centered on the origin, if the whole visible cube is far from the origin, it may be that few or none of the grid lines show up on the plot.



☑ 2: A570-a.t

☑ SphereGrid

Radius: ● ○ 0.799259

Grid Color:

Thickness: 3

Lon Count:

Lat Count:

Position:

Count: 1,022 / 1,022

Plot of the XYZ vertices providing surface of an asteroid (48 Doris) with a spherical grid overplotted.

This control has only one tab, **Style**, with the following fields:

Radius
Defines the radius of the plotted sphere. The slider adjusts the radius as a proportion of the visible plot cube, and the text entry field allows you to enter the radius in data units.

Grid Color
Colour of the plotted grid lines.

Thickness
Thickness of the plotted grid lines.

Lon Count
Determines the number of lines of longitude drawn.

Lat Count
Determines the number of lines of latitude drawn.

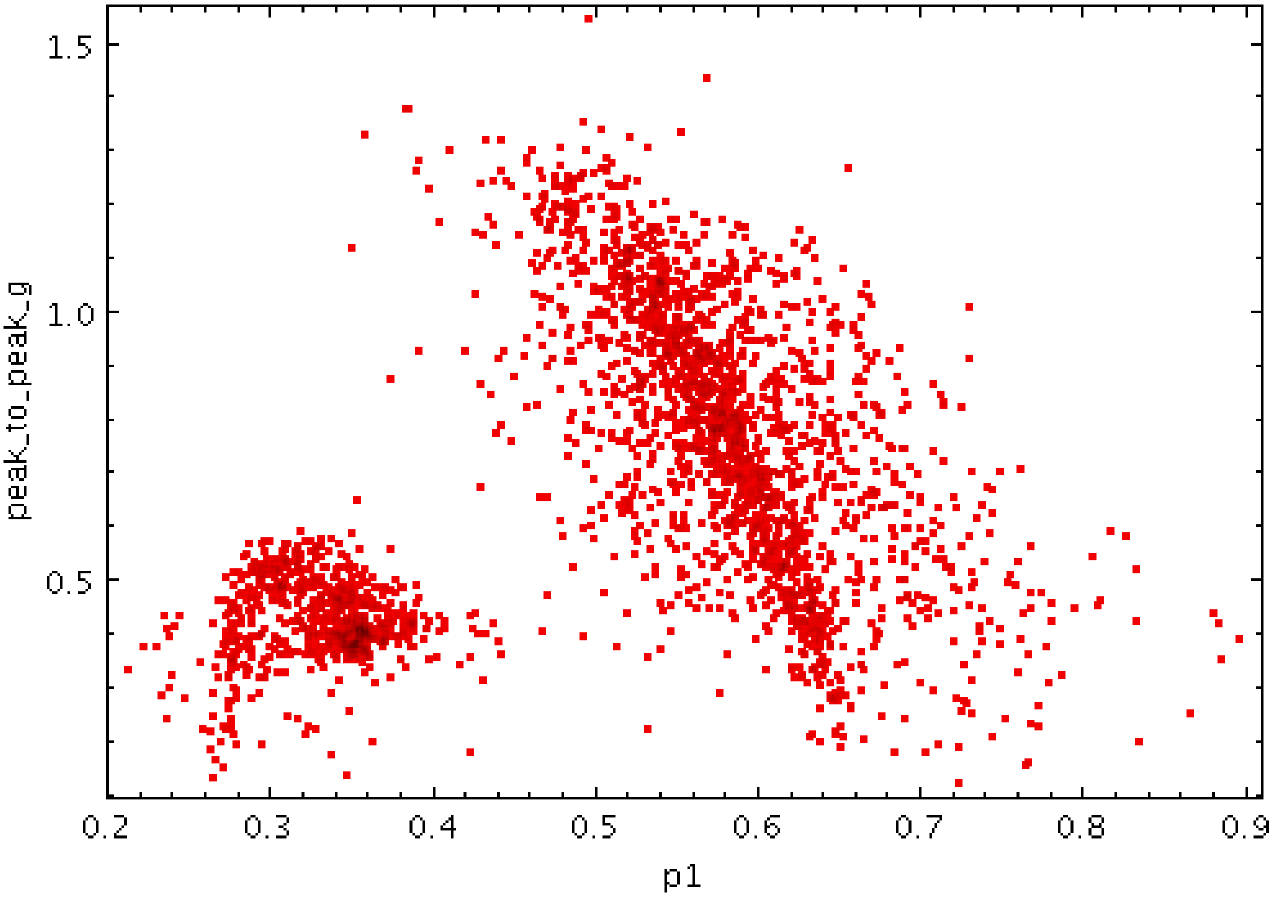
A.4.5 Plot Forms

Plot "forms" are the instructions for what shapes to actually paint on the plotting surface given some positional data from a table. The most obvious (Mark) is simply to plot a marker with a fixed size and shape at each data position. However, there is a range of other things you might want to do, including error bars, vectors, contours, text labels, ...

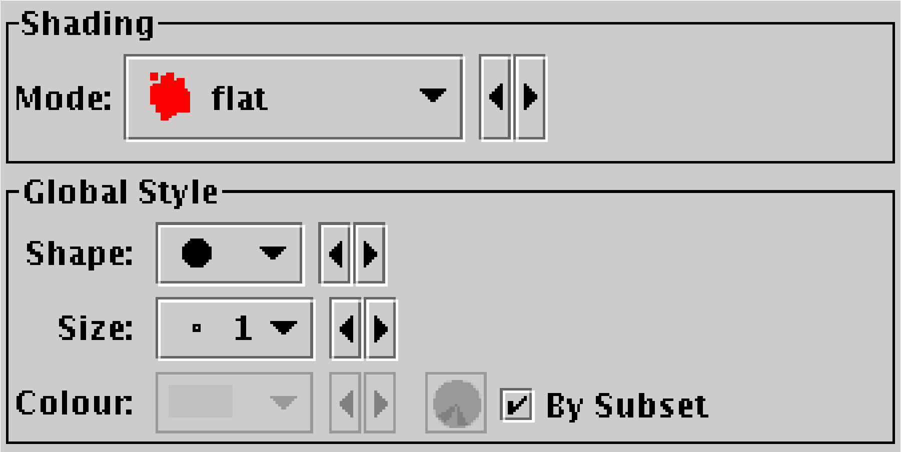
The options provided are described, with the additional configuration information required, in the following sections. Not all are available for every plot type.

A.4.5.1 Mark Form

The **Mark** form (✚) simply plots markers with a fixed shape and size.



Example Mark plot


Shading

Mode: flat

Global Style

Shape: ●

Size: ▪ 1

Colour:

☑ By Subset

Mark form configuration panel

Configuration options are:

Shading Mode
See Appendix A.4.6.

Shape
Marker shape from a list of options.

Size
Marker size in pixels.

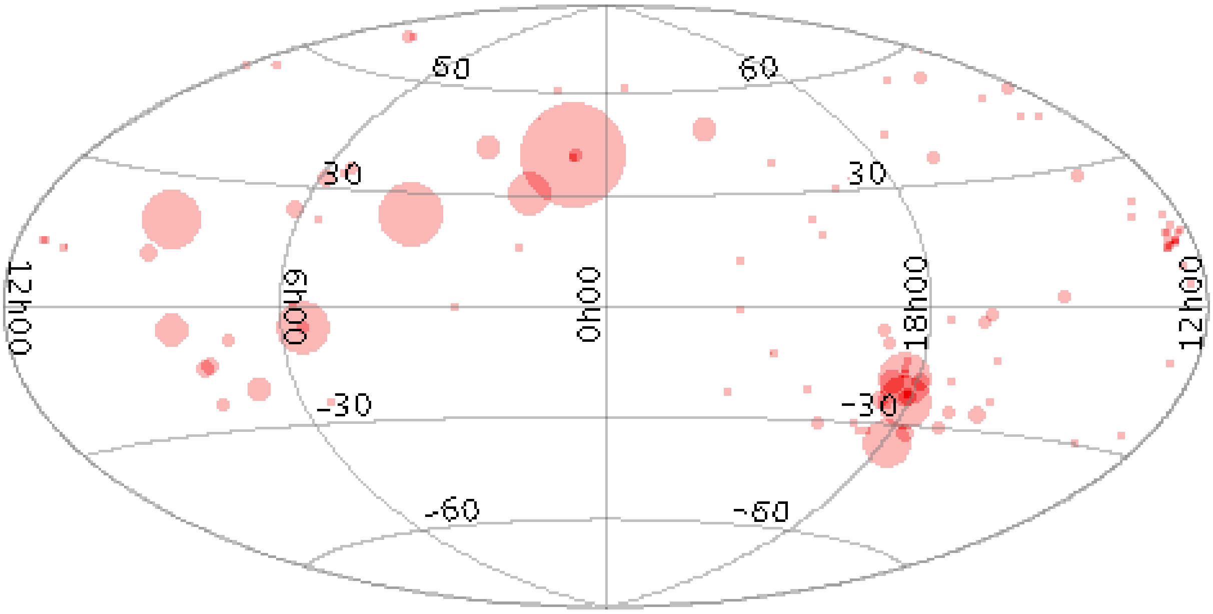
A.4.5.2 Size Form

The **Size** form (□) plots markers of a fixed shape whose size varies according to a supplied data coordinate. The marker size thus represents an additional dimension of the plot.

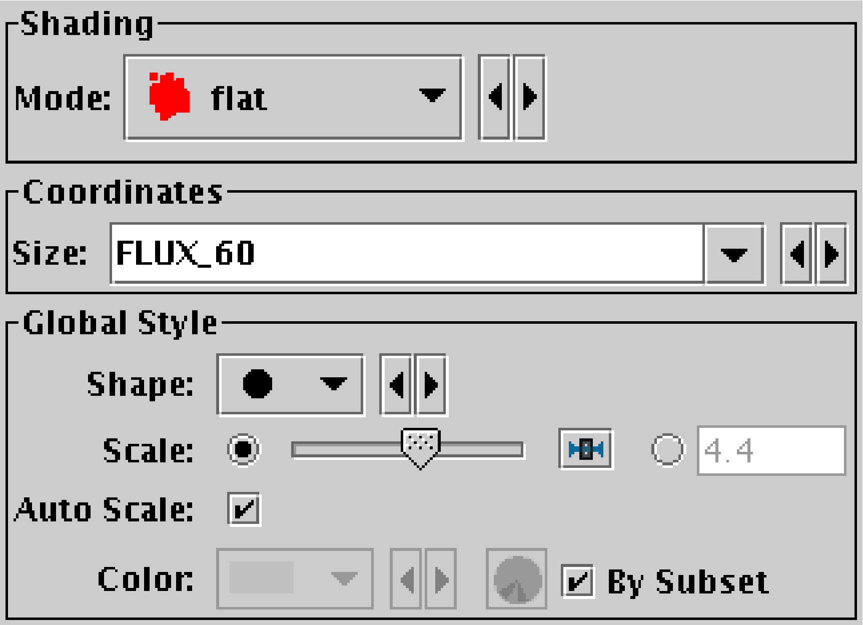
The actual size of the markers depends on the setting of the **Auto Scale** option. If autoscaling is off, then the basic size of each marker is the input data value in units of pixels. If autoscaling is on, then the data values are gathered for all the currently visible points, and a scaling factor is applied so that the largest ones will be a sensible size (a few tens of pixels). This basic size can be further adjusted with the **Scale** slider.

Currently data values of zero always correspond to marker size of zero, negative data values are not represented, and the mapping is linear. An absolute maximum size on markers is also imposed. Other options may be introduced in future.

Note the scaling to size is in terms of screen dimensions (pixels). For sizes that correspond to actual data values, the Error form may be more appropriate.



Example Size plot



Shading

Mode: flat

Coordinates

Size: FLUX_60

Global Style

Shape: ●

Scale: ● ○ 4.4

Auto Scale: ☑

Color: ☑ By Subset

Size form configuration panel

The configuration options are:

Shading Mode
See Appendix A.4.6.

Size
The size coordinate data values. Fill this in with a column name or expression from the table just like for the positional coordinates. The units are either pixels or arbitrary, according to the **Auto Scale** setting.

Shape
Marker shape from a list of options.

Scale
Scales the size of variable-sized markers. Adjusting the slider will make all markers larger or smaller. Alternatively you can enter a scale factor in the text field.

Auto Scale
Determines whether the basic size of variable sized markers is automatically scaled to have a sensible size. If checked, then the sizes of all the plotted markers are examined, and some dynamically calculated factor is applied to them all to make them a sensible size (by default, the largest ones will be a few tens of pixels). If unchecked, the sizes will be the actual input values in units of pixels.

If auto-scaling is off, then markers will keep exactly the same screen size during pan and zoom operations; if it's on, then the visible sizes will change according to what other points are currently plotted.

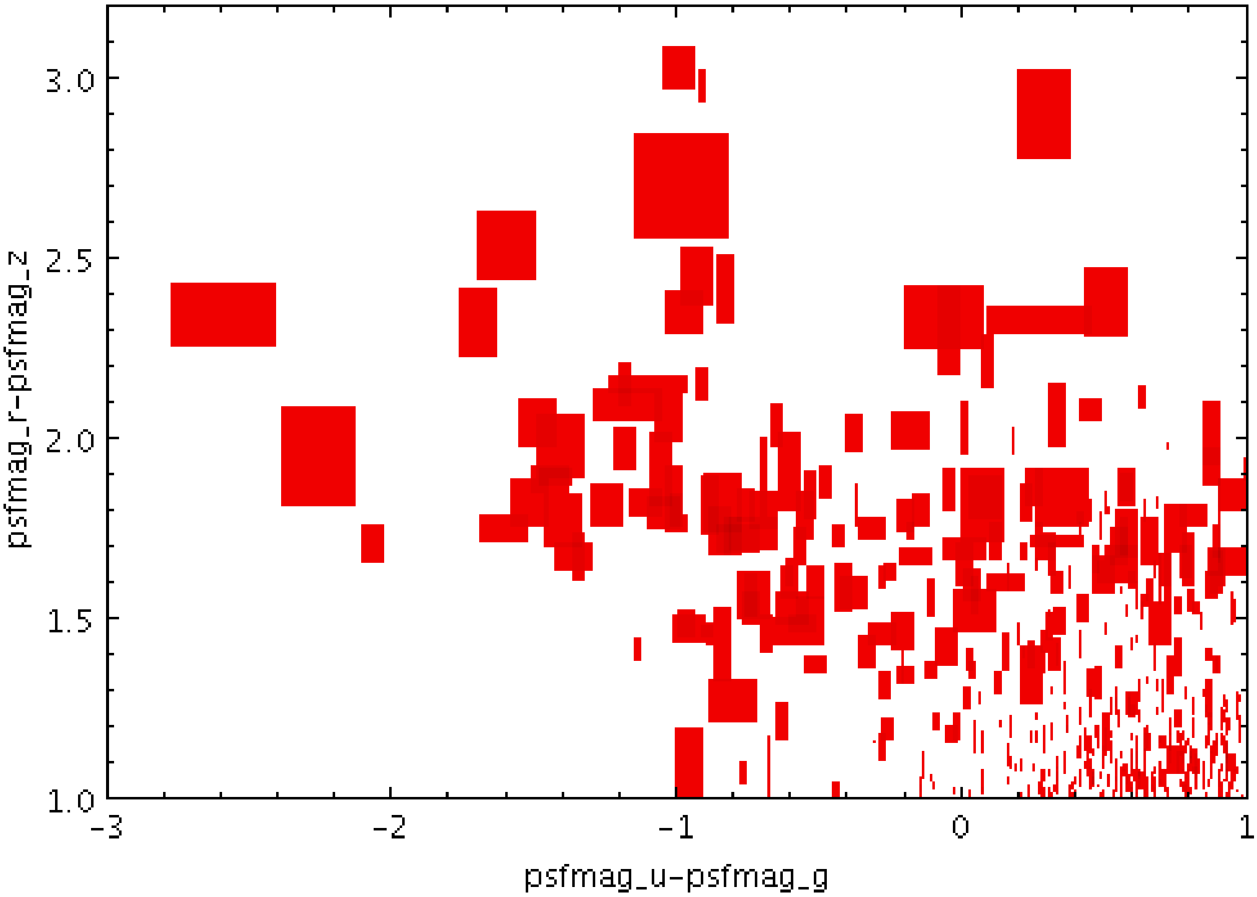
A.4.5.3 SizeXY Form

The **SizeXY** form () plots a shaped marker whose width and height vary independently acoording to two supplied data coordinates. The marker shape can thus encode two additional dimensions of the plot.

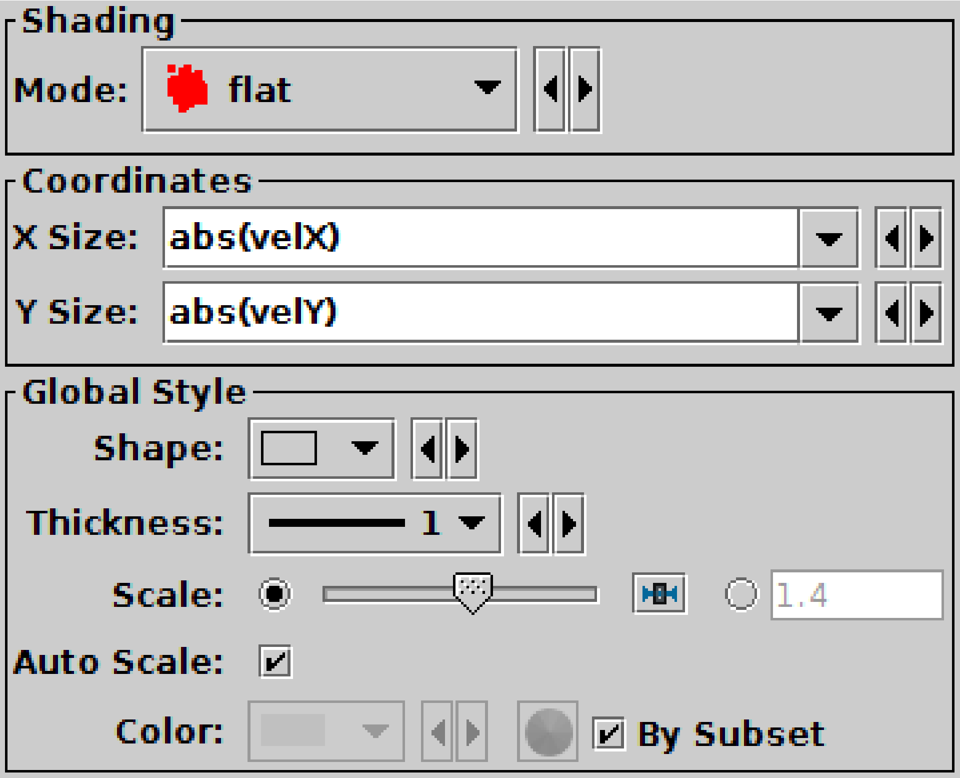
The actual size of the markers depends on the setting of the **Auto Scale** option. If autoscaling is off, the basic dimensions of each marker are given by the input data values in units of pixels. If autoscaling is on, the data values are gathered for all the currently visible points, and scaling factors are applied so that the largest ones will be a sensible size (a few tens of pixels). This autoscaling happens independently for the X and Y directions. The basic sizes can be further adjusted with the **Scale** slider.

Currently data values of zero always correspond to marker extent of zero, negative data values are not represented, and the mapping is linear. An absolute maximum size on markers is also imposed. Other options may be introduced in future.

Note the scaling to size is in terms of screen dimensions (pixels). For sizes that correspond to actual data values, the Error form may be more appropriate.



Example SizeXY plot



Mode: flat

X Size: abs(velX)

Y Size: abs(velY)

Shape: □

Thickness: 1

Scale: ● ○ 1.4

Auto Scale: ☑

Color: ☑ By Subset

SizeXY form configuration panel

The configuration options are:

Shading Mode

See Appendix A.4.6.

X/Y Size

For width and height respectively, the size coordinate data values. Fill in with a column name or expression from the table just like for the positional coordinates. The units are either pixels or arbitrary, according to the **Auto Scale** setting.

Shape

Marker shape from a list of options.

Thickness

Controls the line thickness used when drawing shapes. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled rectangles contain no line drawings.

Scale

Scales the size of variable-sized markers. Adjusting the slider will make all markers larger or smaller. Alternatively you can enter a scale factor in the text field.

Auto Scale

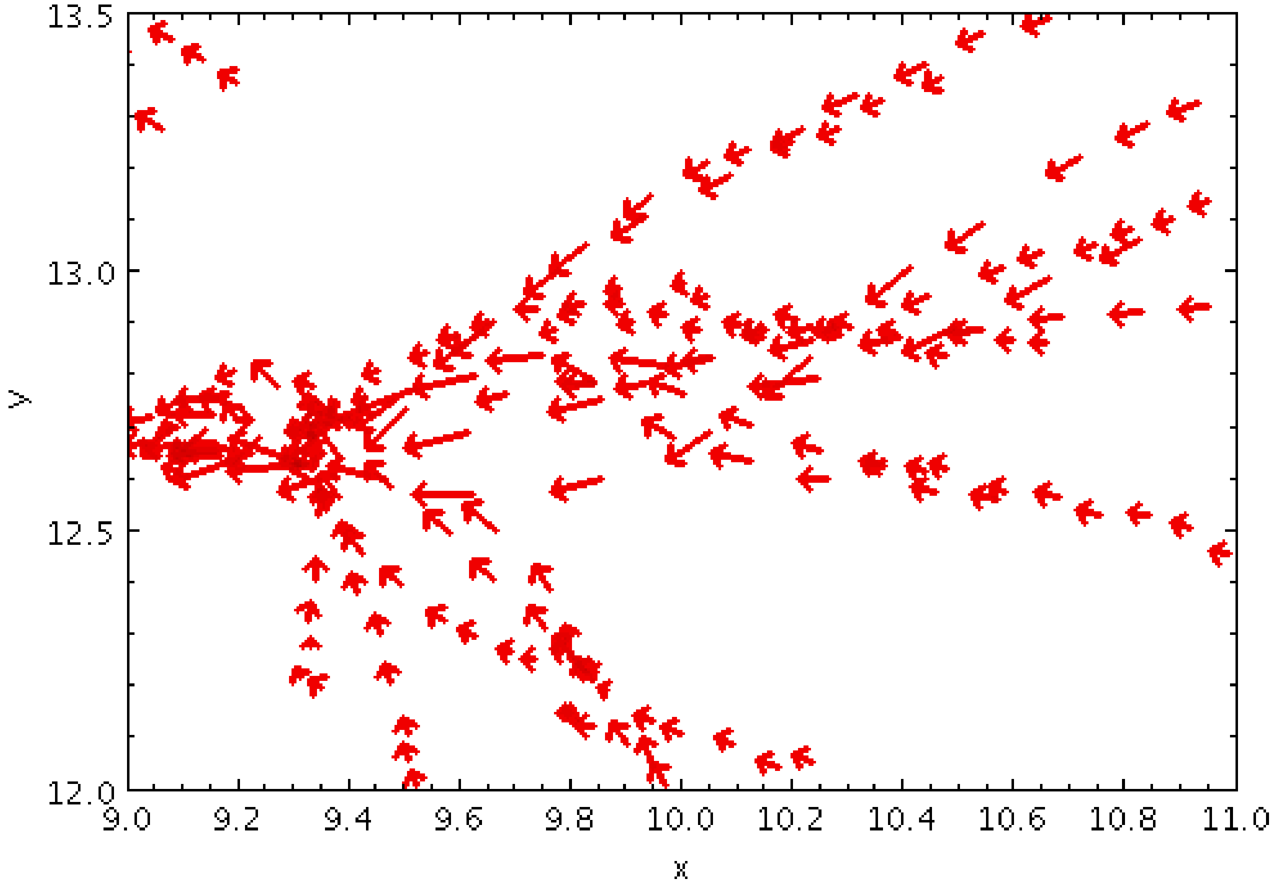
Determines whether the basic size of variable sized markers is automatically scaled to have a sensible size. If checked, then the sizes of all the plotted markers are examined, and some dynamically calculated factor is applied to them all to make them a sensible size (by default, the largest ones will be a few tens of pixels). If unchecked, the sizes will be the actual input values in units of pixels.

If auto-scaling is off, then markers will keep exactly the same screen size during pan and zoom operations; if it's on, then the visible sizes will change according to what other points are currently plotted.

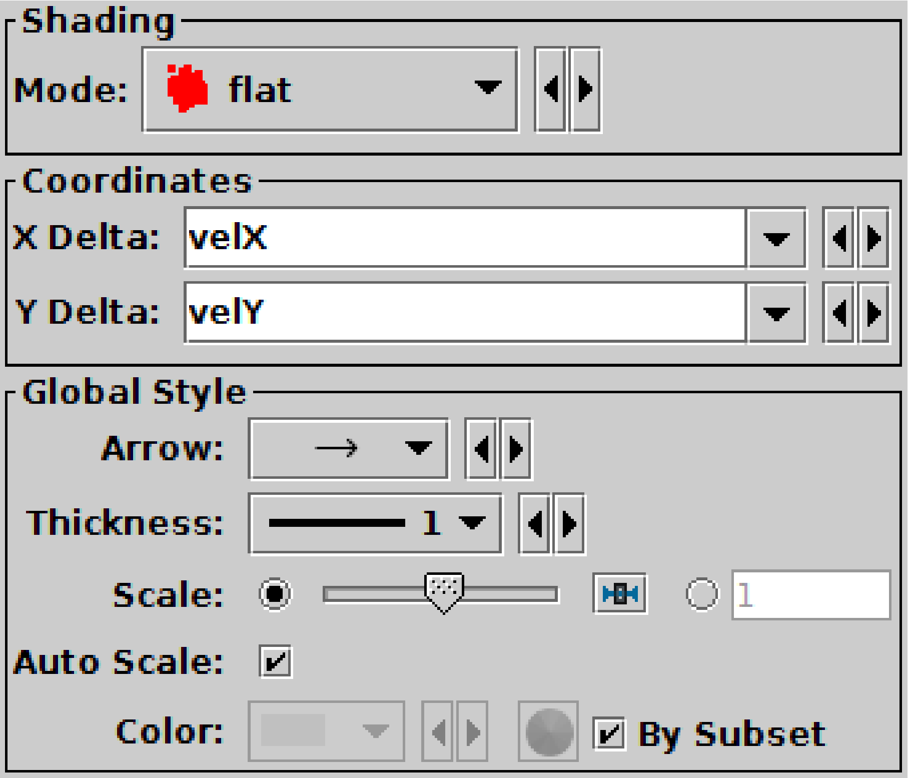
A.4.5.4 Vector Form

The **Vector** form (↗) plots directed lines from the data position given delta values for the coordinates. The plotted markers are typically little arrows, but there are other options.

In some cases such delta values may be the actual magnitude required for the plot, but often the vector data represents a value which has a different magnitude or is in different units to the positional data. As a convenience for this case, the plotter can optionally scale the magnitudes of all the vectors to make them a sensible size (so the largest ones are a few tens of pixels long). This scaling can be adjusted or turned off using the **Scale** and **Auto Scale** options below.



Example XYVector plot


Shading

Mode: flat

Coordinates

X Delta: velX

Y Delta: velY

Global Style

Arrow: →

Thickness: 1

Scale: ● ○ 1

Auto Scale: ☑

Color: ☑ By Subset

Vector form configuration panel

The configuration options are:

Shading Mode

See Appendix A.4.6.

X Delta, Y Delta, ...

The coordinates of the changes in each coordinate which gives the vector. The coordinates here match the coordinates of the plot.

Arrow

Arrow shape selected from a range of options.

Thickness

Controls the line thickness used when drawing arrows. Zero, the default value, means a 1-pixel-wide line is used, and larger values make drawn lines thicker.

Scale

Changes the factor by which all vector sizes are scaled. If the arrows are too small, slide it right, if they are too big, slide it left. The slider scale is logarithmic. Alternatively, enter a fixed value in the text field.

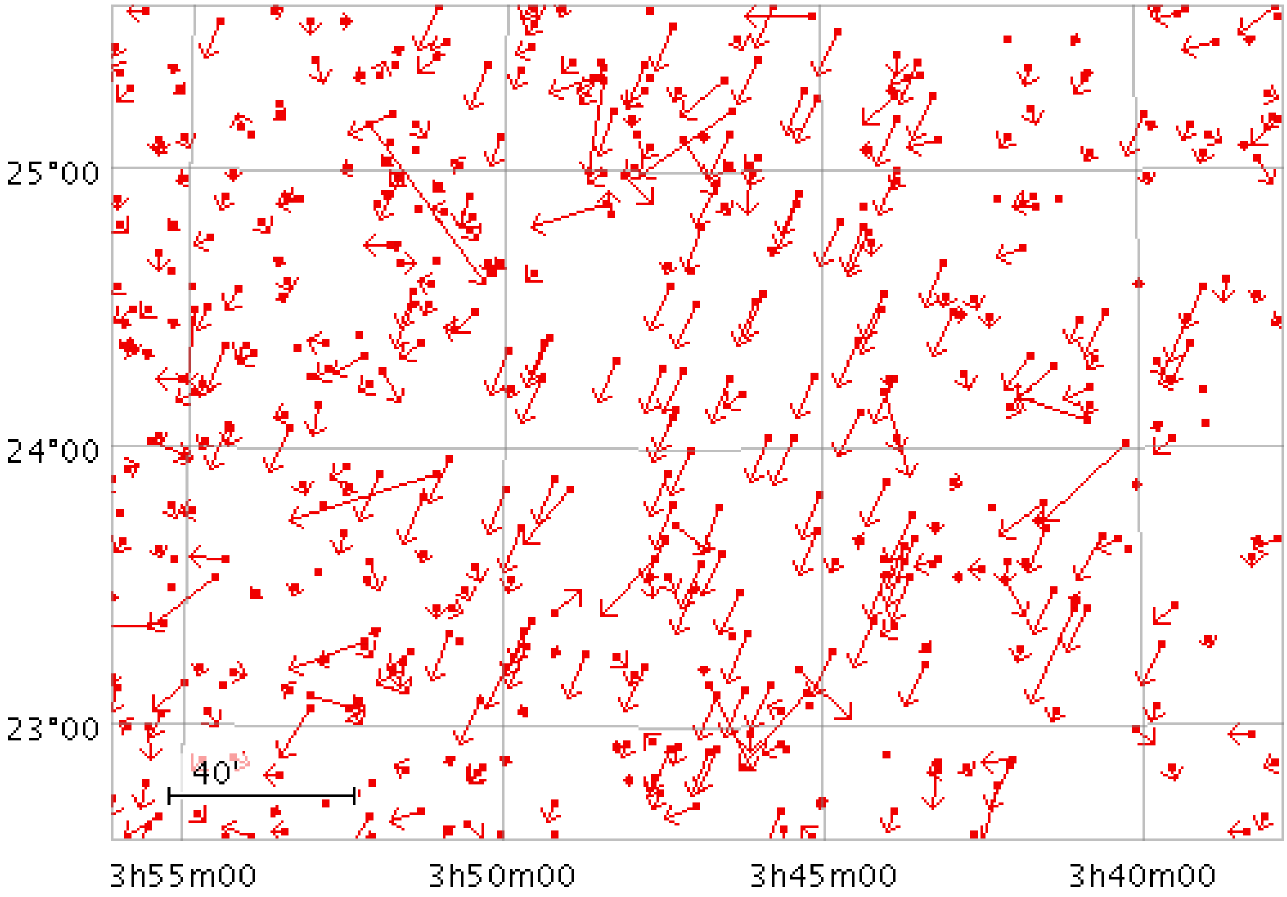
Auto Scale

If selected, this option will determine the default arrow scale size from the data - it will fix it so that the largest arrows are a few tens of pixels long by default. That scaling can then be adjusted using the **Scale** slider. If unselected, then the default position of the Scale slider corresponds to the actual positions given by the submitted delta coordinates.

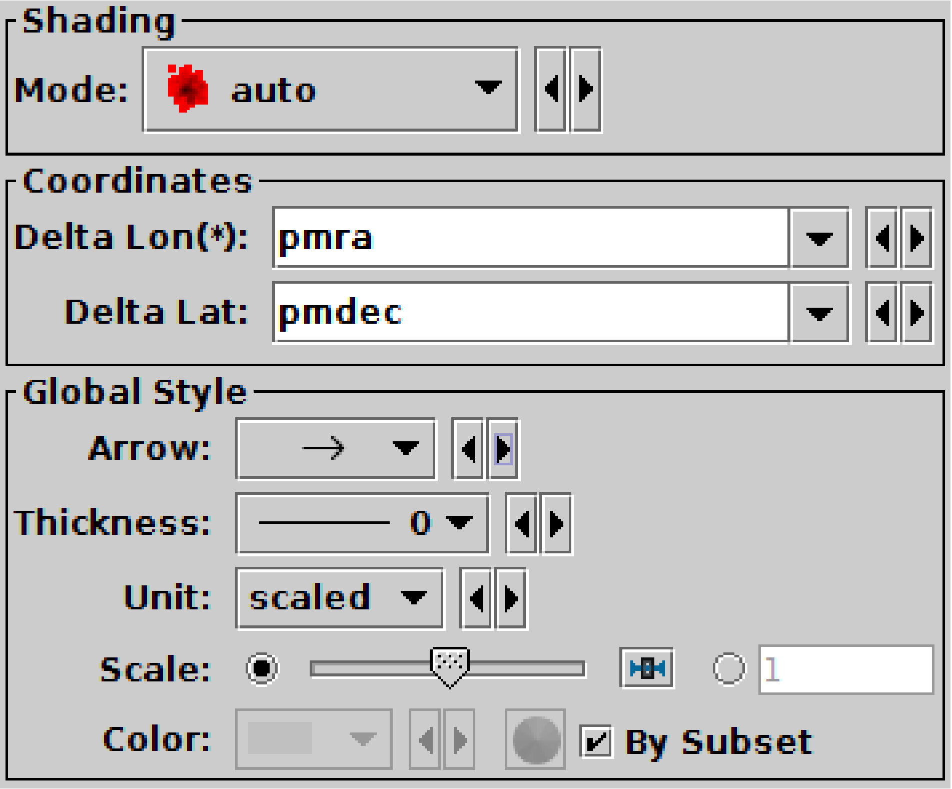
A.4.5.5 SkyVector Form

The **SkyVector** form (↗) plots directed lines from the data position given relative offsets to longitude.cos(latitude) and latitude on the celestial sphere. The plotted markers are typically little arrows, but there are other options.

In some cases such delta values may be the actual magnitude required for the plot, but often the vector data represents a value which has a different magnitude or is in different units to the positional data (for instance proper motions). As a convenience for this case, the plotter can optionally scale the magnitudes of all the vectors to make them a sensible size (so the largest ones are a few tens of pixels long). This scaling can be adjusted or turned off using the **Scale** and **Auto Scale** options below.



Example SkyVector plot



Shading

Mode: auto

Coordinates

Delta Lon(*): pmra

Delta Lat: pmdec

Global Style

Arrow:

Thickness: 0

Unit: scaled

Scale: ● ○ 1

Color: ☑ By Subset

SkyVector form configuration panel

The configuration options are:

Shading Mode

See Appendix A.4.6.

Delta Lon(*)
Change in the longitude coordinate represented by the plotted vector. The supplied value is (if not auto-scaled) an angle in degrees, and **is** considered to be premultiplied by cos(Latitude).

Delta Lat
Change in the latitude coordinate represented by the plotted vector. The supplied value is (if not auto-scaled) an angle in degrees.

Arrow
Arrow shape selected from a range of options.

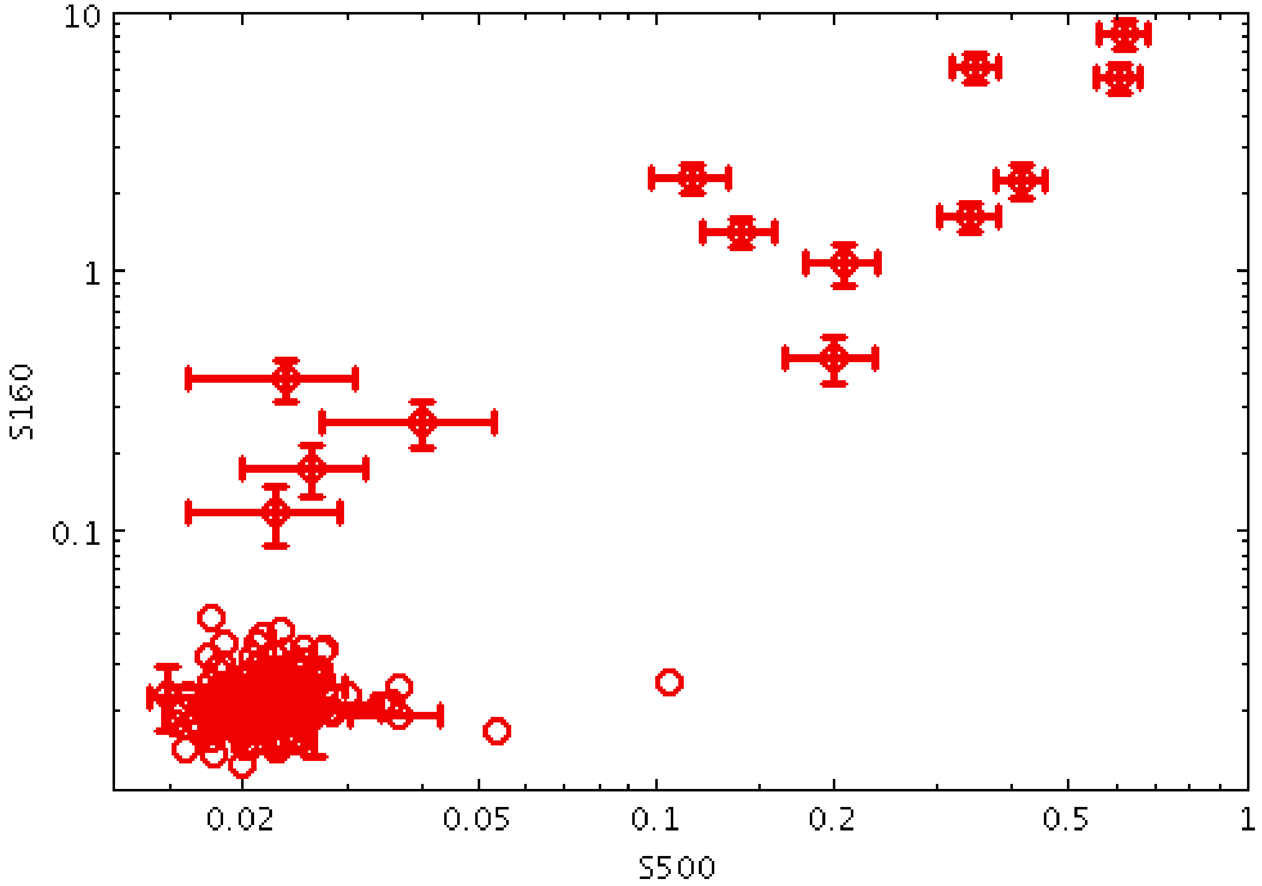
Thickness
Controls the line thickness used when drawing arrows. Zero, the default value, means a 1-pixel-wide line is used, and larger values make drawn lines thicker.

Scale
Changes the factor by which all vector sizes are scaled. If the arrows are too small, slide it right, if they are too big, slide it left. The slider scale is logarithmic. Alternatively, enter a fixed value in the text field.

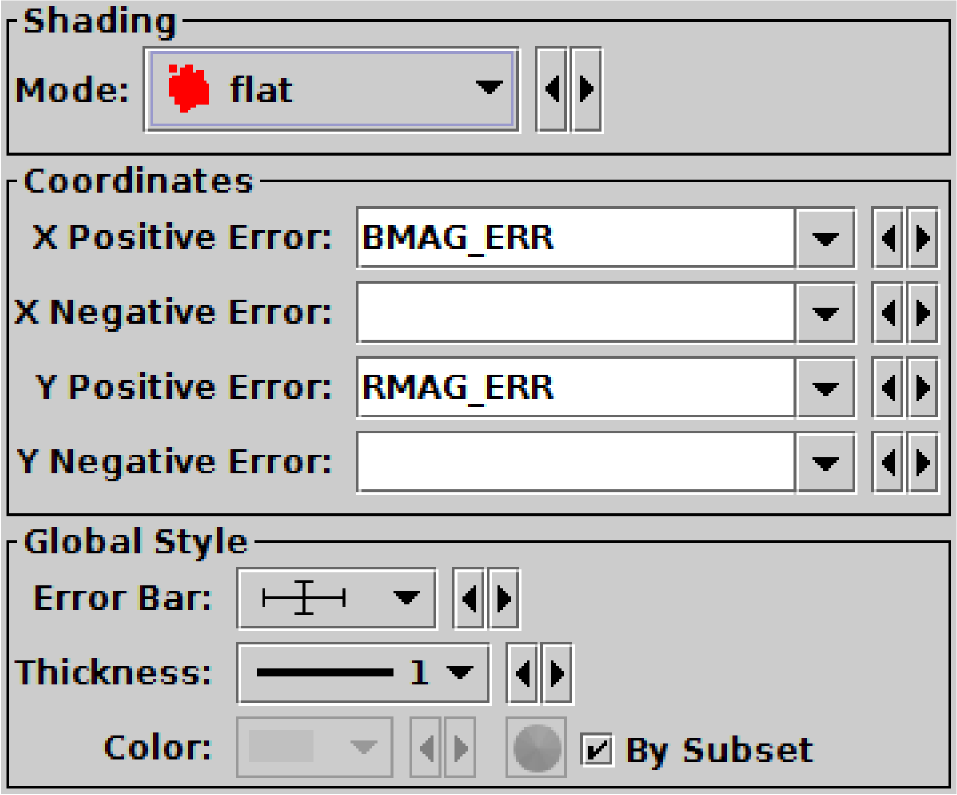
Auto Scale
If selected, this option will determine the default arrow scale size from the data - it will fix it so that the largest arrows are a few tens of pixels long by default. That scaling can then be adjusted using the **Scale** slider. If unselected, then the default position of the Scale slider corresponds to the actual values in degrees given by the submitted Delta Lon/Lat coordinates.

A.4.5.6 Error Bars Form

The **Error Bars** form (┼) draws symmetric or asymmetric error bars in some or all of the dimensions of a 1-, 2- or 3-dimensional Cartesian plot. The shape of the error "bars" is quite configurable, including (for 2- and 3-d errors) ellipses, rectangles etc, aligned with the axes (for rotated ellipses and rectangles, see the XYEllipse and XYCorr forms).



Example XYError plot



Shading
Mode: flat

Coordinates
X Positive Error: BMAG_ERR
X Negative Error:
Y Positive Error: RMAG_ERR
Y Negative Error:

Global Style
Error Bar:
Thickness: 1
Color:
☑ By Subset

Error Bars form configuration panel

The configuration options are:

Shading Mode

See Appendix A.4.6.

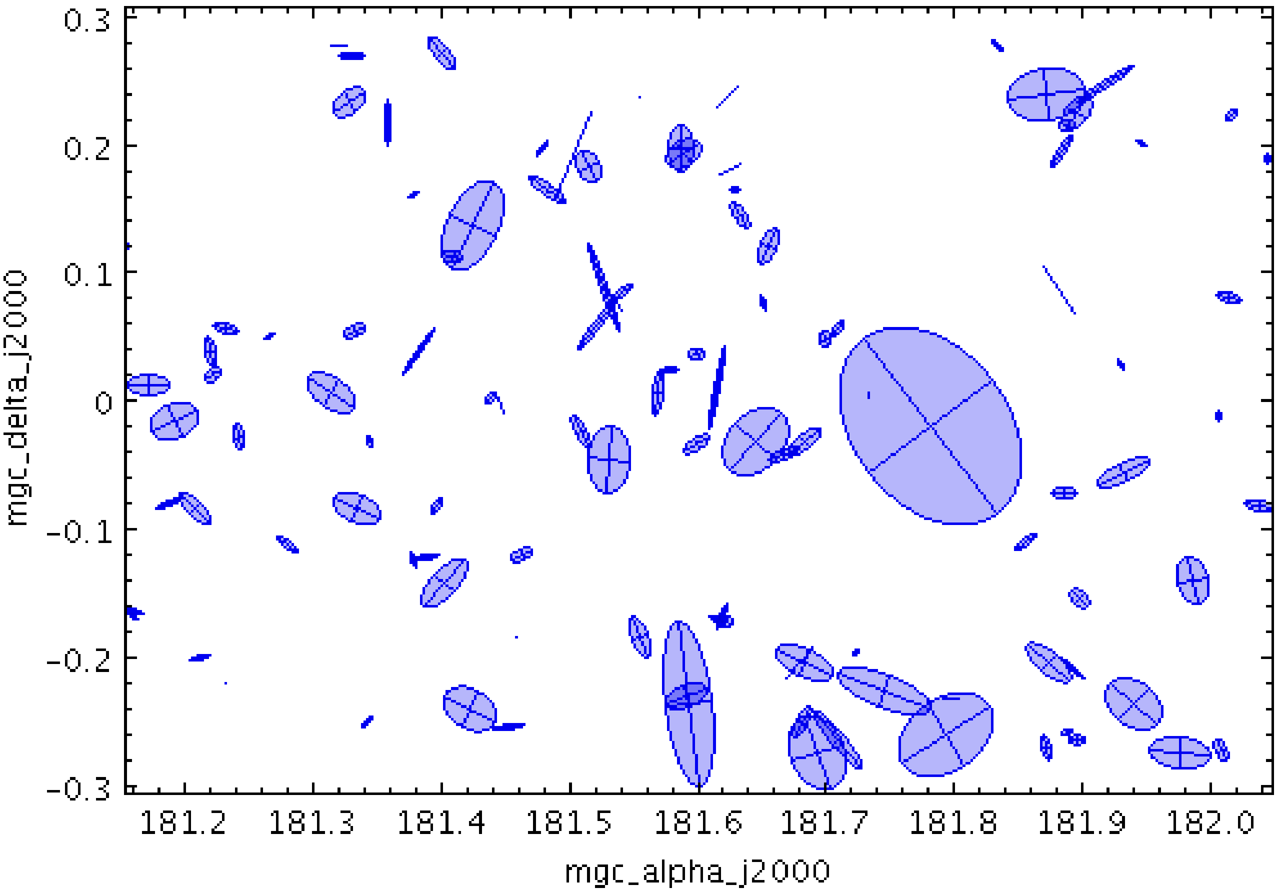
X Positive Error, X Negative Error, ...
For each dimension of the plot, you can enter positive and/or negative error values as columns or expressions from the table. All of these values must be positive; any negative values will be ignored. If both positive and negative values are filled in for an axis, the errors will be asymmetric. If the negative value is blank (either because the coordinate is not filled in, or because its value is blank for that row), the error bars will be symmetric, i.e. the negative error bar will be the same size as the positive one. If you want to ensure only a positive error bar is plotted, enter "`0`" for the corresponding negative error.

Error Bar
Error bar shape from a list of options. Exact appearance will also depend on the dimensionality of the supplied errors.

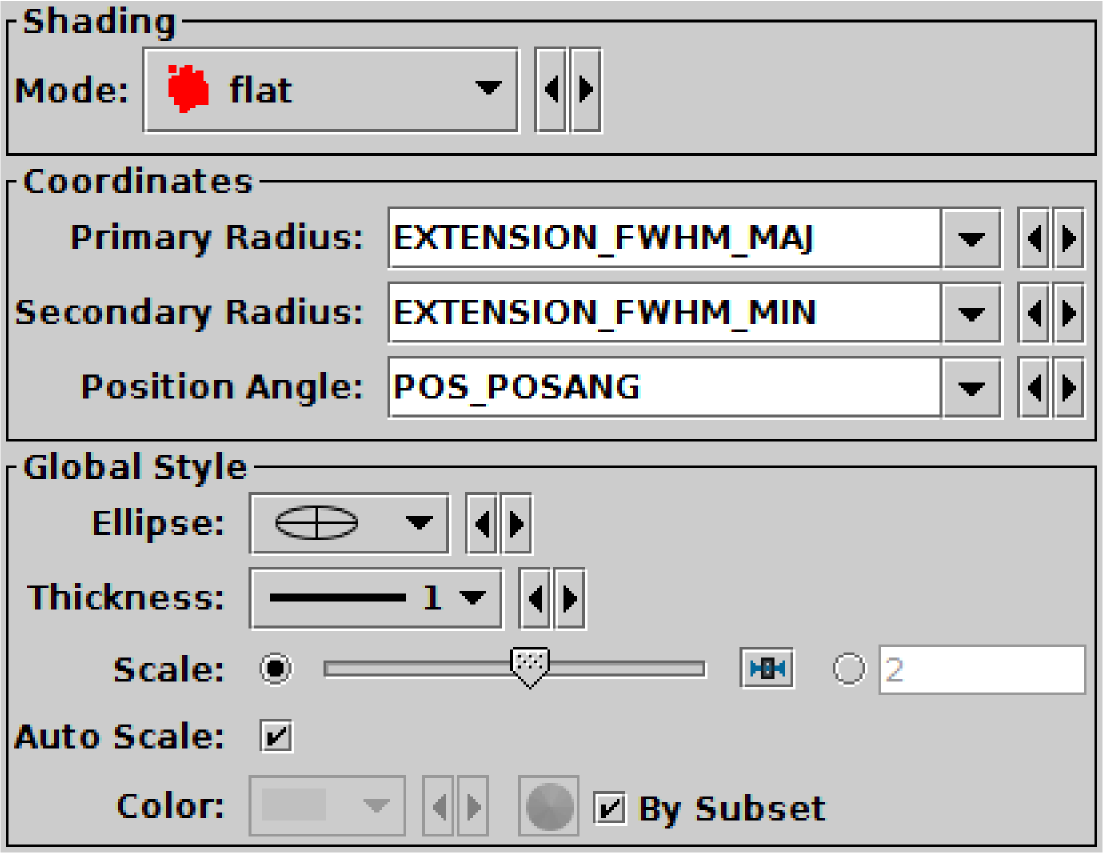
Thickness
Controls the line thickness used when drawing error bars. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled rectangles contain no line drawings.

A.4.5.7 XYEllipse Form

The **XYEllipse** form () plots an ellipse (or rectangle, triangle, or other similar figure) defined by two principal radii and an optional angle of rotation, the so-called position angle. This angle, if specified, is in degrees and gives the angle counterclockwise from the horizontal axis to the first principal radius.



Example XYEllipse plot



Shading

Mode: flat

Coordinates

Primary Radius: EXTENSION_FWHM_MAJ

Secondary Radius: EXTENSION_FWHM_MIN

Position Angle: POS_POSANG

Global Style

Ellipse:

Thickness: 1

Scale: ● ○ 2

Auto Scale: ☑

Color: ☑ By Subset

XYEllipse form configuration panel

The configuration options are:

Shading Mode
See Appendix A.4.6.

Primary Radius, Secondary Radius
The two principal radii for the ellipse, in the same units as the position coordinates. It doesn't matter whether the primary is larger than the secondary. If the Secondary Radius field is left blank, it is assumed to equal the Primary Radius field, i.e. the ellipses are circles.

Position Angle
Position angle in degrees. This is the angle anticlockwise from the X axis to the primary radius. If the value is missing (either this field not filled in or blank in the data) it is considered to be zero.

Ellipse
Ellipse graphical representation, selected from a range of options (includes also rectangles, crosses etc).

Thickness
Controls the line thickness used when drawing ellipses. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled ellipses contain no line drawings.

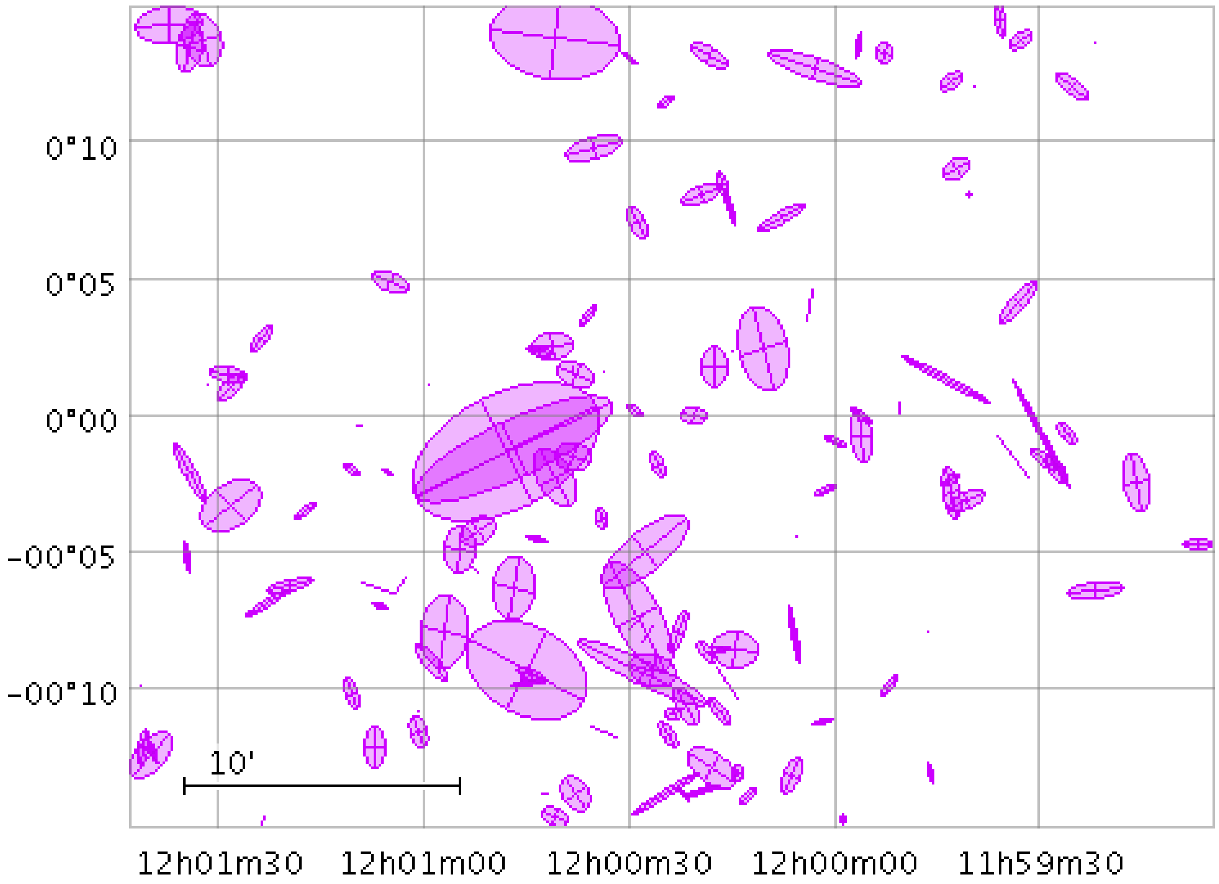
Scale
Changes the factor by which all ellipse sizes are scaled. If the ellipses are too small, slide it right, if they are too big, slide it left. The slider scale is logarithmic. Alternatively, enter a fixed value in the text field.

Auto Scale
If selected, this option will determine the default ellipse scale size from the data - it will fix it so that the largest ellipses are a few tens of pixels long by default. That scaling can then be

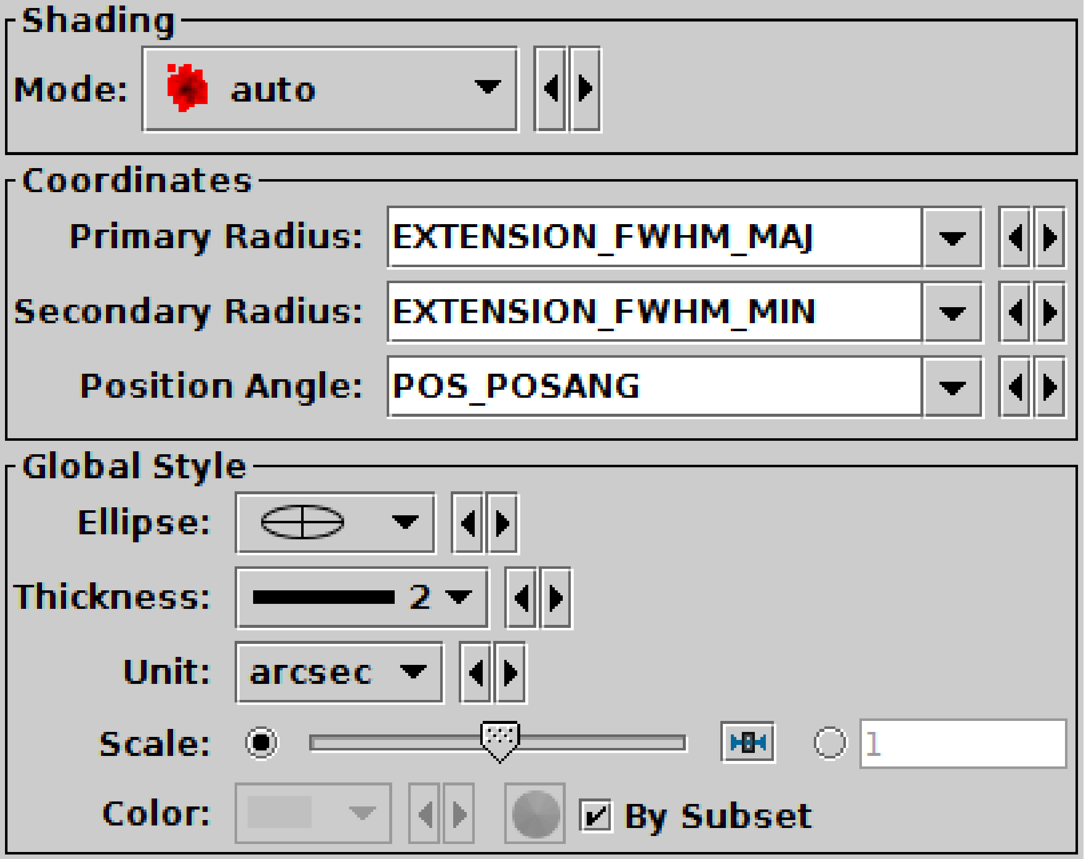
adjusted using the **Scale** slider. If unselected, then the default position of the Scale slider corresponds to the actual positions given by the submitted ellipse radii coordinates.

A.4.5.8 SkyEllipse Form

The **SkyEllipse** ( ) form plots an ellipse (or rectangle, triangle, or other similar figure) defined by two principal radii and an optional angle of rotation, the so-called position angle. This angle, if specified, is in degrees and gives the angle from the North pole towards the direction of increasing longitude on the longitude axis.



Example SkyEllipse plot



Shading

Mode: auto

Coordinates

Primary Radius: EXTENSION_FWHM_MAJ

Secondary Radius: EXTENSION_FWHM_MIN

Position Angle: POS_POSANG

Global Style

Ellipse:

Thickness: 2

Unit: arcsec

Scale: ● ○ 1

Color: ☑ By Subset

SkyEllipse form configuration panel

The configuration options are:

Primary Radius, Secondary Radius
The two principal radii for the ellipse. If auto-scaling is off, these are in units of **degrees**. It doesn't matter whether the primary is larger than the secondary. If the Secondary Radius field is left blank, it is assumed to equal the Primary Radius field, i.e. the ellipses are circles.

Position Angle
Orientation of the ellipse. This is the angle in degrees from the North pole to the primary axis of the ellipse in the direction of increasing longitude. If the value is missing (either this field not filled in or blank in the data) it is considered to be zero.

Ellipse
Ellipse graphical representation, selected from a range of options (includes also rectangles, crosses etc).

Thickness
Controls the line thickness used when drawing ellipses. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled ellipses contain no line drawings.

Scale
Changes the factor by which all ellipse sizes are scaled. If the ellipses are too small, slide it right, if they are too big, slide it left. The slider scale is logarithmic. Alternatively, enter a fixed value in the text field.

Auto Scale
If selected, this option will determine the default ellipse scale size from the data - it will fix it so that the largest ellipses are a few tens of pixels long by default. That scaling can then be adjusted using the **Scale** slider. If unselected, then the default position of the Scale slider corresponds to the actual values in degrees given by the submitted ellipse radii coordinates.

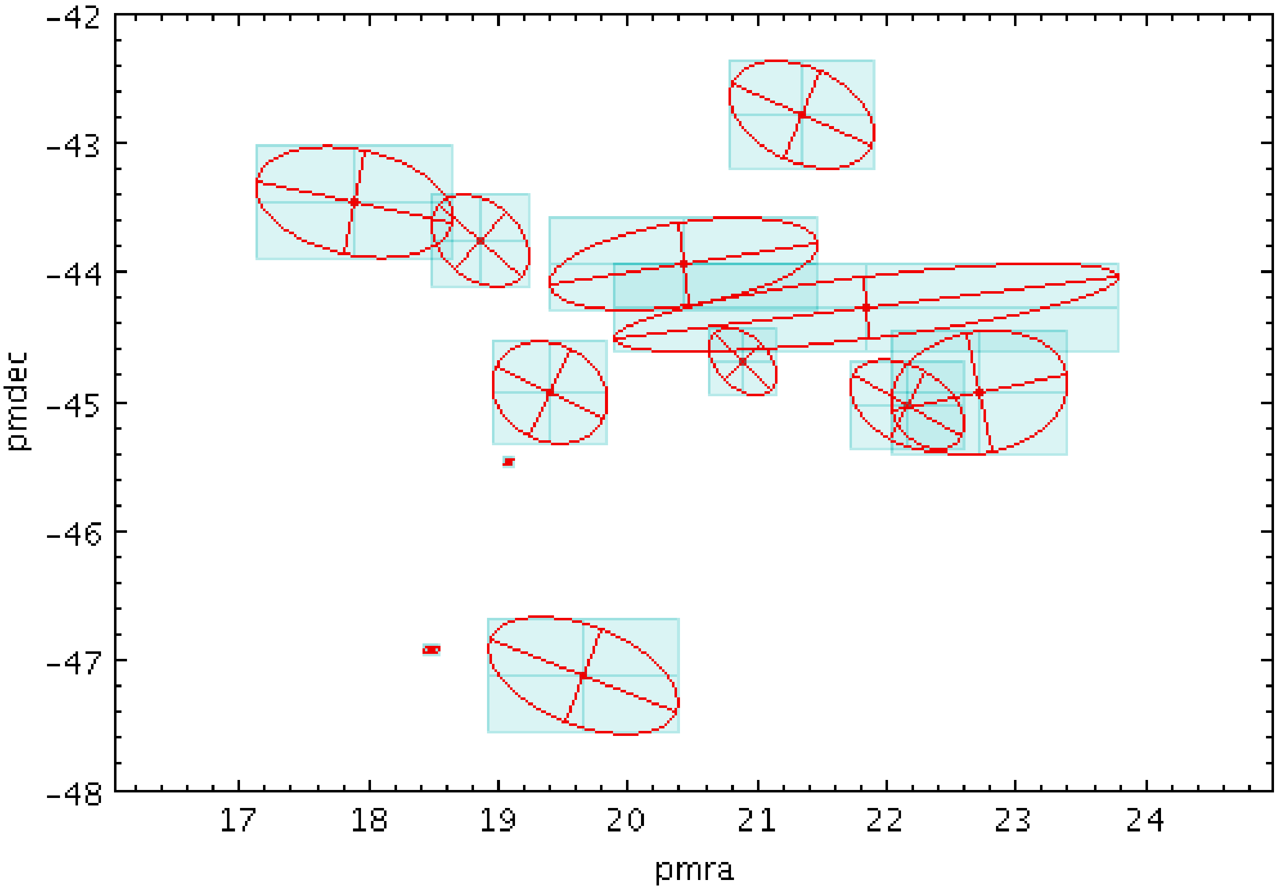
A.4.5.9 XYCorr Form

The **XYCorr** () form plots an error ellipse (or rectangle or other similar figure) defined by errors in the X and Y directions, and a correlation between the two errors.

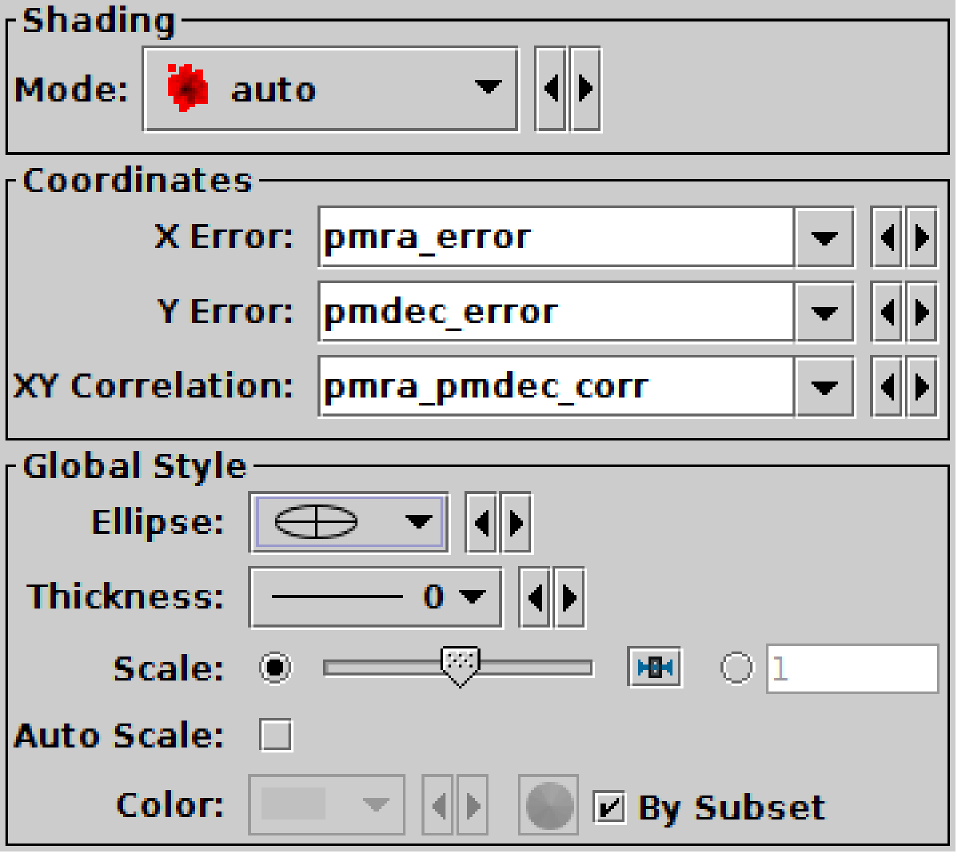
The supplied correlation is a dimensionless value in the range -1..+1 and is equal to the covariance divided by the product of the X and Y errors. The covariance matrix is thus:

```
   [  xerr*xerr         xerr*yerr*xycorr  ]
   [  xerr*yerr*xycorr  yerr*yerr         ]
```

This plot type is suitable for use with the `<x>_error` and `<x>_<y>_corr` columns in the *Gaia* source catalogue.



Example XYCorr plot



Shading

Mode: auto

Coordinates

X Error: pmra_error

Y Error: pmdec_error

XY Correlation: pmra_pmdec_corr

Global Style

Ellipse:

Thickness: 0

Scale: ● ○ 1

Auto Scale: □

Color:

☑ By Subset

XYCorr form configuration panel

The configuration options are:

Shading Mode
See Appendix A.4.6.

X/Y Error
The error along the horizontal and vertical directions, as column names or expressions.

XY Correlation
Correlation betewen the errors in the horizontal and vertical directions. This is a dimensionless quantity in the range -1..+1, and is equivalent to the covariance divided by the product of the X and Y error values themselves. It corresponds to the `x_y_corr` values supplied in the *Gaia* source catalogue.

Ellipse
Ellipse graphical representation, selected from a range of options (includes also rectangles, crosses etc).

Thickness
Controls the line thickness used when drawing polygons. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled polygons contain no line drawings.

Scale
Adjusts the size of variable-sized markers like vectors and ellipses. The default value is 1, smaller or larger values multiply the visible sizes accordingly.

Autoscale
Determines whether plotted ellipses are automatically scaled to have a sensible size. If true, then the sizes of all the ellipse markers are examined, and some dynamically calculated factor is applied to them all to make them a sensible size (by default, the largest ones will be a few tens of pixels). If false, the sizes will be the actual input values interpreted in data coordinates.

If auto-scaling is on, then markers will keep approximately the same screen size during zoom operations; if it's off, they will keep the same size in data coordinates.

A.4.5.10 SkyCorr Form

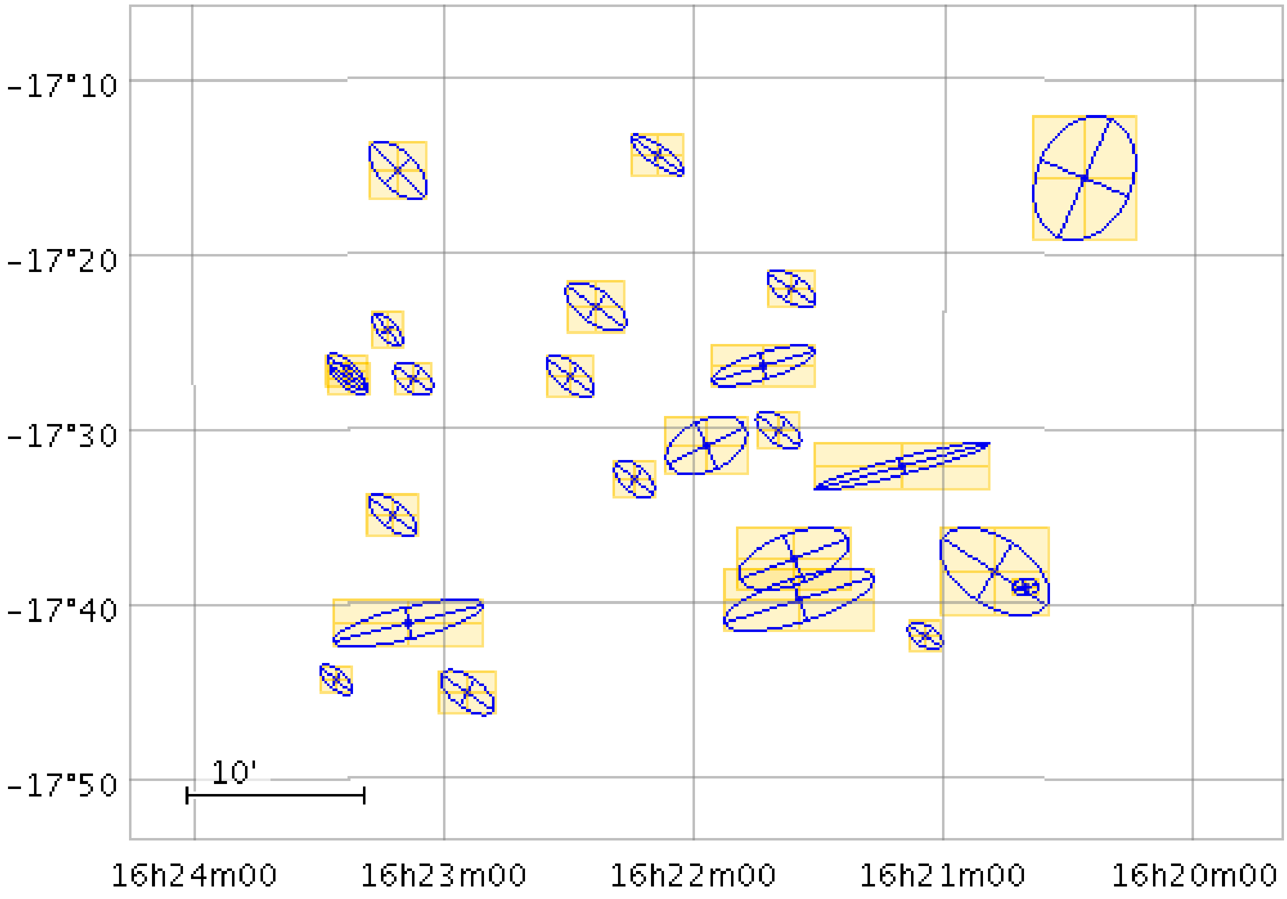
The **SkyCorr** () form plots an error ellipse (or rectangle or other similar figure) on the sky defined by errors in the longitude and latitude directions, and a correlation between the two errors.

The error in longitude is considered to be premultiplied by the cosine of the latitude, i.e. both errors correspond to angular distances along a great circle.

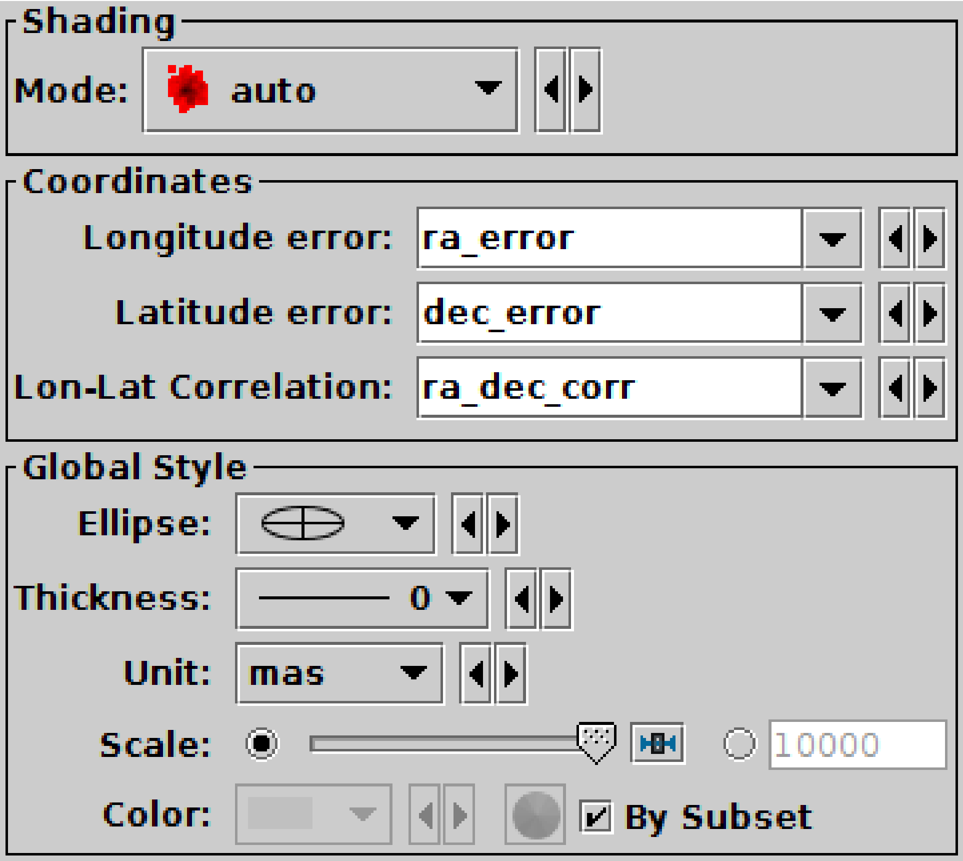
The supplied correlation is a dimensionless value in the range -1..+1 and is equal to the covariance divided by the product of the lon and lat errors. The covariance matrix is thus:

```
   [   lonerr*lonerr        lonerr*laterr*corr   ]
   [   lonerr*laterr*corr   laterr*laterr        ]
```

This plot type is suitable for use with the `ra_error`, `dec_error` and `ra_dec_corr` columns in the *Gaia* source catalogue. Note however that Gaia positional errors are generally quoted in milli-arseconds (mas), while this plotter requires lon/lat errors in degrees, so to plot true error ellipses it is necessary to divide the Gaia error values by 3.6e6. In most plots, Gaia position errors are much too tiny to see!



Example SkyCorr plot



Shading

Mode: auto

Coordinates

Longitude error: ra_error

Latitude error: dec_error

Lon-Lat Correlation: ra_dec_corr

Global Style

Ellipse:

Thickness: 0

Unit: mas

Scale: ● ○ 10000

Color: ☑ By Subset

SkyCorr form configuration panel

The configuration options are:

Shading mode
See Appendix A.4.6.

Longitude error
Error in longitude in degrees, multiplied by the cosine of the latitude.

Latitude error
Error in latitude in degrees.

Lon-Lat correlation
Correlation between the errors in longitude and latitude. This is a dimensionless quantity in the range -1..+1, and is equivalent to the covariance divided by the product of the Longitude and Latitude error values themselves. It corresponds to the `ra_dec_corr` value supplied in the *Gaia* source catalogue.

Ellipse
Ellipse graphical representation, selected from a range of options (includes also rectangles, crosses etc).

Thickness
Controls the line thickness used when drawing ellipses. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled polygons contain no line drawings.

Unit
Gives the units in which the Longitude and Latitude errors are supplied. Options are degrees, arcseconds etc. If the special value **scaled** is given then a non-physical scaling is applied to the input values to make the the largest markers appear at a reasonable size (a few tens of pixels) in the plot. Note that the actual plotted size of the markers can also be scaled using the **scale** option; these two work together to determine the actual plotted sizes.

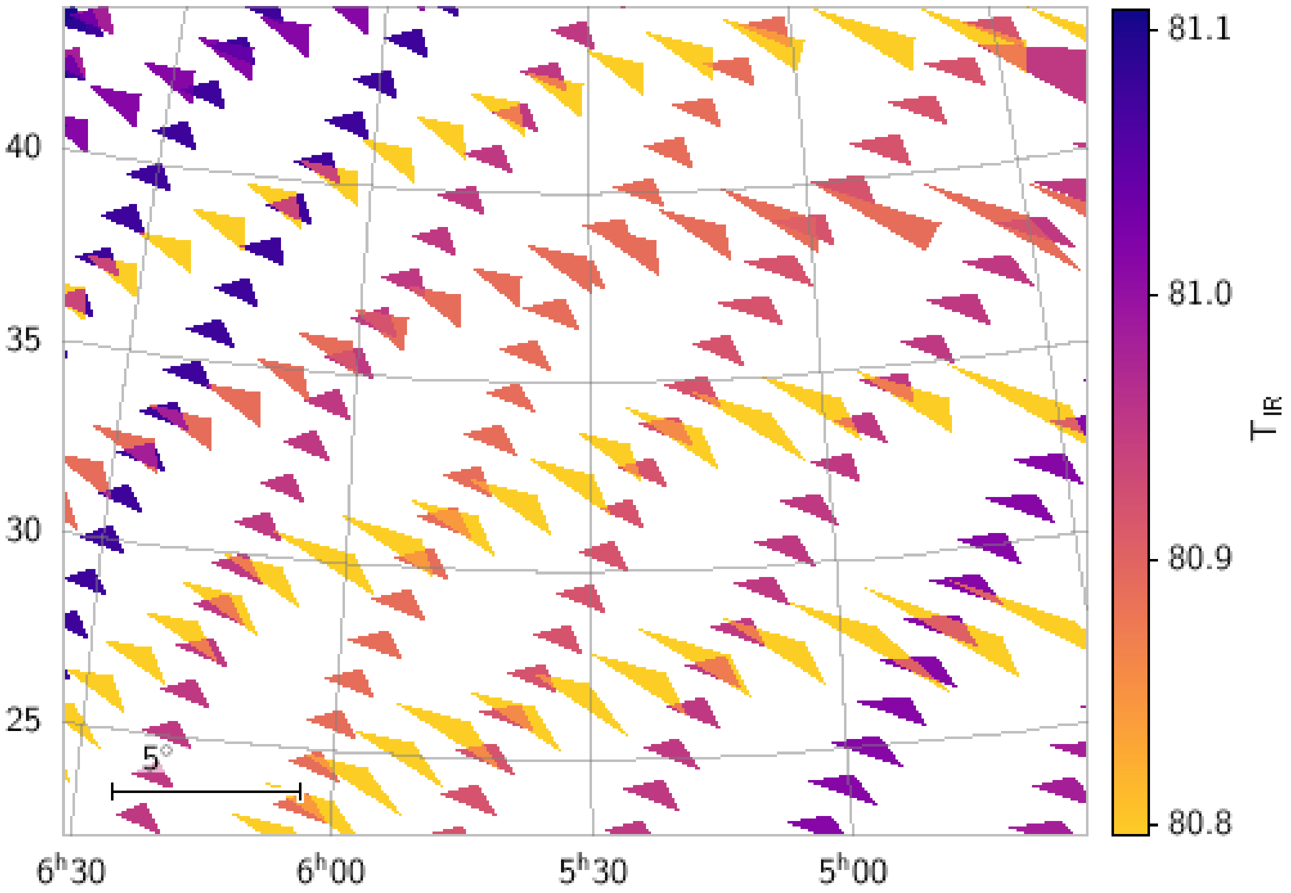
Scale

Scales the size of plotted ellipses. The default value is 1, smaller or larger values multiply the visible sizes accordingly. The main purpose of this option is to tweak the visible sizes of the plotted markers for better visibility. The **unit** option is more convenient to account for the units in which the angular extent coordinates are supplied. If the markers are supposed to be plotted with their absolute angular extents visible, this option should be set to its default value of 1.

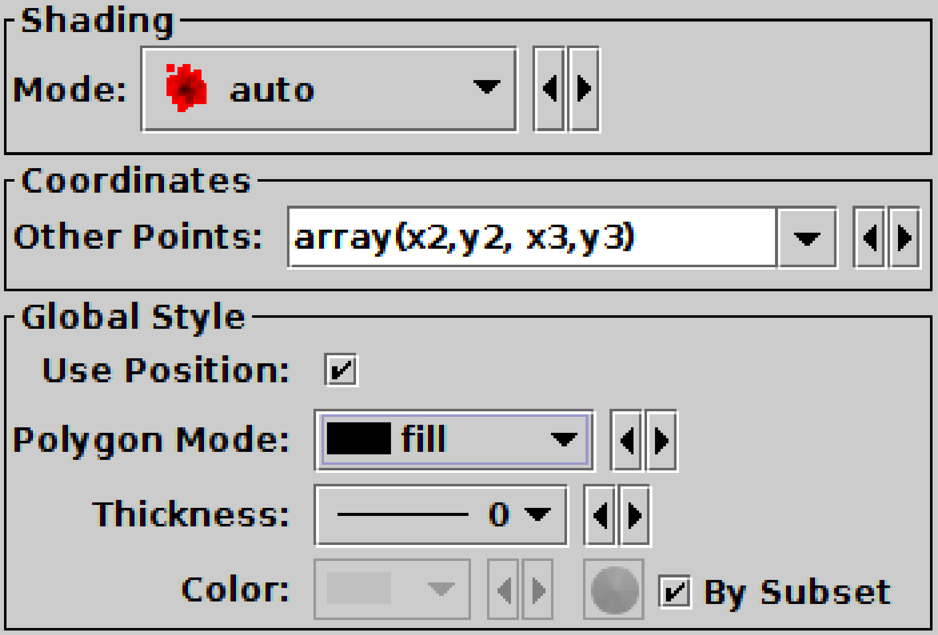
A.4.5.11 Polygon Form

The **Polygon** () form plots filled or outlined polygons with an arbitrary number of vertices.

The first vertex is given in the Position Layer Control's **Position** tab, and the others are given in the Coordinates box within the **Form** tab.



Example Polygon plot



Shading

Mode: auto

Coordinates

Other Points: array(x2,y2, x3,y3)

Global Style

Use Position: ☑

Polygon Mode: fill

Thickness: 0

Color:

☑ By Subset

Polygon form configuration panel

The configuration options are:

Shading mode
See Appendix A.4.6.

Other Points
Array of coordinates giving the polygon vertices *excluding* the first vertex, which is given by the main Position. These coordinates are given as an interleaved array, e.g. `(x2,y2, x3,y3, x4,y4)`. Expression language functions including `array(a,b,c,...)` from the Arrays class, and `parseDoubles(txt)` from the Conversions class, can be useful here. Although the first coordinate pair is supposed to be given in the Position tab and not in this array, if it's more convenient to repeat the first pair here it doesn't usually affect the plot's appearance.

Use Position
Determines whether the basic positional coordinates for the plotted row are included as one of the polygon vertices or not. The polygon has N+1 vertices if this value is set, or N if it is not, where N is the number of vertices supplied by the **Other Points** array. If unset, the reference position is ignored for the purposes of drawing the polygon.

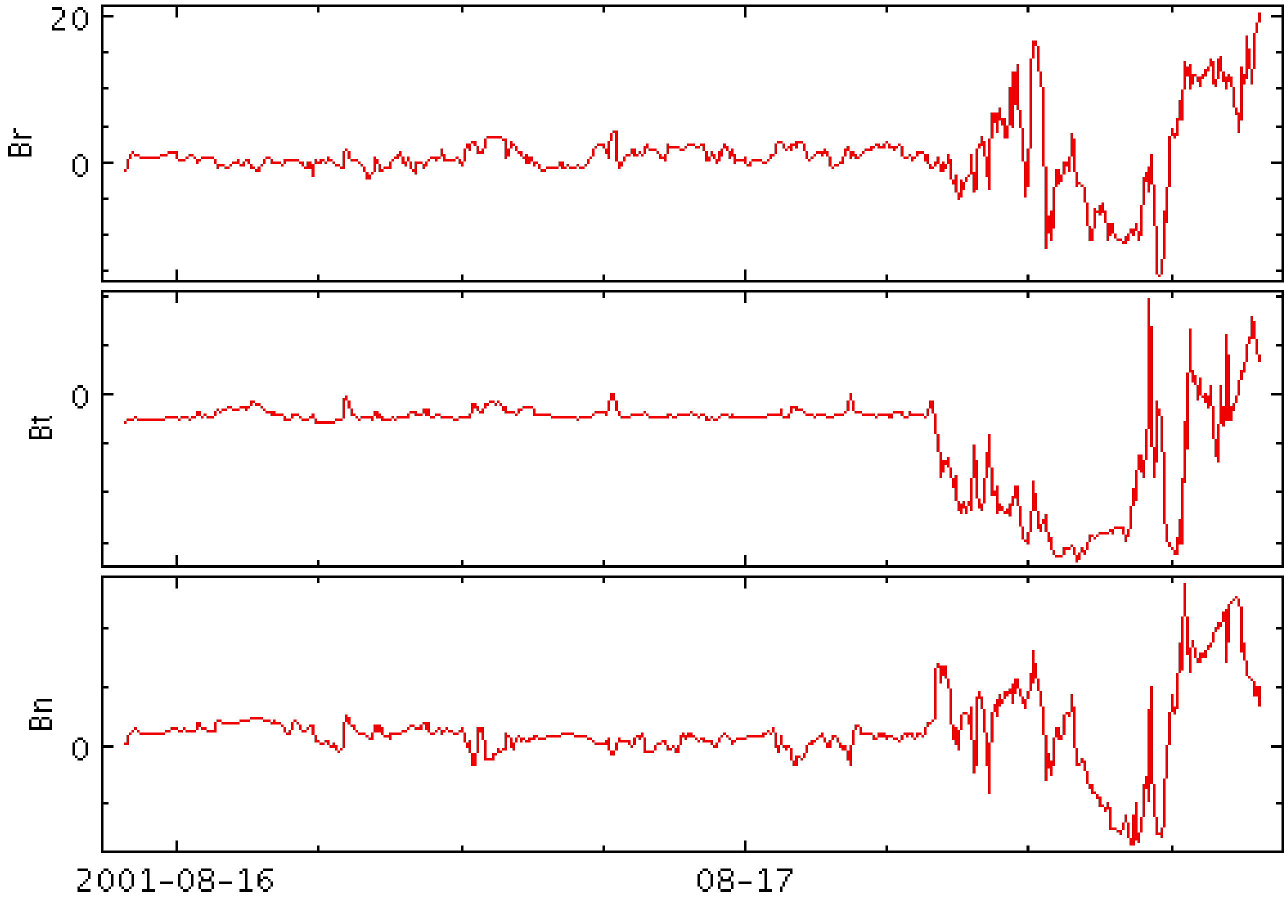
Polygon Mode
Defines how the polygon is drawn. Options are

- `outline`: draws a line round the outside of the polygon
- `fill`: fills the interior of the polygon
- `cross`: draws a line round the outside of the polygon and lines between all the vertices

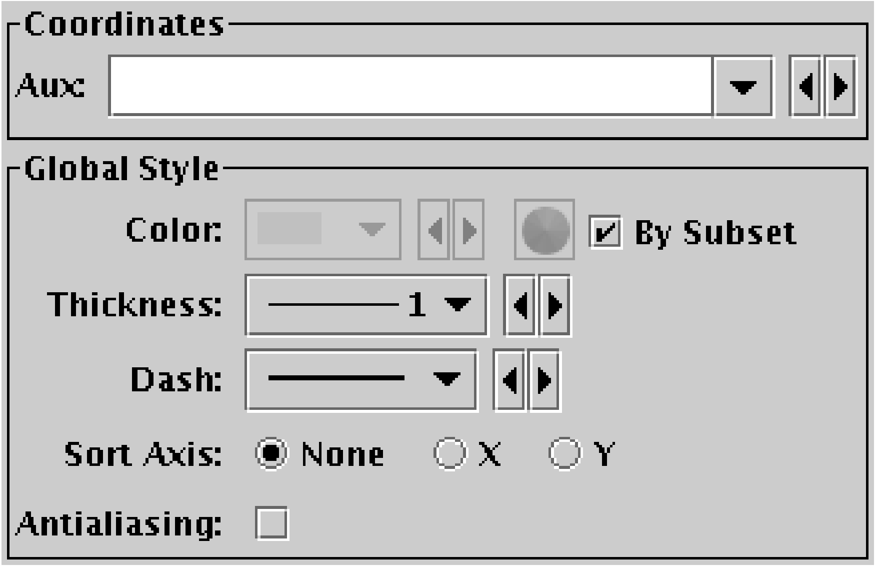
Thickness
Controls the line thickness used when drawing polygons. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled polygons contain no line drawings.

A.4.5.12 Line Form

The **Line** form () draws a point-to-point line joining up the points making up the data set. There are additional options to pre-sort the points according to their order on the X or Y axis (using the **Sort Axis** control), and to vary the colour of the line along its length (using the **Aux** coordinate).



Example Line plot



Coordinates

Aux:

Global Style

Color: ☑ By Subset

Thickness: 1

Dash:

Sort Axis: ● None ○ X ○ Y

Antialiasing: □

Line form configuration panel

The configuration options are:

Aux
If supplied, this controls the colour of the line along its length according to the value given here. As for the Aux shading mode, the details of the colour mapping are controlled using the Aux Axis control.

Thickness

Line thickness in pixels.

Dash

Dash pattern. The line is solid by default.

Sort Axis

May be used to sort the points before the lines are drawn. By default (option **None**) the lines are drawn between the points in the sequence in which they appear in the table. But if you set it to **X** or **Y** the points will be pre-ordered along the given axis, so that lines for unordered rows will come out looking like a function of the X or Y coordinate rather than a scribble.

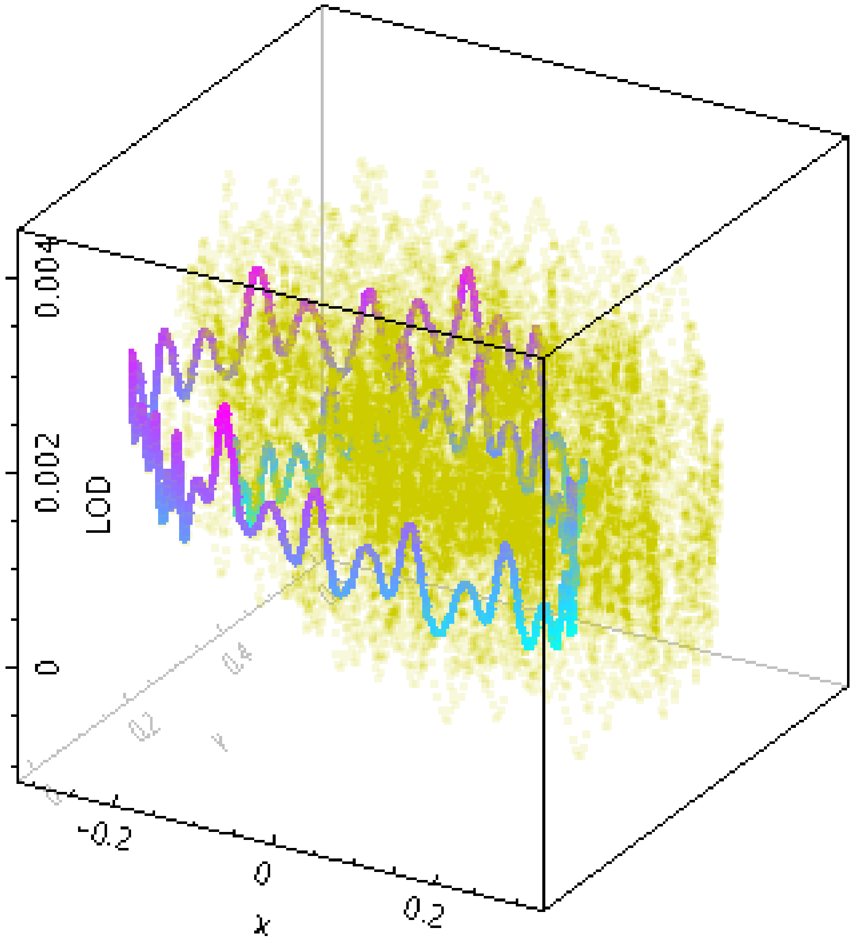
Antialiasing

If true, pixels are blurred to give the line a smoother appearance.

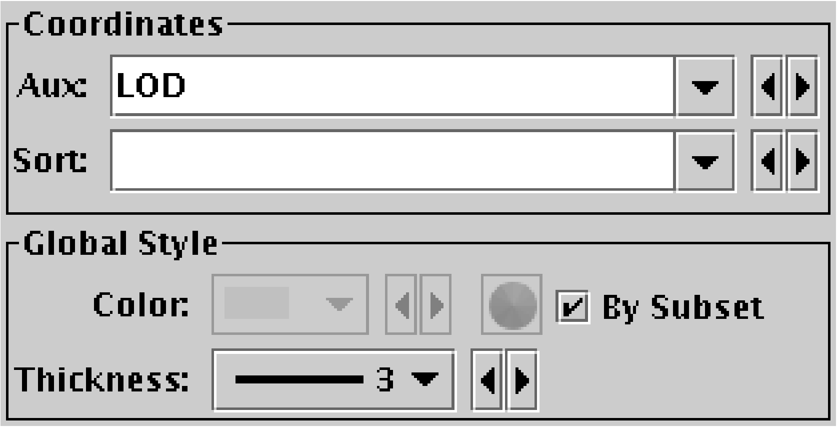
A.4.5.13 Line3d Form

The **Line3d** form ( ) draws a point-to-point line joining up the positions of the points in three dimensions. There are additional options to pre-sort the points by a given quantity before drawing the lines, and to colour the line along its length using an auxiliary coordinate.

Note that the line positioning in 3d and the line segment aux colouring is somewhat approximate. In most cases it is good enough for visual inspection, but pixel-level examination may reveal discrepancies.



Example Line3d plot



Coordinates

Aux: LOD

Sort:

Global Style

Color:

☑ By Subset

Thickness: 3

Line3d form configuration panel

The configuration options are:

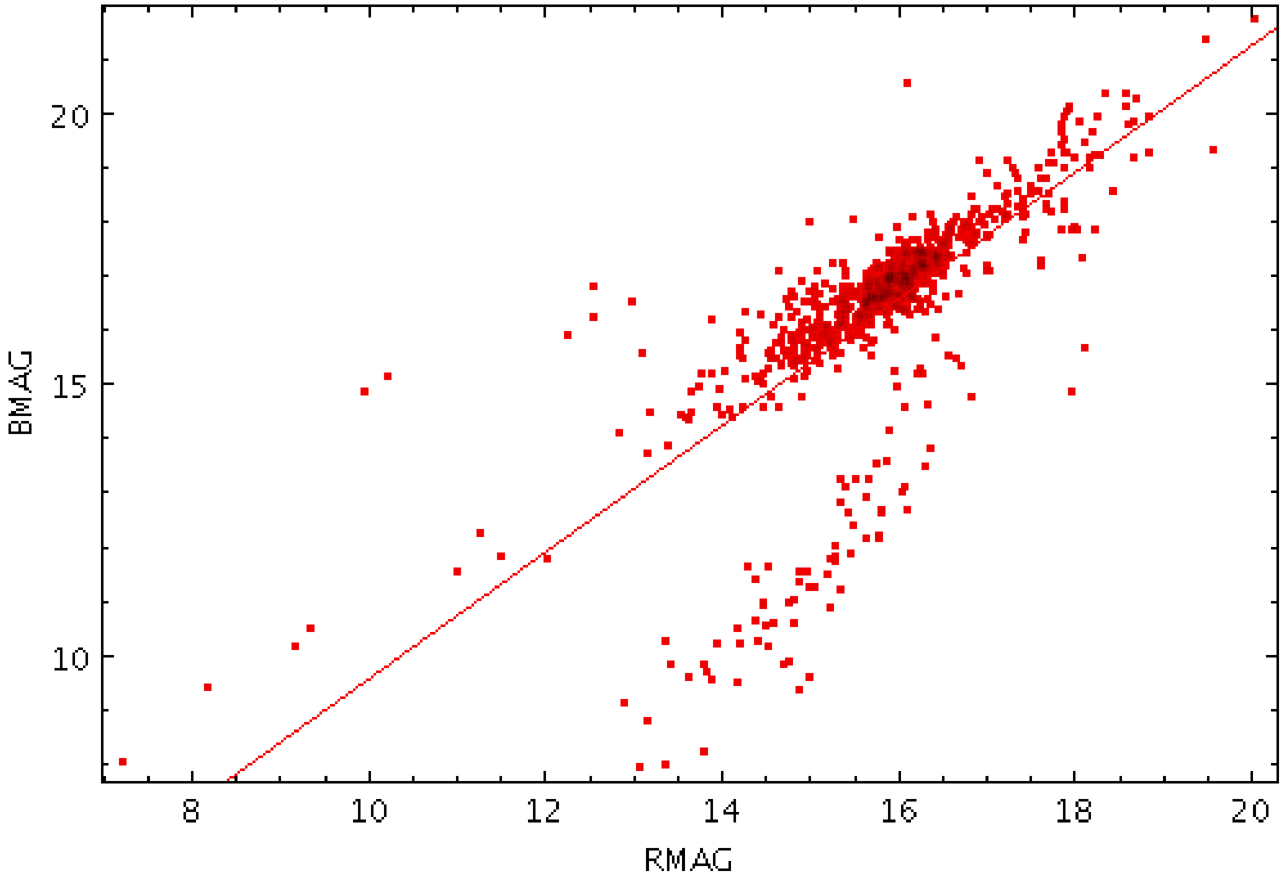
Aux
If supplied, this controls the colour of the line along its length according to the value given here. As for the Aux shading mode, the details of the colour mapping are controlled using the Aux Axis control.

Sort
If supplied, this gives a value to define in what order points are joined together. If no value is given, the natural order is used, i.e. the sequence of rows in the table. Note that if the required order is in fact the natural order of the table, it's better to leave this value blank, since sorting is a potentially expensive step.

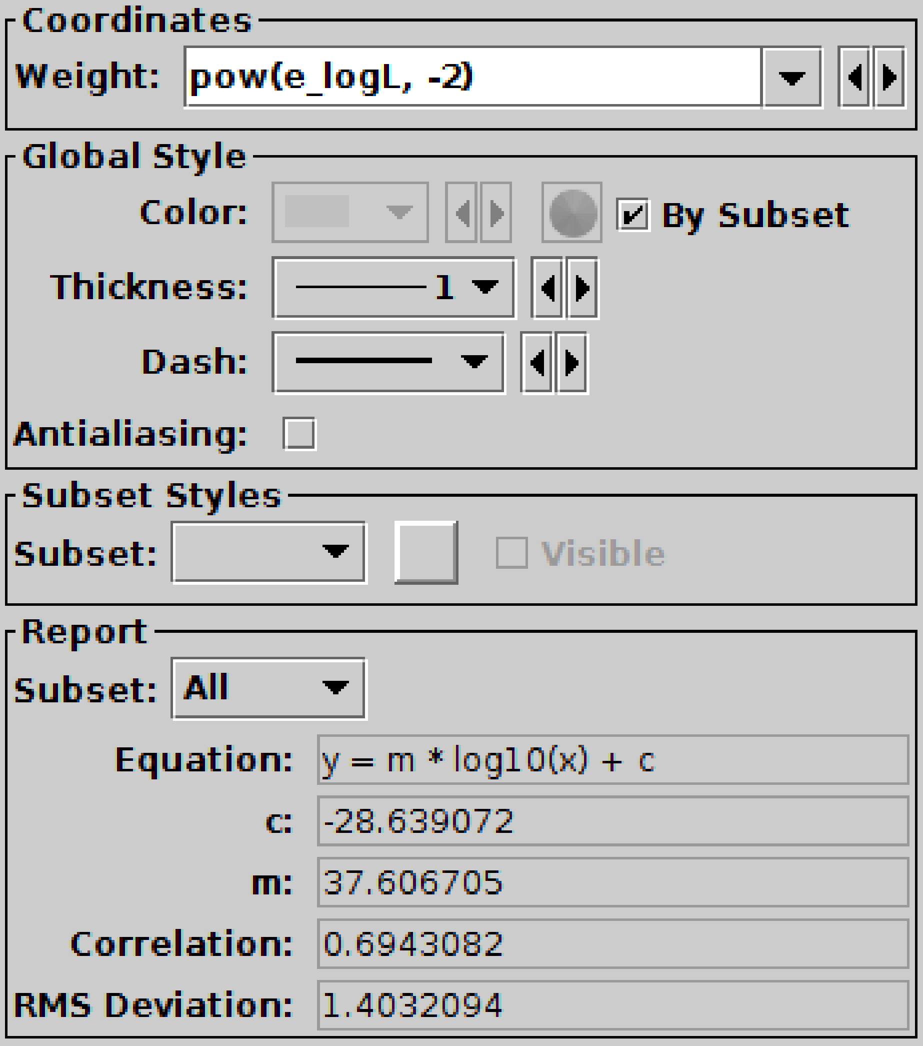
Thickness
Thickness of the line in pixels.

A.4.5.14 Linear Fit Form

The **Linear Fit** form () determines a least-squares best fit of a line to the data points, optionally weighted by a given value, and plots the corresponding line.



Example LinearFit plot



Weight: pow(e_logL, -2)

Color: ☑ By Subset

Thickness: 1

Dash:

Antialiasing: □

Subset: □ Visible

Subset: All

Equation: y = m * log10(x) + c

c: -28.639072

m: 37.606705

Correlation: 0.6943082

RMS Deviation: 1.4032094

Linear Fit configuration panel

The configuration options are:

Weight
Weighting to apply to each point. If left blank, no weighting (equivalent to unit weighting) is applied. If independent sampling errors `E` are available for each point, a suitable value for this may be `1./(E*E)` (equivalently `pow(E,-2)`).

Thickness
Line thickness in pixels.

Dash
Dash pattern. The line is solid by default.

Antialiasing
If true, pixels are blurred to give the line a smoother appearance.

As well as drawing the line onto the plot, the calculated fitting coefficients are displayed at the bottom of the form configuration panel, under the heading **Report**. Note the coefficients are calculated by subset, and are only displayed for one subset at a time. To see the calculated values, select the subset of interest in the **Subset** selector. The reported items are:

Equation
The equation of the line. The basic linear equation is `y=m*x+c`, but if one or both of the axes are plotted logarithmically, the fit is performed to take account of this, and the equation is displayed accordingly.

c
The constant intercept of the line on the Y axis.

m
The line gradient.

Correlation
Pearson's product moment correlation coefficient.

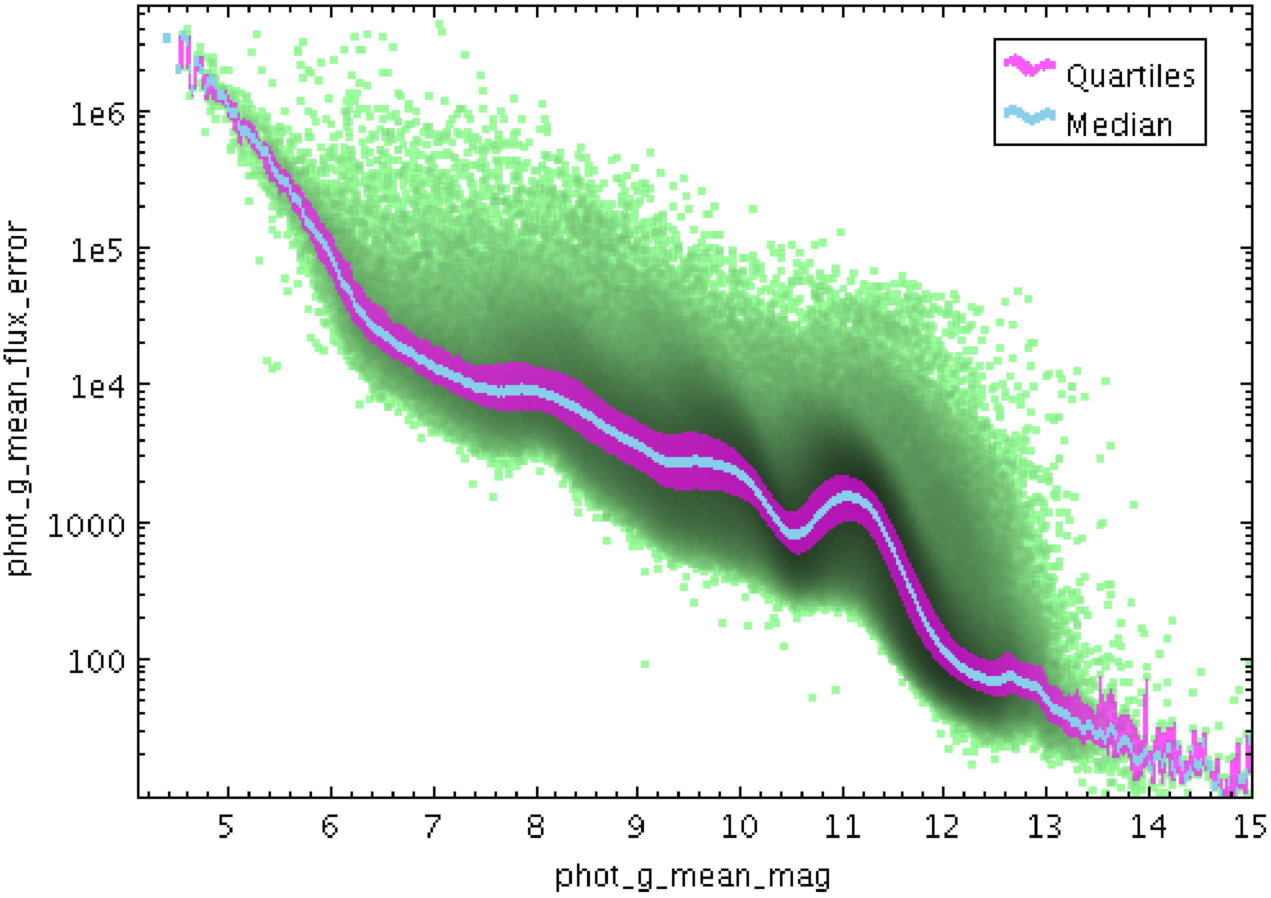
RMS Deviation
The root-mean-squared deviation of the data from the fitted line, i.e. for the case where both axes are linear, Sqrt(Sum[$w_i\ (y_i - m.x_i - c)^2$] / N)

A.4.5.15 Quantile Form

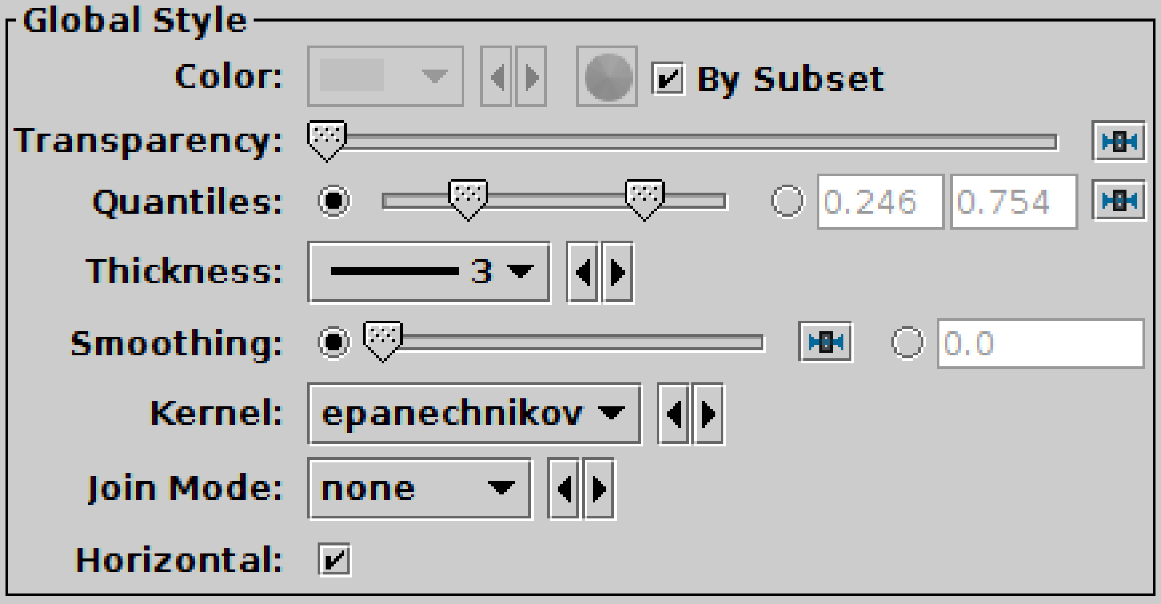
The **Quantile** form ( ) plots a line through a given quantile of the values binned within each pixel column (or row) of a plot. The line is optionally smoothed using a configurable kernel and width, to even out noise from the pixel binning. Instead of a simple line through a given quantile, it is also possible to fill the region between two quantiles.

One way to use this is to draw a line estimating a function *y=f(x)* (or *x=g(y)*) sampled by a noisy set of data points in two dimensions.

Note: In the current implementation, depending on the details of the configuration and the data, there may be some distortions or missing graphics near the edges of the plot. This may be improved in future releases, depending on feedback.



Example Quantile plot



Global Style

Color: ☑ By Subset

Transparency:

Quantiles: ● ○ 0.246 0.754

Thickness: 3

Smoothing: ● ○ 0.0

Kernel: epanechnikov

Join Mode: none

Horizontal: ☑

Quantile configuration panel

The configuration options are:

Transparency
Transparency with which components are plotted, from opaque to invisible.

Quantiles
Defines the quantile or quantile range of values that should be marked in each pixel column (or row). The slider control goes from 0 (minimum in pixel column/row) to 1 (maximum in pixel column/row), so 0.5 indicates the median. This control is a double-slider, so you can drag out a range of values. If the values are the same as each other, a single point will be indicated, but if there is a range then the area between the indicated quantiles will be filled.

The radio buttons let you toggle between using the slider to set the quantile value(s) or entering them in the text fields. If the two values are identical, you can leave the second text field blank.

Thickness
Sets the minimum extent of the markers that are plotted in each pixel column (or row) to indicate the designated value range. If the range is zero sized (two bounding quantiles are equal) this will give the actual thickness of the plotted line. If the range is non-zero however, the line may be thicker than this in places according to the quantile positions.

Smoothing
Configures the smoothing width. This is the characteristic width of the **Kernel** function to be convolved with the density in one dimension to smooth the quantile function.

You can adjust it using the slider (wider smoothing to the right) or enter a value in data coordinates explicitly in the text field. If the smoothed axis is logarithmic, the value is a multiplication factor rather than an additive increment.

Kernel
The functional form of the smoothing kernel. The functions listed refer to the unscaled shape; all kernels are normalised to give a total area of unity.

The available options are:

- **square**: Uniform value: f(x)=1, |x|=0..1
- **linear**: Triangle: f(x)=1-|x|, |x|=0..1
- **epanechnikov**: Parabola: f(x)=1-x*x, |x|=0..1
- **cos**: Cosine: f(x)=cos(x*pi/2), |x|=0..1

- **cos2**: Cosine squared: f(x)=cos^2(x*pi/2), |x|=0..1
- **gauss3**: Gaussian truncated at 3.0 sigma: f(x)=exp(-x*x/2), |x|=0..3
- **gauss6**: Gaussian truncated at 6.0 sigma: f(x)=exp(-x*x/2), |x|=0..6

Join Mode

Defines the graphical style for connecting distinct quantile values. If smoothed samples are packed more closely than the pixel grid the option chosen here doesn't make much difference, but if there are gaps in the data along the sampled axis, it's useful to have a guide to the eye to join one quantile determination to the next.

The available options are:

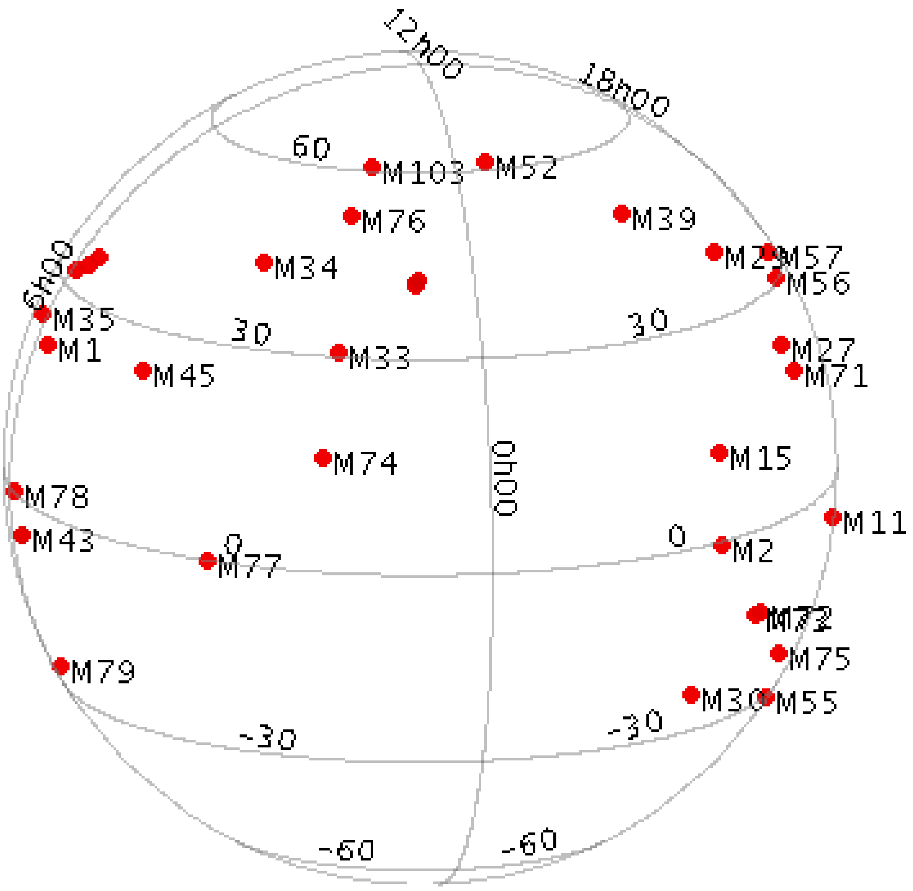
- **none**: displayed quantile ranges are not joined
- **polygon**: the area between a line connecting the upper quantiles and a line connecting the lower quantiles is filled
- **lines**: a line of thickness given by **Thickness** is drawn from the center of each quantile range to the next

Horizontal

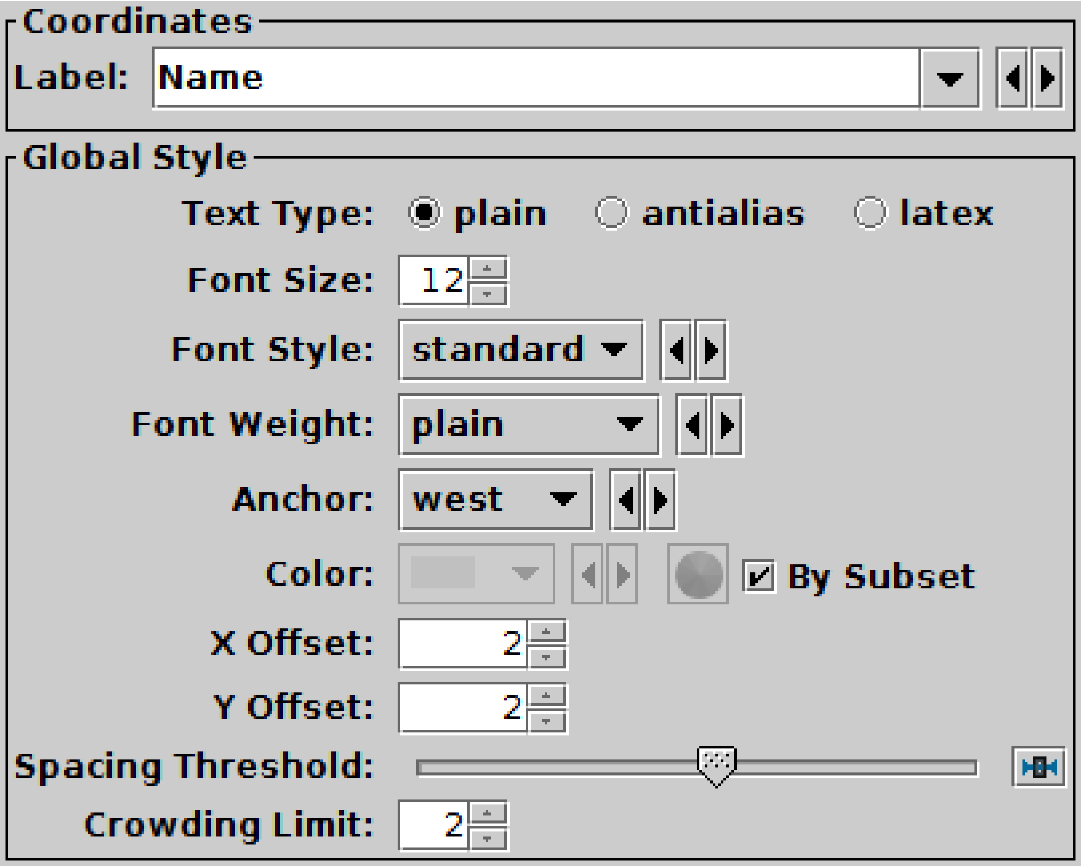
Determines whether the trace bins are horizontal or vertical. If true, *y* quantiles are calculated for each pixel column, and if false, *x* quantiles are calculated for each pixel row.

A.4.5.16 Label Form

The **Label** form (Txt) draws a text label by each point position.



Example Label plot



Coordinates

Label: Name

Global Style

Text Type: ● plain ○ antialias ○ latex

Font Size: 12

Font Style: standard

Font Weight: plain

Anchor: west

Color: ☑ By Subset

X Offset: 2

Y Offset: 2

Spacing Threshold:

Crowding Limit: 2

Label form configuration panel

The configuration options are:

Text
A column or expression from the table supplying the text to write on the plot. Any data type (string or numeric) is permitted.

Text Syntax
How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. At time of writing, on MacOS antialiased text seems to be required to stop the writing coming out upside-down for non-horizontal text. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size
Size of the font in points.

Font Style
Style of the font - standard, serif or monospaced.

Font Weight
Whether the font is plain, bold or italic.

Anchor
The position of the text relative to the data position.

X Offset
Y Offset
Supplies pixel offsets for the label positioning. This allows fine adjustment of where the labels will appear. Each value is in pixels, and may be positive or negative.

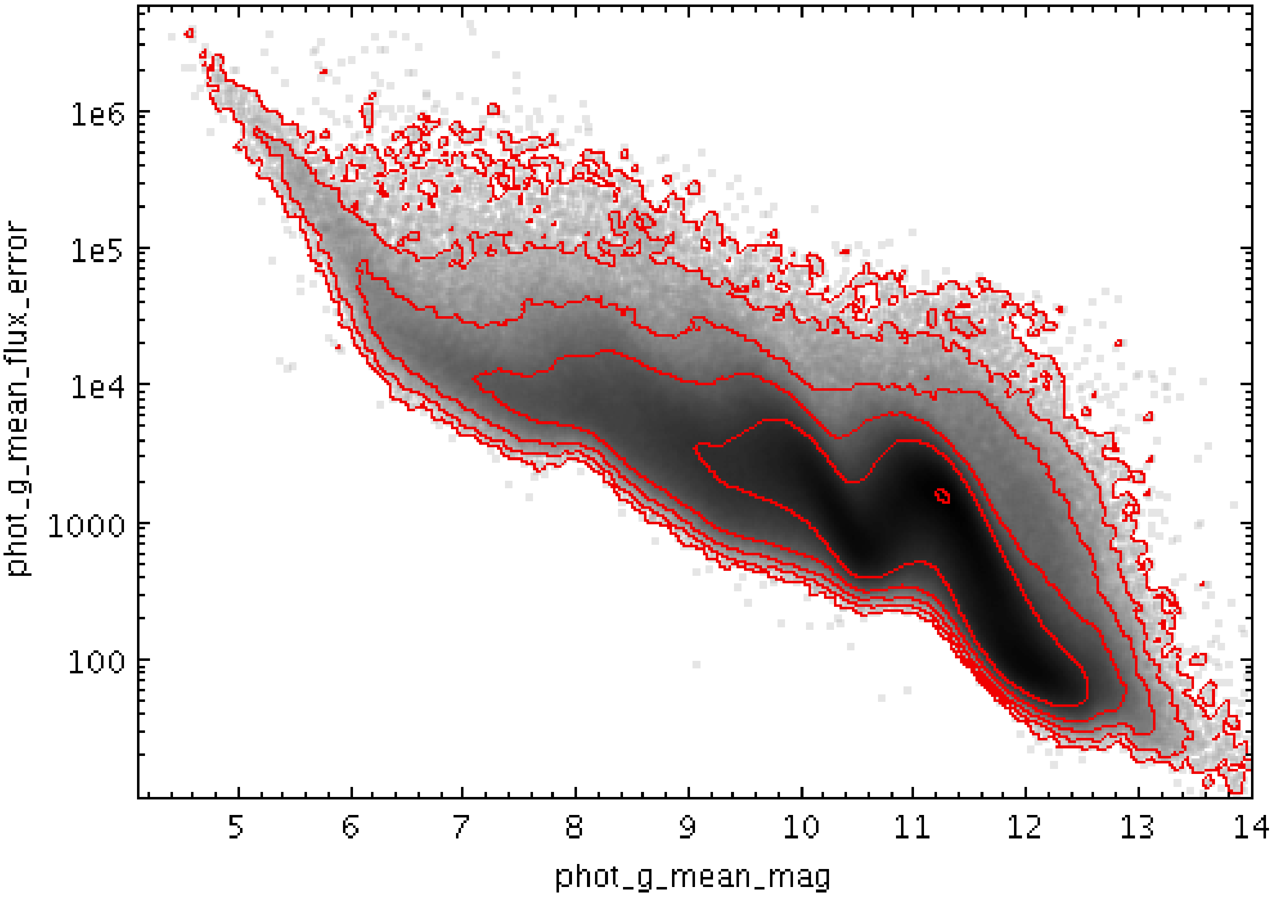
Spacing Threshold
Crowding Limit

These two options control how closely spaced labels can be. Labels which are too closely crowded together will simply not be shown, since overplotting many labels together ends up with them being illegible. The **Spacing Threshold** slider controls the smallest area that a group of labels can have to themselves - if there are too many in the same area, none will be drawn. Sliding it left allows more crowding and right allows less. The **Crowding Limit** controls the largest number of labels that can be in a group. Setting it to 2 for instance is useful if you want to see pairs of labels, even if the pair is close.

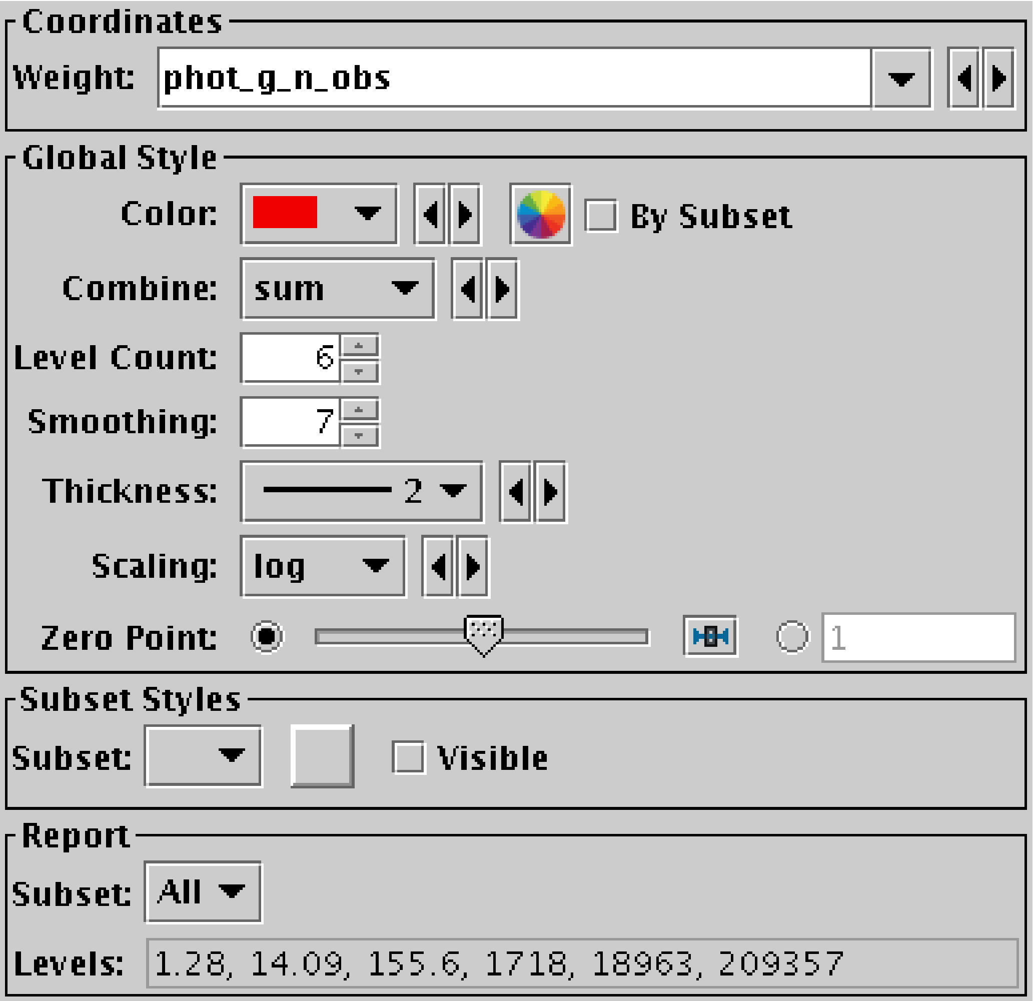
A.4.5.17 Contour Form

The **Contour** form (◎) plots position density contours. These provide another way (alongside the Auto, Density and Weighted shading modes) to visualise the characteristics of overdense regions in a crowded plot.

The contours are currently drawn as pixels rather than lines, so they don't look very beautiful in exported vector output formats (PDF, SVG, PostScript).



Example Contour plot



Coordinates

Weight: phot_g_n_obs

Global Style

Color: 🟥 ☐ By Subset

Combine: sum

Level Count: 6

Smoothing: 7

Thickness: 2

Scaling: log

Zero Point: ● ○ 1

Subset Styles

Subset: ☐ Visible

Report

Subset: All

Levels: 1.28, 14.09, 155.6, 1718, 18963, 209357

Contour form configuration panel

The configuration options are:

Weight
The weighting to apply to each input sample when producing the grid whose values are to be contoured. If supplied this is used in conjunction with the **Combine** selection. If left blank, an effective weighting of unity is used, and only smoothed point densities are contoured.

Color
The colour of the contour lines. If overlaid on top of other plot types that use the same colour, you may want to change this from its default value.

Combine
Defines the way that the weight values are combined when generating the value grid for which the contours will be plotted. If a weighting is supplied, the most useful values are `mean` which traces the mean values of a quantity and `sum` which traces the weighted sum. Other values such as `median` are of dubious validity because of the way that the smoothing is done.

This value is ignored if the weighting coordinate **Weight** is not set.

The available options are:

- `sum`: the sum of all the combined values per bin
- `mean`: the mean of the combined values
- `median`: the median of the combined values (may be slow)
- `stdev`: the sample standard deviation of the combined values
- `min`: the minimum of all the combined values
- `max`: the maximum of all the combined values

- `count`: the number of non-blank values per bin (weight is ignored)

Level Count
The number of contours drawn. In fact this is an upper limit; if there is not enough variation in the plot's density, then fewer contour lines will be drawn.

Smoothing
The size of the smoothing kernel applied to the density before performing the contour determination. If set too low the contours will be too crinkly, and if too high they will lose definition.

Thickness
The thickness in pixels of contour lines.

Scaling
How the smoothed density is treated before the contours levels are determined. Options are linear, logarithmic and equal area.

Zero Point
Determines the level at which the first contour (and hence all the others, which are separated from it by a fixed amount) are drawn. By sliding this from side to side you can sweep the contours over the density range and get a good idea of where interesting features lie.

The plot reports the contour levels it has used:

Levels
The actual values at which contours have been plotted. This can be useful for weighted plots using with a **Combine** value of **mean**, but for other combinations it may not have much physical meaning.

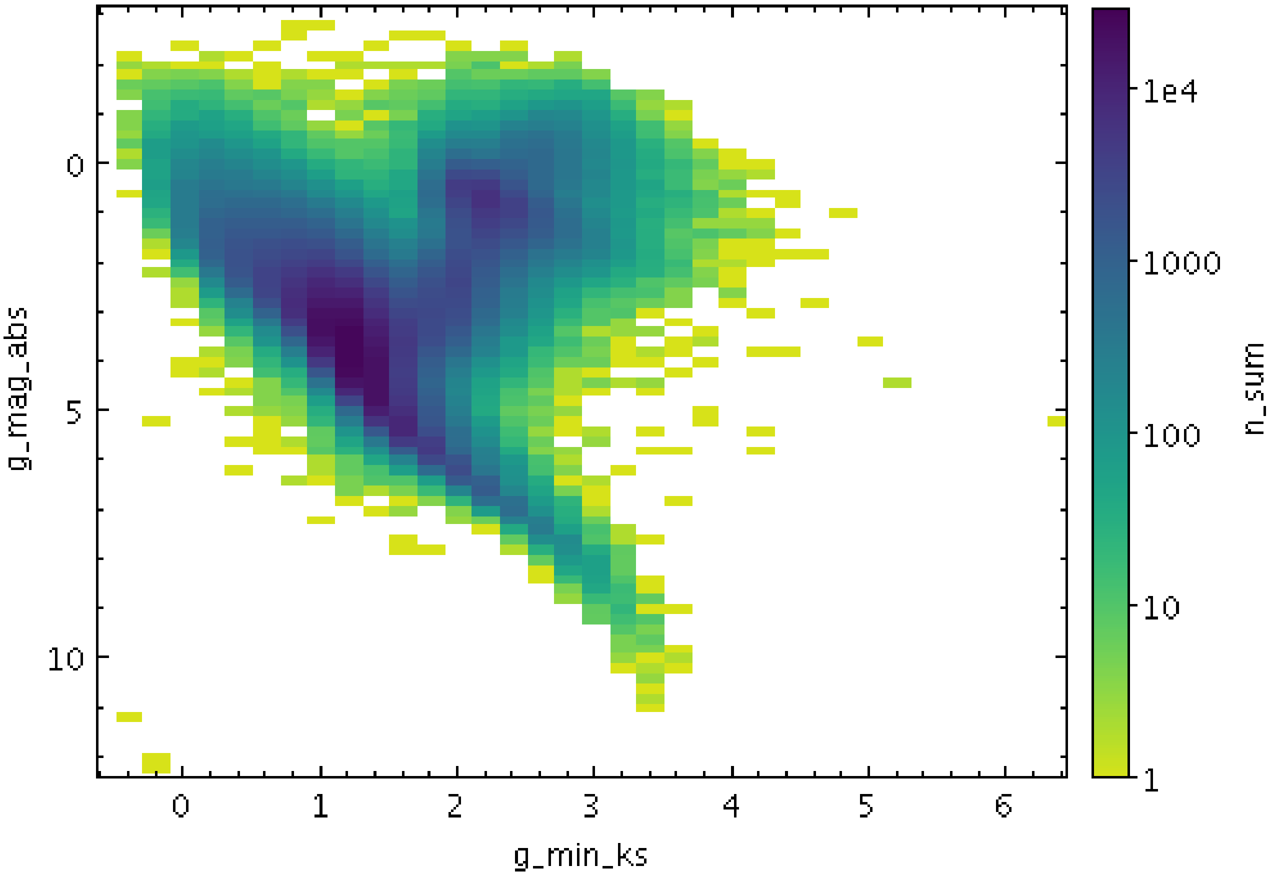
A.4.5.18 Grid Form

The **Grid** form ( ) plots 2-d point data aggregated into rectangular cells on the plotting plane.

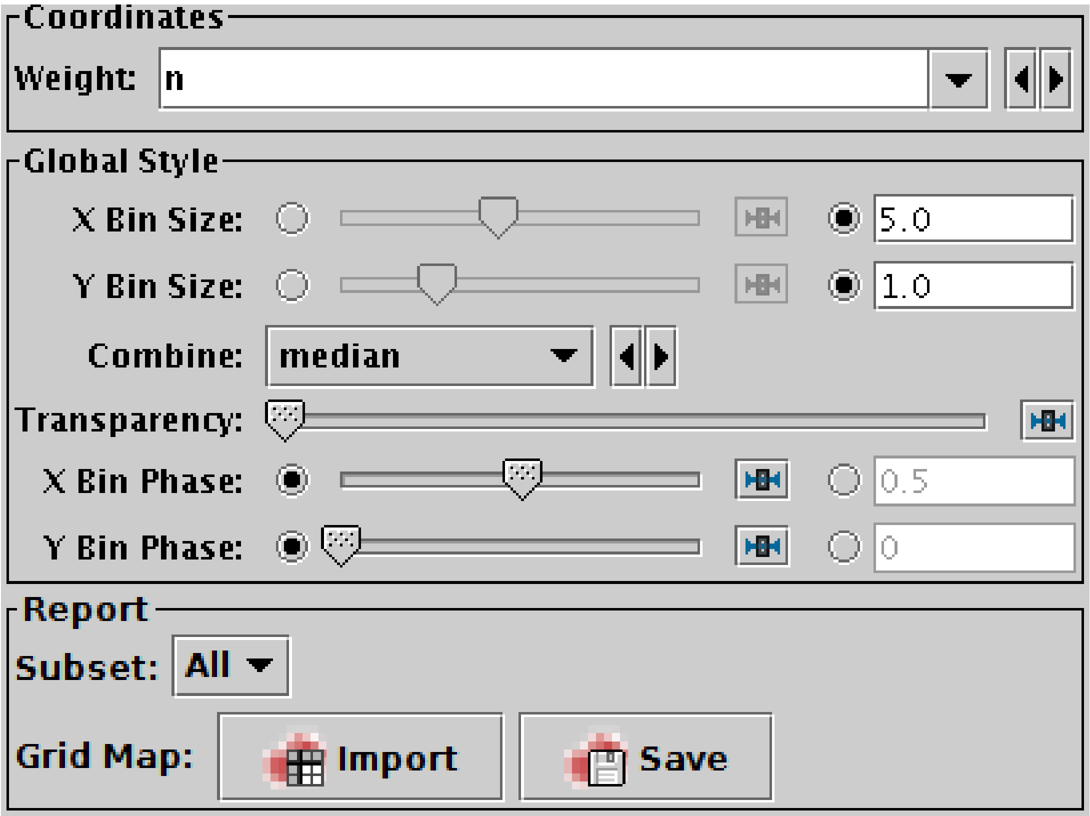
You can optionally use a weighting for the points, and you can configure how the values are combined to produce the output pixel values (colours). You can use this layer type in various ways, including as a 2-d histogram or weighted density map, or to plot gridded data.

The X and Y dimensions of the grid cells (or equivalently histogram bins) can be configured in terms of either the data coordinates or relative to the plot dimensions.

The shading is done using the shared colour map. This colour map is used by all currently visible **Grid**, Aux and Weighted layers. When at least one such layer is being plotted, the Aux Axis control (Appendix A.4.3.5) is visible in the control panel, which allows you to configure the colour map, range, ramp display etc.



Example Grid plot



Coordinates

Weight: n

Global Style

X Bin Size: ○ ● 5.0

Y Bin Size: ○ ● 1.0

Combine: median

Transparency:

X Bin Phase: ● ○ 0.5

Y Bin Phase: ● ○ 0

Report

Subset: All

Grid Map: Import Save

Grid form configuration panel

The configuration options are:

Weight

The weight value applied to each plotted point. Fill this in with a column name or expression from the table just like for a positional coordinate. The exact way this quantity is used depends on the setting of the **Combine** control below. If it's left blank, the weighting is considered to be unity (all values are 1); this makes sense for some combination types (e.g. `sum`) but not others (e.g. `mean`).

X Bin Size
Y Bin Size

A scale for the horizontal/vertical extent of of the rectangular bins into which the data is aggregated. There are two ways to specify this. If the left-hand radio button is selected, the adjacent slider will adjust the bin size, which is also affected by the actual width of the plotting window in pixels. Slide the slider left to get narrower bins or right to get wider ones. If the right-hand radio button is selected, you can enter a numeric value giving the actual extent in data units of the dimension. If the axis in question is logarithmic, this value is a factor.

Combine

Determines how the **Weight** values for the points falling within a given data bin are combined to produce the numeric value used for that bin's colour. For unweighted values (a pure density map) it usually makes sense to use `count` or `sum`. However, if there is a non-blank **Weight** coordinate, one of the other values such as `mean` may be more revealing.

The following options (some are more useful than others) are currently available:

- `sum`: the sum of all weights
- `sum-per-unit`: the sum of all weights per unit of bin size
- `mean`: the mean of all weights
- `median`: the median of all weights (can be slow)
- `q1`: the first quartile of all weights (can be slow)
- `q3`: the third quartile of all weights (can be slow)
- `min`: the minimum weight
- `max`: the maximum weight
- `stdev`: the sample standard deviation of all weights
- `count`: the number of points plotted (weight value is ignored, this is like Density mode)
- `count-per-unit`: the number of points plotted per unit of bin size (weight value is ignored)
- `hit`: one if any point is plotted, blank otherwise (weight value is ignored, this is like Flat mode)

Transparency

Adjusts the transparency of the filled area.

X Bin Phase
Y Bin Phase

Controls where the horizontal/vertical zero point for bins is set. For instance if your X/Y bin size is 1, it controls whether bin boundaries on the X/Y axis are at 0, 1, 2, ... or 0.5, 1.5, 2.5, ... etc. If the slider is at either end of the scale, there will be a bin boundary at X/Y=0 (linear axis) or X/Y=1 (logarithmic axis).

The **Report** panel provides information calculated by the plot:

Grid Map

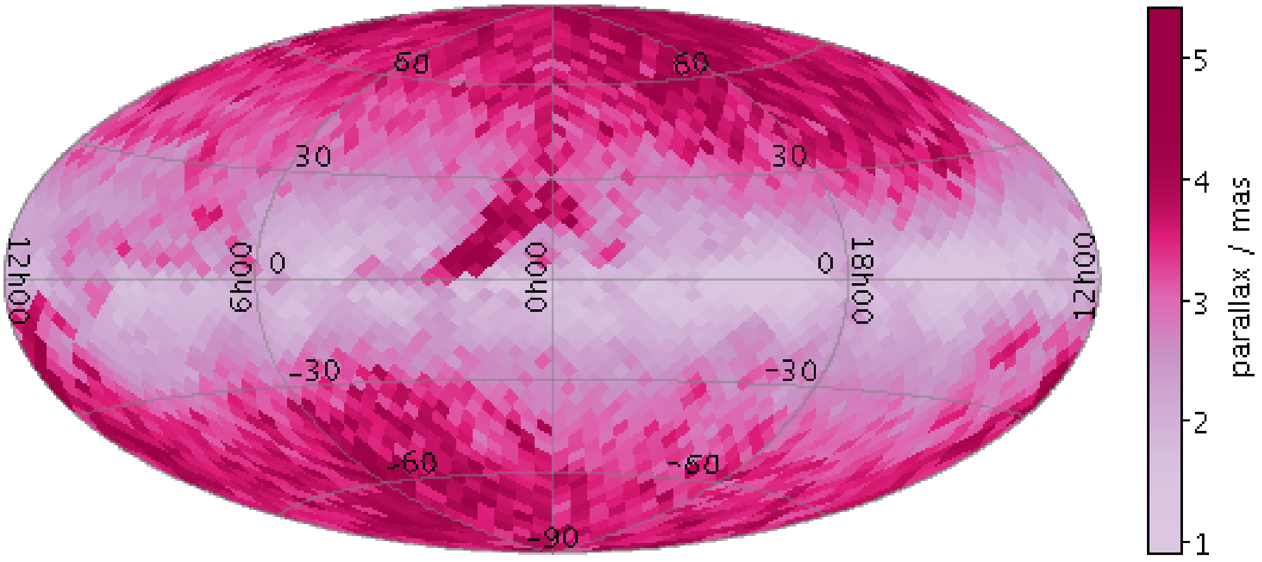
This allows you to export the grid data that you can see in the plot in the form of a table. The table has one row for each plotted cell, with columns giving the central, lower and upper bounds for each the X and Y grid coordinates, as well as a column giving the bin value, corresponding to the colour in the plot. The **Import** (▦) option loads the data as a new table

) option lets you save it directly to disk in one of the available table formats. This information is also available from the **Export** menu.

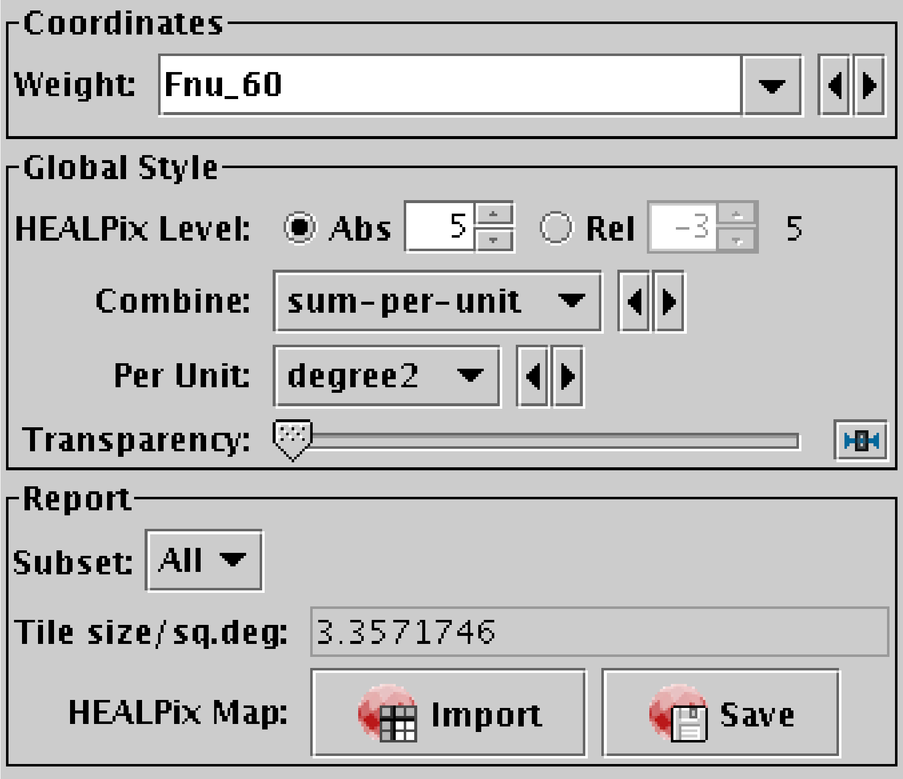
A.4.5.19 SkyDensity Form

The **SkyDensity** form () plots a density map on the sky using a HEALPix grid with a configurable resolution. You can optionally use a weighting for the data values to accumulate within each HEALPix tile, and you can configure how the weighted values are combined to generate the eventual pixel values (and hence colours). HEALPix is a tiling scheme for the sky which uses square-ish pixels of equal area to cover the celestial sphere.

The shading is done using the shared colour map. This colour map is used by all currently visible **SkyDensity**, Grid, Aux and Weighted layers. When at least one such layer is being plotted, the Aux Axis control (Appendix A.4.3.5) is visible in the control panel, which allows you to configure the colour map, range, ramp display etc.



Example SkyDensity plot



Coordinates

Weight: Fnu_60

Global Style

HEALPix Level: ● Abs 5 ○ Rel -3 5

Combine: sum-per-unit

Per Unit: degree2

Transparency:

Report

Subset: All

Tile size/sq.deg: 3.3571746

HEALPix Map: Import Save

SkyDensity form configuration panel

The configuration options are:

Weight
The weight value applied to each plotted point. Fill this in with a column name or expression from the table just like for a positional coordinate. The exact way this quantity is used depends on the setting of the **Combine** control below. If it's left blank, the weighting is considered to be unity (all values are 1); this makes sense for some combination types (e.g. sum) but not others (e.g. mean).

HEALPix Level
This allows you to control the resolution of the HEALPix grid onto which the weighted values are resampled. According to the radio buttons, you can configure this using either a **Abs**olute or **Rel**ative value. In Absolute mode you specify the HEALPix level (*k*) directly; the number of pixels on the sky is $12*2^{2k}$. In Relative mode the level is set so that a HEALPix tile has approximately the given number of screen pixels along a side, and hence the absolute level will change if you zoom in and out. In either case, you can see the absolute level at the right hand side of the control.

Combine
Determines how the weight values associated with markers plotted covering a given HEALPix tile are combined to produce the numeric value used for that tile's colour.

For density-like values (`count-per-unit`, `sum-per-unit`) the scaling is additionally influenced by the **Per Unit** option.

The following options (some are more useful than others) are currently available:

- `sum`: the sum of all the combined values per bin
- `sum-per-unit`: the sum of all the combined values per unit of bin size
- `count`: the number of non-blank values per bin (weight is ignored)
- `count-per-unit`: the number of non-blank values per unit of bin size (weight is ignored)
- `mean`: the mean of the combined values
- `median`: the median of the combined values (may be slow)
- `q1`: the first quartile of the combined values (may be slow)

- `q3`: the third quartile of the combined values (may be slow)
- `min`: the minimum of all the combined values
- `max`: the maximum of all the combined values
- `stdev`: the sample standard deviation of the combined values
- `hit`: 1 if any values present, NaN otherwise (weight is ignored)

Per Unit

Defines the unit of sky area used for scaling density-like **Combine** values (e.g. `count-per-unit` or `sum-per-unit`). If the Combination mode is calculating values per unit area, this configures the area scale in question. For non-density-like combination modes (e.g. `sum` or `mean`) it has no effect.

The available options are:

- `steradian`: steradian
- `degree2`: square degree
- `arcmin2`: square arcminute
- `arcsec2`: square arcsecond
- `mas2`: square milliarcsec
- `uas2`: square microarcsec

Opaque Limit

Determines transparency of the points. By default, they are fully opaque, but if you slide the slider to the right, they will become progressively more transparent.

The **Report** panel provides information calculated by the plot:

HEALPix Level

Reports the actual HEALPix level of the plotted tiles. This is not necessarily the value selected in the style configuration controls, since if tiles would be smaller than screen pixels, the gridding is automatically degraded. This value is also reported to the right of the HEALPix Level configuration control.

Tile size/sq.deg

Reports the size of each plotted tile in square degrees. This is simply a function of the actual HEALPix level.

HEALPix Map

This allows you to export the healpix map that you can see in the plot in the form of a table. The table has a row for each plotted healpix pixel, with two columns: one giving the healpix pixel index, and the other giving the pixel value (corresponding to the colour in the plot). The **Import** () option loads the data as a new table in the TOPCAT application, and the **Save** () option lets you save it directly to disk in one of the available table formats. This information is also available from the **Export** menu.

Some notes apply for this export:

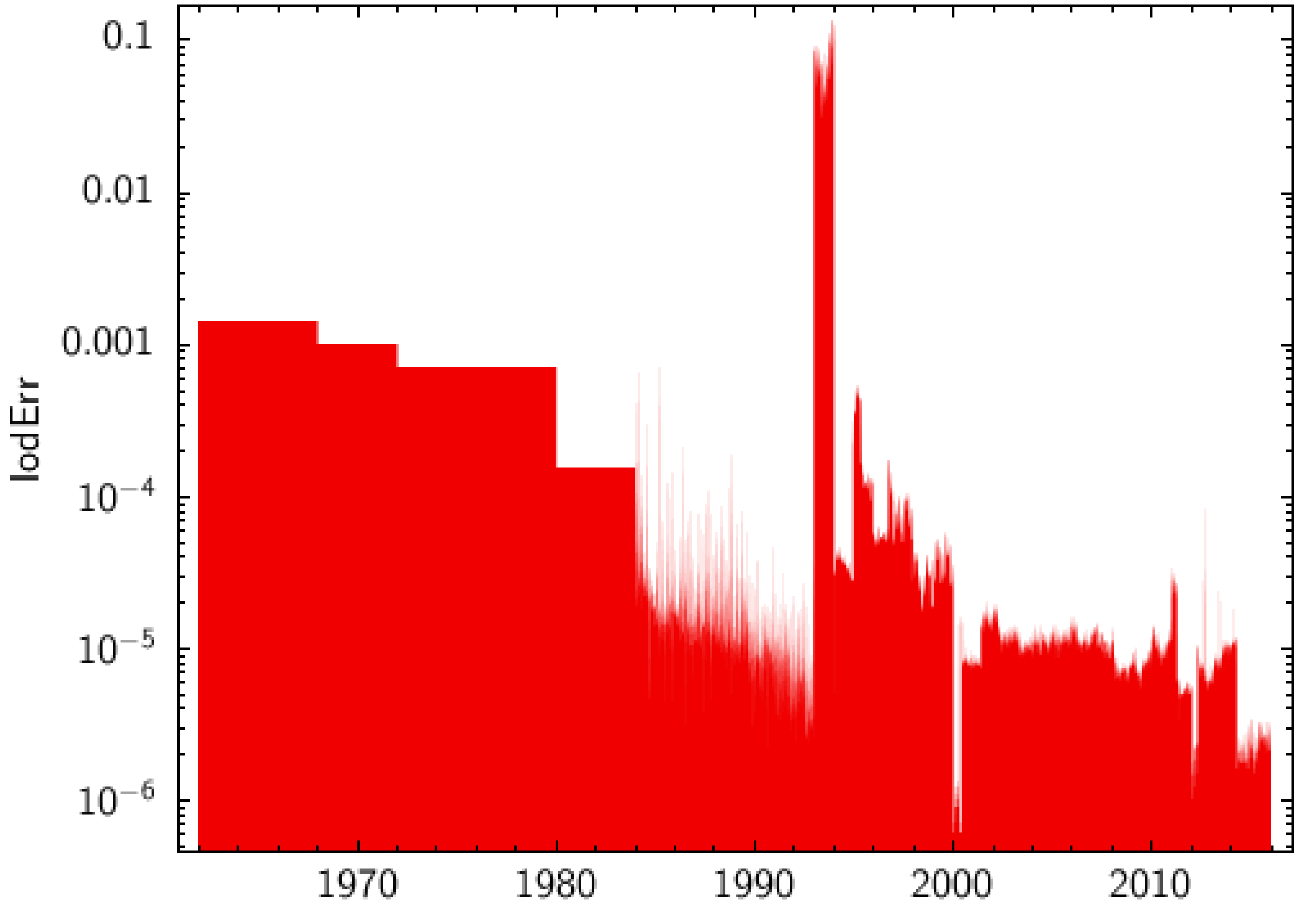
1. If you want to export the table following the HEALPix-FITS convention in a form suitable for use in other applications such as Aladin, you should save it using the output format **fits-healpix** rather than one of the other FITS variants.
2. When exported as FITS, the sky coordinate system (FITS `COORDSYS` header) is determined by the current value of the **View Sky System**, as reported in the Sky Axes Control.
3. The HEALPix level at which the data is exported is that currently plotted as reported by the **HEALPix Level** report item above, not necessarily the one selected in the style configuration controls.

A.4.5.20 Fill Form

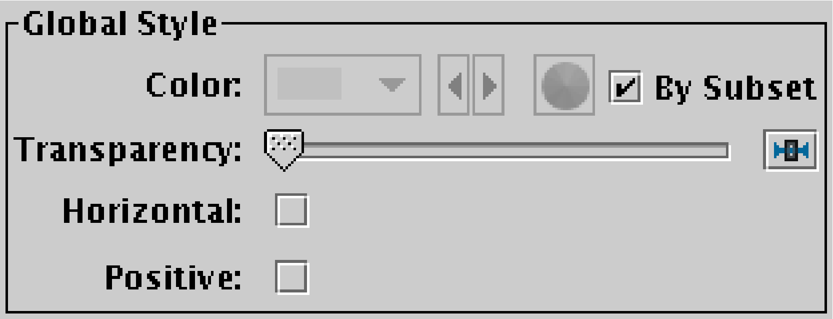
If a two-dimensional dataset represents a single-valued function, the **Fill** form ( ) will fill the area underneath the function's curve with a solid colour. Parts of the surface which would only be partially covered (because of rapid function variation within the width of a single pixel) are represented using appropriate alpha-blending. The filled area may alternatively be that above the curve or (in some plot types) to its left or right.

One example of its use is to reconstruct the appearance of a histogram plot from a set of histogram bins. For X,Y data which is not single-valued, the result may not be very useful.

This form may be used in the Plane or Time plot windows.



Example Fill plot


Global Style

Color: ☑ By Subset

Transparency:

Horizontal: □

Positive: □

Fill form configuration panel

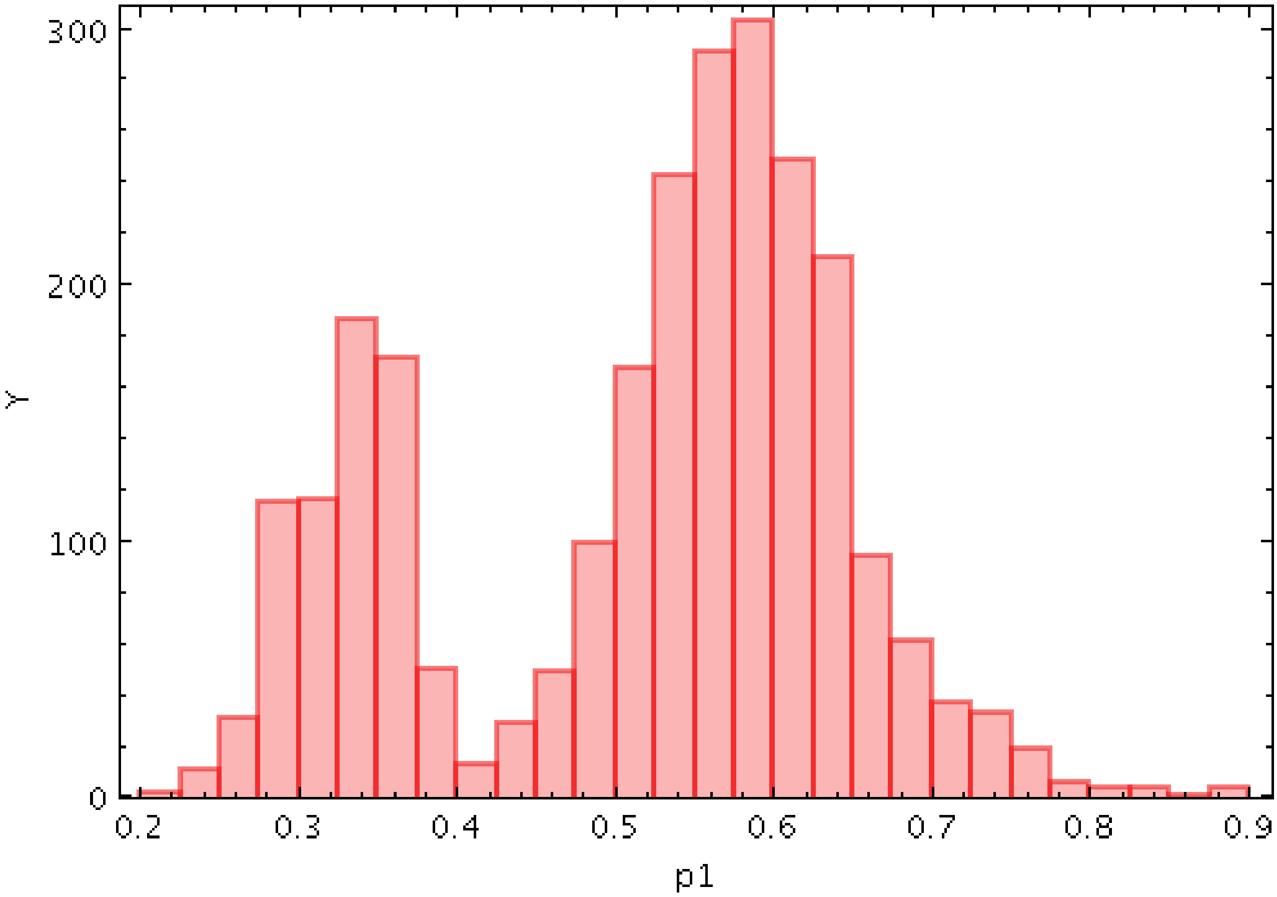
The configuration options are:

Transparency
Adjusts the transparency of the filled area.

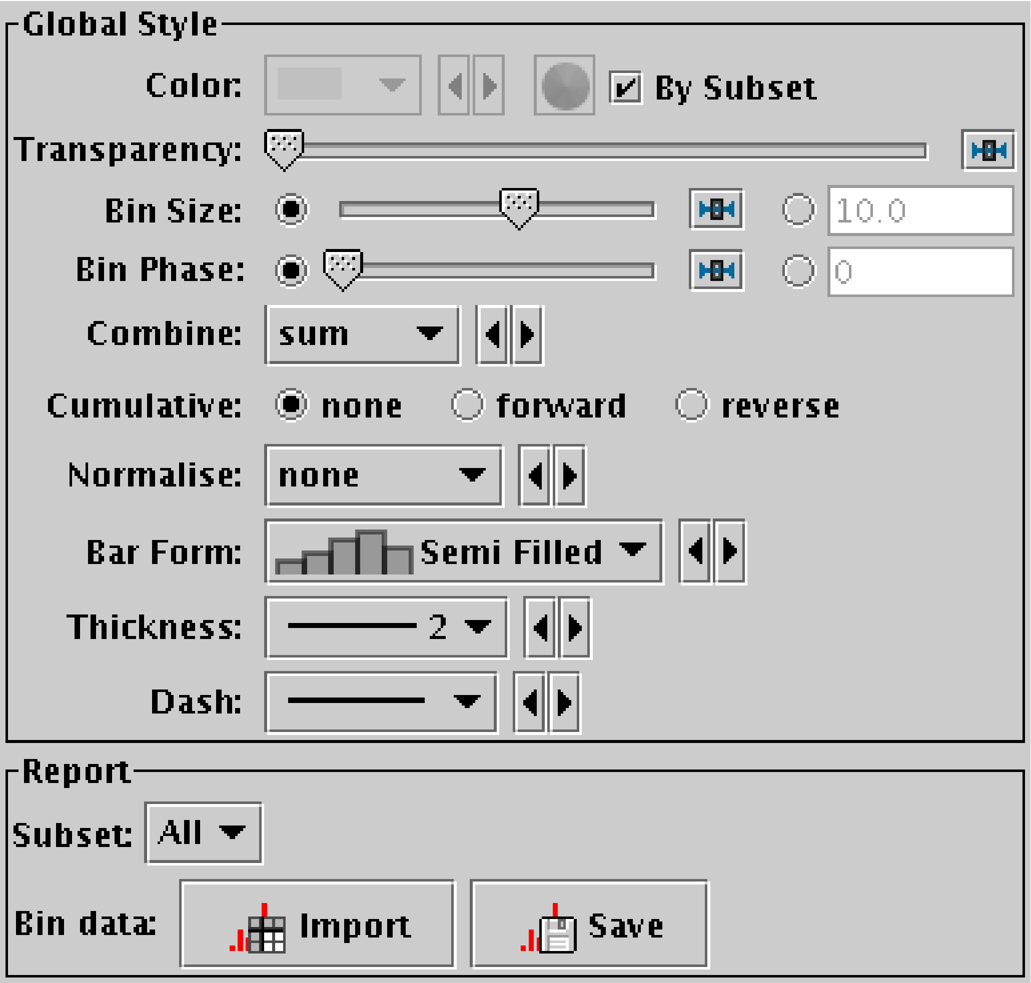
Horizontal
Determines whether the filling is vertical (suitable for functions of the horizontal variable) or horizontal (suitable for functions of the vertical variable). In the Time plot, the fill is always vertical, so this option is not provided.

Positive
Determines the directional sense of the filling. If false, the fill is between the data points and negative infinity along the relevant axis (e.g. down from the data points to the bottom of the plot). If true, the fill is in the other direction.

A.4.5.21 Histogram Form



Example Histogram plot


Global Style

Color: ☑ By Subset

Transparency:

Bin Size: ● ○ 10.0

Bin Phase: ● ○ 0

Combine: sum

Cumulative: ● none ○ forward ○ reverse

Normalise: none

Bar Form: Semi Filled

Thickness: 2

Dash:

Report

Subset: All

Bin data: Import Save

Histogram form configuration panel

The **Histogram** form ( ) plots a histogram generated by binning samples along the X axis.

This form may be used in the Histogram, Plane or Time plot windows.

These options always appear in the form configuration panel:

Colour
Selects the basic colour for the dataset.

Transparency
Adjusts the transparency of the bars or line. For bar styles which are already partially transparent, this fades them further.

Combine
Defines how values contributing to the same bin are combined together to produce the value assigned to that bin, and hence its height. The combined values are those given by the **Weight** coordinate, but if no weight is supplied, a weighting of unity is assumed.

The available options are:

- `sum`: the sum of all the combined values per bin
- `sum-per-unit`: the sum of all the combined values per unit of bin size
- `count`: the number of non-blank values per bin (weight is ignored)
- `count-per-unit`: the number of non-blank values per unit of bin size (weight is ignored)
- `mean`: the mean of the combined values
- `median`: the median of the combined values (may be slow)
- `q1`: the first quartile of the combined values (may be slow)

- `q3`: the third quartile of the combined values (may be slow)
- `min`: the minimum of all the combined values
- `max`: the maximum of all the combined values
- `stdev`: the sample standard deviation of the combined values
- `hit`: 1 if any values present, NaN otherwise (weight is ignored)

Bar Form
Type of histogram bars: filled, open, steps, spikes etc.

Thickness
Controls line thickness where applicable. This is only relevant for bar styles that draw a finite thickness line, so has no effect for solid bars.

Dash
Controls the dash pattern of the line (solid, dots, dashes etc). This is only relevant for bar styles that draw a finite thickness line, so has no effect for solid bars.

And these options appear in the form configuration panel for the Plane window, or the Bins control (📊) for the Histogram window:

Bin Size
A scale for the width of bins that are shown on the screen. There are two ways to specify this. If the left-hand radio button is selected, the adjacent slider will adjust the bin size, which is also affected by the actual width of the plotting window in pixels. Slide the slider left to get narrower bins or right to get wider ones. If the right-hand radio button is selected, you can enter a numeric value giving the actual width in data units of each bar (for a logarithmic X axis this value is a factor).

Bin Phase
Controls where the horizontal zero point for binning is set. For instance if your bin size is 1, it controls whether bin boundaries are at 0, 1, 2, .. or 0.5, 1.5, 2.5, ... etc. If the slider is at either end of the scale, there will be a bin boundary at X=0 (linear X axis) or X=1 (logarithmic X axis).

Cumulative
If set to **forward** or **reverse**, the histogram bars are plotted cumulatively; each bin includes the counts from all previous bins in the direction of negative or positive infinity.

Normalise
Defines how, if at all, the bars are normalised. The available options are:

- **none**: No normalisation is performed.
- **area**: The total area of histogram bars is normalised to unity. For cumulative plots, this behaves like **height**.
- **unit**: Histogram bars are scaled by the inverse of the bin width in data units. For cumulative plots, this behaves like **none**.
- **maximum**: The height of the tallest histogram bar is normalised to unity. For cumulative plots, this behaves like **height**.
- **height**: The total height of histogram bars is normalised to unity.

When used in the Time plot only, additional options **per_second**, **per_day** etc are available corresponding to the frequency over the named time unit.

The **Report** panel gives you access to the bin values calculated by the plot:

Bin Data
This allows you to export the bin values that you can see in the plot in the form of a table. The table has a row for each bin in the plotted histogram, with columns giving the mid-point, lower

bound and upper bound of each bin on the X axis, and its height. The **Import** ( ) option loads the data as a new table in the TOPCAT application, and the **Save** ( ) option lets you save it directly to disk in one of the available table formats. This information is also available from the **Export** menu.

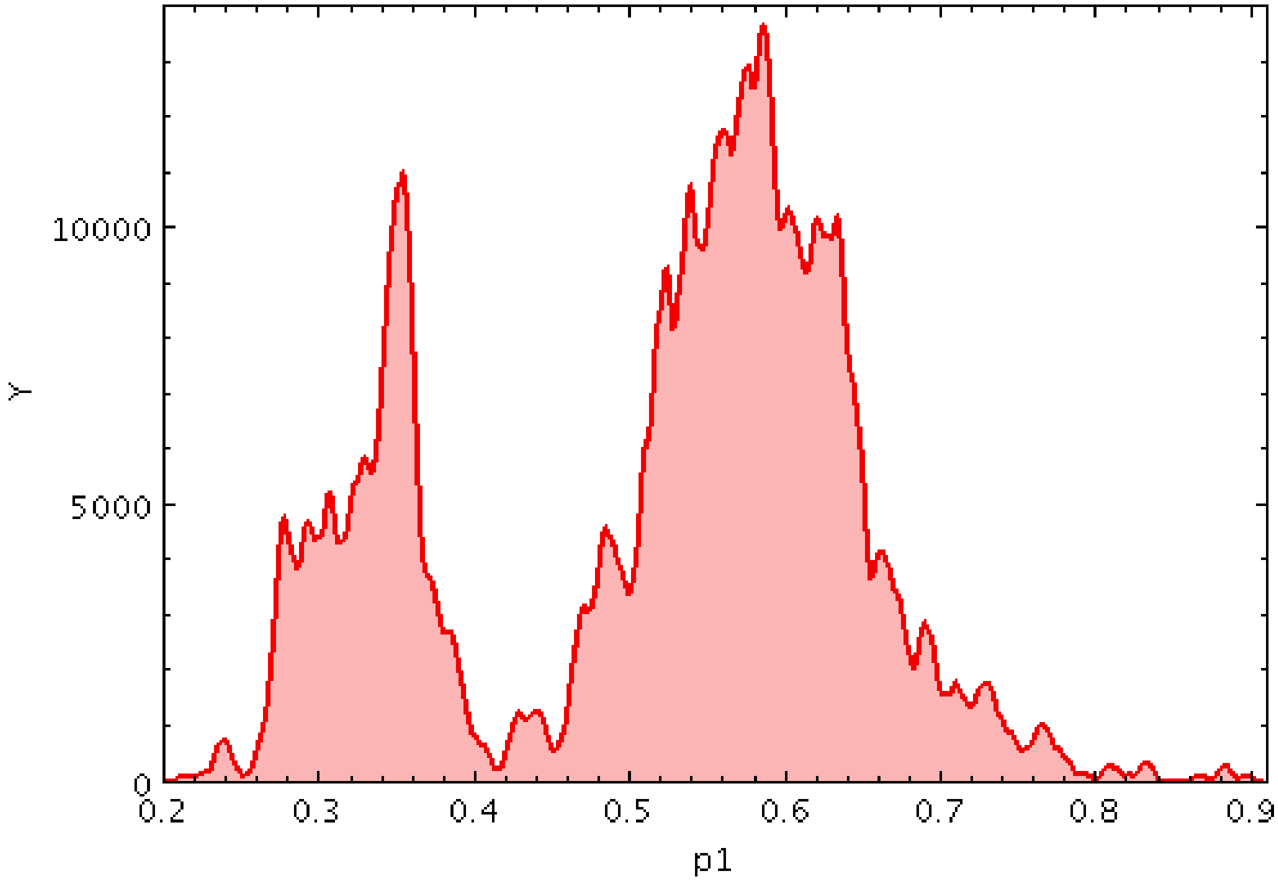
A.4.5.22 KDE Form

The **KDE** form ( ) plots a discrete Kernel Density Estimate giving a smoothed frequency of data values along the horizontal axis, using a fixed-width smoothing kernel. (for a variable-bandwidth kernel, see KNN). This is a generalisation of a histogram in which the bins are always 1 pixel wide, and a smoothing kernel is applied to each bin. The width and shape of the kernel may be varied.

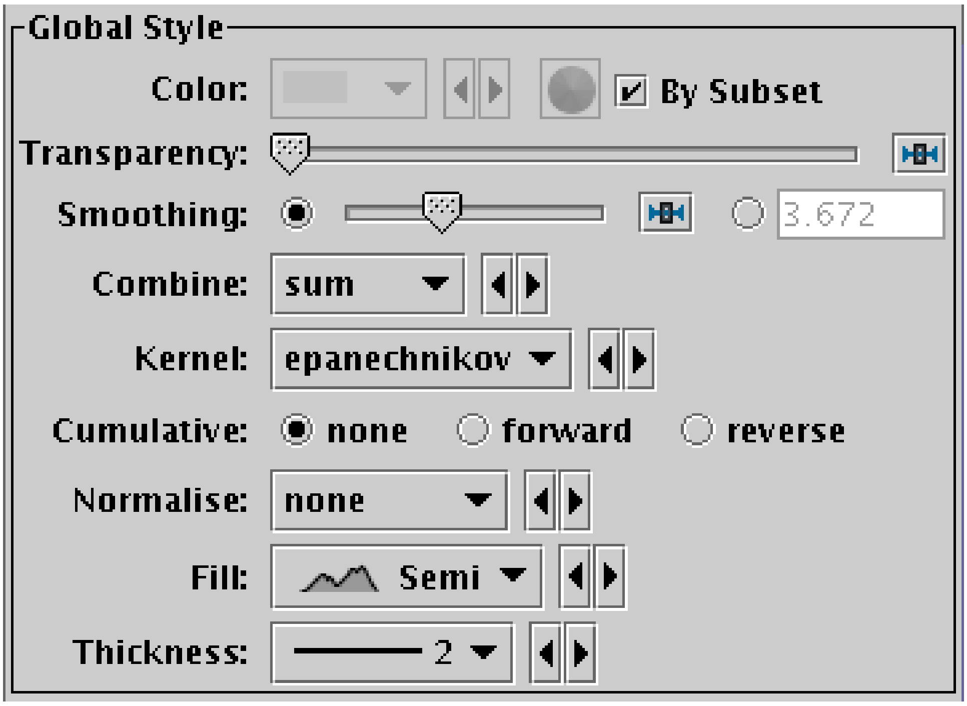
This is suitable for cases where the division into discrete bins done by a normal histogram is unnecessary or troublesome.

Note this is not a true Kernel Density Estimate, since, for performance reasons, the smoothing is applied to the (pixel-width) bins rather than to each data sample. The deviation from a true KDE caused by this quantisation will be at the pixel level, hence in most cases not visually apparent.

This form may be used in the Histogram, Plane or Time plot windows.



Example KDE plot



Color: ☑ By Subset

Transparency:

Smoothing: ● ○ 3.672

Combine: sum

Kernel: epanechnikov

Cumulative: ● none ○ forward ○ reverse

Normalise: none

Fill: Semi

Thickness: 2

KDE form configuration panel

These options always appear in the form configuration panel:

Colour
Selects the basic colour for the dataset.

Transparency
Adjusts the transparency of the bars or line. For bar styles which are already partially transparent, this fades them further.

Combine
Defines how values contributing to the same bin are combined together to produce the value assigned to that bin, and hence its height. The bins in this case are 1-pixel wide, so lack much physical significance. This means that while some combination modes, such as `sum-per-unit` and `mean` make sense, others such as `sum` do not.

The combined values are those given by the **Weight** coordinate, but if no weight is supplied, a weighting of unity is assumed.

The available options are:

- `sum`: the sum of all the combined values per bin
- `sum-per-unit`: the sum of all the combined values per unit of bin size
- `count`: the number of non-blank values per bin (weight is ignored)
- `count-per-unit`: the number of non-blank values per unit of bin size (weight is ignored)
- `mean`: the mean of the combined values
- `median`: the median of the combined values (may be slow)
- `q1`: the first quartile of the combined values (may be slow)
- `q3`: the third quartile of the combined values (may be slow)
- `min`: the minimum of all the combined values
- `max`: the maximum of all the combined values
- `stdev`: the sample standard deviation of the combined values
- `hit`: 1 if any values present, NaN otherwise (weight is ignored)

Fill
Determines how the density function is represented. The options are:

- **solid**: area between level and axis is filled with solid colour

- **line**: level is marked by a wiggly line
- **semi**: level is marked by a wiggly line, and area below it is filled with a transparent colour

Thickness
Controls line thickness where applicable. This is only relevant for bar styles that draw a finite thickness line, so has no effect for **solid** filling.

And these options appear in the form configuration panel for the Plane window, or the Bins control ( ) for the Histogram window:

Smoothing
Configures the smoothing width for kernel density estimation. This is the characteristic width of the kernel function to be convolved with the density to produce the visible plot.

Sliding the slider to the right makes the kernel width larger. The width in data units is shown in the text field on the right (if the X axis is logarithmic, this is a factor). Alternatively you can click the radio button near the text field, and enter the width in data units directly.

Kernel
The functional form of the smoothing kernel. The functions listed refer to the unscaled shape; all kernels are normalised to give a total area of unity.

The available options are:

- **square**: Uniform value: f(x)=1, |x|=0..1
- **linear**: Triangle: f(x)=1-|x|, |x|=0..1
- **epanechnikov**: Parabola: f(x)=1-x*x, |x|=0..1
- **cos**: Cosine: f(x)=cos(x*pi/2), |x|=0..1
- **cos2**: Cosine squared: f(x)=cos^2(x*pi/2), |x|=0..1
- **gauss3**: Gaussian truncated at 3.0 sigma: f(x)=exp(-x*x/2), |x|=0..3
- **gauss6**: Gaussian truncated at 6.0 sigma: f(x)=exp(-x*x/2), |x|=0..6

Cumulative
If set to **forward** or **reverse**, the heights are plotted cumulatively; each bin includes the counts from all previous bins in the direction of negative or positive infinity.

Normalise
Defines how, if at all, the bars are normalised. The available options are:

- **none**: No normalisation is performed.
- **area**: The total area of histogram bars is normalised to unity. For cumulative plots, this behaves like **height**.
- **unit**: Histogram bars are scaled by the inverse of the bin width in data units. For cumulative plots, this behaves like **none**.
- **maximum**: The height of the tallest histogram bar is normalised to unity. For cumulative plots, this behaves like **height**.
- **height**: The total height of histogram bars is normalised to unity.

When used in the Time plot only, additional options **per_second**, **per_day** etc are available corresponding to the frequency over the named time unit.

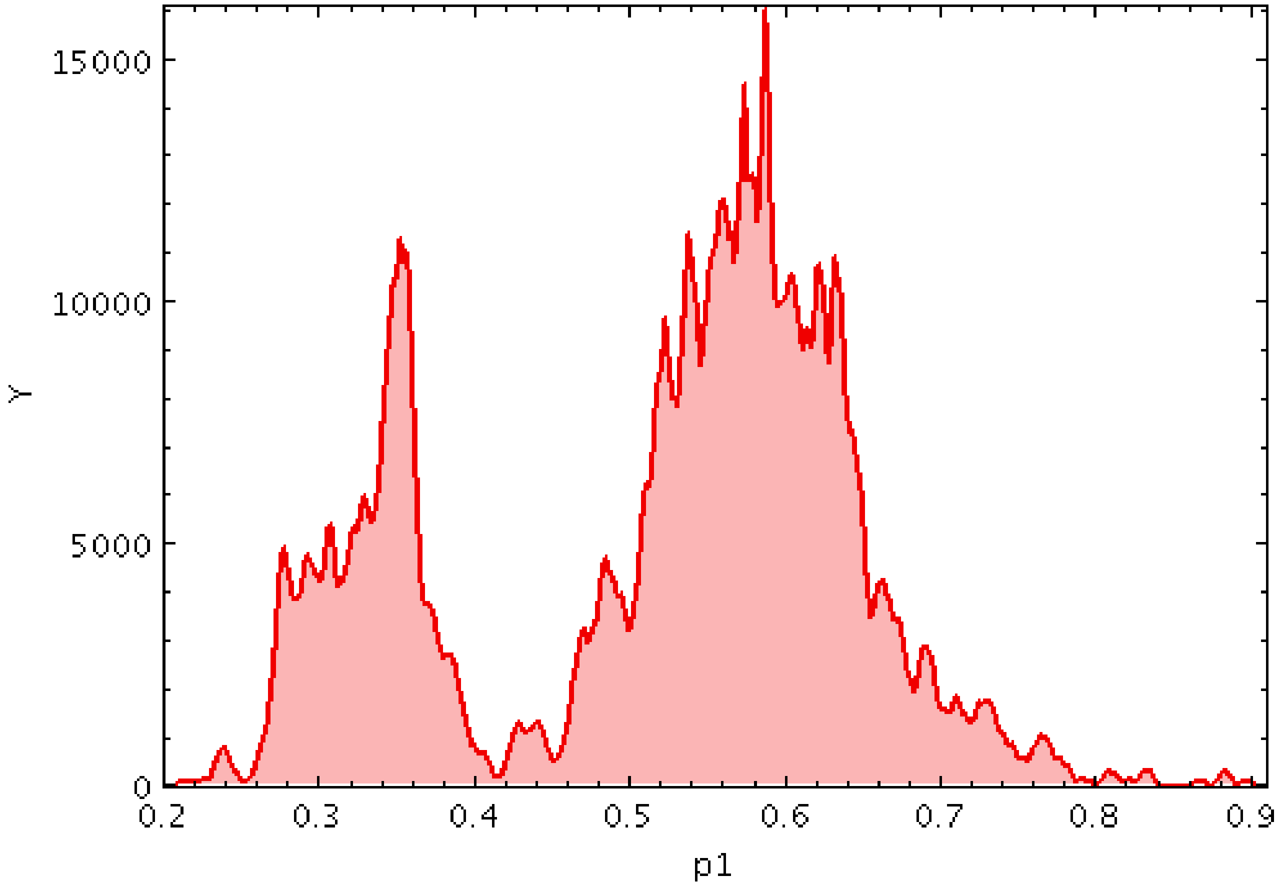
A.4.5.23 KNN Form

The **KNN** form ( ) plots a discrete Kernel Density Estimate giving a smoothed frequency of data values along the horizontal axis, using an adaptive (K-Nearest-Neighbours) smoothing kernel. This is a generalisation of a histogram in which the bins are always 1 pixel wide, and a variable-bandwidth smoothing kernel is applied to each bin (for a fixed-bandwidth kernel, see KDE).

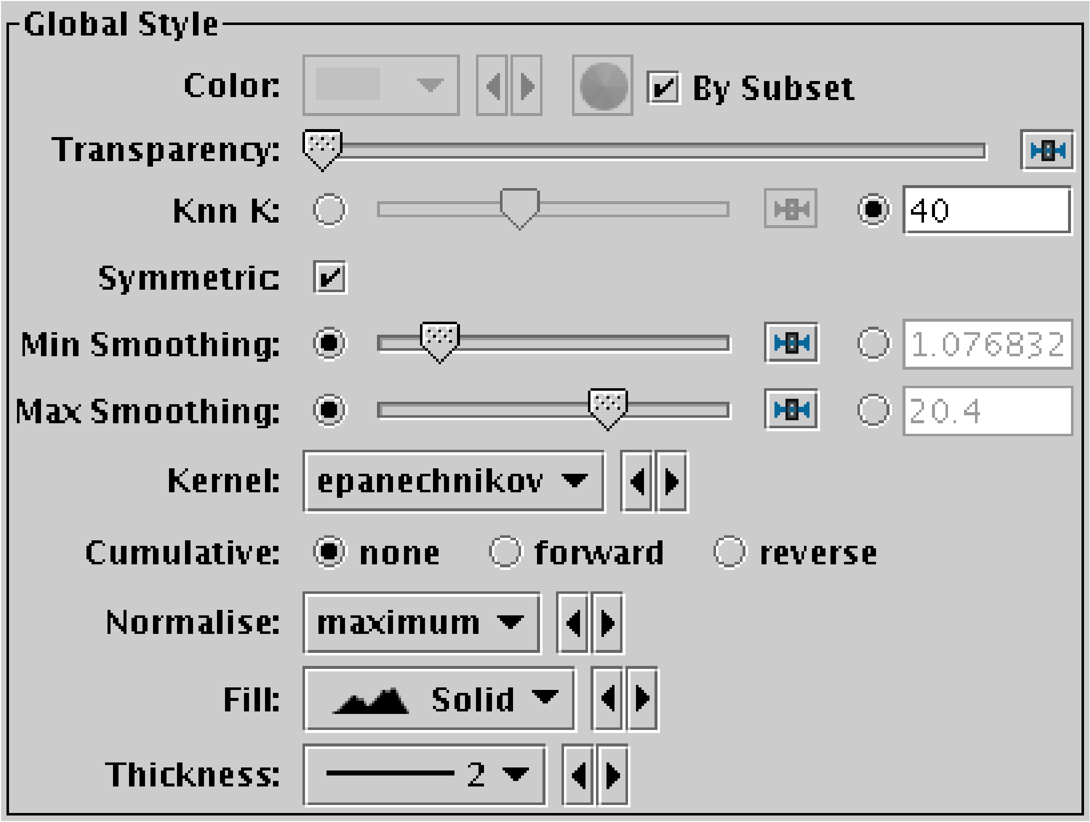
The K-Nearest-Neighbour figure gives the number of points in each direction to determine the width of the smoothing kernel for smoothing each bin. Upper and lower limits for the kernel width are also supplied; if the upper and lower limits are equal, this is equivalent to a fixed-width kernel.

Note this is not a true Kernel Density Estimate, since, for performance reasons, the smoothing is applied to the (pixel-width) bins rather than to each data sample. The deviation from a true KDE caused by this quantisation will be at the pixel level, hence in most cases not visually apparent.

This form may be used in either the Histogram, Plane or Time plot windows.



Example KNN plot



Global Style

Color: ☑ By Subset

Transparency:

Knn K: ○ ● 40

Symmetric ☑

Min Smoothing: ● ○ 1.076832

Max Smoothing: ● ○ 20.4

Kernel: epanechnikov

Cumulative: ● none ○ forward ○ reverse

Normalise: maximum

Fill: Solid

Thickness: 2

KNN form configuration panel

These options always appear in the form configuration panel:

Colour
Selects the basic colour for the dataset.

Transparency
Adjusts the transparency of the bars or line. For bar styles which are already partially transparent, this fades them further.

Knn K
Sets the number of nearest neighbours to count away from a sample point to determine the width of the smoothing kernel at that point. For the symmetric case this is the total of nearest neighbours summed over both directions, and for the asymmetric case it is the number in a single direction. The threshold is actually the weighted total of samples; for unweighted (weight=1) bins that is equivalent to the number of samples.

Symmetric
If checked, the nearest neigbour search is carried out in both directions, and the kernel is symmetric. If unchecked, the nearest neigbour search is carried out separately in the positive and negative directions, and the kernel width is accordingly different in the positive and negative directions.

Min Smoothing
Fixes the minimum size of the smoothing kernel. This functions as a lower limit on the distance that is otherwise determined by searching for the K nearest neighbours at each sample point.

You can either use the slider, or check the radio button on the right and enter the value in data units directly.

Max Smoothing
Fixes the maximum size of the smoothing kernel. This functions as an upper limit on the distance that is otherwise determined by searching for the K nearest neighbours at each sample

point.

You can either use the slider, or check the radio button on the right and enter the value in data units directly.

Fill

Determines how the density function is represented. The options are:

- **solid**: area between level and axis is filled with solid colour
- **line**: level is marked by a wiggly line
- **semi**: level is marked by a wiggly line, and area below it is filled with a transparent colour

Thickness

Controls line thickness where applicable. This is only relevant for bar styles that draw a finite thickness line, so has no effect for **solid** filling.

And these options appear in the form configuration panel for the Plane window, or the Bins control ( ) for the Histogram window:

Kernel

The functional form of the smoothing kernel. The functions listed refer to the unscaled shape; all kernels are normalised to give a total area of unity.

The available options are:

- **square**: Uniform value: f(x)=1, |x|=0..1
- **linear**: Triangle: f(x)=1-|x|, |x|=0..1
- **epanechnikov**: Parabola: f(x)=1-x*x, |x|=0..1
- **cos**: Cosine: f(x)=cos(x*pi/2), |x|=0..1
- **cos2**: Cosine squared: f(x)=cos^2(x*pi/2), |x|=0..1
- **gauss3**: Gaussian truncated at 3.0 sigma: f(x)=exp(-x*x/2), |x|=0..3
- **gauss6**: Gaussian truncated at 6.0 sigma: f(x)=exp(-x*x/2), |x|=0..6

Cumulative

If set to **forward** or **reverse**, the heights are plotted cumulatively; each bin includes the counts from all previous bins in the direction of negative or positive infinity.

Normalise

Defines how, if at all, the bars are normalised. The available options are:

- **none**: No normalisation is performed.
- **area**: The total area of histogram bars is normalised to unity. For cumulative plots, this behaves like **height**.
- **unit**: Histogram bars are scaled by the inverse of the bin width in data units. For cumulative plots, this behaves like **none**.
- **maximum**: The height of the tallest histogram bar is normalised to unity. For cumulative plots, this behaves like **height**.
- **height**: The total height of histogram bars is normalised to unity.

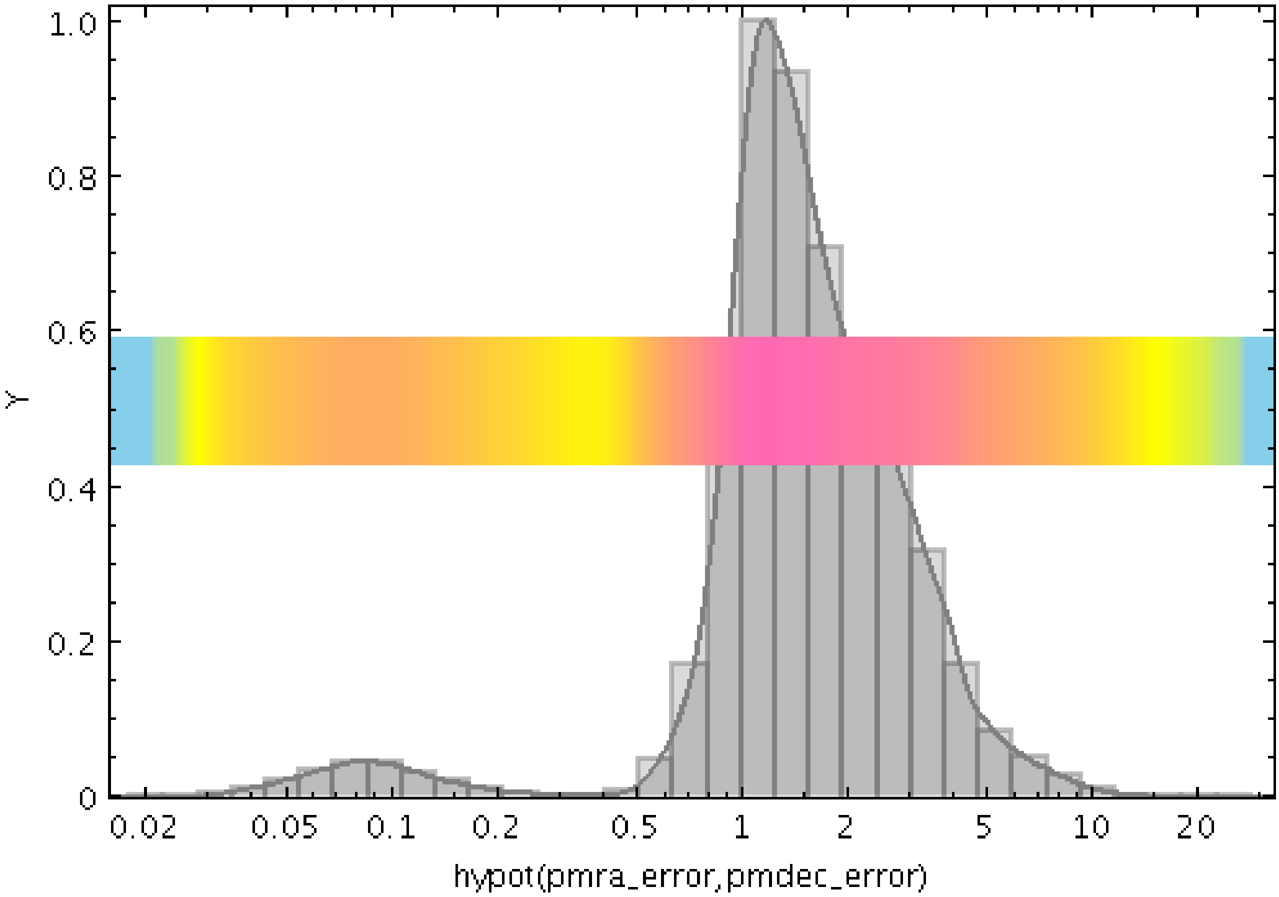
When used in the Time plot only, additional options **per_second**, **per_day** etc are available corresponding to the frequency over the named time unit.

A.4.5.24 Densogram Form

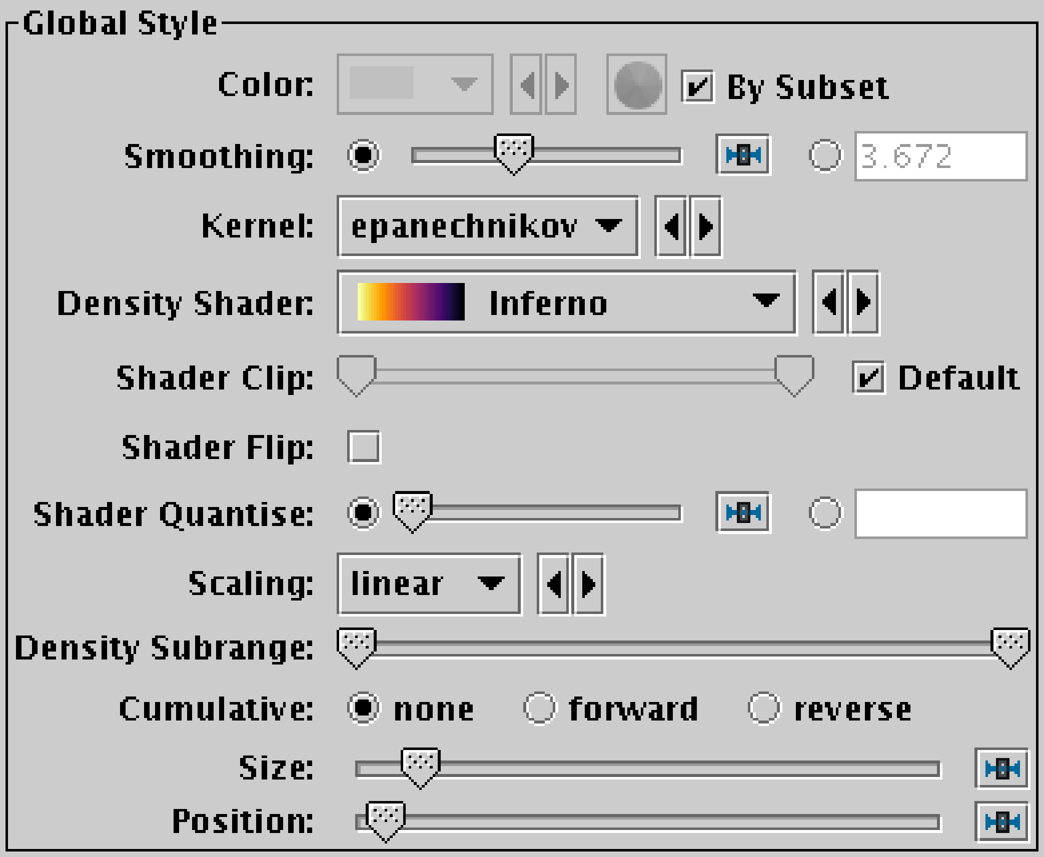
The **Densogram** form ( ) represents smoothed density of data values along the horizontal axis using a colourmap. This is like a Kernel Density Estimate (smoothed histogram with bins 1 pixel wide), but instead of representing the data extent vertically as bars or a line, values are represented by a fixed-size pixel-width column of a colour from a colour map. A smoothing kernel, whose width and shape may be varied, is applied to each data point.

This is a rather unconventional way to represent density data, and this plotting mode is probably not very useful. But hey, nobody's forcing you to use it.

This form may be used in either the Histogram, Plane or Time plot windows.



Example Densogram plot



Global Style

Color: ☑ By Subset

Smoothing: ● ○ 3.672

Kernel: epanechnikov

Density Shader: Inferno

Shader Clip: ☑ Default

Shader Flip: □

Shader Quantise: ● ○

Scaling: linear

Density Subrange:

Cumulative: ● none ○ forward ○ reverse

Size:

Position:

Densogram form configuration panel

These options always appear in the form configuration panel:

Colour
The base colour for the data set. Only certain shader colour maps respect this value, for others it is ignored.

Density Shader
The colour map for displaying density values. There are two types, relative and absolute. Relative maps have names marked by a star ("*"), and alter the basic dataset colour, for instance by darkening or lightening it, while absolute maps (the rest) ignore the basic dataset colour altogether. For a single-dataset plot, the absolute maps are best, but for multiple subsets it may be less confusing to use a relative one. Colour maps are listed in Appendix A.4.7.

Shader Clip
Select a sub-range of the full colour map above. If the **Default** checkbox is checked, then all or most of the colour ramp from the **Shader** control is used. If you want to configure the range of colours from the map yourself, uncheck the Default checkbox, and slide the handles in from the end of the slider to choose exactly the range you want.

The default range is clipped at one end for colour maps that fade to white, so that all the plotted colours will be distinguishable against a white background. If you don't want that, you can uncheck Default and leave the handles at the extreme ends of the slider.

Shader Flip
Whether the density scale should map forwards or backwards into the colour map.

Shader Quantise
Allows the colour map to be quantised. By default, the colour map is effectively continuous. If you slide the slider to the right, or enter a value in the text field, the map will be split into a decreasing number of discrete colours. This can be used to generate a contour-like effect, and may make it easier to trace the boundaries of regions of interest by eye.

Scaling

Determines the function used to map the range of density values onto the colour map. Options are **linear**, **logarithmic**, **square** and **square root**, **arc cosine**, **cosine**.

Density Subrange

Adjusts the density range over which the colour map is applied. By default the colour map is scaled using limits found from the data density in the plot (the most dense few pixels are ignored), but you can restrict the range using this slider.

Size

Height of the density bar in pixels.

Position

Determines where on the plot region the density bar appears. The value should be in the range 0..1; zero corresponds to the bottom of the plot and one to the top.

And these options appear in the form configuration panel for the Plane window, or the Bins control ( ) for the Histogram window:

Smoothing

Configures the smoothing width for kernel density estimation. This is the characteristic width of the kernel function to be convolved with the density to produce the visible plot.

Sliding the slider to the right makes the kernel width larger. The width in data units is shown in the text field on the right (if the X axis is logarithmic, this is a factor). Alternatively you can click the radio button near the text field, and enter the width in data units directly.

Kernel

The functional form of the smoothing kernel. The functions listed refer to the unscaled shape; all kernels are normalised to give a total area of unity.

The available options are:

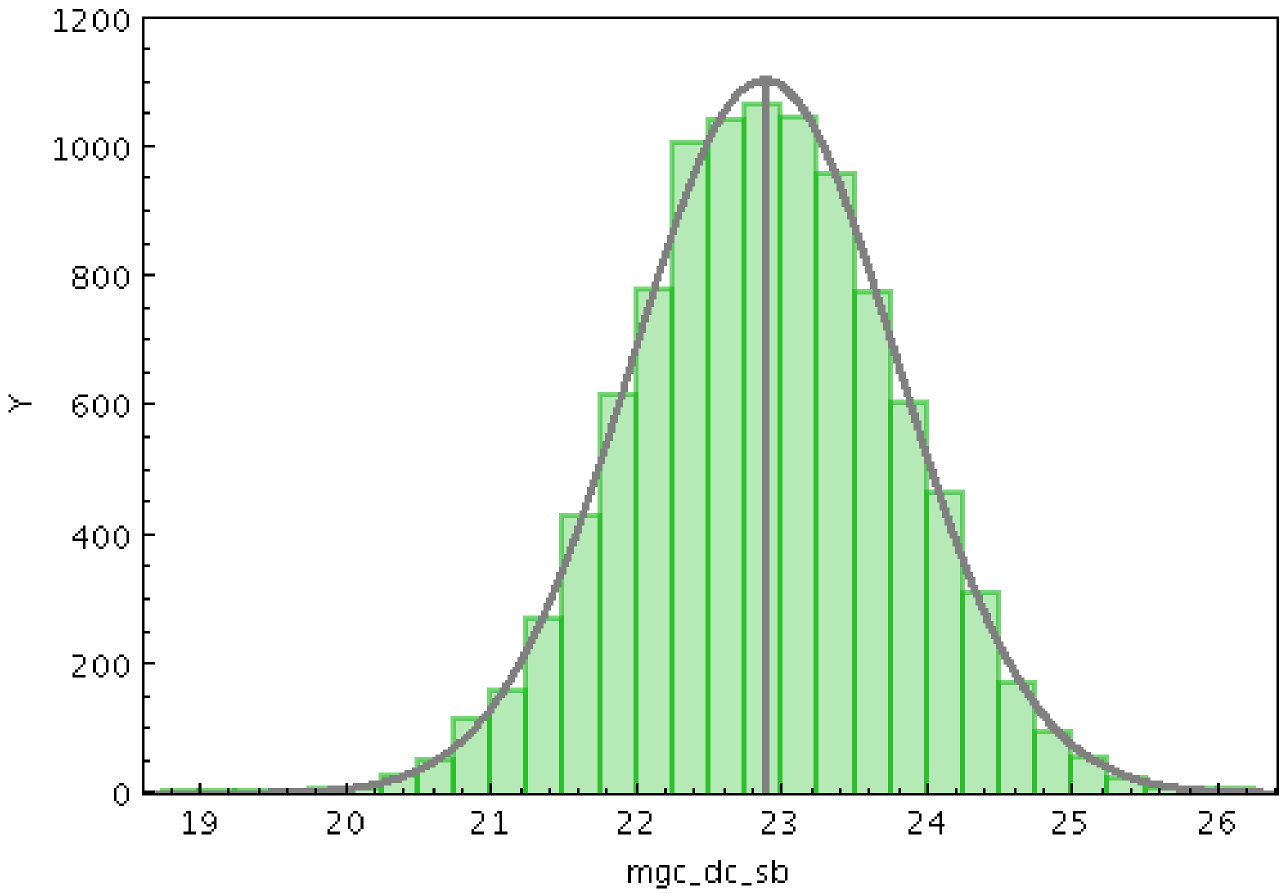
- **square**: Uniform value: f(x)=1, |x|=0..1
- **linear**: Triangle: f(x)=1-|x|, |x|=0..1
- **epanechnikov**: Parabola: f(x)=1-x*x, |x|=0..1
- **cos**: Cosine: f(x)=cos(x*pi/2), |x|=0..1
- **cos2**: Cosine squared: f(x)=cos^2(x*pi/2), |x|=0..1
- **gauss3**: Gaussian truncated at 3.0 sigma: f(x)=exp(-x*x/2), |x|=0..3
- **gauss6**: Gaussian truncated at 6.0 sigma: f(x)=exp(-x*x/2), |x|=0..6

Cumulative

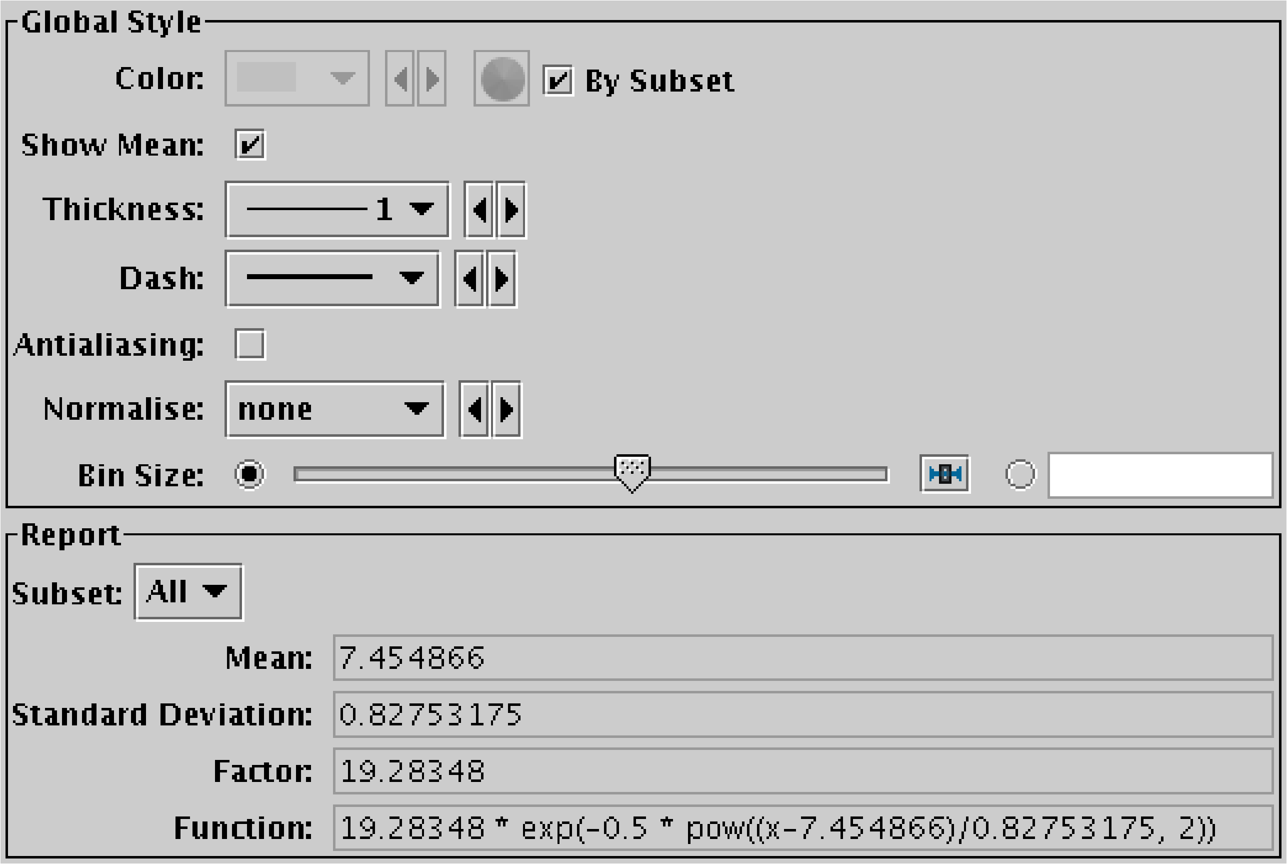
If set to **forward** or **reverse**, the levels are calculated cumulatively; each bin includes the counts from all previous bins in the direction of negative or positive infinity.

A.4.5.25 Gaussian Form

The **Gaussian** form ( ) plots a best fit Gaussian to the histogram of a sample of data. In fact, all it does is to calculate the mean and standard deviation of the sample, and plot the corresponding Gaussian curve. The mean and standard deviation values are reported by the plot (see below).



Example Gaussian plot



Global Style

Color: ☑ By Subset

Show Mean: ☑

Thickness: 1

Dash:

Antialiasing: □

Normalise: none

Bin Size: ● ○

Report

Subset: All

Mean: 7.454866

Standard Deviation: 0.82753175

Factor: 19.28348

Function: 19.28348 * exp(-0.5 * pow((x-7.454866)/0.82753175, 2))

Gaussian fit configuration panel

These options always appear in the form configuration panel:

Show Mean
If true, this will cause the mean value to be indicated by a vertical line.

Thickness
Controls line thickness.

Dash
Controls the dash pattern of the line (solid, dots, dashes etc).

Antialiasing
If true, lines are antialiased, which makes them look smoother on the screen or bitmapped export images. Has no effect on vector export images (PDF, SVG, EPS).

And these options appear in the form configuration panel for the Plane window, or the Bins control ( ) for the Histogram window:

Normalise
Defines how the histogram is scaled vertically to map its height to data coordinates. The normalisation options match those for the histogram form, so that if the same normalisation and bin size is chosen here, the plotted curve will be a best fit to the shape of the corresponding histogram bars.

Bin Size
Defines the notional size of the bins of a histogram which the plotted Gaussian should match. This option is used only to affect the vertical scaling, and only has effect for certain values of the **Normalise** option.

As well as drawing the line onto the plot, the calculated fitting coefficients are displayed at the bottom of the form configuration panel, under the heading **Report**. Note the coefficients are calculated by subset, and are only displayed for one subset at a time. To see the calculated values, select the subset of interest in the **Subset** selector. The reported items are:

Mean
The mean of the data set.

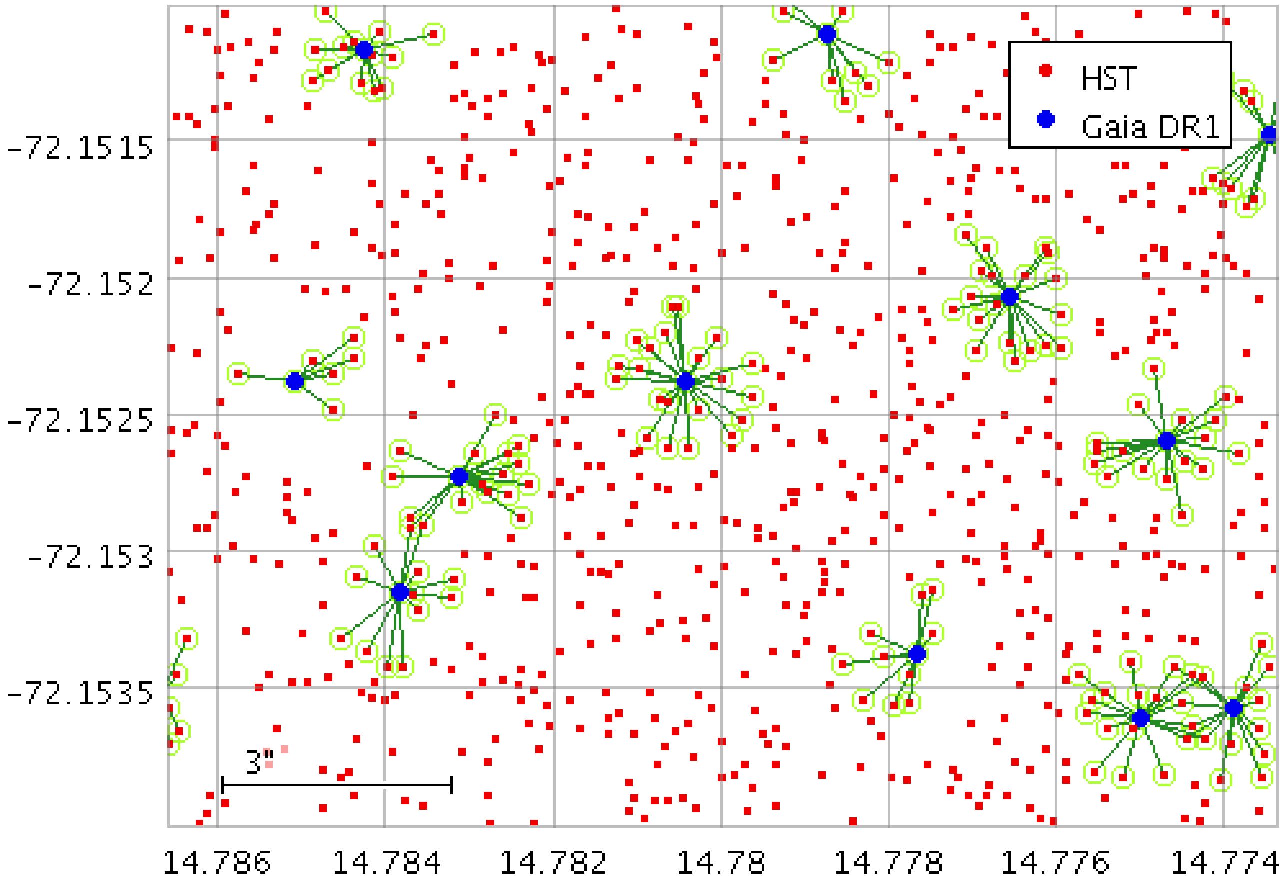
Standard Deviation
The standard deviation of the data set.

Factor
The scaling factor applied to the basic exponential function to yield the actual function plotted in data coordinates.

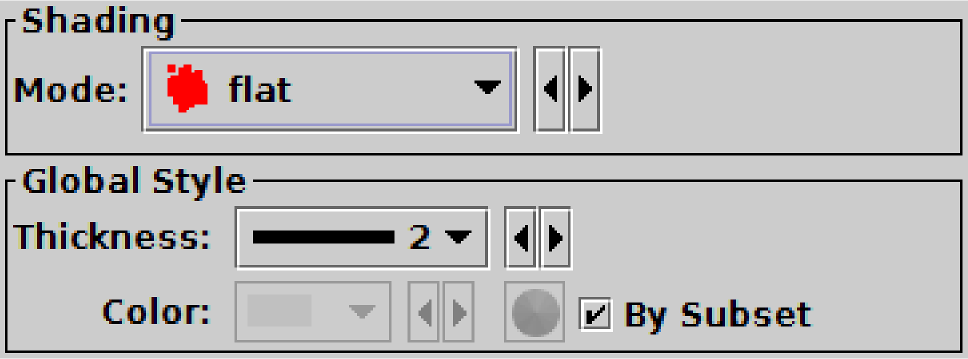
Function
The actual function plotted; this includes the numeric values shown by the other report items, and defines exactly what they mean. This expression uses topcat's expression language, and can be used (for instance) directly in the Function plotter.

A.4.5.26 Link2 Form

The **Link2** form ( ) is available from the pair position layer control, and plots a line linking the two points in a position pair.



Example Link2 plot



Shading

Mode: flat

Global Style

Thickness: 2

Color: ☑ By Subset

Link2 form configuration panel

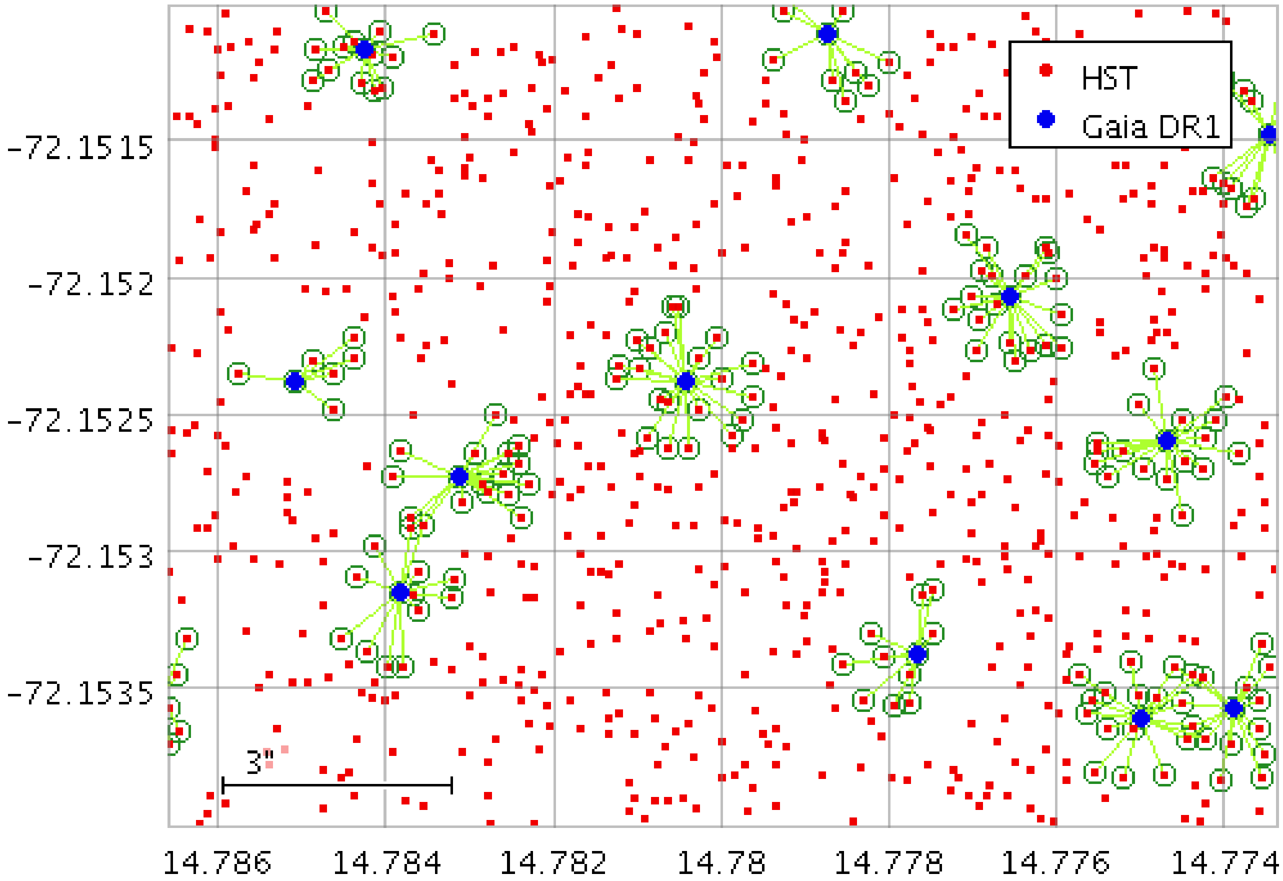
Configuration options are:

Shading Mode
See Appendix A.4.6.

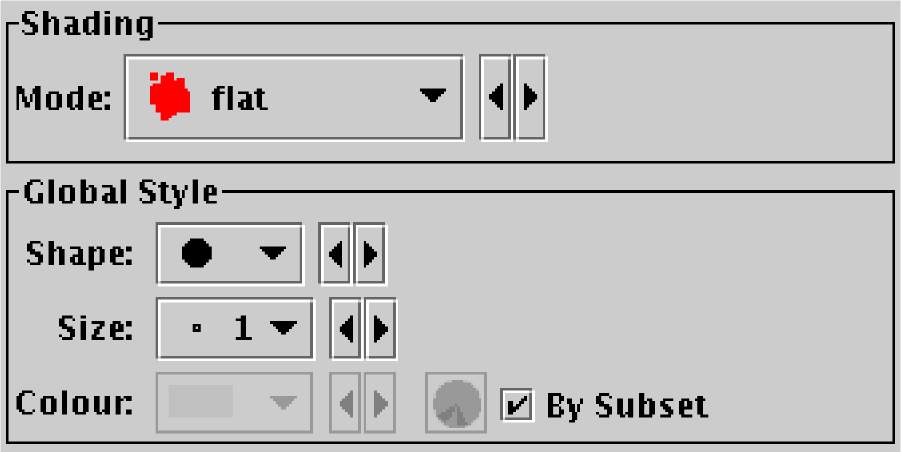
Thickness
Controls the line thickness. Zero, the default value, means a 1-pixel-wide line is used, and larger values make the lines thicker.

A.4.5.27 Mark2 Form

The **Mark2** (○ ○) form is available from the pair position layer control, and plots a marker of configurable shape and size at both points in a position pair. The same marker is used for both ends; if you want different points you can use two single Mark forms.



Example Mark2 plot



Shading

Mode: flat

Global Style

Shape: ●

Size: 1

Colour:

☑ By Subset

Mark2 form configuration panel

Configuration options are the same as for Mark:

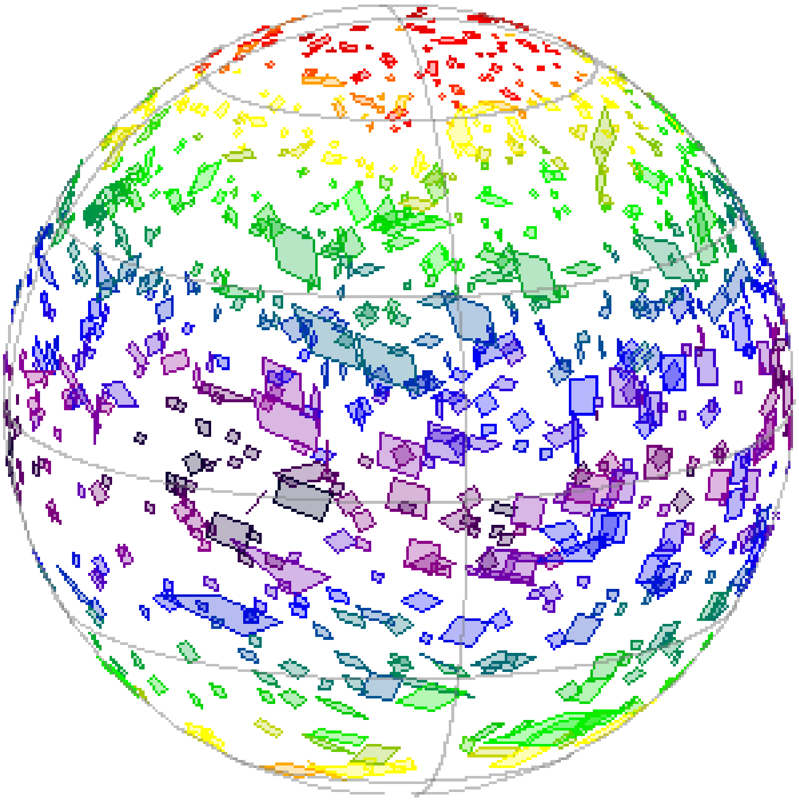
Shading Mode
See Appendix A.4.6.

Shape
Marker shape from a list of options.

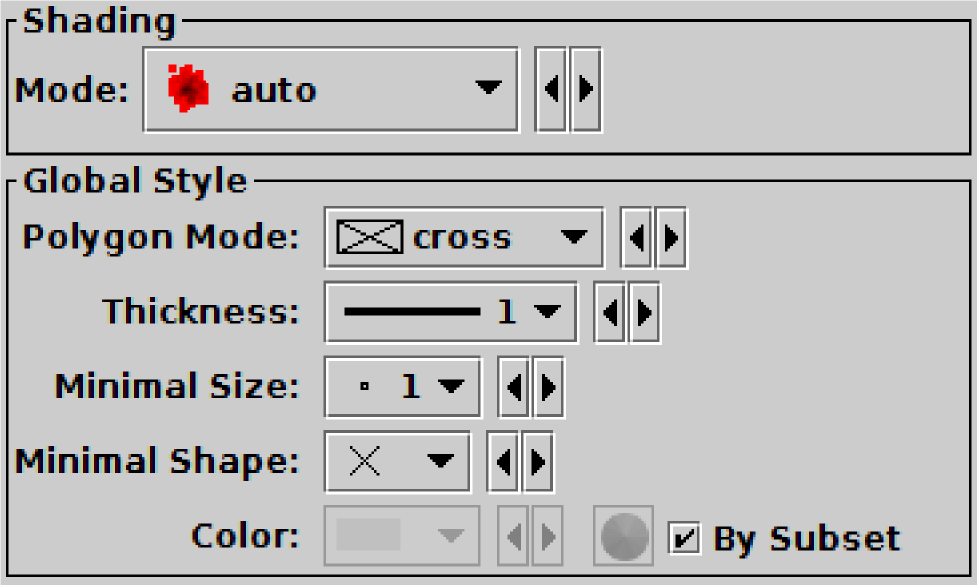
Size
Marker size in pixels.

A.4.5.28 Poly4 Form

The **Poly4** ( ) form is available from the quad position layer control, and draws a quadrilateral defined by the coordinates of its vertices supplied as 4 separate positions. Various options for how it is drawn, such as filled or outlined, are available.



Example Poly4 plot



Shading

Mode: auto

Global Style

Polygon Mode: cross

Thickness: 1

Minimal Size: 1

Minimal Shape: ×

Color:

☑ By Subset

Poly4 form configuration panel

Configuration options are:

Shading Mode
See Appendix A.4.6.

Polygon Mode
Defines how the polygon is drawn. Options are

- `outline`: draws a line round the outside of the polygon
- `fill`: fills the interior of the polygon
- `cross`: draws a line round the outside of the polygon and lines between all the vertices

Thickness

Controls the line thickness used when drawing polygons. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled rectangles contain no line drawings.

Minimal Size

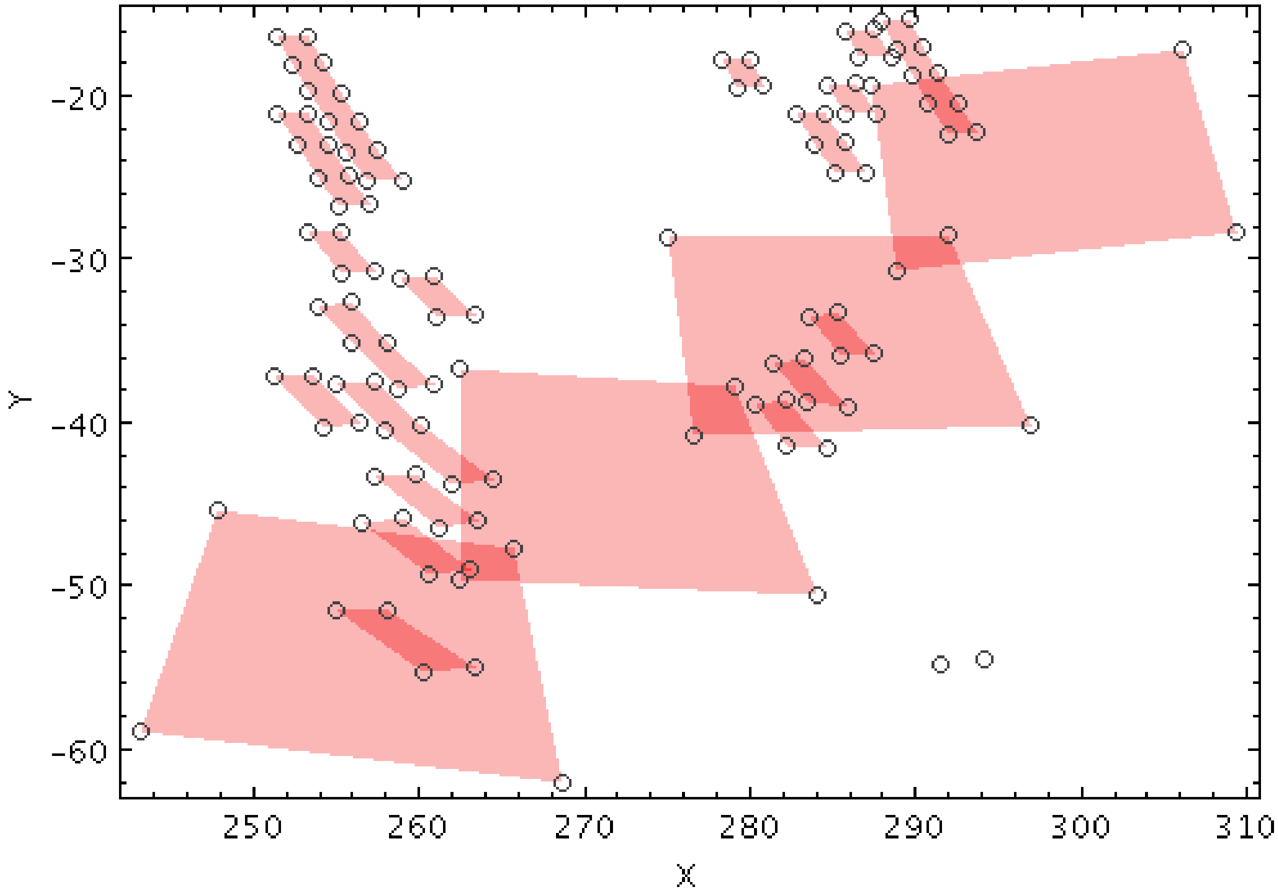
Defines a threshold size in pixels below which, instead of the shape defined by the polygon coordinates, a replacement marker will be painted instead. If this is set to zero, then only the shape itself will be plotted, which may mean it appears as only a single pixel. By setting a larger value, you can ensure that the position of even small polygons is easily visible, at the expense of giving them an artificial shape and size. This value also defines the size of the replacement markers.

Minimal Shape

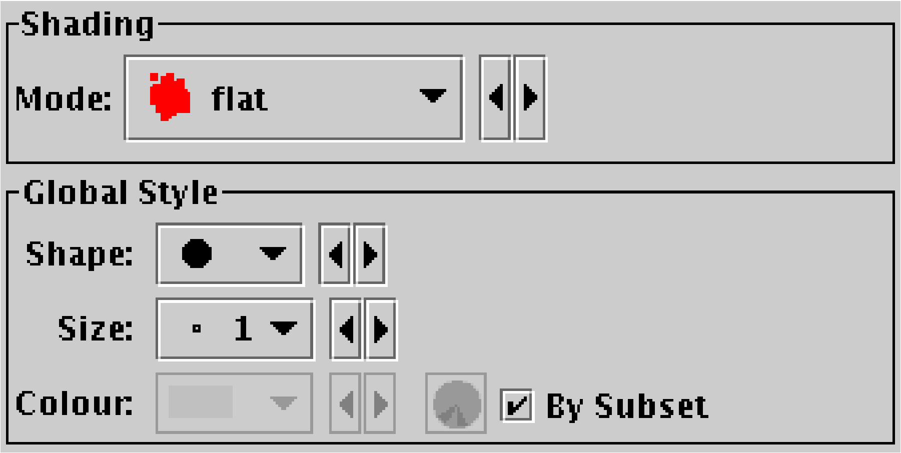
Selects the shape of replacement markers that get plotted instead of very small polygons, as controlled by the **Minimal Size** setting.

A.4.5.29 Mark4 Form

The **Mark4** form (⁘) is available from the Quad Position layer control, and plots 4 similar markers of fixed size and shape representing 4 separate positions from the same table row. This is a convenience (you could do the same thing by plotting the four markers separately) that makes it easy to mark the corners of polygons plotted from the Quad layer control.



Example Mark4 plot



Shading

Mode: flat

Global Style

Shape: ●

Size: 1

Colour:

☑ By Subset

Mark4 form configuration panel

Configuration options are the same as for Mark:

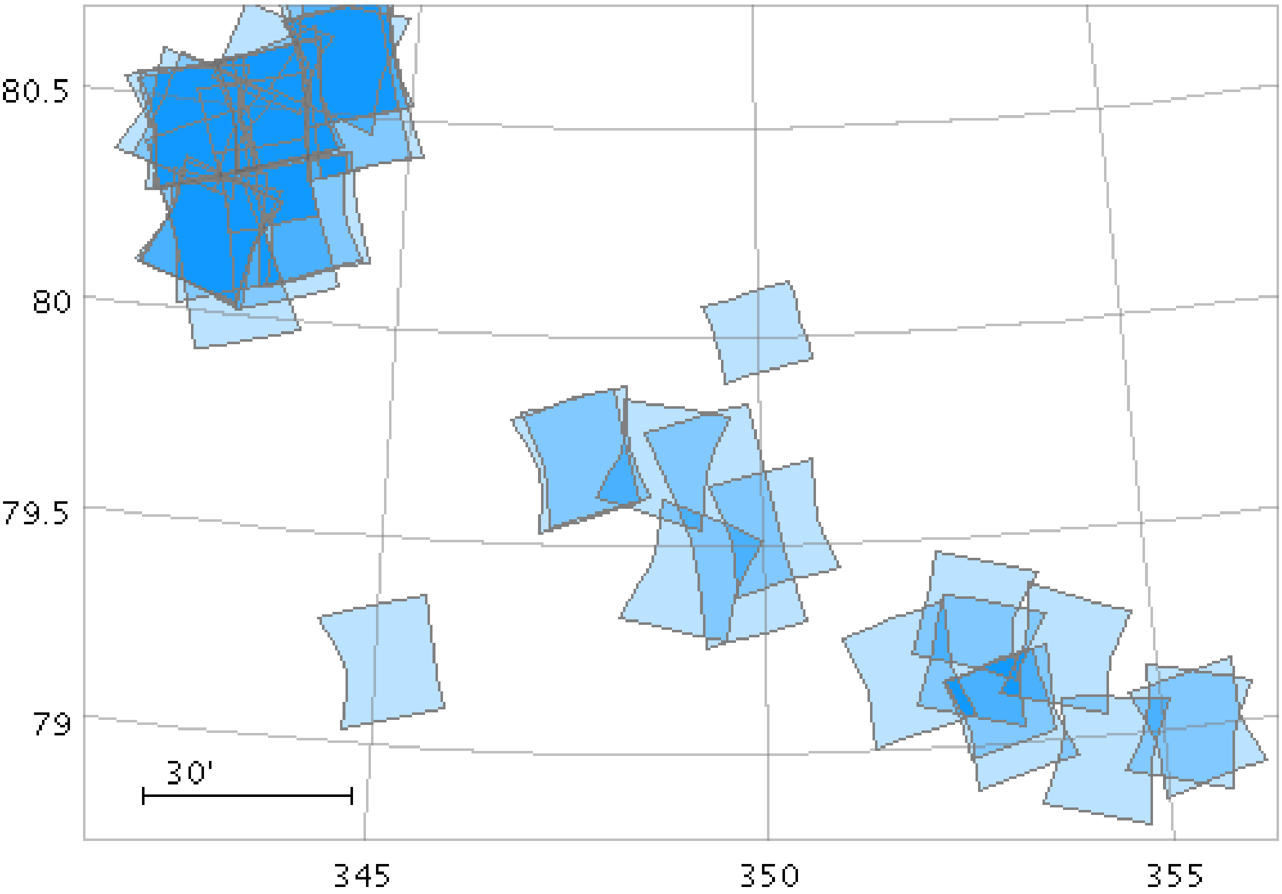
Shading Mode
See Appendix A.4.6.

Shape
Marker shape from a list of options.

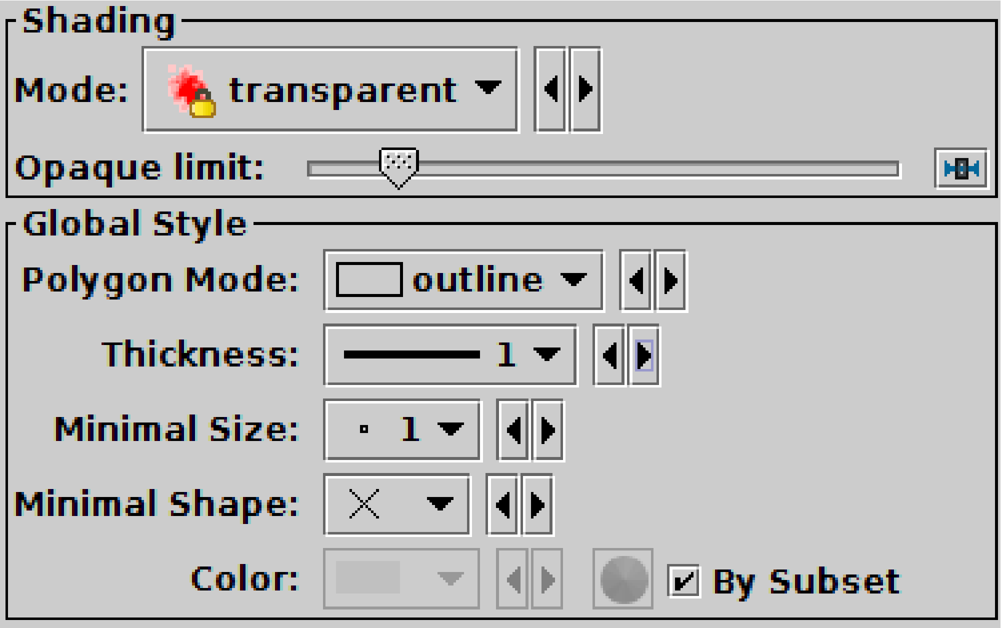
Size
Marker size in pixels.

A.4.5.30 Area Form

The **Area** form () is available from the Area layer control, and outlines or fills areas that are defined in a table row.



Example Area plot



Mode: transparent

Opaque limit:

Polygon Mode: outline

Thickness: 1

Minimal Size: 1

Minimal Shape:

Color:

☑ By Subset

Area form configuration panel

Configuration options are:

Shading Mode
See Appendix A.4.6.

Polygon Mode
Defines how the polygon is drawn. Options are

- `outline`: draws a line round the outside of the polygon

- `fill`: fills the interior of the polygon
- `cross`: draws a line round the outside of the polygon and lines between all the vertices

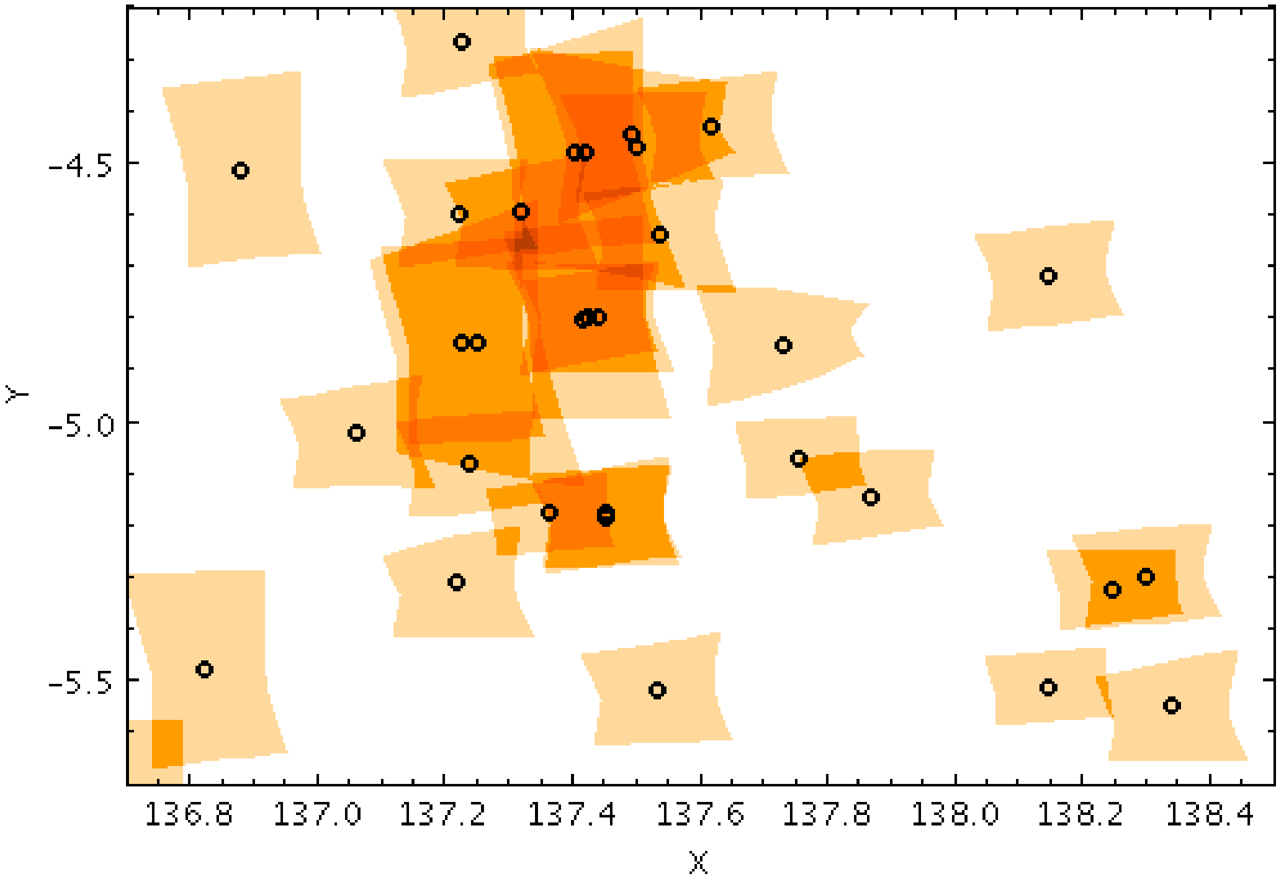
Thickness
Controls the line thickness used when drawing polygons. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled polygons contain no line drawings.

Minimal Size
Defines a threshold size in pixels below which, instead of the shape defined by the Area coordinate, a replacement marker will be painted instead. If this is set to zero, then only the shape itself will be plotted, which may mean it appears as only a single pixel. By setting a larger value, you can ensure that the position of even small areas is easily visible, at the expense of giving them an artificial shape and size. This value also defines the size of the replacement markers.

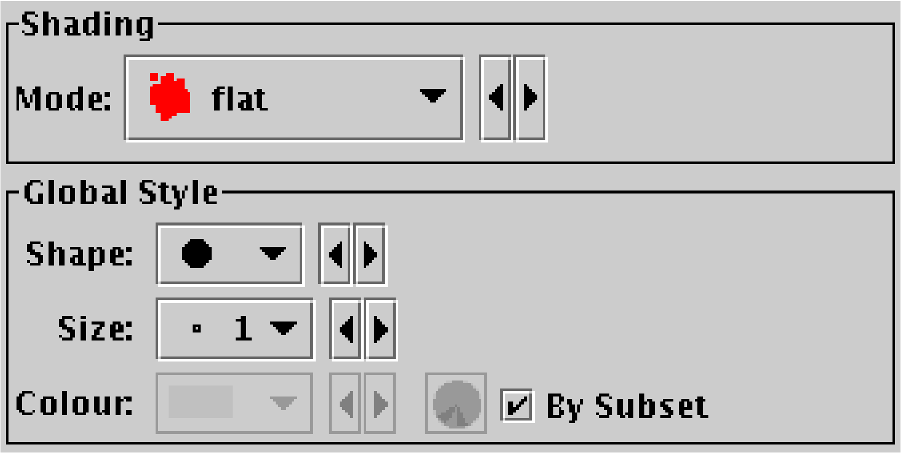
Minimal Shape
Selects the shape of replacement markers that get plotted instead of very small areas, as controlled by the **Minimal Size** setting.

A.4.5.31 Central Form

The **Central** form (✚) is available from the Area layer control, and plots a point at the nominal center of the given area. The effect is just the same as for the Mark form, but it can be used from the Area layer control rather than having to specify positional coordinates separately. The plotted position is determined by the plotting code from the shape information; it may or may not correspond to the shape's actual center.



Example Central plot



Shading

Mode: flat

Global Style

Shape: ●

Size: 1

Colour:

☑ By Subset

Central form configuration panel

Configuration options are:

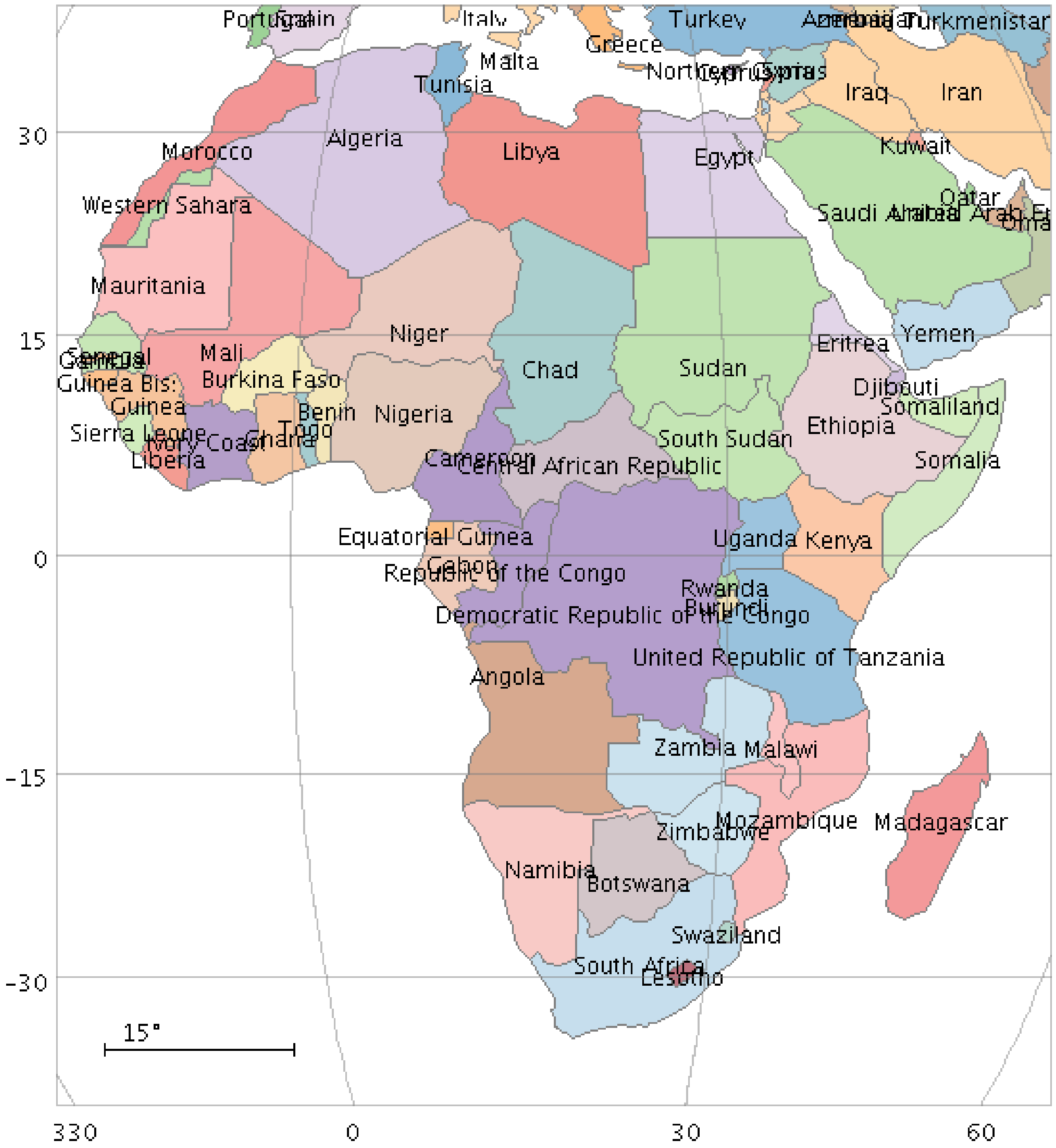
Shading Mode
See Appendix A.4.6.

Shape
Marker shape from a list of options.

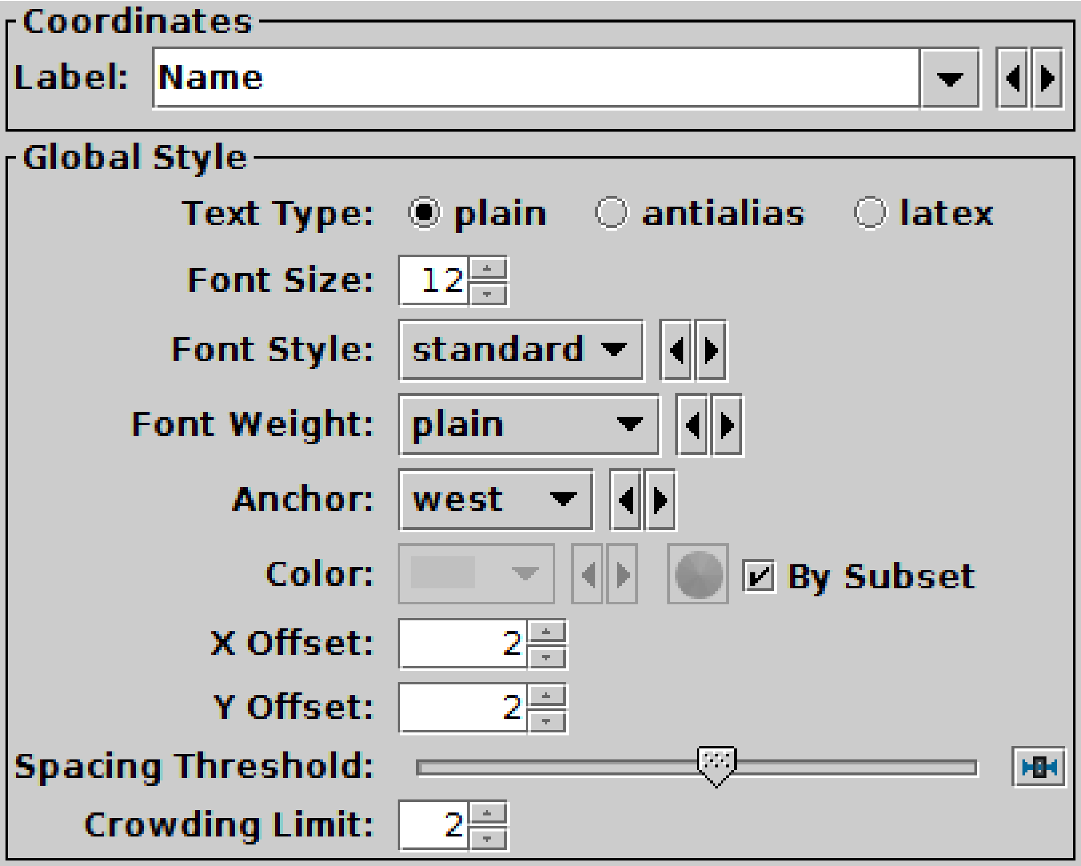
Size
Marker size in pixels.

A.4.5.32 AreaLabel Form

The **AreaLabel** form (Txt) is available from the Area layer control, and draws a text label near the nominal center of each plotted area. The effect is just the same as for the Label form, but it can be used from the Area layer control rather than having to specify positional coordinates separately. The plotted position is determined by the plotting code from the shape information; it may or may not correspond to the shape's actual center.



Example AreaLabel plot



Coordinates

Label: Name

Global Style

Text Type: ● plain ○ antialias ○ latex

Font Size: 12

Font Style: standard

Font Weight: plain

Anchor: west

Color: ☑ By Subset

X Offset: 2

Y Offset: 2

Spacing Threshold:

Crowding Limit: 2

AreaLabel form configuration panel

The configuration options are:

Text
: A column or expression from the table supplying the text to write on the plot. Any data type (string or numeric) is permitted.

Text Syntax
: How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. At time of writing, on MacOS antialiased text seems to be required to stop the writing coming out upside-down for non-horizontal text. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size
: Size of the font in points.

Font Style
: Style of the font - standard, serif or monospaced.

Font Weight
: Whether the font is plain, bold or italic.

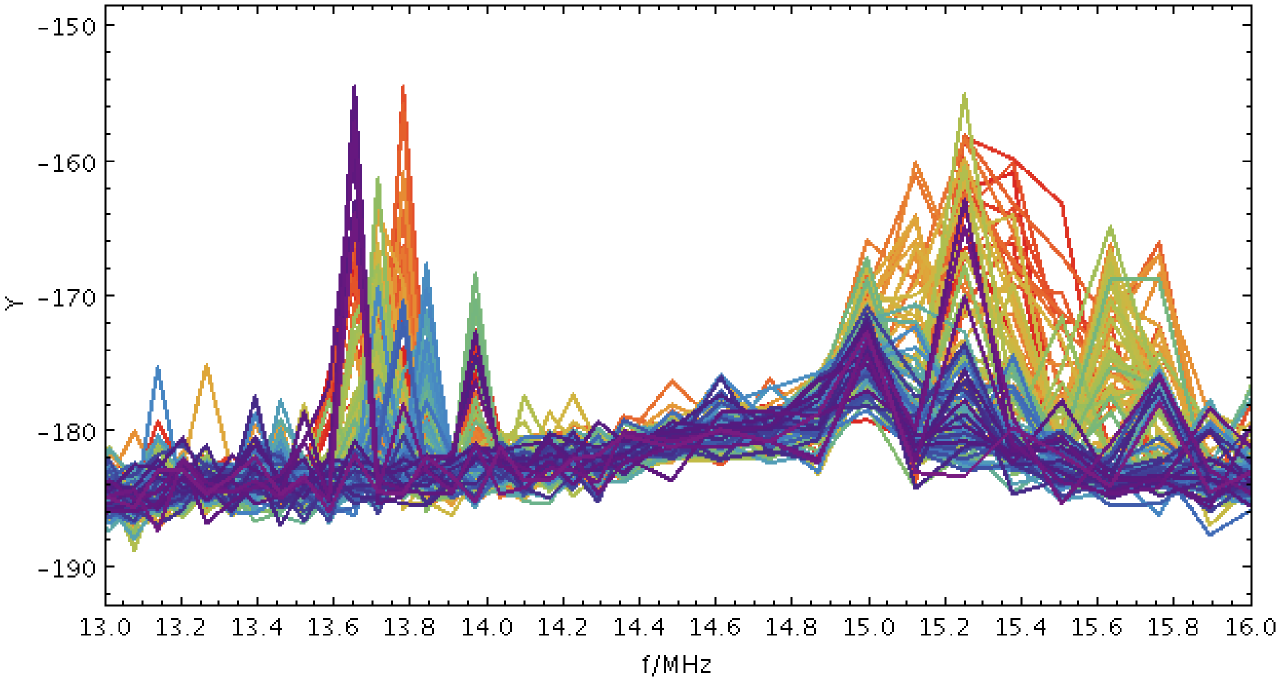
Anchor
: The position of the text relative to the data position.

Spacing Threshold
Crowding Limit
: These two options control how closely spaced labels can be. Labels which are too closely crowded together will simply not be shown, since overplotting many labels together ends up with them being illegible. The **Spacing Threshold** slider controls the smallest area that a group of labels can have to themselves - if there are too many in the same area, none will be drawn. Sliding it left allows more crowding and right allows less. The **Crowding Limit**

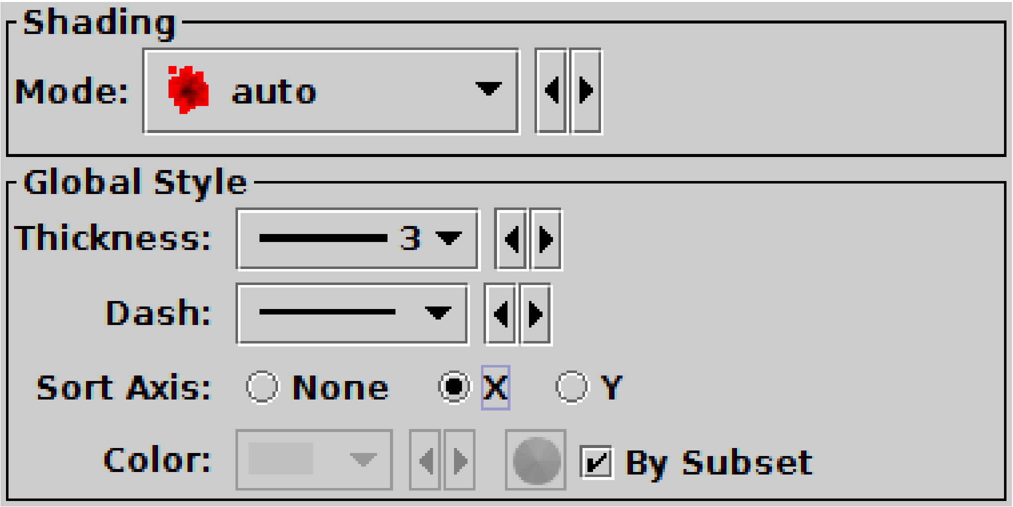
controls the largest number of labels that can be in a group. Setting it to 2 for instance is useful if you want to see pairs of labels, even if the pair is close.

A.4.5.33 Lines Form

The **Lines** form ( ), available from the XYArray Layer Control, draws a point-to-point line joining all the elements of X, Y array-valued coordinates, resulting in one line for each table row. It is typically used to plot a number of spectra or time series.



Example Lines plot



Shading

Mode: auto

Global Style

Thickness: 3

Dash:

Sort Axis: ○ None ● X ○ Y

Color: ☑ By Subset

Lines form configuration panel

The configuration options are:

Shading mode
See Appendix A.4.6.

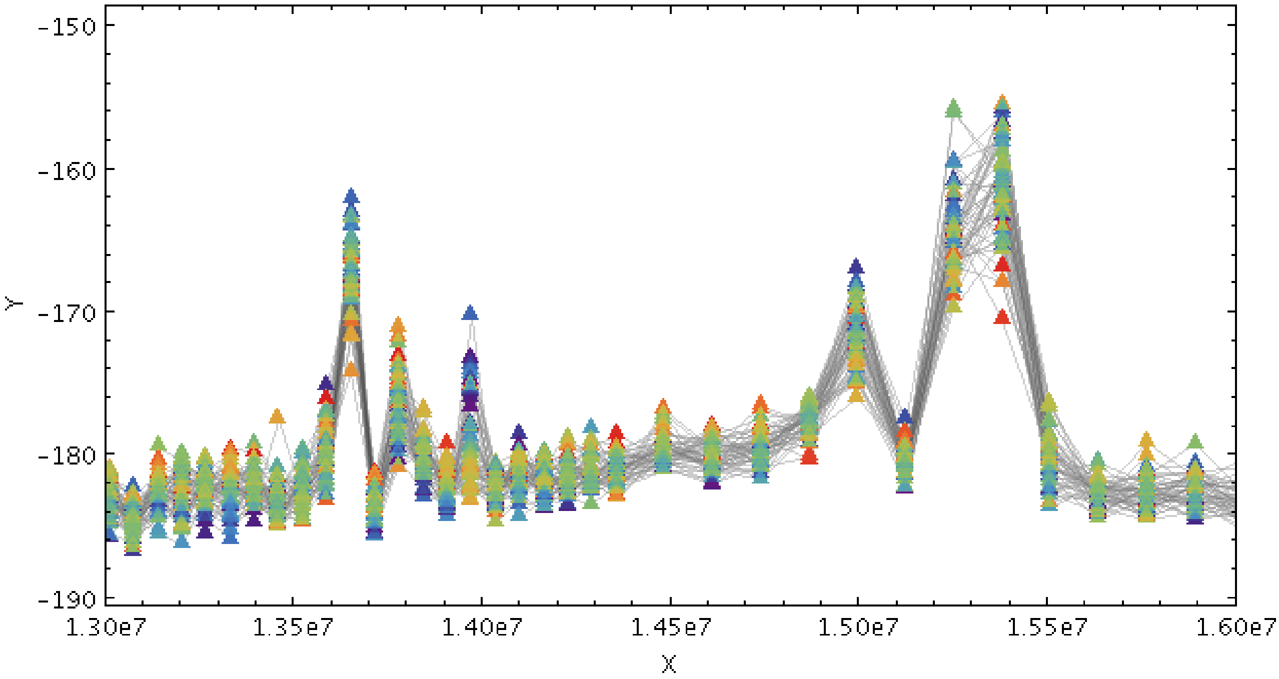
Thickness
Line thickness in pixels.

Dash
Dash pattern. The line is solid by default.

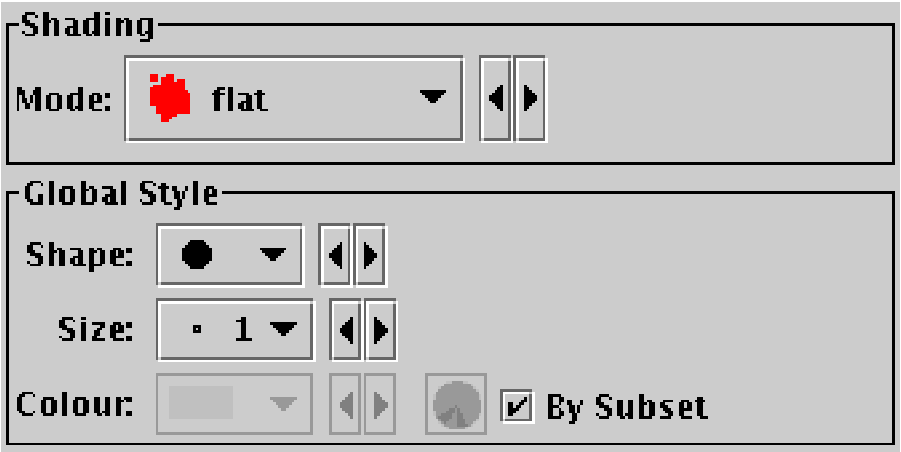
Sort Axis
May be used to sort the points before the lines are drawn. By default (option **None**) the lines are drawn between the points in the sequence in which they appear in each array. But if you set it to **X** or **Y** the points will be pre-ordered along the given axis, so that lines for unordered arrays will come out looking like a function of the X or Y coordinate rather than a scribble.

A.4.5.34 Marks Form

The **Marks** form (●), available from the XYArray Layer Control, draws a set of points representing all the elements of X, Y array-valued coordinates, resulting in a sequence of points for each table row.



Example Marks plot



Shading

Mode: flat

Global Style

Shape: ●

Size: 1

Colour:

☑ By Subset

Mark form configuration panel

The configuration options are:

Shading Mode

See Appendix A.4.6.

Shape
Marker shape from a list of options.

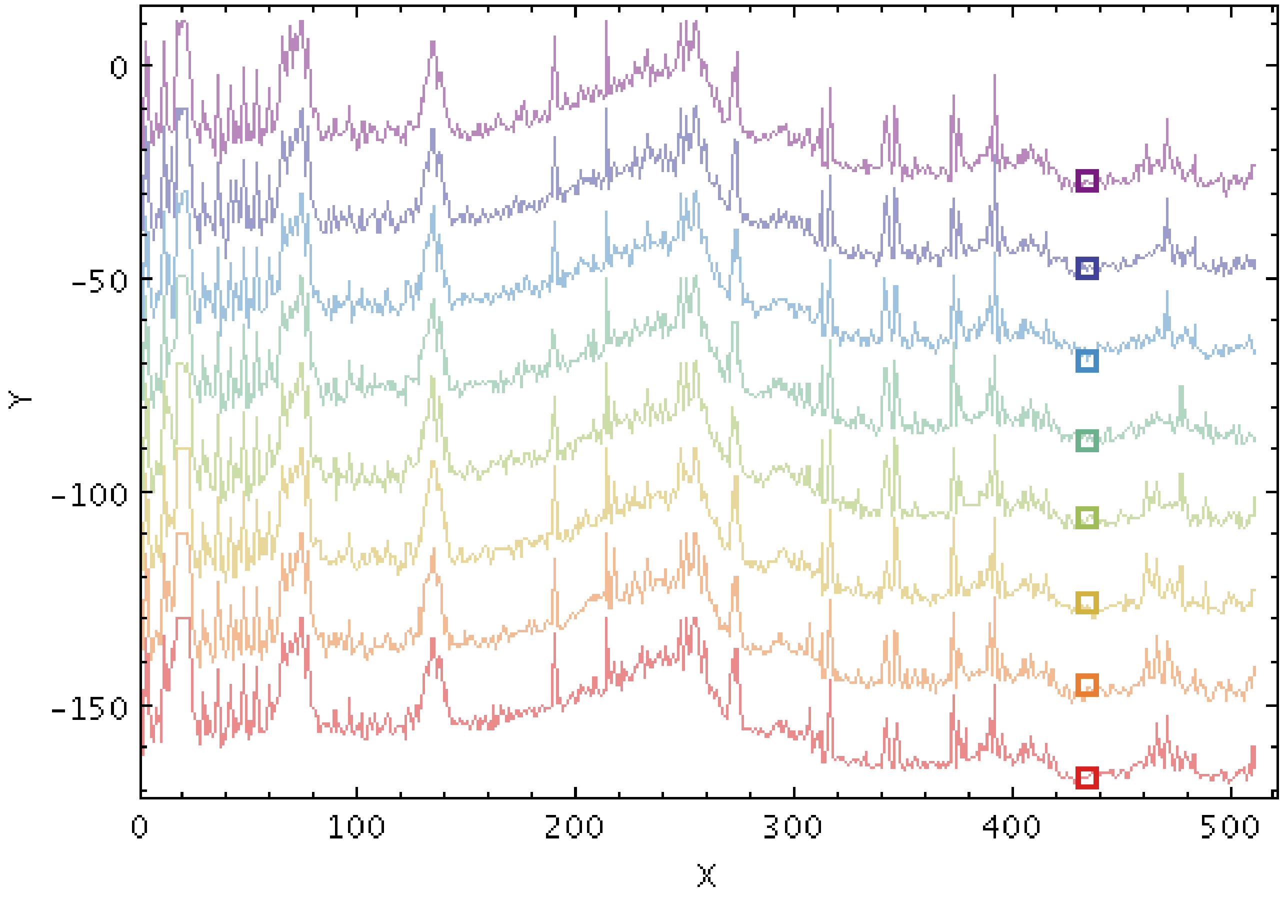
Size
Marker size in pixels.

A.4.5.35 Handles Form

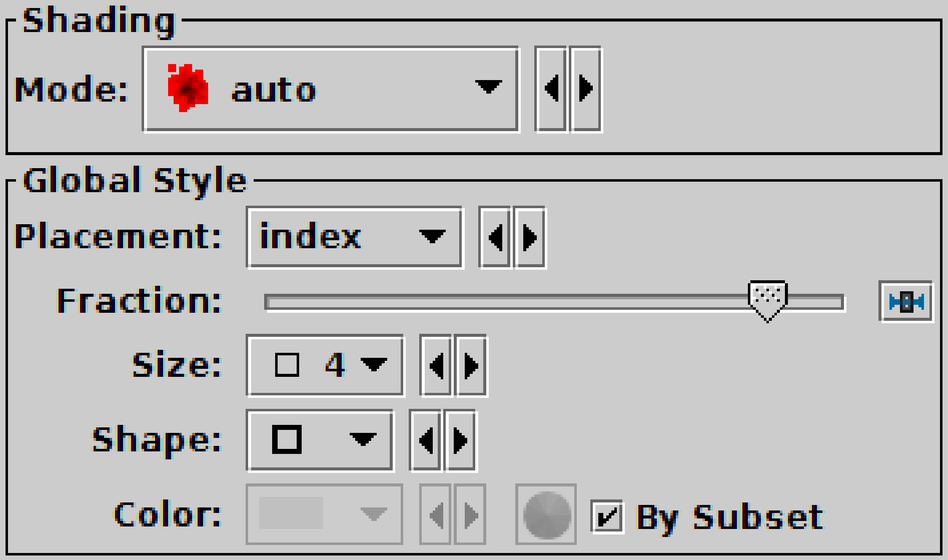
The **Handles** form ( ), available from the XYArray Layer Control, draws a single pointer (a "handle") somewhere near each group of points defined by pair of X/Y array coordinates. This doesn't do much to show the shape of the line formed by the array values, but it does provide a reference point for each line. This can be used by the parts of TOPCAT that associated a fixed position with each table row, in particular:

- you can click on the handle to activate the row it relates to, for instance to highlight the table row in the Data Window
- if the row is activated by some other action, its handle will be highlighted with the activation cursor
- you can define subsets graphically using the reference positions defined by the handles

Since array coordinates don't normally have a single per-row position, adding a Handles layer like this is the only way to perform those position-related activities with the plots from the XYArray Layer Control. Because of its general usefulness, a deactivated Handles control is added automatically to the layer control, so you just need to click the checkbox to display handles in this way.



Example Handles plot



Shading

Mode: auto

Global Style

Placement: index

Fraction:

Size: □ 4

Shape: □

Color:

☑ By Subset

Handles form configuration panel

The configuration options are:

Shading Mode
See Appendix A.4.6.

Placement
Specifies where to draw the handle marker in relation to the X/Y array values. The options are:

- `index`: (X,Y) position at a certain fraction of the way through the arrays, as given by the **Fraction** value; Fraction=0.0 is the first element, Fraction=1.0 is the last
- `ymax`: (X,Y) position at which the maximum Y value is located (**Fraction** is ignored)
- `ymin`: (X,Y) position at which the minimum Y value is located (**Fraction** is ignored)
- `xmax`: (X,Y) position at which the maximum X value is located (**Fraction** is ignored)
- `xmin`: (X,Y) position at which the minimum X value is located (**Fraction** is ignored)
- `xymean`: center of gravity of all the (X,Y) points (**Fraction** is ignored)

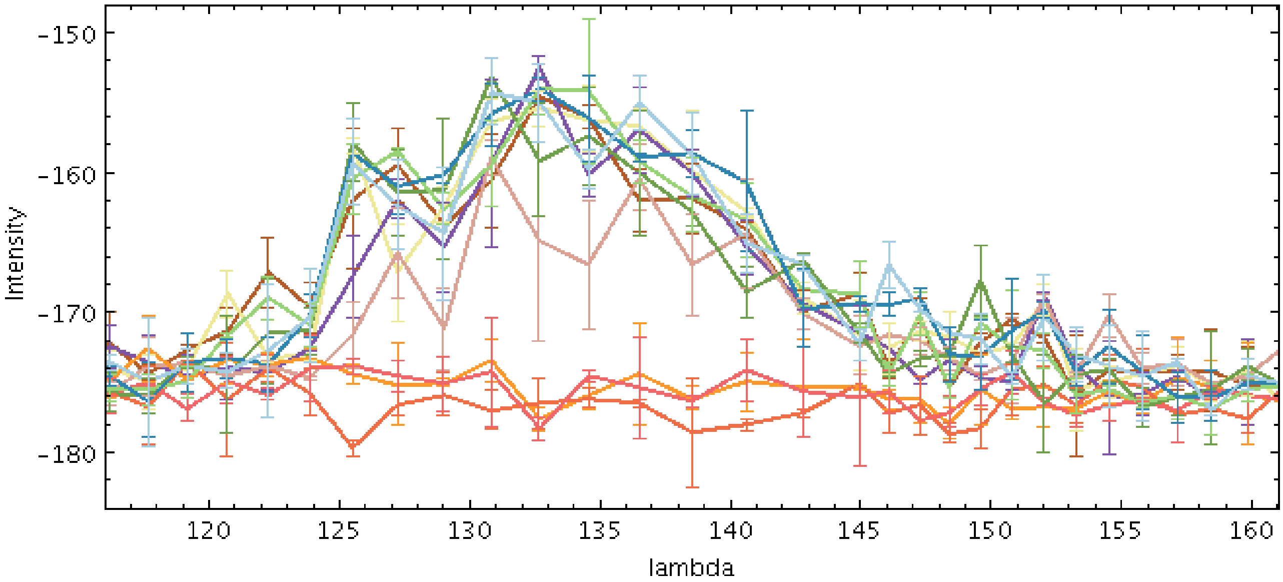
Fraction
Provides a numeric value in the range 0..1 that may influence where the handle is placed. Currently, this is only relevant for a **Placement** value of **index**, where it indicates how far through the array the reference (X,Y) position should be taken (0.0 means the first element, 1.0 means the last). For other values of placement it is ignored.

Size
Marker size in pixels.

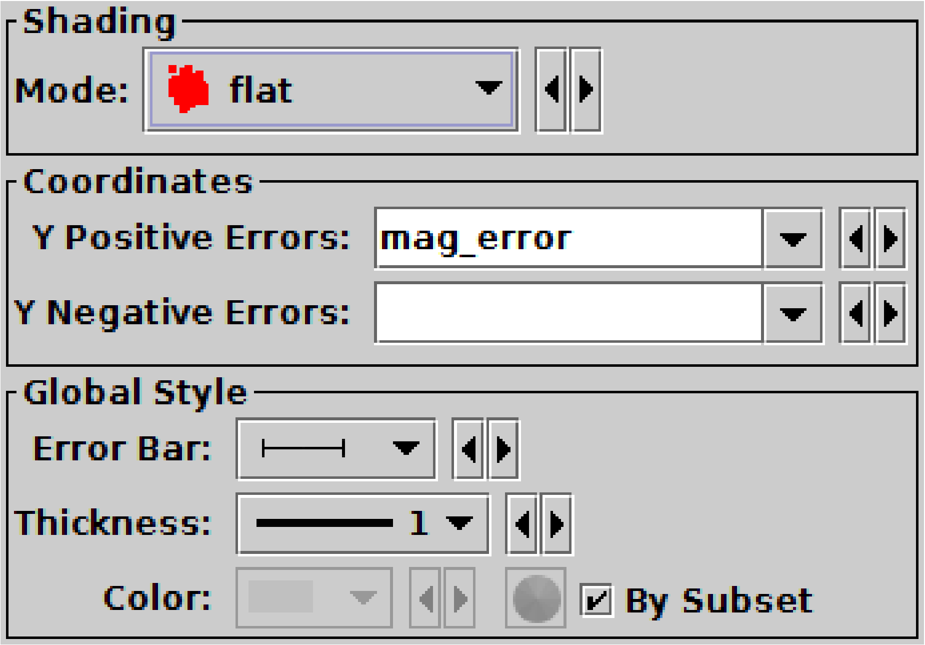
Shape
Marker shape from a list of options.

A.4.5.36 YErrors Form

The **YErrors** form ( ), available from the XYArray Layer Control, draws a set of symmetric or asymmetric vertical error bars representing all the elements of X, Y array-valued coordinates, resulting in a sequence of error bars for each table row.



Example YErrors plot



Shading

Mode: flat

Coordinates

Y Positive Errors: mag_error

Y Negative Errors:

Global Style

Error Bar:

Thickness: 1

Color: ☑ By Subset

YErrors form configuration panel

The configuration options are:

Shading Mode
See Appendix A.4.6.

Y Positive Errors
Y Negative Errors
Array values giving vertical error extents corresponding to the plotted data position arrays. If both positive and negative values are filled in, the errors will be asymmetric. If the negative value is blank (either because the coordinate is not filled in, or because its value is NaN for that row), the error bars will be symmetric, i.e. the negative error bar will be the same size as the positive one. If you want to ensure only a positive error bar is plotted, supply zero for the corresponding negative errors. The error extents must be positive; negative array elements are ignored.

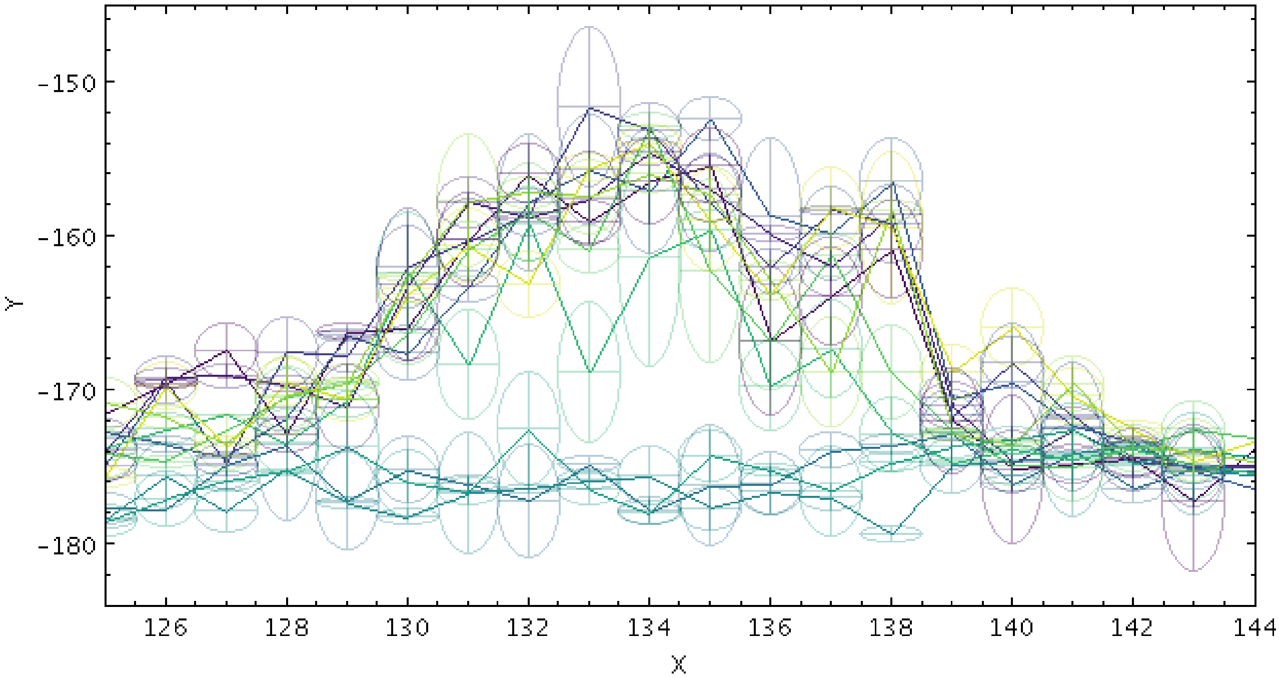
The column names or expressions used here must be array values, matching the length of the data array values. If you need to create or manipulate array values, the functions in the Arrays class may be of use here.

Error Bar
Error bar shape from a list of options.

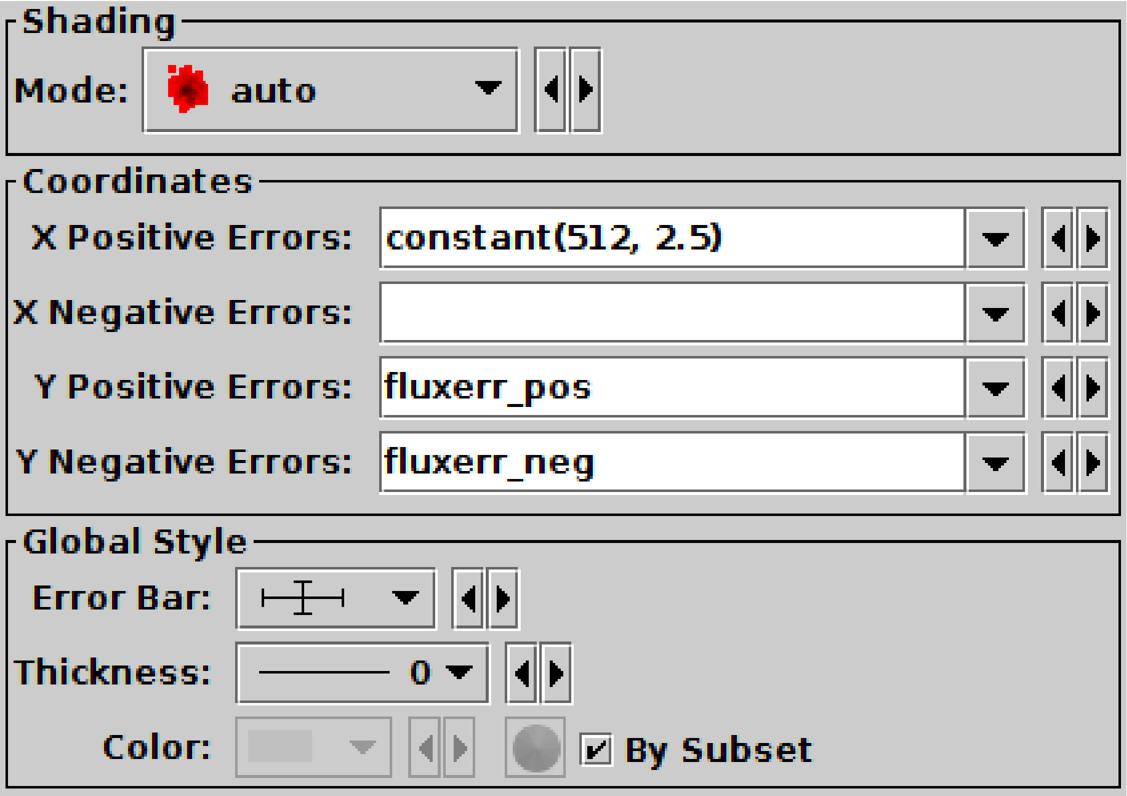
Thickness
Controls the line thickness used when drawing error bars.

A.4.5.37 XYErrors Form

The **XYErrors** form (┼), available from the XYArray Layer Control, draws a set of error bars representing all the elements of X, Y array-valued coordinates, resulting in a sequence of error bars for each table row. Symmetric or asymmetric vertical and/or horizontal error bars may be drawn, and the shape of the error "bars" is quite configurable, including error ellipses, rectangles etc.



Example XYErrors plot



Shading

Mode: auto

Coordinates

X Positive Errors: constant(512, 2.5)

X Negative Errors:

Y Positive Errors: fluxerr_pos

Y Negative Errors: fluxerr_neg

Global Style

Error Bar:

Thickness: 0

Color: ☑ By Subset

XYErrors form configuration panel

The configuration options are:

Shading Mode
See Appendix A.4.6.

X Positive Errors
X Negative Errors
Y Positive Errors
Y Negative Errors
Array values giving error extents in X and Y directions corresponding to the plotted data position arrays. For each axis, if both positive and negative values are filled in, the errors will be asymmetric. If the negative value is blank (either because the coordinate is not filled in, or because its value is NaN for that row), the error bars will be symmetric, i.e. the negative error bar will be the same size as the positive one. If you want to ensure only a positive error bar is plotted, supply zero for the corresponding negative errors. Either X, Y or both error values may be supplied.

The column names or expressions used here must be array values, matching the length of the data array values. If you need to create or manipulate array values, the functions in the Arrays class may be of use here.

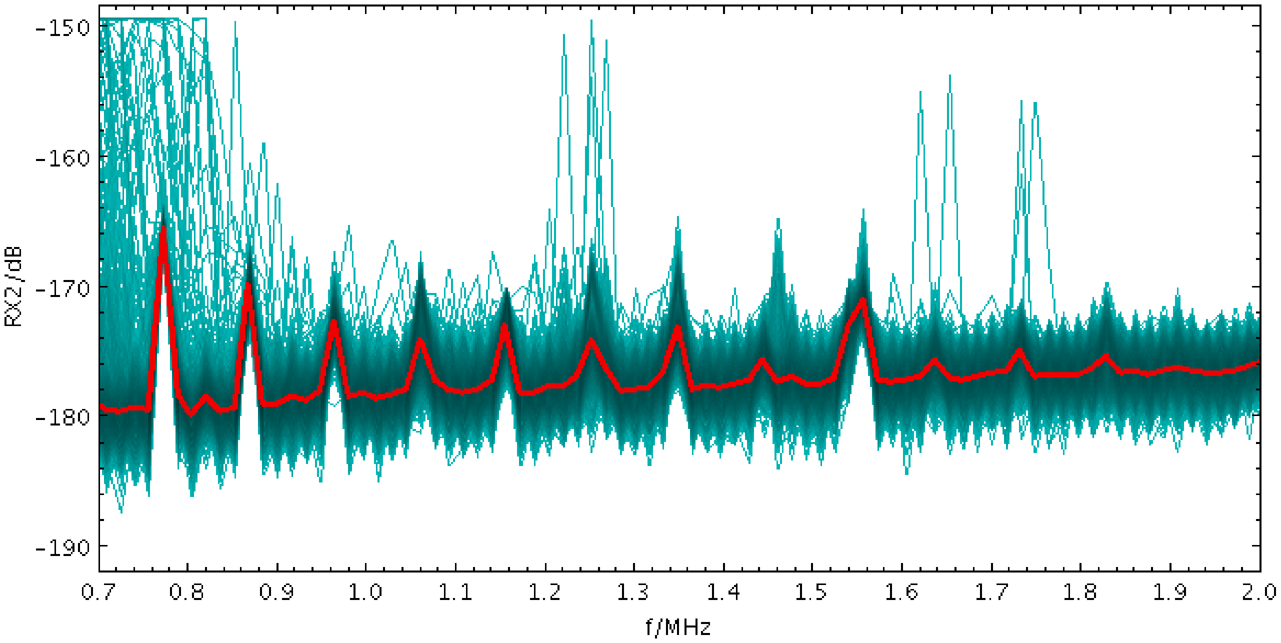
Error Bar
Error bar shape from a list of options.

Thickness
Controls the line thickness used when drawing error bars. Zero, the default value, means a 1-pixel-wide line is used. Larger values make drawn lines thicker, but note changing this value will not affect all shapes, for instance filled rectangles contain no line drawings.

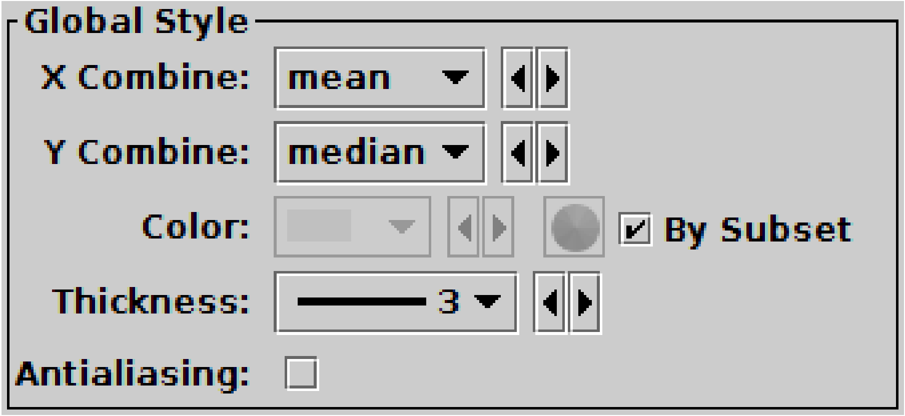
A.4.5.38 StatLine Form

), available from the XYArray Layer Control, plots a single line based on a combination (typically the mean) of input array-valued coordinates. The input X and Y coordinates must be fixed-length arrays of length N; a line with N points is plotted, each point representing the mean (or median, minimum, maximum, ...) of all the input array elements at the corresponding position.

Note that because the X and Y arrays must be of a fixed size for all rows, and because combination is performed in both X and Y directions, this is typically only suitable for plotting combined spectra if they all share a common horizontal axis, e.g. are all sampled into the same wavelength bins. To visually combine spectra with non-uniform sampling, the ArrayQuantile form may be more useful.



Example StatLine plot



X Combine: mean

Y Combine: median

Color: ☑ By Subset

Thickness: 3

Antialiasing: □

StatLine form configuration panel

The configuration options are:

X Combine
Y Combine

Defines how corresponding array elements on the X/Y axis are combined together to produce the plotted value. The following options are currently available:

- `mean`: the mean of the combined values
- `median`: the median of the combined values (may be slow)
- `min`: the minimum of all the combined values
- `max`: the maximum of all the combined values

- `q.01`: the 1st percentile of the combined values (may be slow)
- `q1`: the first quartile of the combined values (may be slow)
- `q3`: the third quartile of the combined values (may be slow)
- `q.99`: the 99th percentile of the combined values (may be slow)
- `stdev`: the sample standard deviation of the combined values
- `sum`: the sum of all the combined values
- `count`: the number of non-blank values

Color
Color of the plotted line.

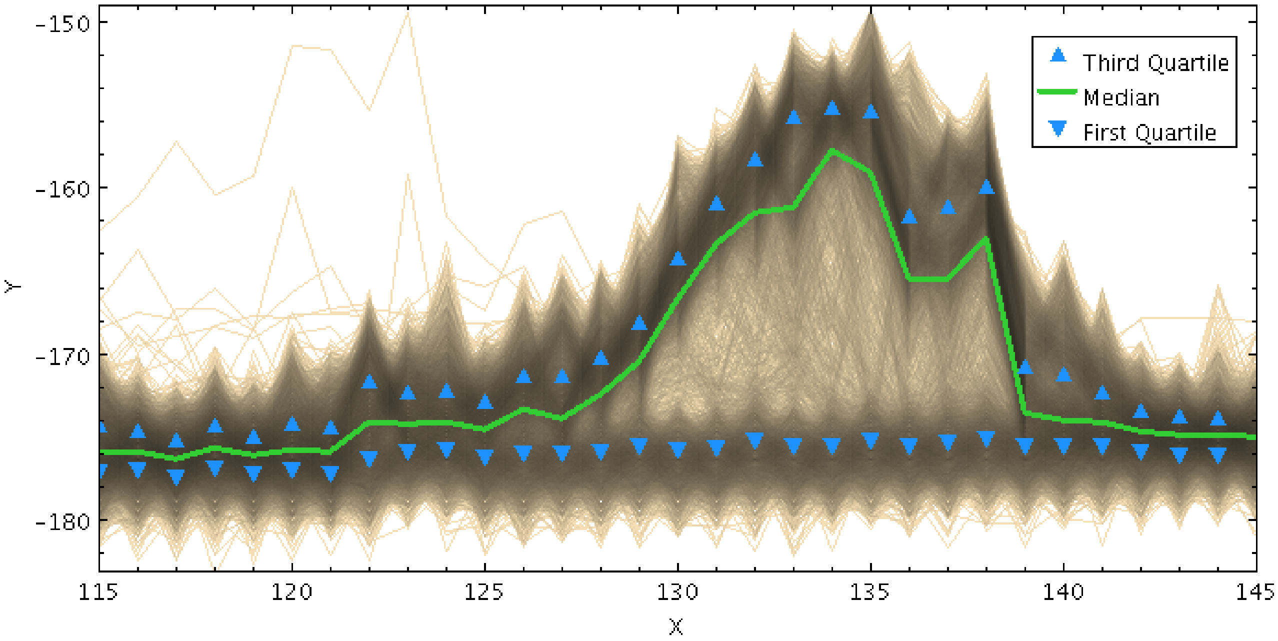
Thickness
Thickness in pixels of the plotted line.

Antialiasing
Controls whether grid lines will be drawn antialiased (smoothed) or not. This option does not affect exported plots.

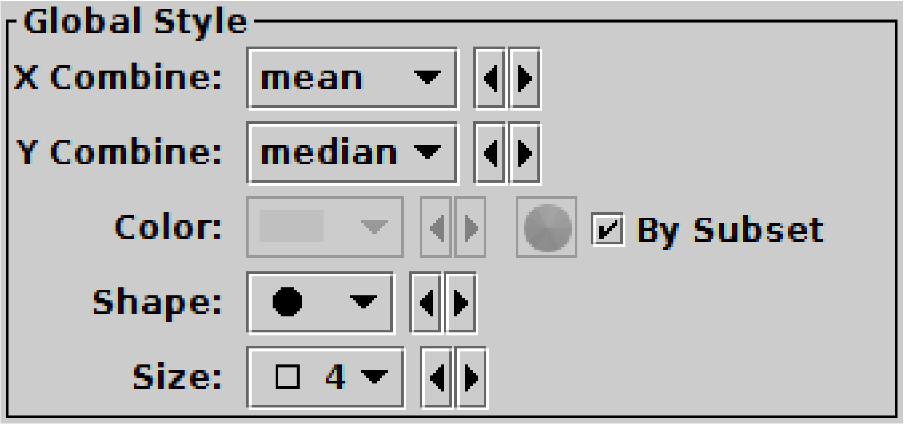
A.4.5.39 StatMark Form

The **StatMark** form (), available from the XYArray Layer Control, plots a set of markers based on a combination (typically the mean) of input array-valued coordinates. The input X and Y coordinates must be fixed-length arrays of length N; N markers are plotted, each one representing the mean (or median, minimum, maximum, ...) of all the input array elements at the corresponding position.

Note that because the X and Y arrays must be of a fixed size for all rows, and because combination is performed in both X and Y directions, this is typically only suitable for plotting combined spectra if they all share a common horizontal axis, e.g. are all sampled into the same wavelength bins. To visually combine spectra with non-uniform sampling, the ArrayQuantile form may be more useful.



Example StatMark plot



Global Style

X Combine: mean

Y Combine: median

Color: ☑ By Subset

Shape: ●

Size: □ 4

StatLine form configuration panel

The configuration options are:

X Combine
Y Combine
Defines how corresponding array elements on the X/Y axis are combined together to produce the plotted value. The following options are currently available:

- `mean`: the mean of the combined values
- `median`: the median of the combined values (may be slow)
- `min`: the minimum of all the combined values
- `max`: the maximum of all the combined values
- `q.01`: the 1st percentile of the combined values (may be slow)
- `q1`: the first quartile of the combined values (may be slow)
- `q3`: the third quartile of the combined values (may be slow)
- `q.99`: the 99th percentile of the combined values (may be slow)
- `stdev`: the sample standard deviation of the combined values
- `sum`: the sum of all the combined values
- `count`: the number of non-blank values

Color
Color of the plotted markers.

Shape
Shape of the plotted markers.

Size
Size of the plotted markers in pixels.

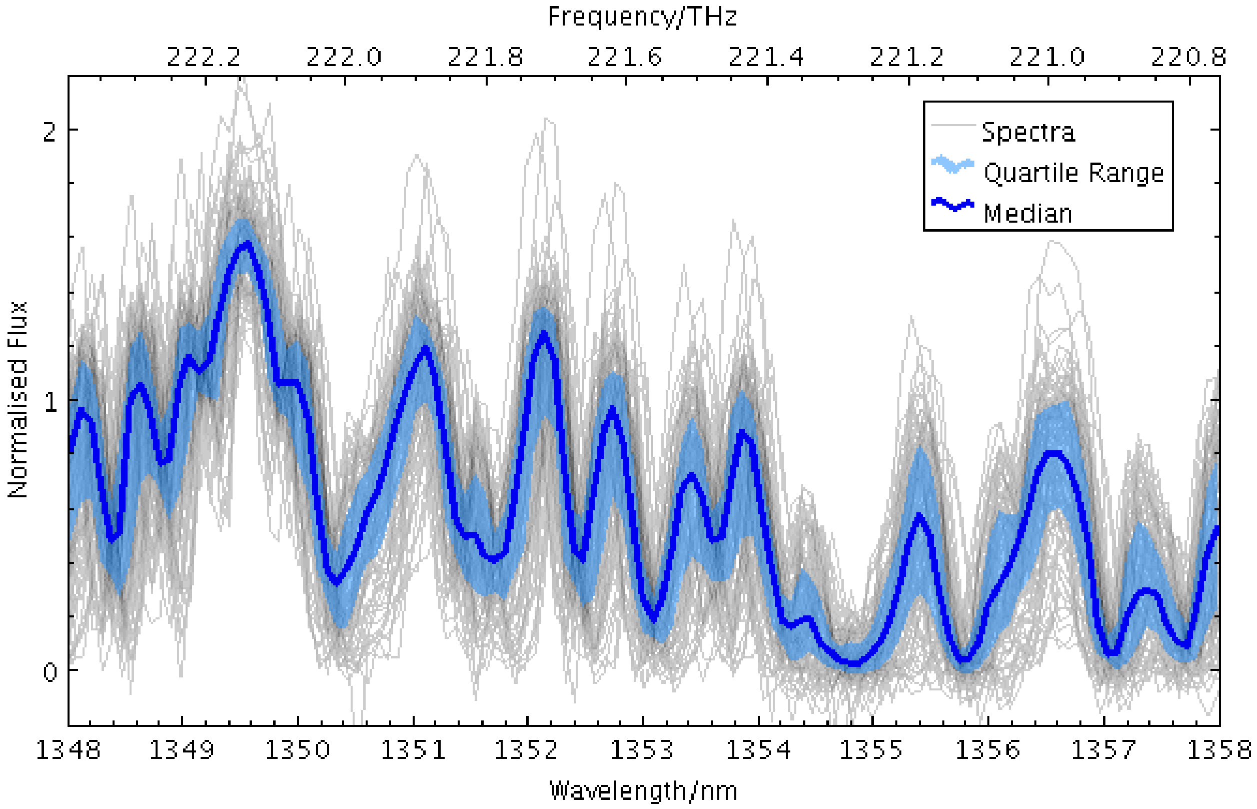
A.4.5.40 ArrayQuantile Form

The **ArrayQuantile** form ( ), available from the XYArray Layer Control, displays a quantile or quantile range for a set of plotted X/Y array pairs. If a table contains one spectrum per row in array-valued wavelength and flux columns, this plotter can be used to display a median of all the spectra, or a range between two quantiles. Smoothing options are available to even out noise arising from the pixel binning.

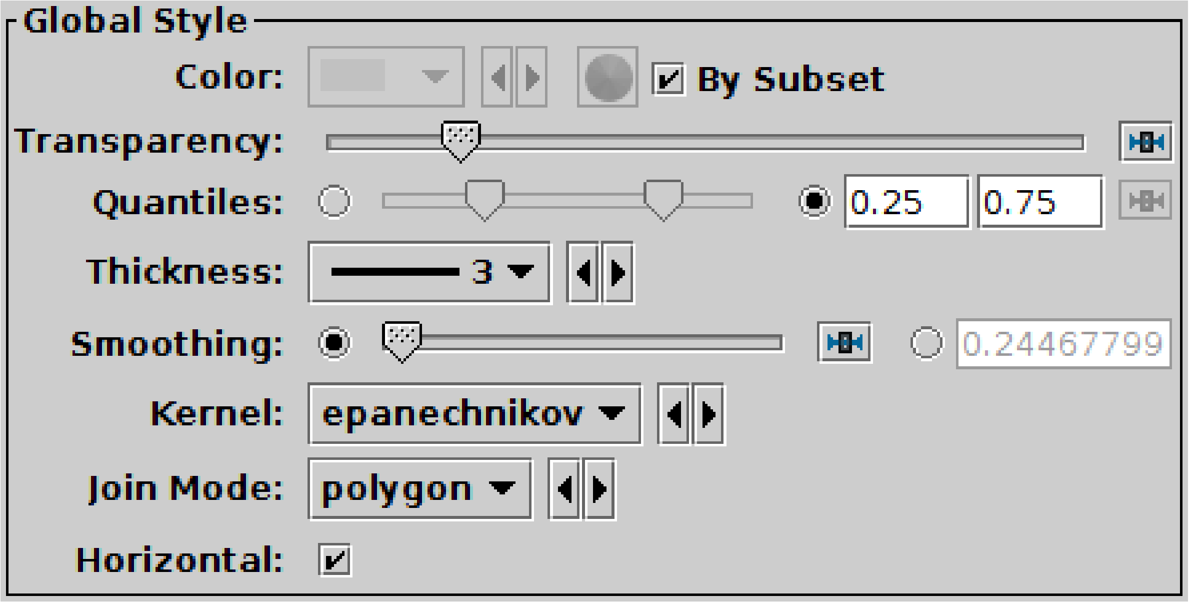
For each row, the **X Values** and **Y Values** arrays must be the same length as each other, but this plot type does not (unlike StatMark and StatLine) require the arrays to be sampled into the same bins for each row.

The algorithm calculates quantiles for all the X,Y points plotted in each column of pixels. This means that more densely sampled spectra have more influence on the output than sparser ones.

Note: In the current implementation, depending on the details of the configuration and the data, there may be some distortions or missing graphics near the edges of the plot. This may be improved in future releases, depending on feedback.



Example ArrayQuantile plot



Global Style

Color: ☑ By Subset

Transparency:

Quantiles: ○ ● 0.25 0.75

Thickness: 3

Smoothing: ● ○ 0.24467799

Kernel: epanechnikov

Join Mode: polygon

Horizontal: ☑

ArrayQuantile form configuration panel

The configuration options are:

Transparency
Transparency with which components are plotted, from opaque to invisible.

Quantiles

Defines the quantile or quantile range of values that should be marked in each pixel column (or row). The slider control goes from 0 (minimum in pixel column/row) to 1 (maximum in pixel column/row), so 0.5 indicates the median. This control is a double-slider, so you can drag out a range of values. If the values are the same as each other, a single point will be indicated, but if there is a range then the area between the indicated quantiles will be filled.

The radio buttons let you toggle between using the slider to set the quantile value(s) or entering them in the text fields. If the two values are identical, you can leave the second text field blank.

Thickness

Sets the minimum extent of the markers that are plotted in each pixel column (or row) to indicate the designated value range. If the range is zero sized (two bounding quantiles are equal) this will give the actual thickness of the plotted line. If the range is non-zero however, the line may be thicker than this in places according to the quantile positions.

Smoothing

Configures the smoothing width. This is the characteristic width of the **Kernel** function to be convolved with the density in one dimension to smooth the quantile function.

You can adjust it using the slider (wider smoothing to the right) or enter a value in data coordinates explicitly in the text field. If the smoothed axis is logarithmic, the value is a multiplication factor rather than an additive increment.

Kernel

The functional form of the smoothing kernel. The functions listed refer to the unscaled shape; all kernels are normalised to give a total area of unity.

The available options are:

- **square**: Uniform value: f(x)=1, |x|=0..1
- **linear**: Triangle: f(x)=1-|x|, |x|=0..1
- **epanechnikov**: Parabola: f(x)=1-x*x, |x|=0..1
- **cos**: Cosine: f(x)=cos(x*pi/2), |x|=0..1
- **cos2**: Cosine squared: f(x)=cos^2(x*pi/2), |x|=0..1
- **gauss3**: Gaussian truncated at 3.0 sigma: f(x)=exp(-x*x/2), |x|=0..3
- **gauss6**: Gaussian truncated at 6.0 sigma: f(x)=exp(-x*x/2), |x|=0..6

Join Mode

Defines the graphical style for connecting distinct quantile values. If smoothed samples are packed more closely than the pixel grid the option chosen here doesn't make much difference, but if there are gaps in the data along the sampled axis, it's useful to have a guide to the eye to join one quantile determination to the next.

The available options are:

- **none**: displayed quantile ranges are not joined
- **polygon**: the area between a line connecting the upper quantiles and a line connecting the lower quantiles is filled
- **lines**: a line of thickness given by **Thickness** is drawn from the center of each quantile range to the next

Horizontal

Determines whether the trace bins are horizontal or vertical. If true, *y* quantiles are calculated for each pixel column, and if false, *x* quantiles are calculated for each pixel row.

A.4.6 Shading Modes

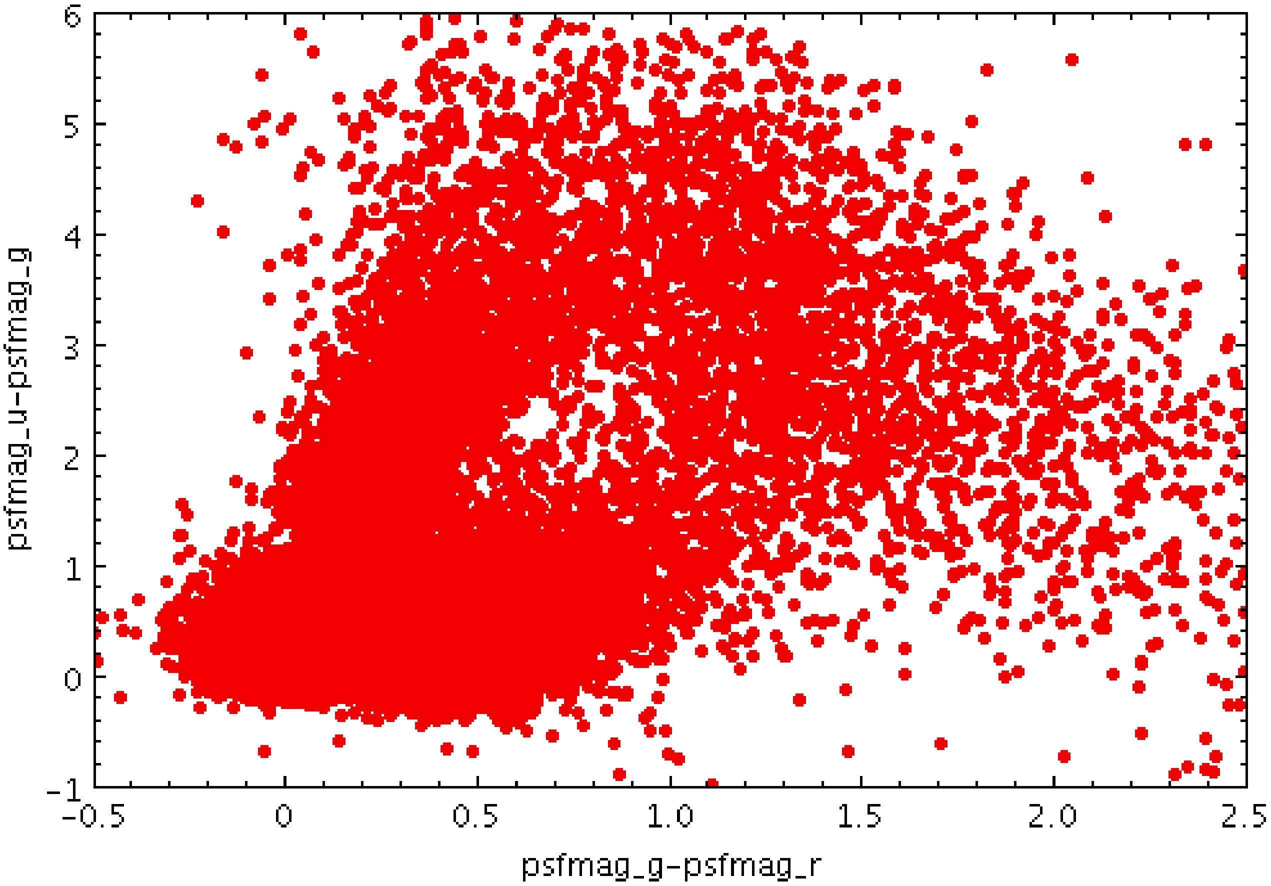
Most of the Plot Forms (Appendix A.4.5) have a style colour associated with them for each data set.

This defines the basic colour used to plot the shape at each data point. However, many of the forms also ask you to select a **Shading Mode**, which determines the actual colour displayed in the plot for the plotted points. The shaded colour is based on the selected style colour, but may also be influenced by the number of points plotted there, some extra data coordinate, or other configuration information.

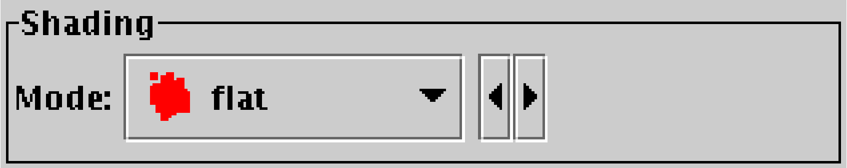
When exporting plots to an external vector graphics format (PDF, SVG or EPS), some of the shading modes may not behave the same as in a bitmap (on the screen, or to a bitmapped format such as GIF or PNG). Any such anomalies are noted in the in the mode descriptions below.

The different mode shading mode options are described in the following subsections.

A.4.6.1 Flat Mode



Example Flat shading mode plot


Shading

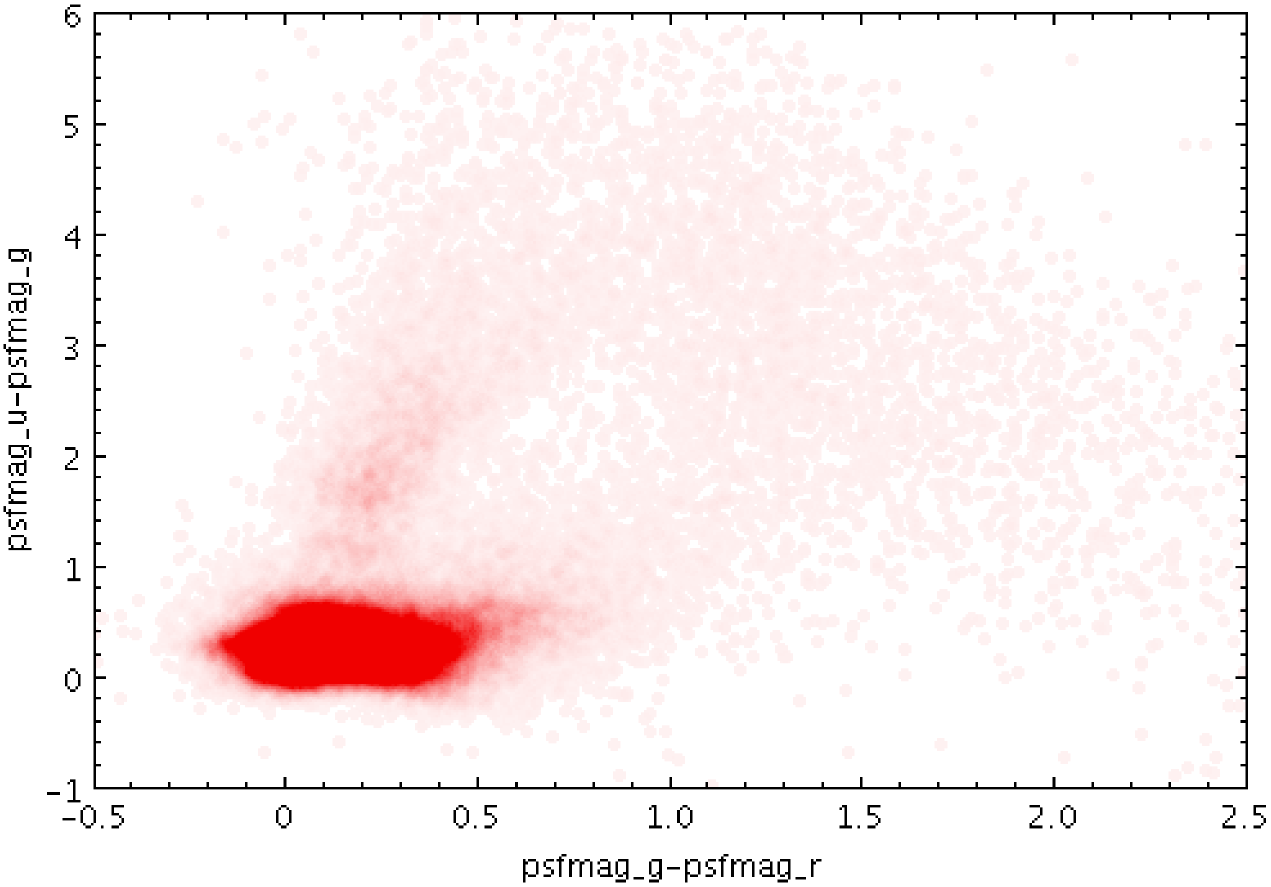
Mode: ● flat

Flat shading mode selection

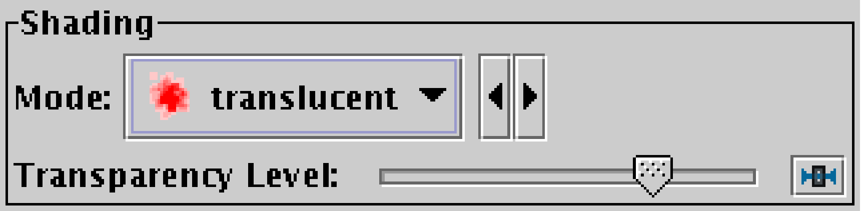
The **Flat** shading mode (●) simply colours points in the colour selected by their style. It has no additional parameters or coordinates.

Exporting: This mode works without problem for both bitmapped and vector output.

A.4.6.2 Translucent Mode



Example Translucent shading mode plot



Shading

Mode: translucent

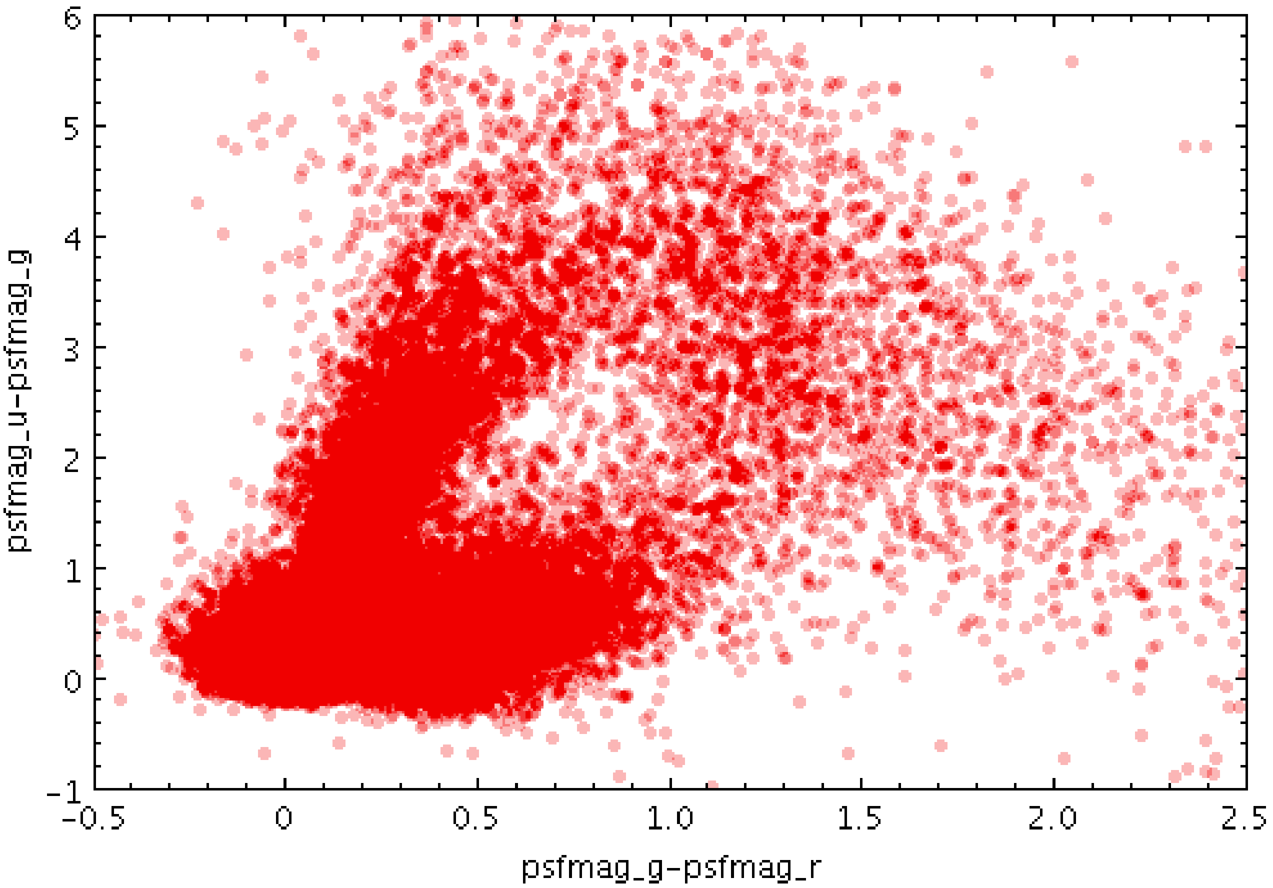
Transparency Level:

Translucent shading mode selection

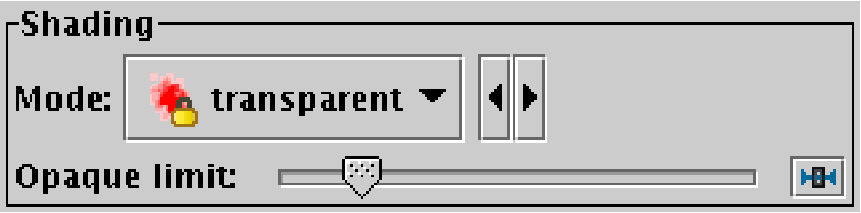
The **Translucent** shading mode ( ) colours shapes in a transparent version of the colour selected by their style. The degree of transparency is determined by how many points are currently being plotted on top of each other, and by the **Transparency Level** slider; as you slide further to the right, points get more transparent. Unlike transparent mode, the transparency varies according to the current point density, so you can usually leave the setting the same as you zoom in and out.

Exporting: When the points are opaque, this mode works without problem for both bitmapped and vector output, but when the transparency is set there may be anomalies. Transparent points are rendered in PDF output, though the transparency levels may not be exactly the same as on the screen. This can be fixed by using the **Force Bitmap** option in the Plot Export (Appendix A.4.2.5) dialogue. For PostScript, transparent points are rendered as opaque. You can use **Force Bitmap** with PostScript which will get transparency right for this layer, but then any earlier layers will be completely obscured.

A.4.6.3 Transparent Mode



Example Transparent shading mode plot



Shading

Mode: transparent

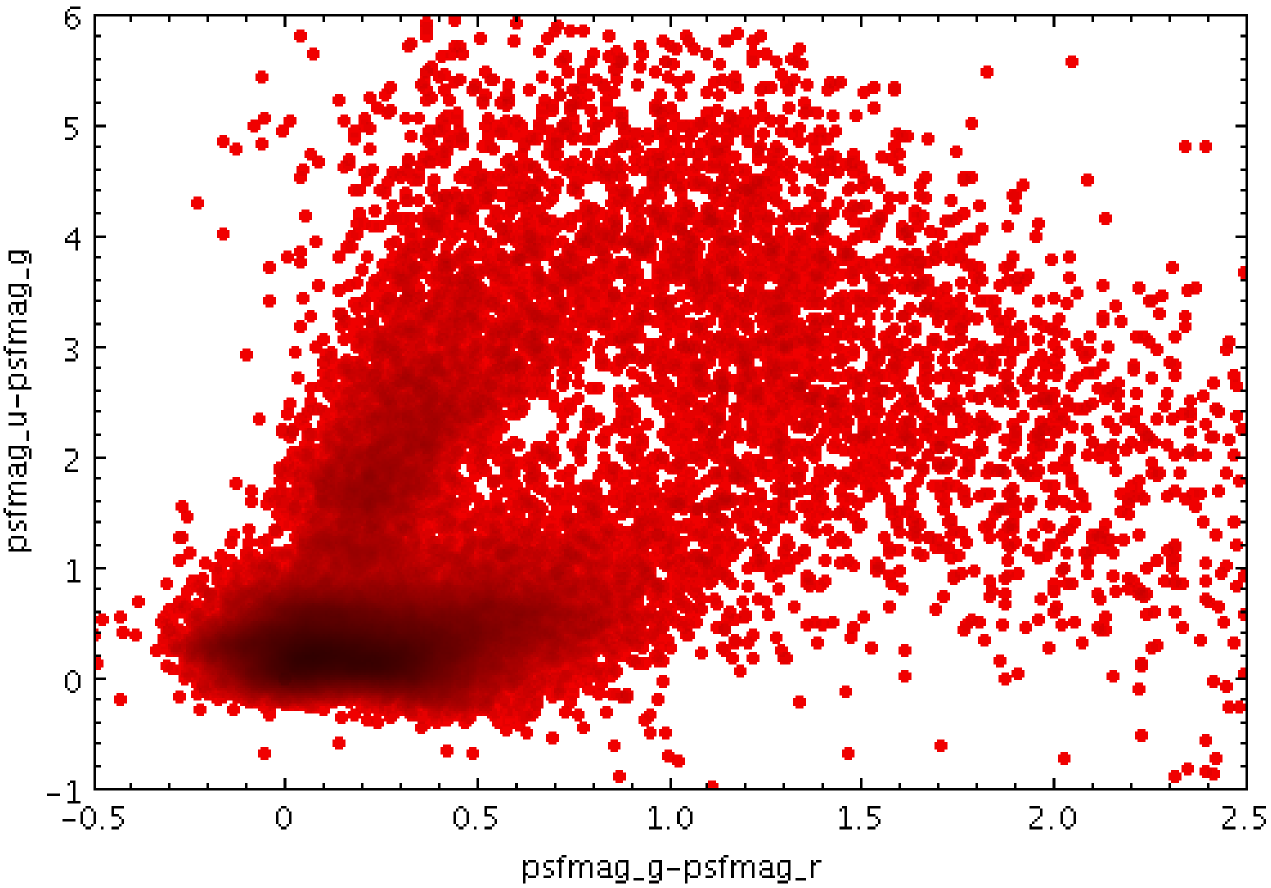
Opaque limit:

Transparent shading mode selection

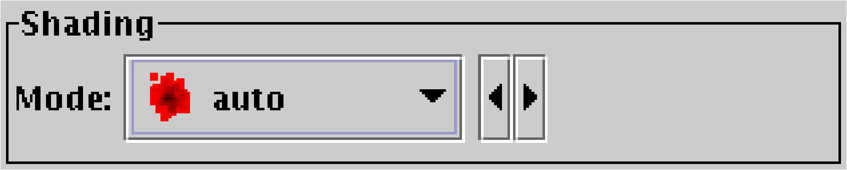
The **Transparent** shading mode ( ) colours shapes in a transparent version of the colour selected by their style. The degree of transparency is determined by the **Opaque Limit** slider - at the left end, points are fully opaque and this is equivalent to Flat mode, and as you slide further to the right, the points get more transparent. The higher the opaque limit, the more points have to be plotted on top of each other to reach colour that fully obscures the background. Unlike translucent mode, transparency of each colour is fixed by the opaque limit, rather than adjusting depending on the density of points currently plotted.

Exporting: When the points are opaque, this mode works without problem for both bitmapped and vector output, but when the transparency is set there may be anomalies. Transparent points are rendered in PDF output, though the transparency levels may not be exactly the same as on the screen. This can be fixed by using the **Force Bitmap** option in the Plot Export (Appendix A.4.2.5) dialogue. For PostScript, transparent points are rendered as opaque. You can use **Force Bitmap** with PostScript which will get transparency right for this layer, but then any earlier layers will be completely obscured.

A.4.6.4 Auto Mode



Example Auto shading mode plot


Shading

Mode: auto

Auto shading mode selection

The **Auto** shading mode (●) colours isolated points in their selected colour, but where multiple points *from the same data set* overlap it adjusts the colour by darkening it. This means for that isolated points (most or all points in a non-crowded plot, or outliers in a crowded plot) it behaves just like Flat mode, but it's easy to see where overdense regions lie.

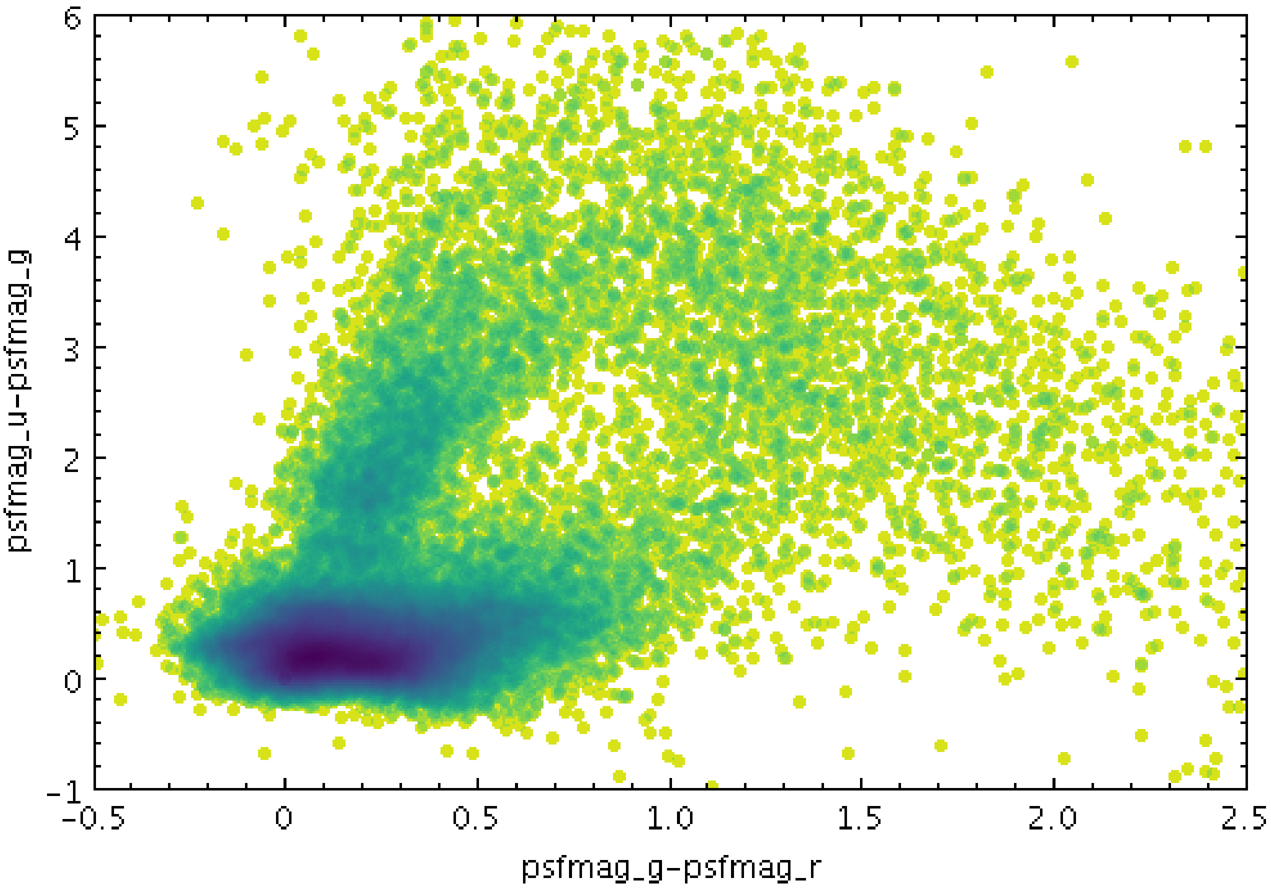
This is like Density mode, but with no user-configurable options.

This is the default mode for 2d plots, since it gives you a good first idea of what the data is doing. For 3d plots it can be used, and it works well for single dataset plots, but in the case of multiple datasets it can be misleading since the coloured pixels can't be placed sensibly in the 3d space.

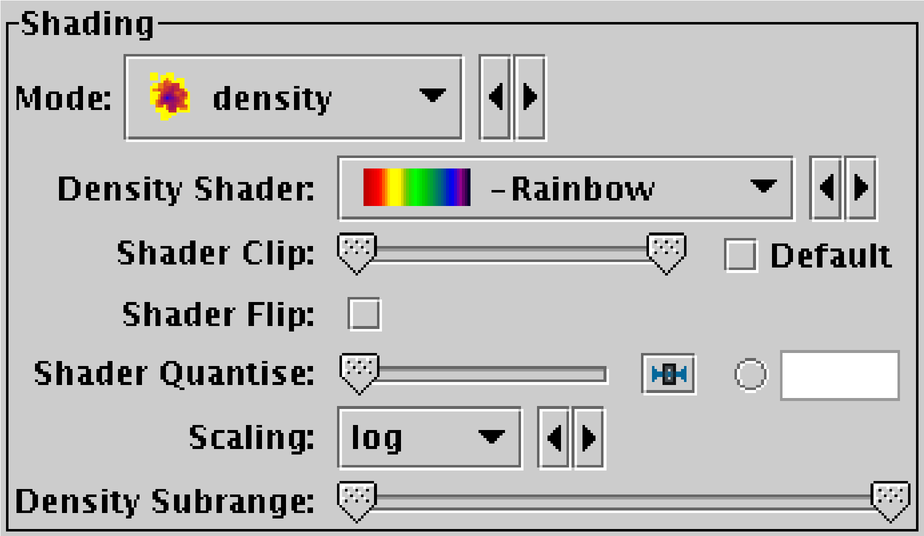
The colour darkening is based on the asinh function; the intention is that two points overlaid should be just enough different in colour for the difference to be visible, and the mapping is scaled so that if there are very dense regions they will come out nearly black.

Exporting: When exported to vector formats, the output is automatically forced to a bitmap for Auto-mode layers. In the case of PostScript, this completely obscures any previous layers.

A.4.6.5 Density Mode



Example Density shading mode plot


Shading

Mode: density

Density Shader: -Rainbow

Shader Clip: □ Default

Shader Flip: □

Shader Quantise: ○

Scaling: log

Density Subrange:

Density mode selection

The **Density** shading mode ( ) uses a configurable colour map to indicate how many points are plotted over each other. Specifically, it colours each pixel according to how many times that pixel has has been covered by one of the shapes plotted by the layer in question. To put it another way, it generates a false-colour density map with pixel granularity using a smoothing kernel of the form of the shapes plotted by the layer. The upshot is that you can see the plot density of points or other shapes plotted.

This is like Auto mode, but with more user-configurable options. The options are:

Density Shader
The colour map for displaying density values. There are two types, relative and absolute. Relative maps have names marked by a star ("*"), and alter the basic dataset colour, for instance by darkening or lightening it, while absolute maps (the rest) ignore the basic dataset colour altogether. For a single-dataset plot, the absolute maps are best, but for multiple subsets it may be less confusing to use a relative one. Colour maps are listed in Appendix A.4.7.

Shader Clip
Select a sub-range of the full colour map above. If the **Default** checkbox is checked, then all or most of the colour ramp from the **Shader** control is used. If you want to configure the range of colours from the map yourself, uncheck the Default checkbox, and slide the handles in from the end of the slider to choose exactly the range you want. The default range is clipped at one end for colour maps that fade to white, so that all the plotted colours will be distinguishable against a white background. If you don't want that, you can uncheck Default and leave the handles at the extreme ends of the slider.

Shader Flip
Whether the density scale should map forwards or backwards into the colour map.

Shader Quantise
Allows the colour map to be quantised. By default, the colour map is effectively continuous. If you slide the slider to the right, or enter a value in the text field, the map will be split into a decreasing number of discrete colours. This can be used to generate a contour-like effect, and may make it easier to trace the boundaries of regions of interest by eye.

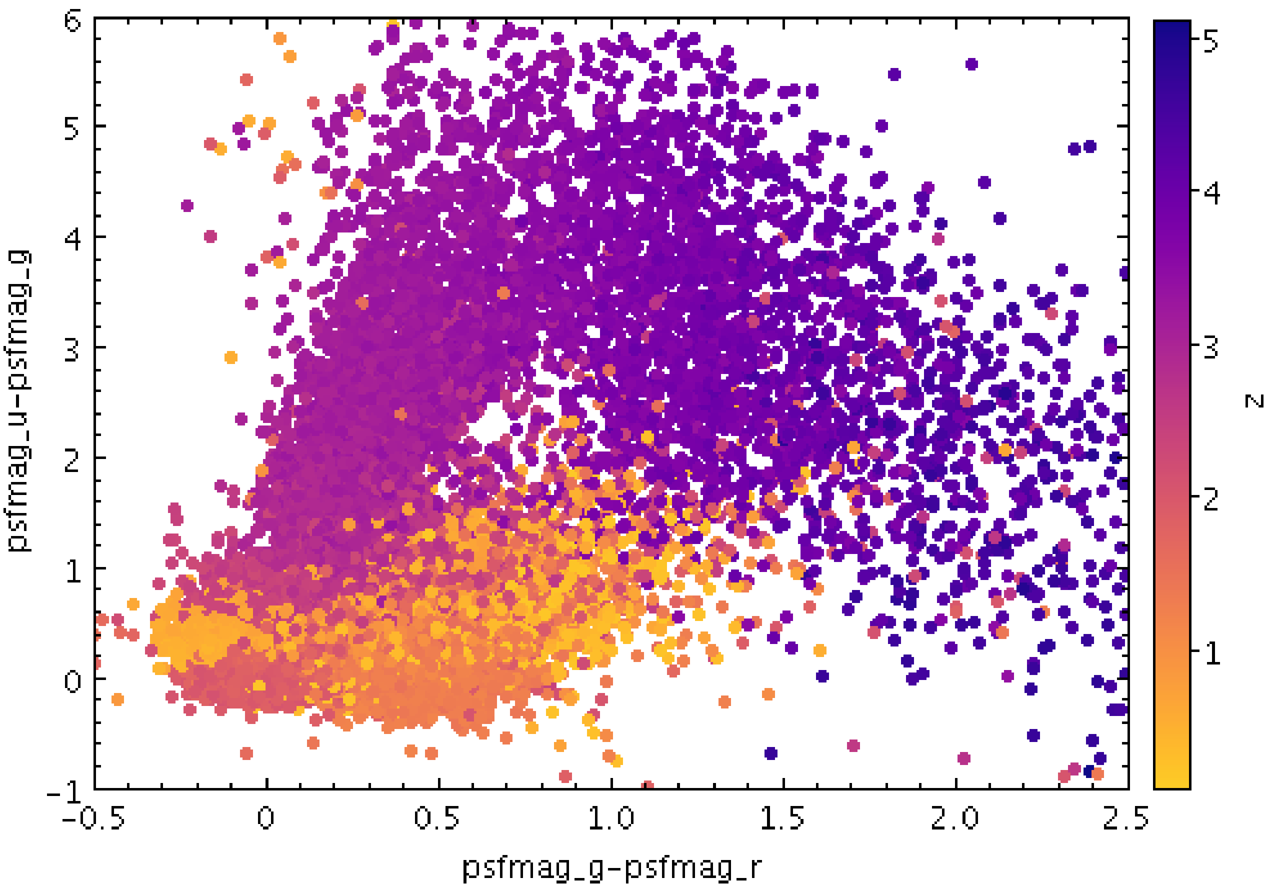
Scaling
Determines the function used to map the range of density values onto the colour map. Options are **linear**, **logarithmic**, **histogram**, **logarithmic histogram**, **square** and **square root**, **arc cosine**, **cosine**.

Density Subrange
Adjusts the density range over which the colour map is applied. By default the colour map is scaled using limits found from the data density in the plot (the most dense few pixels are ignored), but you can restrict the range using this slider.

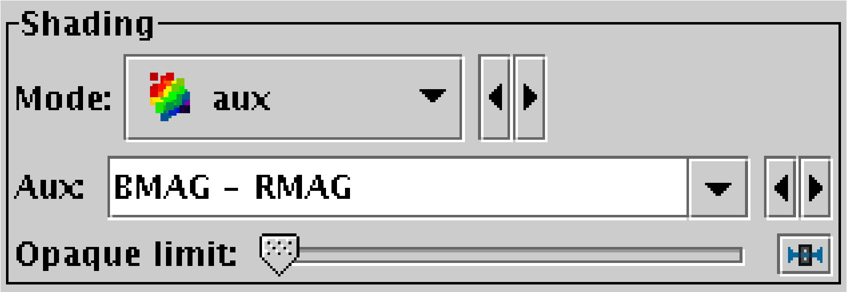
Although these options give you quite some control over how densities are mapped to colours, this mode does not display the colour mapping in a way that shows you quantitatively which colours correspond to which numeric density values. If you want that kind of visual feedback, you should use the Weighted shading mode, which can be configured to display point densities (as well as other quantities), and also causes a colour ramp to be displayed under control of the Aux Axis control.

Exporting: When exported to vector formats, the output is automatically forced to a bitmap for Density-mode layers. In the case of PostScript, this completely obscures any previous layers.

A.4.6.6 Aux Mode



Example Aux shading mode plot



Shading

Mode: aux

Aux: BMAG - RMAG

Opaque limit:

Aux mode selection

The **Aux** shading mode ( ) colours each point according to the value of an additional data coordinate, using a colour map shared with other layers in the plot. The point colours then represent an additional dimension. There is an additional option to draw the points with a fixed transparency.

The shading is done using the shared colour map. This colour map is used by all currently visible **Aux**, Weighted, Grid and SkyDensity layers. When at least one such layer is being plotted, the Aux Axis control (Appendix A.4.3.5) is visible in the control panel, which allows you to configure the colour map, range, ramp display etc.

The options are:

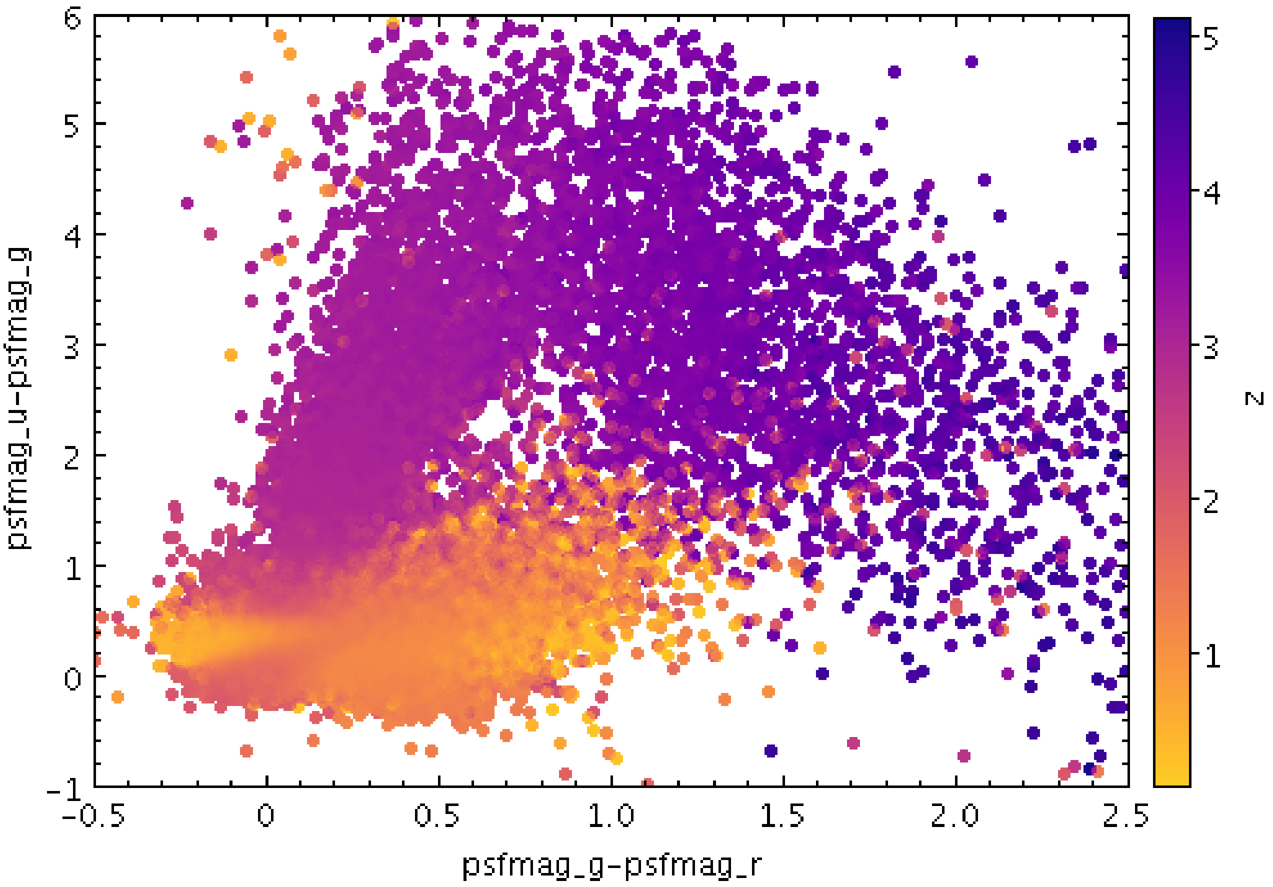
Aux
: The auxiliary coordinate data values. Fill this in with a column name or expression from the table just like for a positional coordinate.

Opaque Limit
: Determines transparency of the points. By default, they are fully opaque, but if you slide the

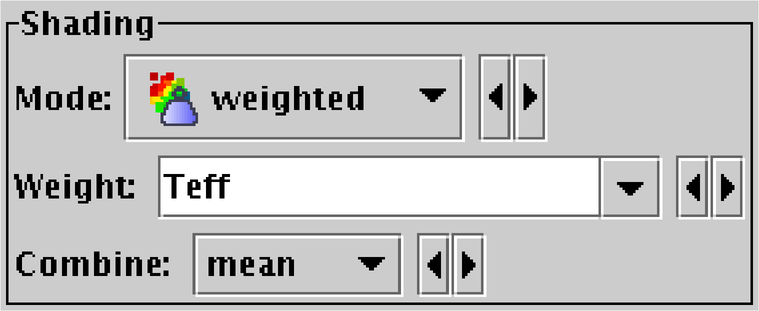
slider to the right, they will become progressively more transparent.

Exporting: Transparent points are rendered in PDF output, though the transparency levels may not be exactly the same as on the screen. This can be fixed by using the **Force Bitmap** option in the Plot Export (Appendix A.4.2.5) dialogue. For PostScript, transparent points are rendered as opaque. You can use **Force Bitmap** with PostScript which will get transparency right for this layer, but then any earlier layers will be completely obscured.

A.4.6.7 Weighted Mode



Example Weighted shading mode plot



Shading

Mode: weighted

Weight: Teff

Combine: mean

Weighted mode selection

The **Weighted** shading mode ( ) paints markers using colours indicating density at each pixel like the Density mode, but with an optional weighting coordinate, using a colour map shared with other layers in the plot. You can configure how the weighted coordinates are combined at each pixel to give the final weighted result. Depending on the configuration and actual point density, this can behave in some ways like Aux mode and in some ways like Density mode, but allows you to do things that are not possible in either.

The shading is done using the shared colour map. This colour map is used by all currently visible **Weighted**, Aux, Grid and SkyDensity layers. When at least one such layer is being plotted, the Aux Axis control (Appendix A.4.3.5) is visible in the control panel, which allows you to configure the colour map, range, ramp display etc.

The options are:

Weight

The weight value applied to each plotted point. Fill this in with a column name or expression from the table just like for a positional coordinate. The exact way this quantity is used depends on the setting of the **Combine** control below. If it's left blank, the weighting is considered to be unity (all values are 1); this makes sense for some combination types (e.g. sum) but not others (e.g. mean).

Combine

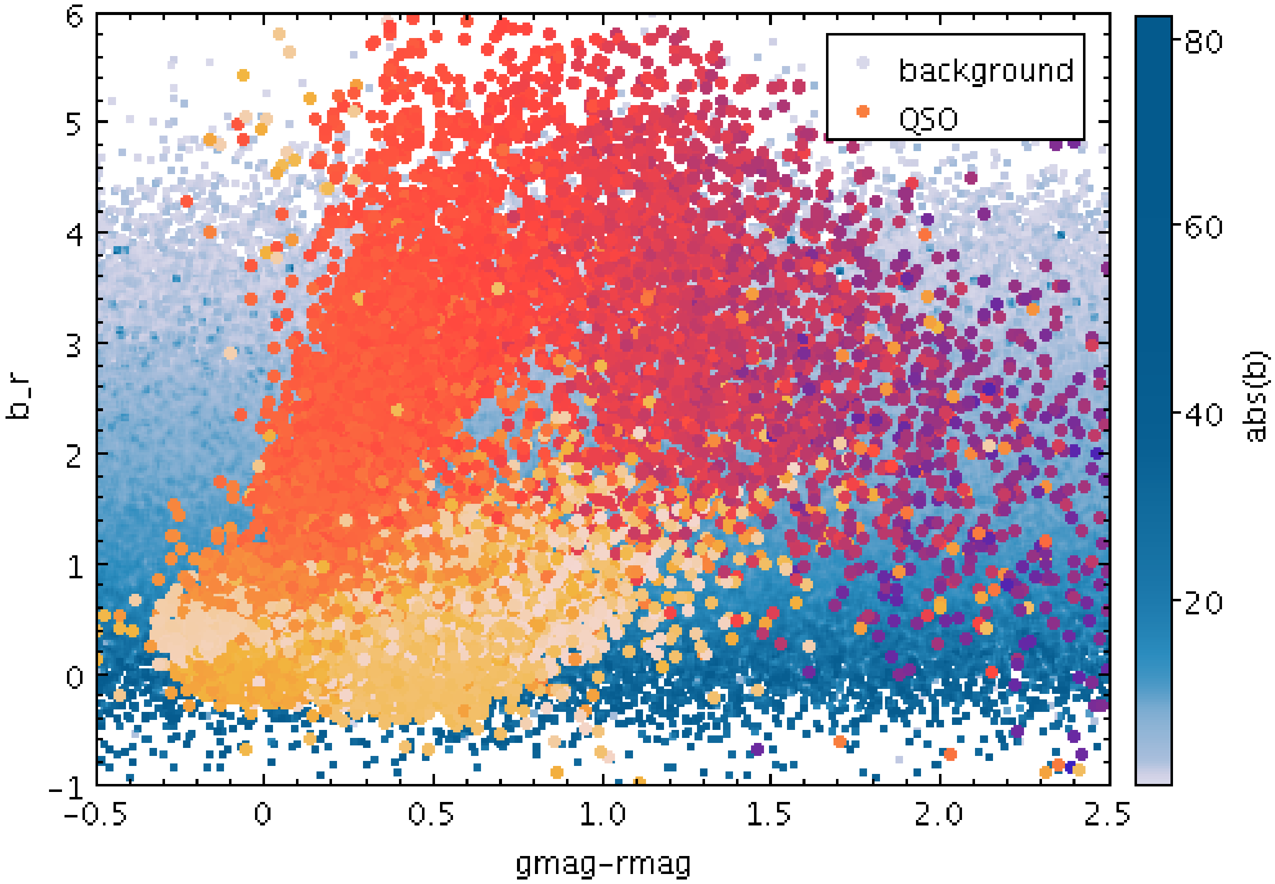
Determines how the weight values associated with markers plotted covering a given screen pixel are combined to produce the numeric value used for that pixel's colour.

The following options (some are more useful than others) are currently available:

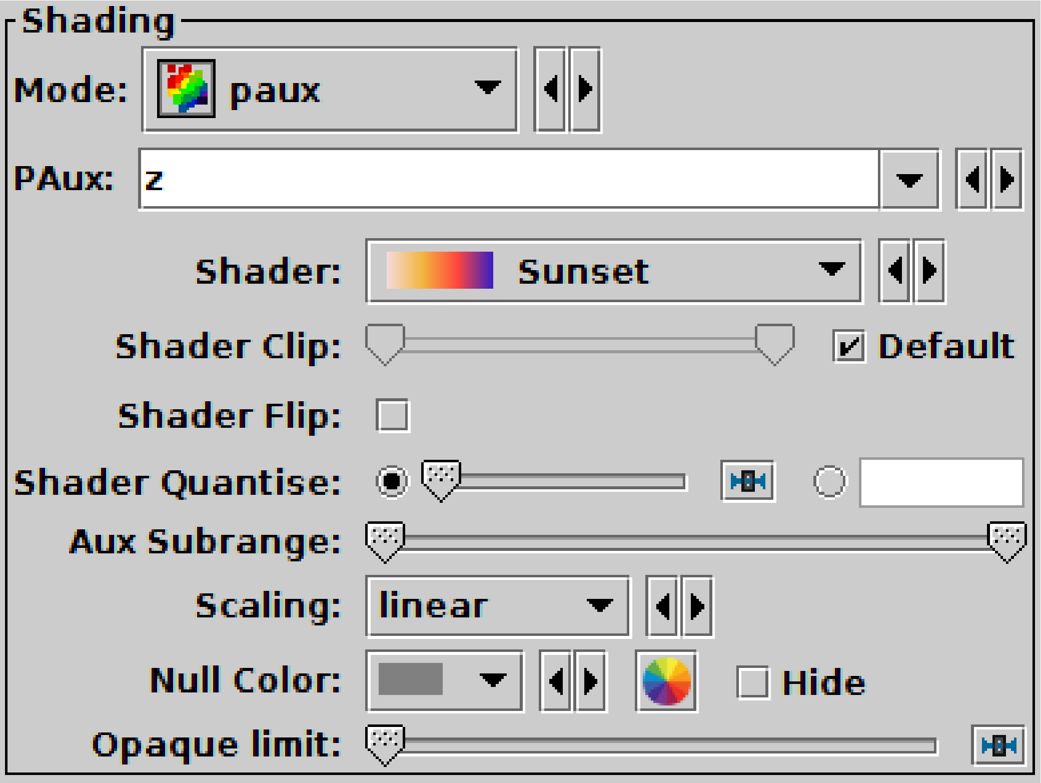
- `sum`: the sum of all weights
- `mean`: the mean of all weights
- `median`: the median of all weights (may be slow)
- `q1`: the first quartile of all weights (may be slow)
- `q3`: the third quartile of all weights (may be slow)
- `min`: the minimum weight
- `max`: the maximum weight
- `stdev`: the sample standard deviation of all weights
- `count`: the number of points plotted (weight value is ignored, this is like Density mode)
- `hit`: one if any point is plotted, blank otherwise (weight value is ignored, this is like Flat mode)

Exporting: When exported to vector formats, the output is automatically forced to a bitmap for Density-mode layers. In the case of PostScript, this completely obscures any previous layers.

A.4.6.8 PAux Mode



Example PAux shading mode plot



Shading

Mode: paux

PAux: z

Shader: Sunset

Shader Clip: ☑ Default

Shader Flip: □

Shader Quantise: ● ○

Aux Subrange:

Scaling: linear

Null Color: □ Hide

Opaque limit:

PAux mode selection

The **PAux** (private-auxiliary) shading mode (▦) colours each point according to the value of an additional data coordinate, but using a private colour map. The point colours then represent an additional dimension. There is an additional option to draw the points with a fixed transparency.

This mode is like Aux mode, except that the colour map is not shared with other layers, and the colour ramp is not displayed. So, by using this mode alongside **Aux** or Weighted modes you can make a plot that uses multiple different colour maps, though only one can have an associated visible ramp.

The options are:

PAux
The auxiliary coordinate data values. Fill this in with a column name or expression from the table just like for a positional coordinate.

Shader
The colour map for displaying density values. There are two types, relative and absolute. Relative maps have names marked by a star ("*"), and alter the basic dataset colour, for instance by darkening or lightening it, while absolute maps (the rest) ignore the basic dataset colour altogether. For a single-dataset plot, the absolute maps are best, but for multiple subsets it may be less confusing to use a relative one. Colour maps are listed in Appendix A.4.7.

Shader Clip
Select a sub-range of the full colour map above. If the **Default** checkbox is checked, then all or most of the colour ramp from the **Shader** control is used. If you want to configure the range of colours from the map yourself, uncheck the Default checkbox, and slide the handles in from the end of the slider to choose exactly the range you want. The default range is clipped at one end for colour maps that fade to white, so that all the plotted colours will be distinguishable against a white background. If you don't want that, you can uncheck Default and leave the handles at the extreme ends of the slider.

Shader Flip
Whether the density scale should map forwards or backwards into the colour map.

Shader Quantise
Allows the colour map to be quantised. By default, the colour map is effectively continuous. If you slide the slider to the right, or enter a value in the text field, the map will be split into a decreasing number of discrete colours. This can be used to generate a contour-like effect, and may make it easier to trace the boundaries of regions of interest by eye.

Aux Subrange
Adjusts the density range over which the colour map is applied. By default the colour map is scaled using limits found from the aux coordinate in the plot, but you can restrict the range using this slider.

Scaling
Determines the function used to map the range of aux coordinate values onto the colour map. Options are **linear**, **logarithmic**, **histogram**, **logarithmic histogram**, **square** and **square root**, **arc cosine**, **cosine**.

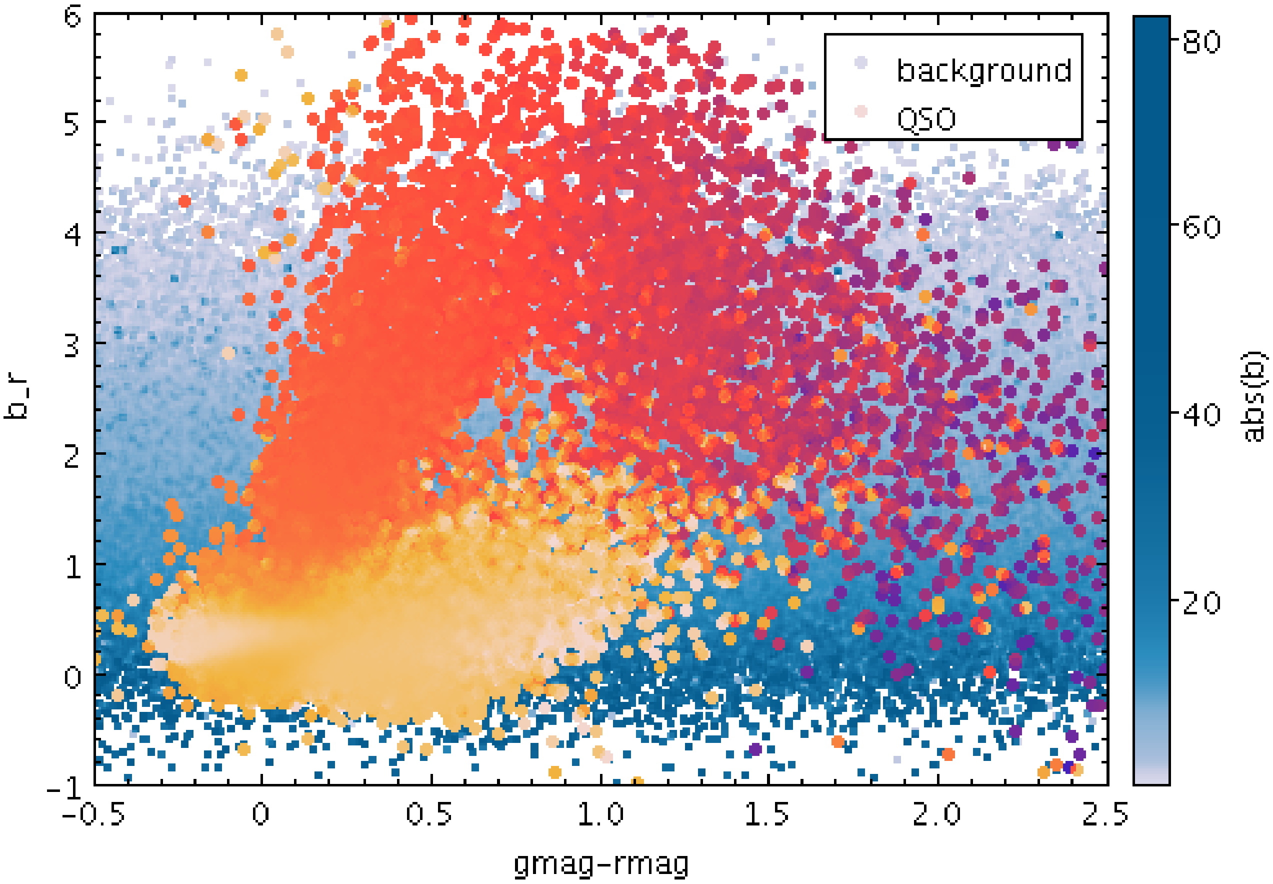
Null Colour
What colour should be used to represent points with a null value for the aux data coordinate. If the associated **Hide** option is selected, then those points will not appear in the plot at all.

Opaque Limit
Determines transparency of the points. By default, they are fully opaque, but if you slide the slider to the right, they will become progressively more transparent.

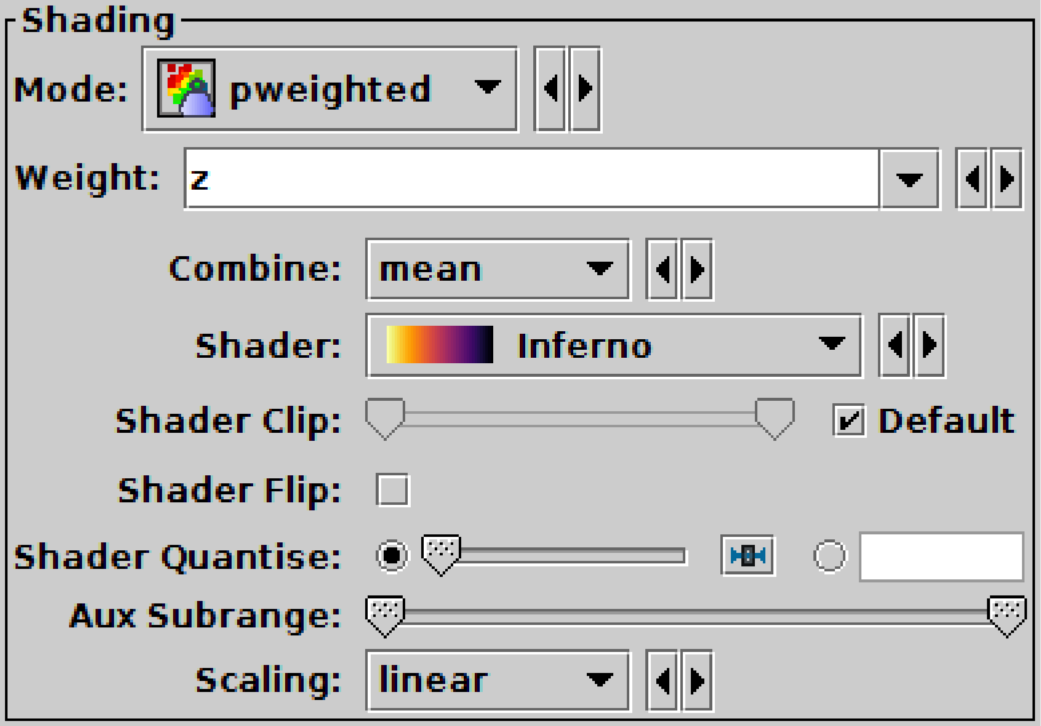
Exporting: Transparent points are rendered in PDF output, though the transparency levels may not be exactly the same as on the screen. This can be fixed by using the **Force Bitmap** option in the Plot Export (Appendix A.4.2.5) dialogue. For PostScript, transparent points are rendered as opaque.

You can use **Force Bitmap** with PostScript which will get transparency right for this layer, but then any earlier layers will be completely obscured.

A.4.6.9 PWeighted Mode



Example PWeighted shading mode plot


Shading

Mode: pweighted

Weight: z

Combine: mean

Shader: Inferno

Shader Clip: ☑ Default

Shader Flip: □

Shader Quantise: ● ○

Aux Subrange:

Scaling: linear

PWeighted mode selection

The **PWeighted** (private-weighted) shading mode () paints markers using colour indicating weighted density at each pixel, using a private colour map.

This mode is like Weighted mode, except that the colour map is not shared with other layers, and the colour ramp is not displayed. So, by using this mode alongside Aux or **Weighted** modes you can make a plot that uses multiple different colour maps, though only one can have an associated visible ramp.

The options are:

Weight

The weight value applied to each plotted point. Fill this in with a column name or expression from the table just like for a positional coordinate. The exact way this quantity is used depends on the setting of the **Combine** control below. If it's left blank, the weighting is considered to be unity (all values are 1); this makes sense for some combination types (e.g. sum) but not others (e.g. mean).

Combine

Determines how the weight values associated with markers plotted covering a given screen pixel are combined to produce the numeric value used for that pixel's colour.

The following options (some are more useful than others) are currently available:

- `sum`: the sum of all weights
- `mean`: the mean of all weights
- `median`: the median of all weights (may be slow)
- `q1`: the first quartile of all weights (may be slow)
- `q3`: the third quartile of all weights (may be slow)
- `min`: the minimum weight
- `max`: the maximum weight
- `stdev`: the sample standard deviation of all weights
- `count`: the number of points plotted (weight value is ignored, this is like Density mode)
- `hit`: one if any point is plotted, blank otherwise (weight value is ignored, this is like Flat mode)

Shader

The colour map for displaying density values. There are two types, relative and absolute. Relative maps have names marked by a star ("*"), and alter the basic dataset colour, for instance by darkening or lightening it, while absolute maps (the rest) ignore the basic dataset colour altogether. For a single-dataset plot, the absolute maps are best, but for multiple subsets it may be less confusing to use a relative one. Colour maps are listed in Appendix A.4.7.

Shader Clip

Select a sub-range of the full colour map above. If the **Default** checkbox is checked, then all or most of the colour ramp from the **Shader** control is used. If you want to configure the range of colours from the map yourself, uncheck the Default checkbox, and slide the handles in from the end of the slider to choose exactly the range you want. The default range is clipped at one end for colour maps that fade to white, so that all the plotted colours will be distinguishable against a white background. If you don't want that, you can uncheck Default and leave the handles at the extreme ends of the slider.

Shader Flip

Whether the density scale should map forwards or backwards into the colour map.

Shader Quantise

Allows the colour map to be quantised. By default, the colour map is effectively continuous. If you slide the slider to the right, or enter a value in the text field, the map will be split into a decreasing number of discrete colours. This can be used to generate a contour-like effect, and may make it easier to trace the boundaries of regions of interest by eye.

Aux Subrange
Adjusts the density range over which the colour map is applied. By default the colour map is scaled using limits found from the aux coordinate in the plot, but you can restrict the range using this slider.

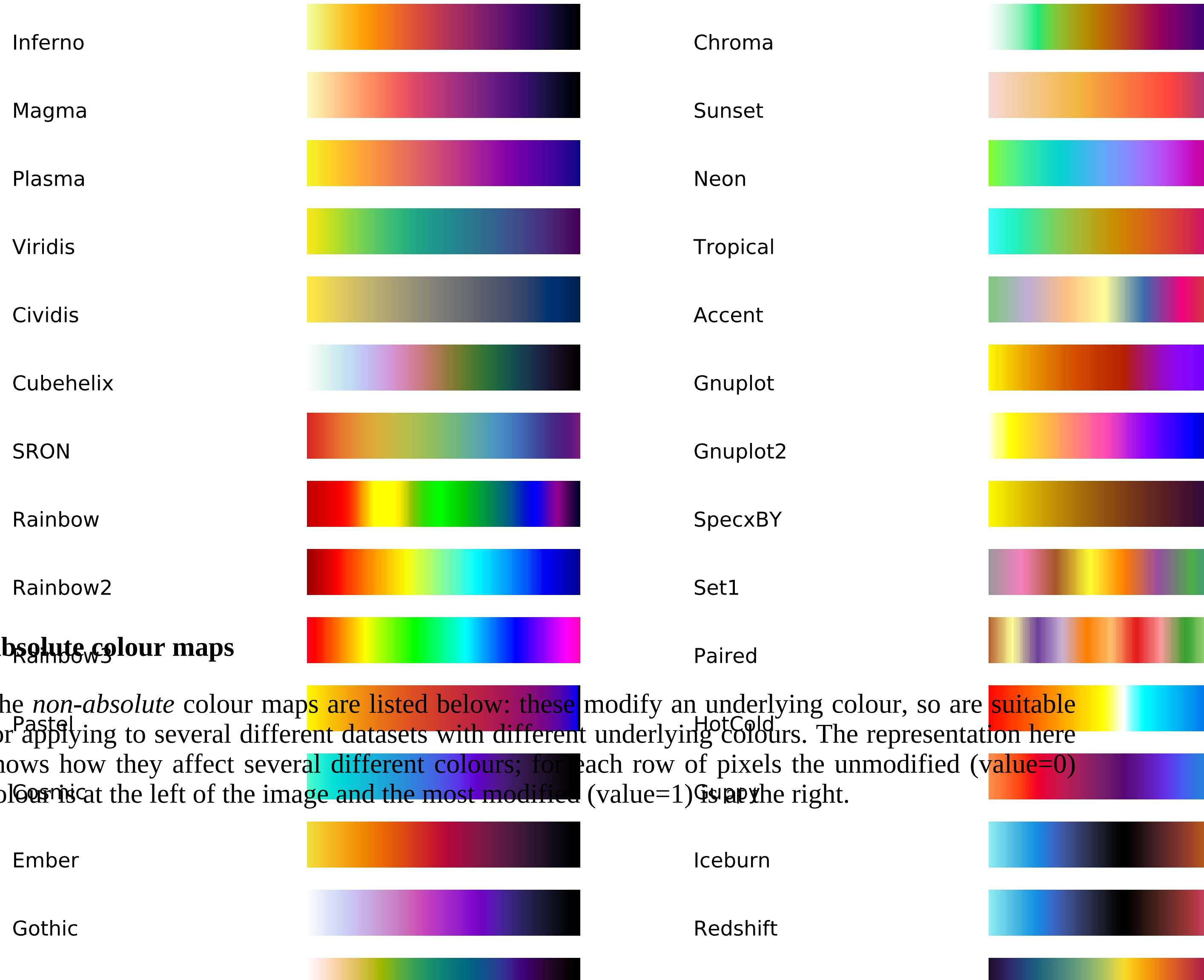
Scaling
Determines the function used to map the range of aux coordinate values onto the colour map. Options are **linear**, **logarithmic**, **histogram**, **logarithmic histogram**, **square** and **square root**, **arc cosine**, **cosine**.

Exporting: When exported to vector formats, the output is automatically forced to a bitmap for PWeighted-mode layers. In the case of PostScript, this completely obscures any previous layers.

A.4.7 Colour Maps

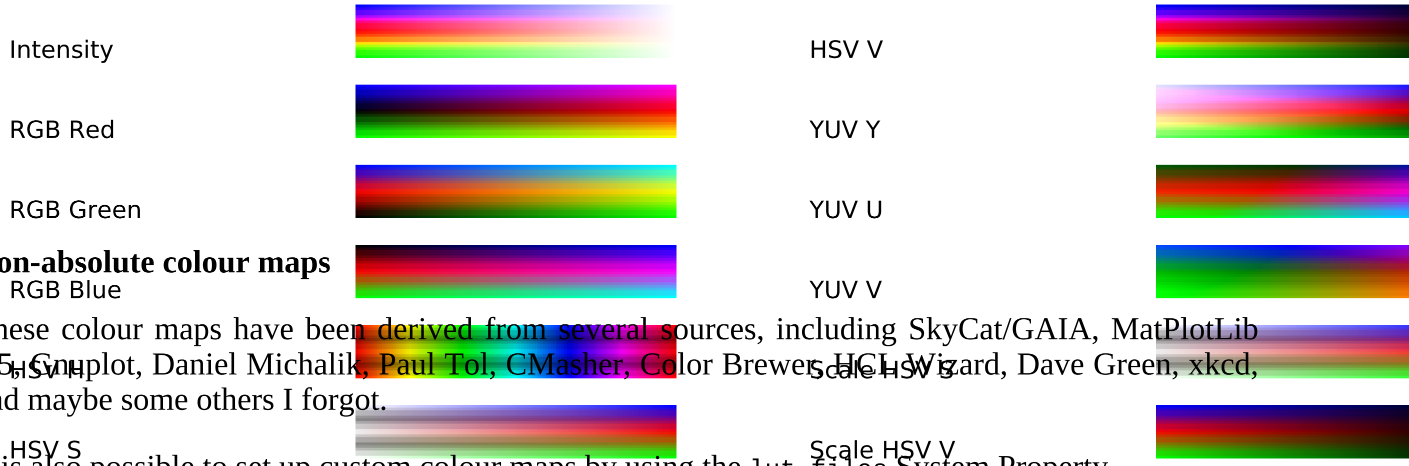
A number of colour maps are available, and used for instance with the Aux Axis Control and Density Shading Mode. Not all colour maps are suitable/available in all contexts, and in some cases the maps are by default clipped at one end to avoid for instance white-on-white plotting, but the lists below give an overview of which named colourmaps can be used.

The *absolute* colour maps are listed below: these do not depend on the underlying colour of the plotted symbols, so are suitable when only one dataset is being plotted.



Absolute colour maps

The *non-absolute* colour maps are listed below: these modify an underlying colour, so are suitable for applying to several different datasets with different underlying colours. The representation here shows how they affect several different colours; for each row of pixels the unmodified (value=0) colour is at the left of the image and the most modified (value=1) is at the right.

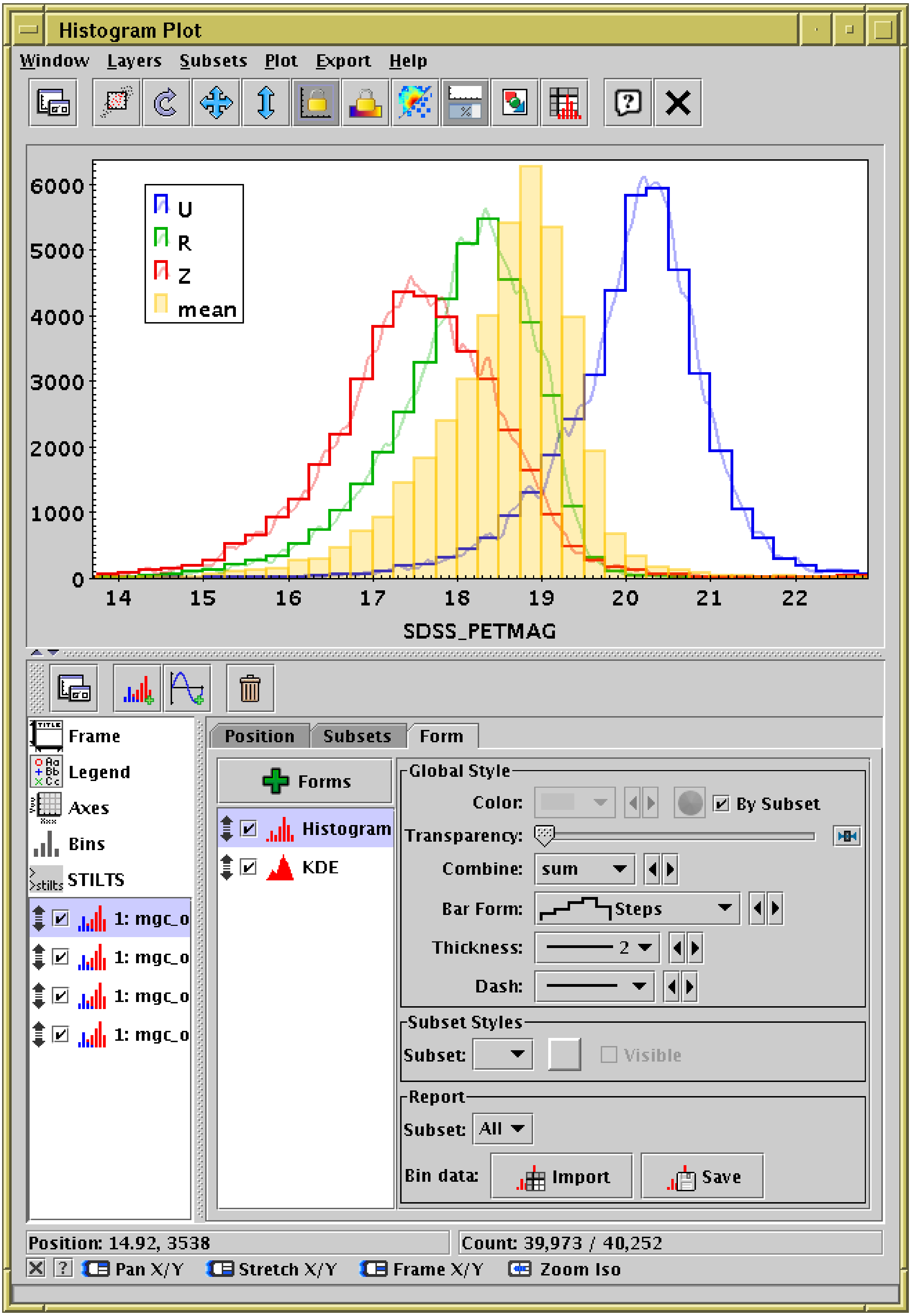


Non-absolute colour maps

These colour maps have been derived from several sources, including SkyCat/GAIA, MatPlotLib 1.5, Gnuplot, Daniel Michalik, Paul Tol, CMasher, Color Brewer, HCL Wizard, Dave Green, xkcd, and maybe some others I forgot.

It is also possible to set up custom colour maps by using the `lut.files` System Property.

A.4.8 Histogram Plot Window



Global Style

Color: ☑ By Subset

Transparency:

Combine: sum

Bar Form: Steps

Thickness: 2

Dash:

Subset Styles

Subset: ☐ Visible

Report

Subset: All

Bin data: Import Save

Position: 14.92, 3538

Count: 39,973 / 40,252

Histogram Plot Window

The **Histogram Plot** () plots 1-dimensional histograms and some variants on the idea of a 1-dimensional Kernel Density Estimate. In many respects it works like the Plane Plot, but it has a restricted set of plot types and an additional fixed control Bins, and the scrolling works a bit differently.

See the Window Overview (Appendix A.4.2) for features common to all plotting windows.

As well as the standard actions, this window additionally provides the following toolbar buttons:

Rescale Y

Adjusts the vertical range of the visible data region to accommodate all the histogram bars in the currently visible horizontal region of the plot. The horizontal range is not changed.

Measure Distance

In addition, the histogram window lets you export the binned data as a new table, either saving it or loading it directly into TOPCAT's table list. The following actions are available in the **Export** menu; note they only apply to histograms proper, not to KDEs:

Save as Table

The bin counts/sums corresponding to the currently plotted histogram will be written to disk in tabular form. The first two columns give the lower and upper bounds of each bin, and the subsequent columns give the occupancies of each bin for each plotted data set. If only one dataset is plotted, there will only be three columns.

Import as Table

Assembles a table as per the **Save** option above, but rather than writing it to disk imports it directly into TOPCAT, where it can be manipulated in all the usual ways.

The Histogram Plot offers the following plot controls:

- Histogram Layer Control, with these form options:
  - Histogram
  - KDE
  - KNN
  - Densogram
  - Gaussian
- Function Layer Control

The following subsections describe navigation, axis configuration and bin configuration.

A.4.8.1 Histogram Navigation

For general comments on plot navigation, see Appendix A.4.2.1. Additional configuration options are available in the **Navigation** tab of the Axes control.

The navigation actions for this window are:

Left drag
Pan. On the body of the plot this moves it around up/down/left/right. To move it only vertically, drag on the left of the Y axis. To move it only horizontally, drag below the X axis. You can configure dragging on the body of the plot to be only vertical or horizontal by setting the **Pan/Zoom Axes** in the **Navigation** axis configuration tab.

Right drag (CTRL-drag)
Stretch zoom. On the body of the plot, dragging up/down stretches/squashes the plot vertically, and dragging left/right stretches/squashes it horizontally. To zoom only vertically, drag on the left of the Y axis. To zoom only horizontally, drag below the X axis.

Zooming horizontally will normally adjust the width of the histogram bars appropriately.

Normally, the zoom will be centered horizontally at the mouse position and vertically on the X axis, so the zoom will not move the bottom of the histogram. You can adjust that with the **Anchor X/Y Axis** options in the **Navigation** axis configuration tab.

Wheel
Isotropic zoom. Spinning the mouse wheel forwards/backwards will zoom in/out just like dragging with the right button up-and-right or down-and-left.

You can also manually fix the plot bounds using the **Range** tab of the Axes control.

A.4.8.2 Histogram Axes Control

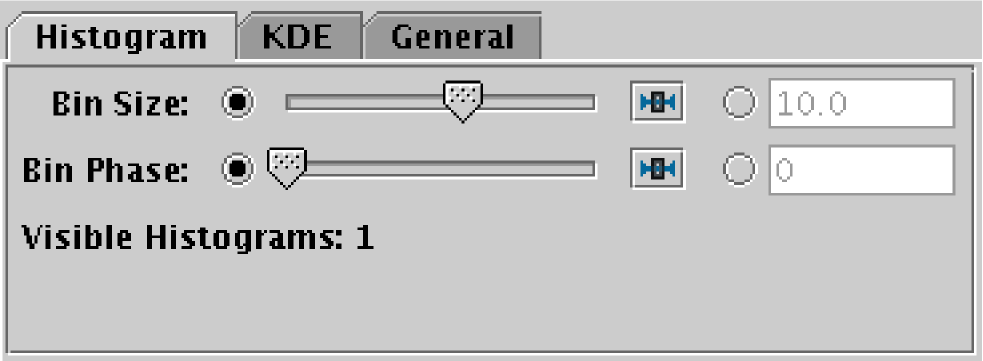
The Histogram Plot axis control is very similar to the Plane Plot axis control described in Appendix A.4.9.2. The only difference is in the **Navigation** tab. For the histogram, the **Anchor X Axis** option is set **on** by default, since you will usually want the Y=0 line to stay anchored to the bottom of the plot when zooming.

A.4.8.3 Bins Control

The **Bins** control ( ) is found in the control stack of the Histogram window. It configures the common placement and calculation of some options for all the histogram-like layers displayed. Being able to set these values in common for all displayed layers of a similar type can be convenient, but if you need to use different parameters for different datasets, you can plot the same layer forms in the Plane window (which has no fixed Bins control) instead.

There are three tabs: **Histogram**, **KDE** and **General**, described below.

Histogram Tab



Bin Size: ● ○ 10.0

Bin Phase: ● ○ 0

Visible Histograms: 1

Histogram tab of histogram window Bins fixed control

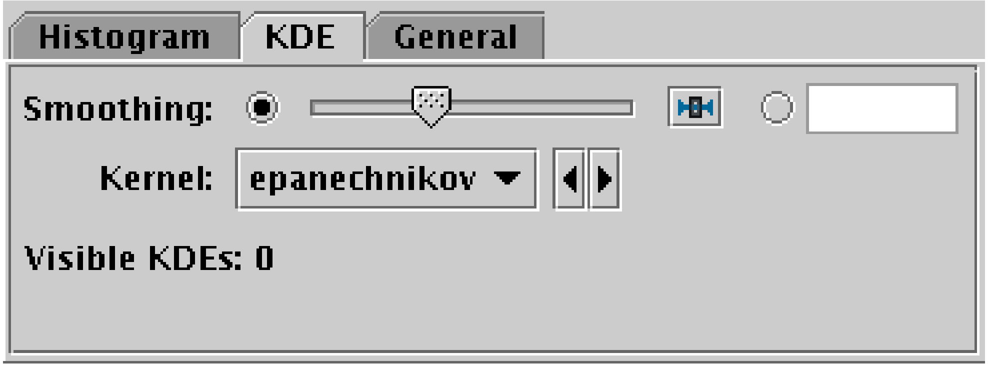
The **Histogram** tab affects all Histogram layers, and to some extent the Gaussian layer, and has the following controls:

Bin Size
: A scale for the width of bins that are shown on the screen. There are two ways to specify this. If the left-hand radio button is selected, the adjacent slider will adjust the bin size, which is also affected by the actual width of the plotting window in pixels. Slide the slider left to get narrower bins or right to get wider ones. If the right-hand radio button is selected, you can enter a numeric value giving the actual width in data units of each bar (for a logarithmic X axis this value is a factor).

  Although Gaussian layers don't have bars, the value of this control can affect the scaling of plotted gaussian fits for some normalisation options, since the Gaussian plots try to scale themselves to match the height of corresponding histograms.

Bin Phase
: Controls where the horizontal zero point for binning is set. For instance if your bin size is 1, it controls whether bin boundaries are at 0, 1, 2, .. or 0.5, 1.5, 2.5, ... etc. If the slider is at either end of the scale, there will be a bin boundary at X=0 (linear X axis) or X=1 (logarithmic X axis).

KDE Tab



Smoothing: ● ○

Kernel: epanechnikov

Visible KDEs: 0

KDE tab of histogram window Bins fixed control

The **KDE** (Kernel Density Estimate) tab affects all KDE, KNN and Densogram layers, and has the following controls:

Smoothing
: Configures the smoothing width for kernel density estimation. This is the characteristic width of the kernel function to be convolved with the density to produce the visible plot.

  Sliding the slider to the right makes the kernel width larger. The width in data units is shown in the text field on the right (if the X axis is logarithmic, this is a factor). Alternatively you can click the radio button near the text field, and enter the width in data units directly.

Note this affects KDE and Densogram layers, but *not* KNN layers, which have their own smoothing controls.

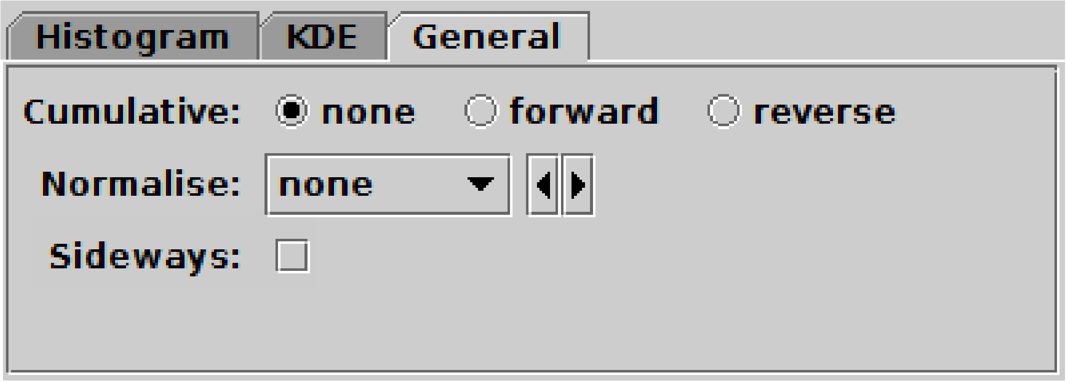
Kernel

The functional form of the smoothing kernel. The functions listed refer to the unscaled shape; all kernels are normalised to give a total area of unity.

The available options are:

- **square**: Uniform value: f(x)=1, |x|=0..1
- **linear**: Triangle: f(x)=1-|x|, |x|=0..1
- **epanechnikov**: Parabola: f(x)=1-x*x, |x|=0..1
- **cos**: Cosine: f(x)=cos(x*pi/2), |x|=0..1
- **cos2**: Cosine squared: f(x)=cos^2(x*pi/2), |x|=0..1
- **gauss3**: Gaussian truncated at 3.0 sigma: f(x)=exp(-x*x/2), |x|=0..3
- **gauss6**: Gaussian truncated at 6.0 sigma: f(x)=exp(-x*x/2), |x|=0..6

General Tab



Cumulative: ● none ○ forward ○ reverse

Normalise: none

Sideways: □

General tab of histogram window Bins fixed control

The **General** tab affects all histogram-like layers, and has the following controls:

Cumulative

If set to **forward** or **reverse**, the bin values are calculated cumulatively; each bin includes the counts from all previous bins in the direction of negative or positive infinity.

Normalise

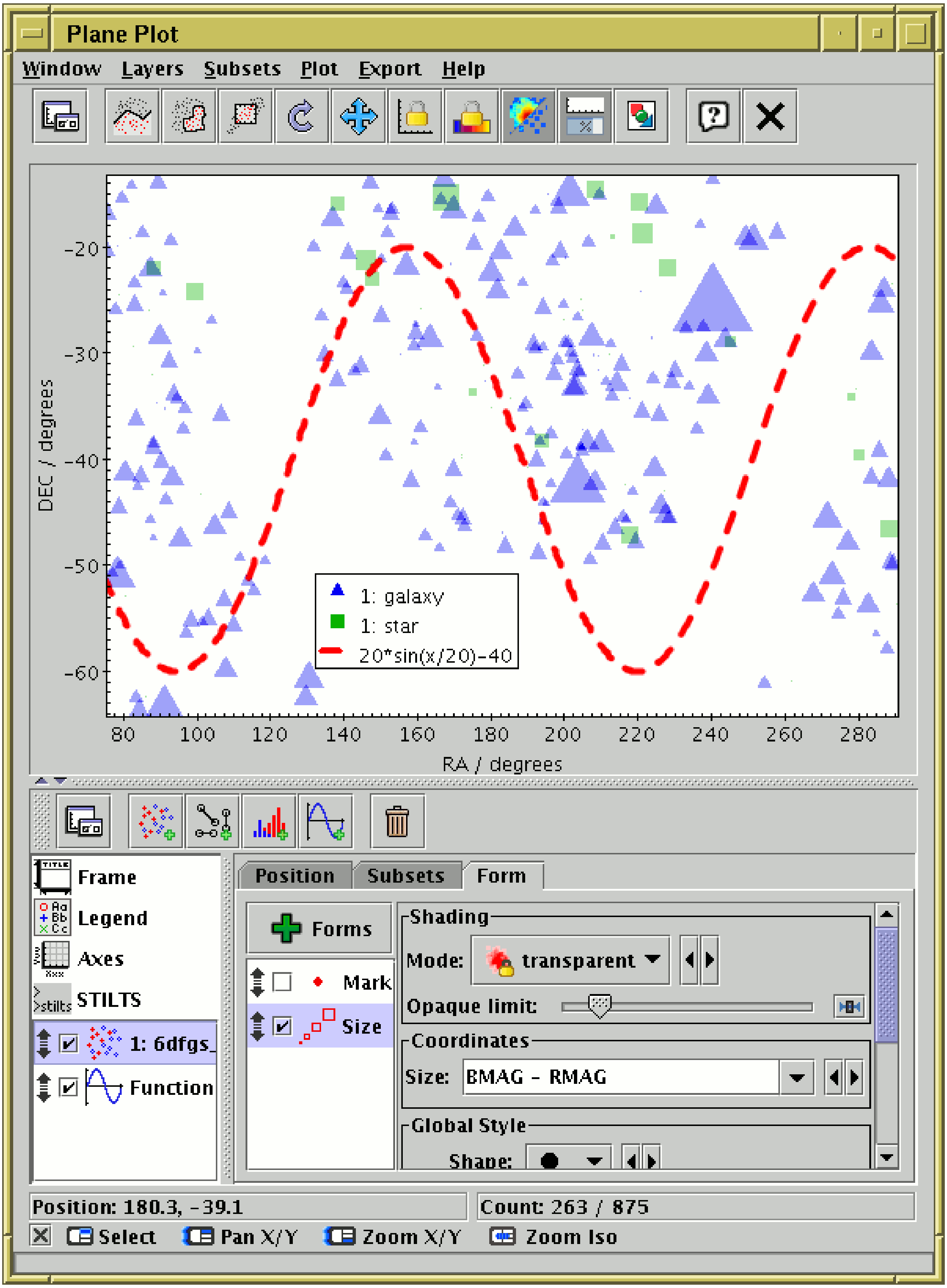
Defines how, if at all, the bars are normalised. The available options are:

- **none**: No normalisation is performed.
- **area**: The total area of histogram bars is normalised to unity. For cumulative plots, this behaves like **height**.
- **unit**: Histogram bars are scaled by the inverse of the bin width in data units. For cumulative plots, this behaves like **none**.
- **maximum**: The height of the tallest histogram bar is normalised to unity. For cumulative plots, this behaves like **height**.
- **height**: The total height of histogram bars is normalised to unity.

Sideways

Controls the orientation of the histogram. By default the quantity being accumulated is on the horizontal axis and the frequencies are represented vertically. But if this option is set, the quantity accumulated is on the vertical axis and the frequencies are represented horizontally, so the chart is displayed reflected in the X=Y line.

A.4.9 Plane Plot Window



☐ Mark

☑ Size

☑ 1: 6dfgs_

☑ Function

Shading

Mode: transparent

Opaque limit:

Coordinates

Size: BMAG - RMAG

Global Style

Shape: ●

Position: 180.3, -39.1

Count: 263 / 875

Plane Plot Window

The **Plane Plot** ( ) plots 2-dimensional Cartesian positions on a plane. The positional coordinates are **X** and **Y**. To control the direction and linear/log scaling of the axes, see the **Coords** tab of the Axes control.

The Plane Plot offers the following plot controls:

- Position Layer Control, with these form options:
  - Mark
  - Size
  - SizeXY
  - Vector
  - Error Bars
  - XYEllipse
  - XYCorr
  - Polygon
  - Line
  - Linear Fit
  - Label
  - Contour
  - Grid
  - Fill
  - Quantile
- Pair Layer Control, with these form options:
  - Link2
  - Mark2
- Area Layer Control, with these form options:
  - Area
  - Central

  - Area Label
- Quad Layer Control, with these form options:
  - Poly4
  - Mark4
- XYArray Layer Control, with these form options:
  - Lines
  - Marks
  - Handles
  - YErrors
  - XYErrors
  - StatLine
  - StatMark
  - ArrayQuantile
- Histogram Layer Control, with these form options:
  - Histogram
  - KDE
  - KNN
  - Densogram
  - Gaussian
- Function Layer Control

As well as the standard actions, this window additionally provides the following toolbar buttons:

- Measure Distance
- Draw Algebraic Subset

and **Subsets** menu item

- Algebraic Subset From Visible

See the Window Overview (Appendix A.4.2) for features common to all plotting windows. The following subsections describe navigation and axis configuration.

A.4.9.1 Plane Navigation

For general comments on plot navigation, see Appendix A.4.2.1. Additional configuration options are available in the **Navigation** tab of the Axes control.

The navigation actions for this window are:

Left drag
Pan. On the body of the plot this moves it around up/down/left/right. To move it only vertically, drag on the left of the Y axis. To move it only horizontally, drag below the X axis. You can configure dragging on the body of the plot to be only vertical or horizontal by setting the **Pan/Zoom Axes** in the **Navigation** axis configuration tab.

Right drag (CTRL-drag)
Stretch zoom. On the body of the plot, dragging up/down stretches/squashes the plot vertically, and dragging left/right stretches/squashes it horizontally. Zooming is around the mouse position at the start of the drag. To zoom only vertically, drag on the left of the Y axis. To zoom only horizontally, drag below the X axis.

Center drag (SHIFT-drag)
Frame zoom. On the body of the plot, dragging right-and-down or right-and-up drags out a frame; when the button is released, the plot will be zoomed in to cover the area enclosed by the frame. Dragging left (and up or down) does something like the opposite, you can zoom out using a similar (though not quite the same) mechanism. To zoom in/out only horizontally, drag right/left below the X axis. To zoom in/out only vertically, drag up/down on the left of the Y axis.

Wheel
Isotropic zoom. Spinning the mouse wheel forwards/backwards will zoom in/out around the mouse position, just like dragging with the right button up-and-right or down-and-left.

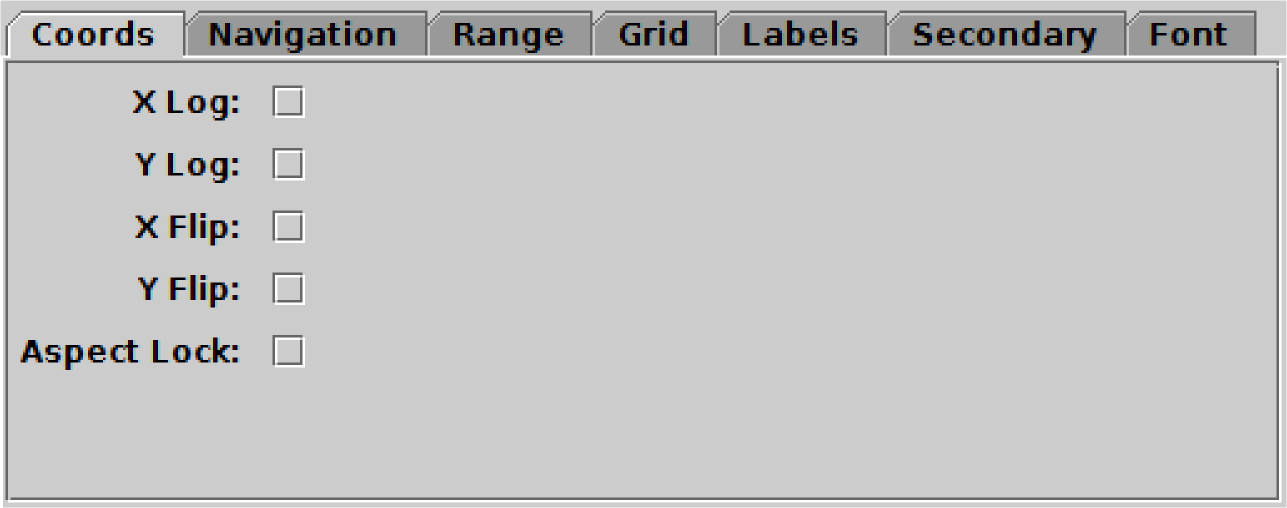
Left click
Select. If there is a plotted point near the cursor, it will plot a marker on it and activate (Section 8) it.

You can also manually fix the plot bounds using the **Range** tab of the Axes control.

A.4.9.2 Plane Axes Control

The **Axes** control () for the plane plot window has the following tabs:

Coords Tab



X Log: □

Y Log: □

X Flip: □

Y Flip: □

Aspect Lock: □

Coords tab of plane Axes control

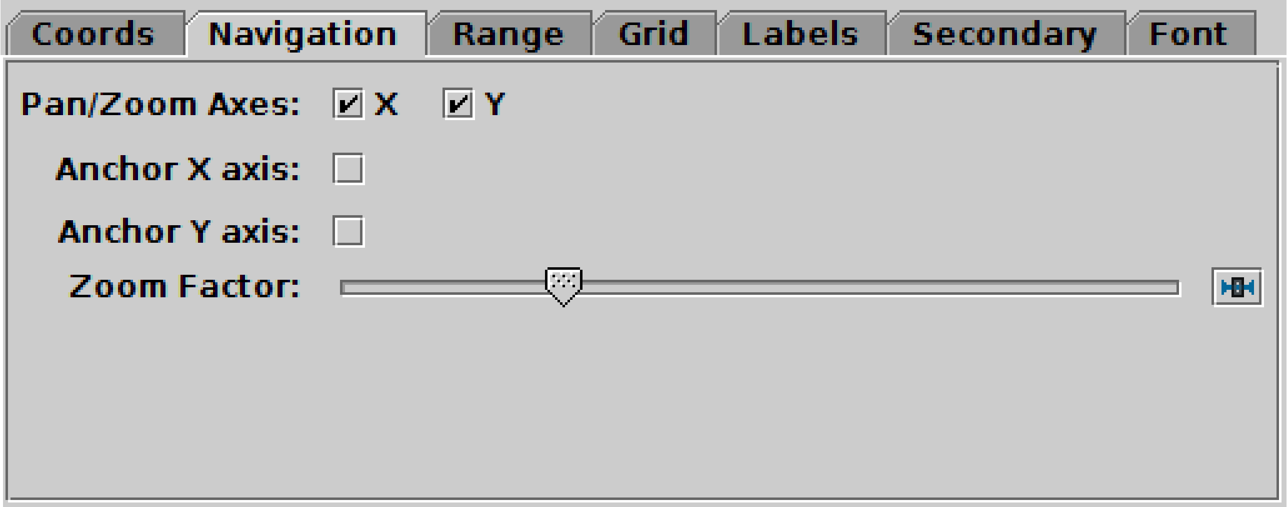
The **Coords** tab controls the axis coordinates. It has the following options:

X/Y Log
If selected, horizontal/vertical axis coordinates are logarithmic, otherwise they are linear.

X/Y Flip
If selected, horizontal/vertical axis coordinate axes run in the opposite direction to normal.

Aspect Lock
If selected, the number of pixels per unit is always the same on both axes, i.e. the unit square is always a square. Otherwise, there is no constraint on the relative sizes of the X and Y axis units.

Navigation Tab



Pan/Zoom Axes: ☑ X ☑ Y

Anchor X axis: □

Anchor Y axis: □

Zoom Factor:

Navigation tab of plane Axes control

The **Navigation** tab controls details of how the navigation works. It has the following options:

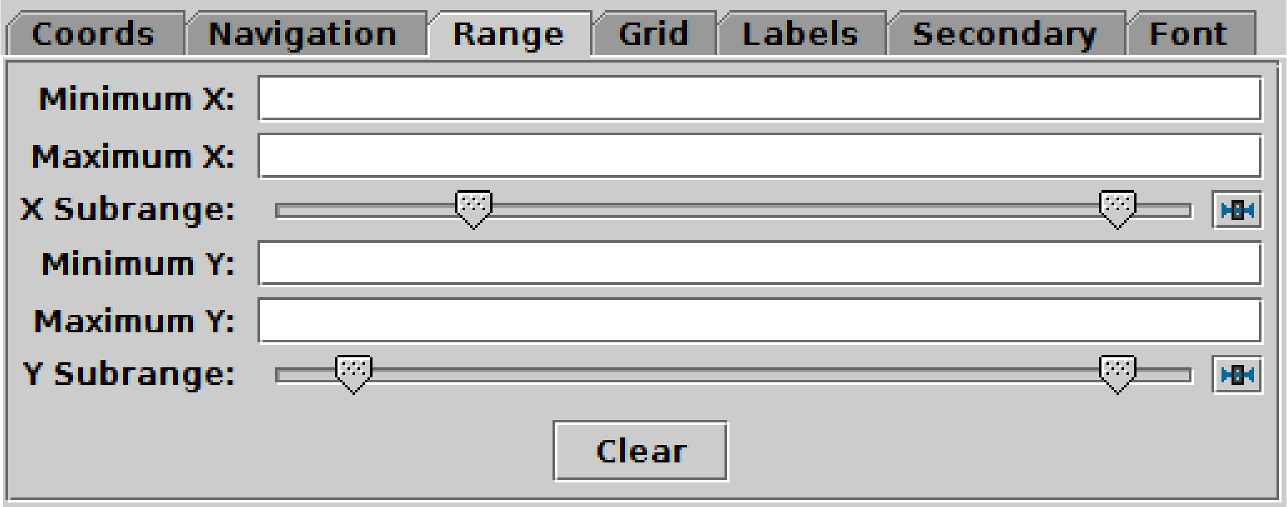
Pan/Zoom Axes
Normally, dragging with the left/right mouse buttons or using the mouse wheel on the main part of the plot will pan/zoom it in both X and Y directions. By unchecking the X or Y checkbox here, you can prevent pan/zoom in the corresponding direction, so if the Y box is unchecked, pan/zoom will only affect the vertical direction (note the same effect can be

achieved by dragging to the left of the Y axis).

Anchor X/Y axis
Normally, zoom operations zoom around the position of the mouse at the start of the wheel/drag gesture. Checking these boxes fixes the X/Y reference coordinate for zooms to be the Y=0 or X=0 lines. This can be useful if you want the X or Y axis to stay put (e.g. at the edge of the plot) during zoom actions.

Zoom Factor
Controls the factor by which each zoom action zooms the plot. Moving this slider to the left/right makes the mouse more/less sensitive (one wheel click or dragging a fixed distance has more/less zoom effect).

Range Tab



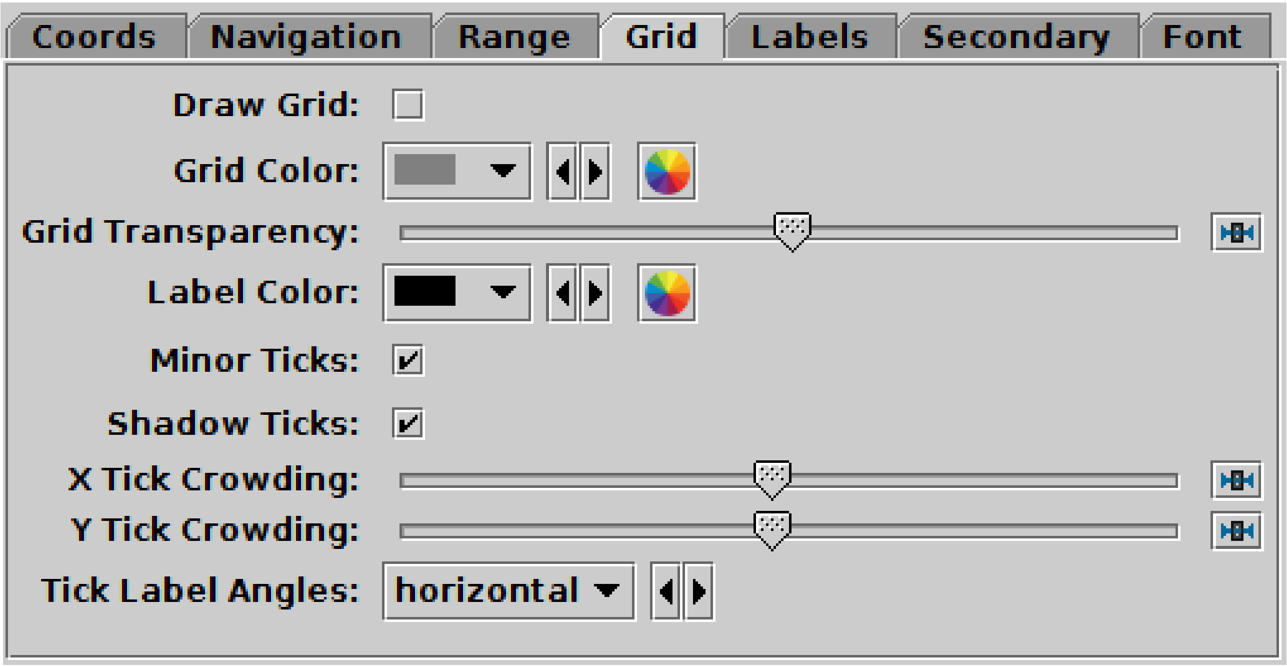
Range tab of plane Axes control

The **Range** tab provides manual configuration of the visible range of the plot. Making changes to this tab will reset the visible plot range, but not vice versa - zooming and panning in the usual way will not change the settings of this panel.

Filling in the **Minimum**/**Maximum** fields for either or both axes will constrain the corresponding range of the visible data. The limits corresponding to any of those fields that are left blank will initially be worked out from the data. The **Subrange** double-sliders restrict the ranges within the (explicit or automatic) min/max ranges. Note you can move both sliders at once by grabbing a position between the two.

The **Clear** button resets all the fields.

Grid Tab



Draw Grid: □

Grid Color:

Grid Transparency:

Label Color:

Minor Ticks: ☑

Shadow Ticks: ☑

X Tick Crowding:

Y Tick Crowding:

Tick Label Angles: horizontal

Grid tab of plane Axes control

The **Grid** tab configures the appearance of the axis grid. It has the following options:

Draw Grid
If true, grid lines will be drawn across the plot for every tick mark.

Grid Colour
Selects the colour with which grid lines will be drawn.

Grid Transparency
Controls the transparency of the grid lines, which are drawn over the plot content.

Label Colour
Selects the colour in which axis label text will be written.

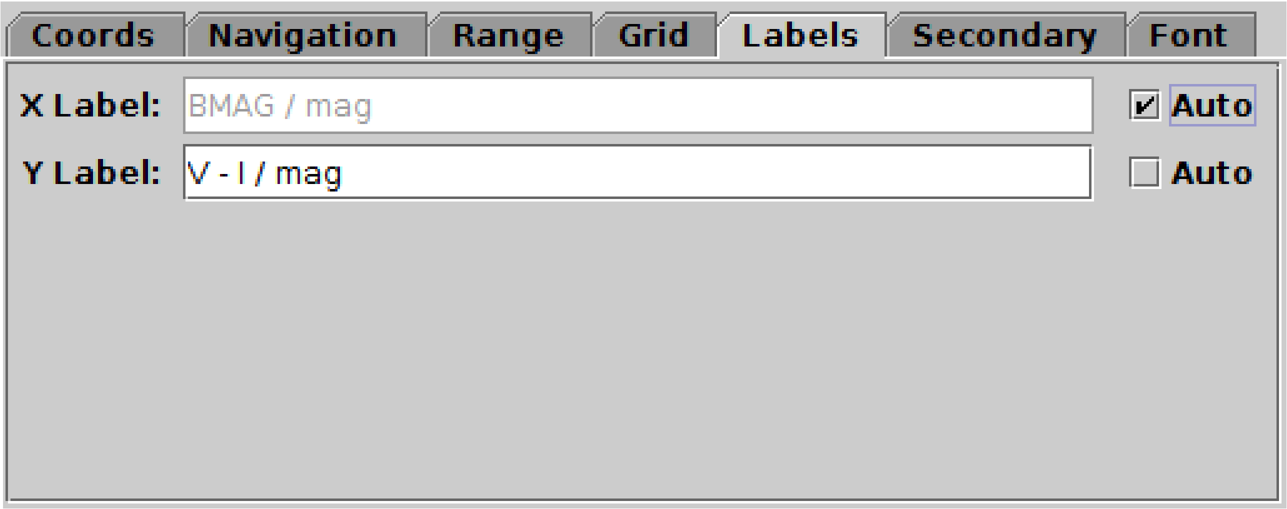
Minor Ticks
If set, minor (unlabelled) tick marks will be drawn between the major (labelled) ones.

Shadow Ticks
If set and no secondary axis is in use, then tick marks without numeric labels are painted along the axis opposite to the primary axis, so that tick marks are visible along all edges not just the ones with numeric labels. If a secondary axis is in use, this setting is ignored.

X/Y Tick Crowding
Use the slider to influence how many tick marks are draw on each axis.

Tick Label Angles
Controls orientation of numeric labels on the axes. By default they are drawn with **horizontal** alignment, but you can also choose **angled**. If **adaptive** is selected, they will be horizontal where possible, but may be angled to accommodate more labels if crowding is high; note this option is currently not perfect and can result in suboptimal border placement.

Labels Tab



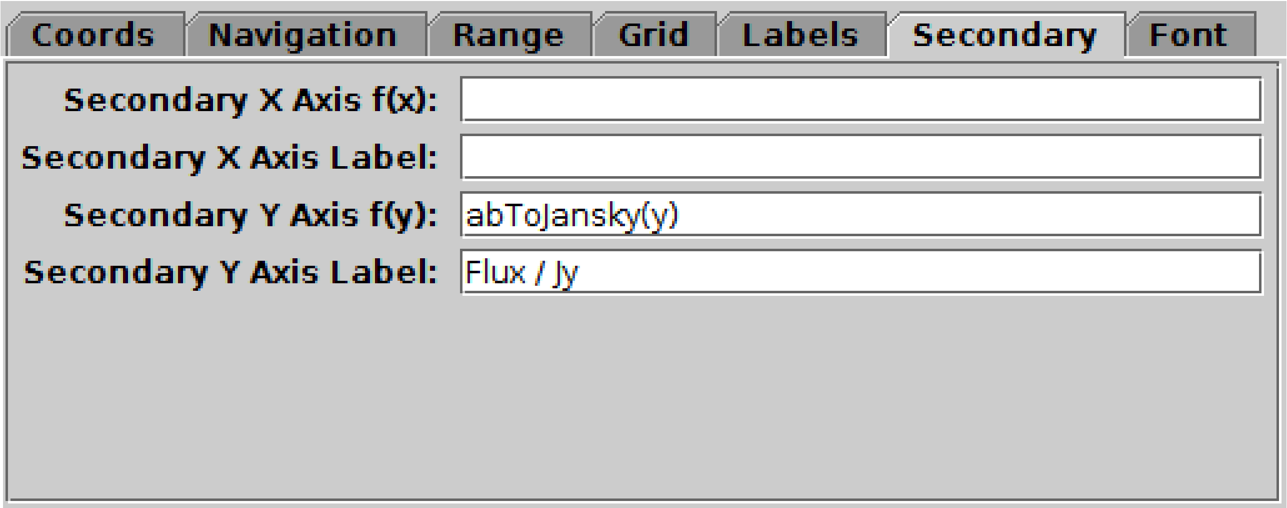
X Label: BMAG / mag ☑ Auto

Y Label: V - I / mag □ Auto

Labels tab of plane Axes control

The **Labels** tab controls the text labels on the axes. If the **Auto** checkbox is set, the text will be taken from one of the data coordinates being plotted on that axis. To override those with your own axis labels, unset Auto and type text in to the **Label** fields.

Secondary Axes Tab



Secondary X Axis f(x):

Secondary X Axis Label:

Secondary Y Axis f(y): abToJansky(y)

Secondary Y Axis Label: Flux / Jy

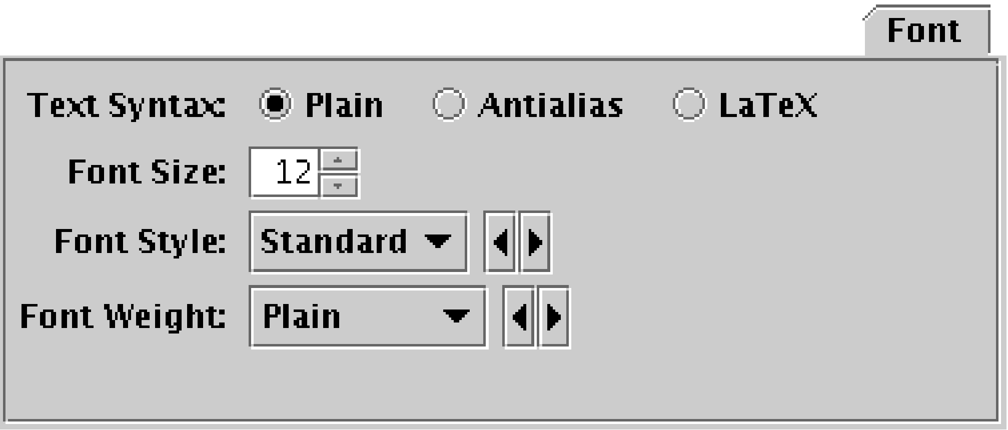
Secondary tab of plane Axes control

The **Secondary** tab controls optional secondary X and Y axes at the top and right edges of the plot, to go with the standard (primary) X and Y axes at the bottom and left edges, so you can annotate a plot for instance with both magnitudes and fluxes, or both frequency and wavelength.

Secondary X/Y Axis f(x/y)
: Defines the secondary axis in relation to the primary one by means of a supplied function that maps primary to secondary axis values, written using TOPCAT's expression language. For the Secondary X axis this is given as a function of the dummy variable **x**, and for the Secondary Y axis as a function of the dummy variable **y**. The function supplied should be monotonic and reasonably well-behaved, otherwise the secondary axis annotation may not work well. TOPCAT will attempt to make a sensible decision about whether to use linear or logarithmic tick marks.

Secondary X/Y Axis Label
: Provides a textual annotation near the secondary axis. This can be supplied whether or not the axis mapping functions are actually present.

Font Tab



Text Syntax: ● Plain ○ Antialias ○ LaTeX

Font Size: 12

Font Style: Standard

Font Weight: Plain

Font tab

The **Font** tab configures the font used for axis annotation. It also affects some other things like the legend.

Text Syntax
How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. At time of writing, on MacOS antialiased text seems to be required to stop the writing coming out upside-down for non-horizontal text. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size
Size of the font in points.

Font Style
Style of the font.

Font Weight
Whether the font is plain, bold or italic.

A.4.10 Sky Plot Window

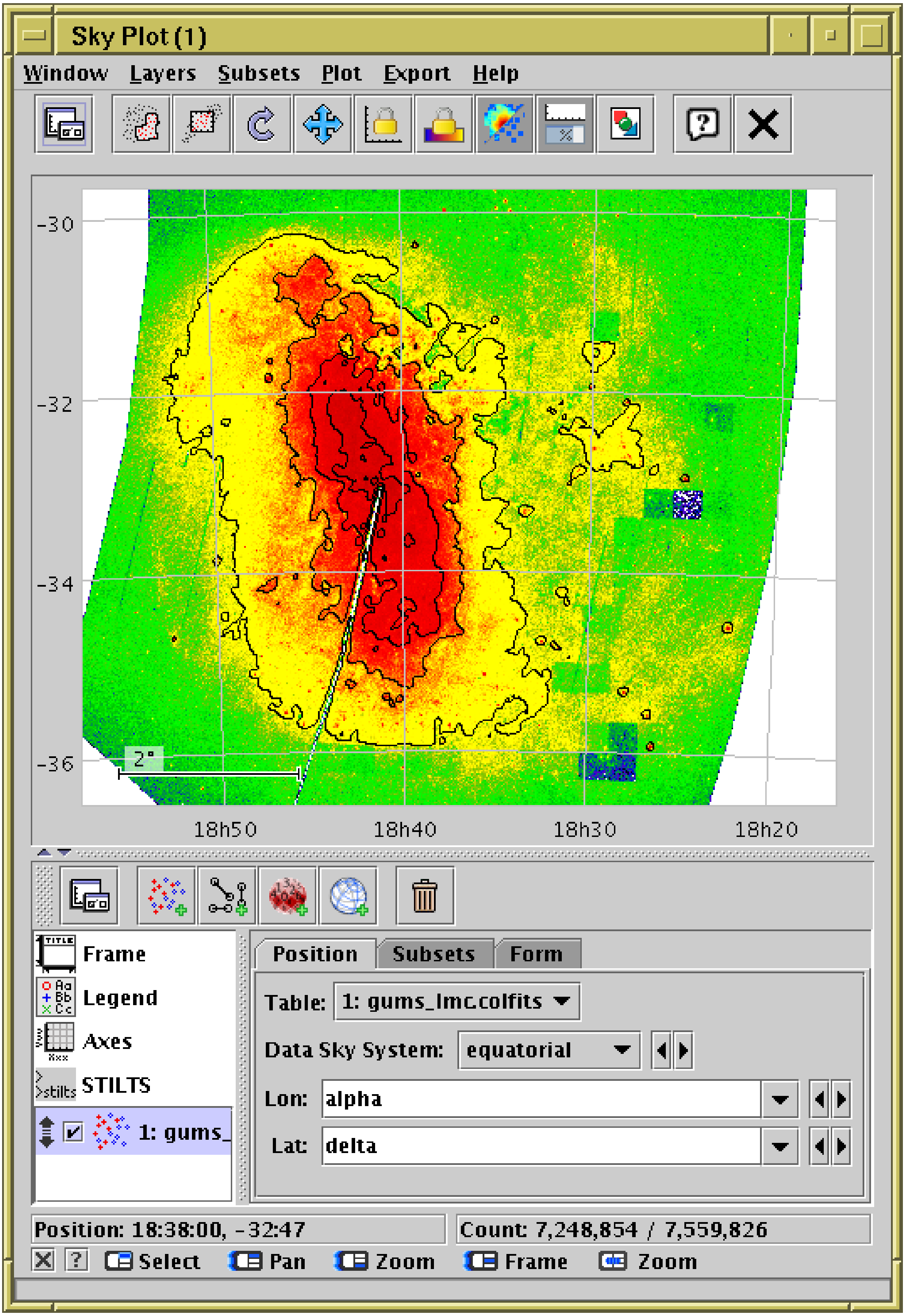


Table: 1: gums_lmc.colfits

Data Sky System: equatorial

Lon: alpha

Lat: delta

Position: 18:38:00, -32:47

Count: 7,248,854 / 7,559,826

Sky Plot Window

The **Sky Plot** () plots longitude/latitude positions onto the celestial sphere. It can plot to a number of projections (currently Sin, Hammer-Aitoff and Plate Carrée).

The positional coordinates are **Longitude** and **Latitude**, specified in degrees. When supplying them, you can specify an associated **Data Sky System** (Equatorial, Galactic, Supergalactic or Ecliptic2000). Note that this is the sky system of the data coordinates, not (necessarily) of the plot you want to see. To specify the coordinates you want the data to be plotted on, use the **View Sky System** option in the **Projection** tab of the Axes control. However, if you just want to view the data using the same system as the coordinates you are supplying, you can ignore leave these values as their default (both Equatorial) and no conversion will be done.

The sky plot offers the following plot controls:

- Position Layer Control, with these form options:
  - Mark
  - Size
  - SizeXY
  - SkyVector
  - SkyEllipse
  - SkyCorr
  - Polygon
  - Label
  - Contour
  - SkyDensity
- Pair Layer Control, with the following plot controls:
  - Link2
  - Mark2
- Area Layer Control, with the following plot controls:
  - Area
  - Central
  - Area Label
- Quad Layer Control, with the following plot controls:

  - Poly4
  - Mark4
- Healpix Layer Control
- SkyGrid Layer Control

As well as the standard actions, this window additionally provides the following toolbar buttons:

- Measure Distance
- Draw Algebraic Subset

See the Window Overview (Appendix A.4.2) for features common to all plotting windows. The following subsections describe navigation and axis configuration.

A.4.10.1 Sky Navigation

For general comments on plot navigation, see Appendix A.4.2.1. Additional configuration options are available in the **Navigation** tab of the Axes control.

The navigation options for this window are:

Left drag

Pan. In the Sin projection, this attempts to rotate the sphere, with north staying vertical on the screen, in such a way that the same part of the sky stays under the cursor position. At low magnifications this is sometimes not possible; if the mouse starts on the sphere and ends up off it the navigation will be jerky.

The Hammer-Aitoff and Plate Carrée projections act like a piece of paper that can be dragged around in two dimensions in the usual way.

Wheel

Zoom. Spinning the mouse wheel forwards/backwards will zoom in/out around the mouse position. For this Sin projection this will also have the effect of rotating the sphere to keep North vertical on the screen.

Right drag (CTRL-drag)

Stretch zoom. Dragging with the right mouse button up-and-right/down-and-left has just the same effect as spinning the mouse button forwards/backwards.

Center drag (SHIFT-drag)

Frame zoom. Dragging with the middle button to the right (and either up or down) drags out a frame; when the button is released, the plot will be zoomed in to cover (roughly) the area enclosed by the frame. Dragging left (and either up or down) does something like the opposite, you can zoom out using a similar (though not quite the same) mechanism.

Left click

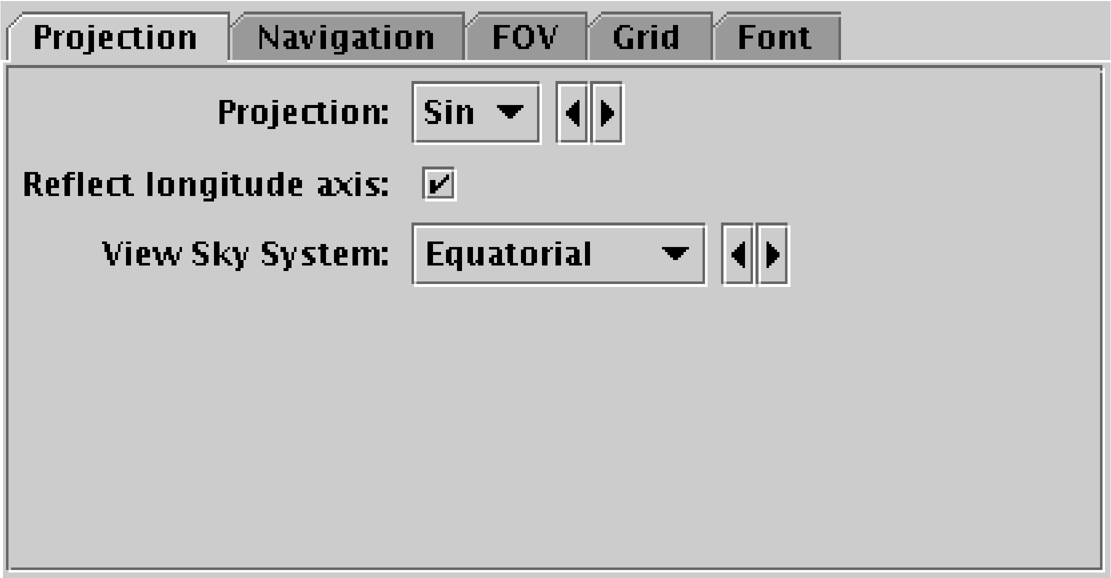
Select. If there is a plotted point near the cursor, it will plot a marker on it and activate (Section 8) it.

You can also navigate to a known sky position by coordinates or object name using the **FOV** tab of the Axes control.

A.4.10.2 Sky Axes Control

The **Axes** control () for the sky plot window has the following tabs:

Projection Tab



Projection | Navigation | FOV | Grid | Font

Projection: Sin

Reflect longitude axis: ☑

View Sky System: Equatorial

Projection tab of the sky Axes control

The **Projection** tab controls how the sky position coordinates are projected onto the screen.

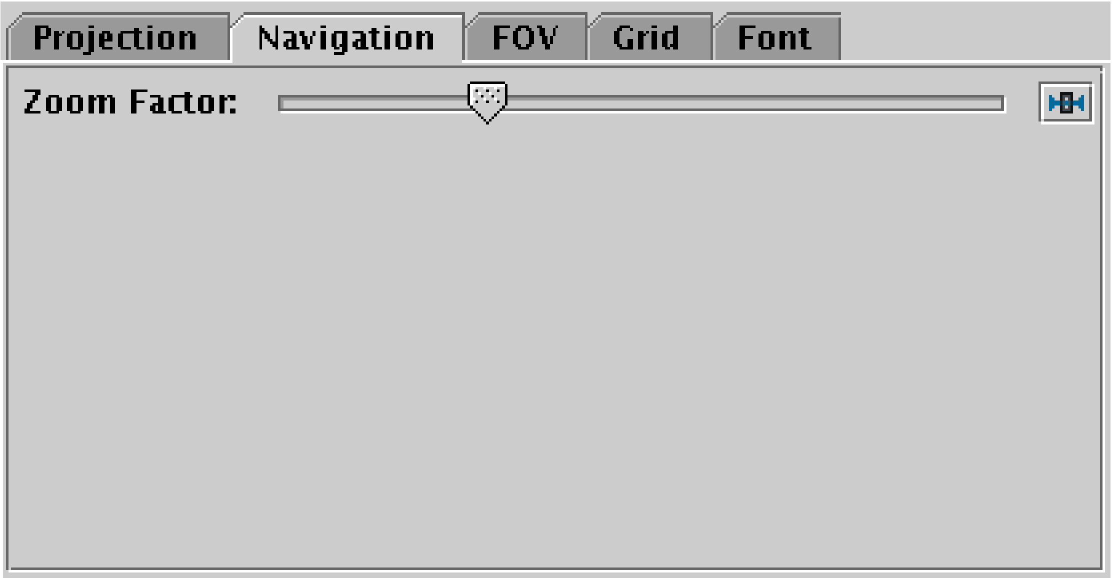
Projection
Selects the sky projection to use. The options are **Sin**; **Aitoff** or **Aitoff0** for Hammer-Aitoff; and **Car** or **Car0** for Plate Carrée. Sin is rotatable, the other two are essentially flat all-sky projections. Car and Aitoff have longitude=0 at the center of the plot, while Car0 and Aitoff0 have it at the left/right edge. Note that for historical reasons the projections named Aitoff denote the equal-area Hammer-Aitoff projection and not the Aitoff projection itself. The Hammer-Aitoff projection is defined as:

```
      x = [2sqrt(2)/sqrt(1+cos(lat)cos(lon/2))]cos(lat)sin(lon/2)
      y = [sqrt(2)/sqrt(1+cos(lat)cos(lon/2))]sin(lat)
```

Reflect longitude axis
Determines whether longitude increases left to right or right to left.

View Sky System
Determines the sky coordinate system in which the data positions will be viewed. This interacts with the **Data Sky System** selected in the **Positions** tab of the table layer control for data coordinates; supplied data points are projected from the data system to the view system before being plotted. If you have (for instance) data in equatorial coordinates that you want to view in galactic coordinates, then select the Data Sky System as Equatorial and the View Sky System as Galactic. If the data and view systems are the same, it's OK to leave both as their defaults, even if they're not equatorial - the only effect that the chosen Data and View sky systems has is from the transformation between them.

Navigation Tab



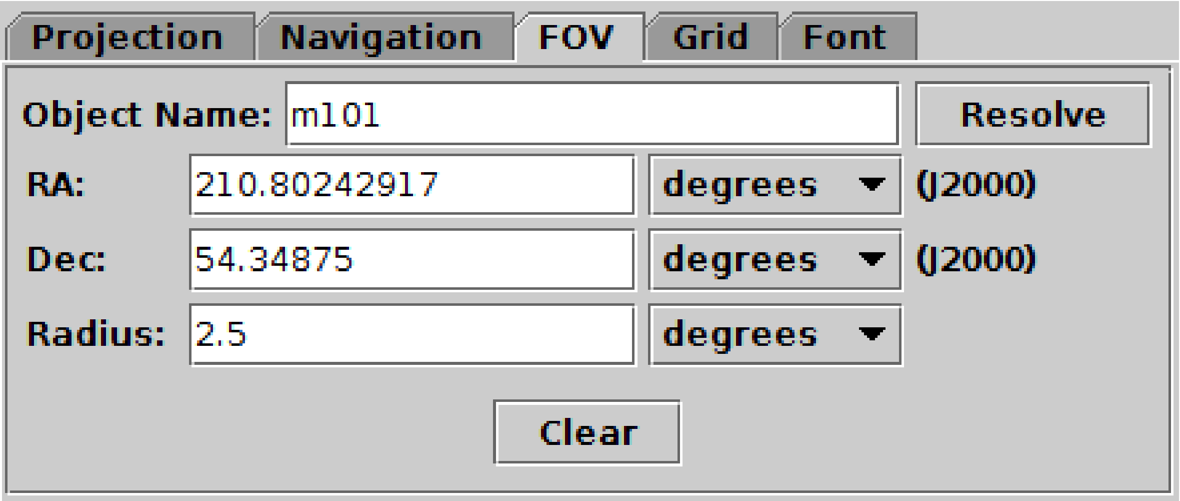
Zoom Factor:

Navigation tab of sky Axes control

The **Navigation** tab controls details of how the navigation works. It has the following option:

Zoom Factor
Controls the factor by which each zoom action zooms the plot. Moving this slider to the left/right makes the mouse more/less sensitive (one wheel click or dragging a fixed distance has more/less zoom effect).

Field of View Tab



Object Name: m101

RA: 210.80242917 degrees (J2000)

Dec: 54.34875 degrees (J2000)

Radius: 2.5 degrees

Field Of View tab of the sky Axes control

The **FOV** (Field Of View) tab allows you to enter a sky position or object name and a radius and positions the view at that region. Filling in the position and radius fields and hitting return will reposition the sky view, but not vice versa; normal pan/zoom operations will not affect the content of this panel.

Object Name
If you fill this in with the name of a celestial object and hit the **Resolve** button, a Simbad query will execute to fill in the **RA** and **Dec** fields with its position.

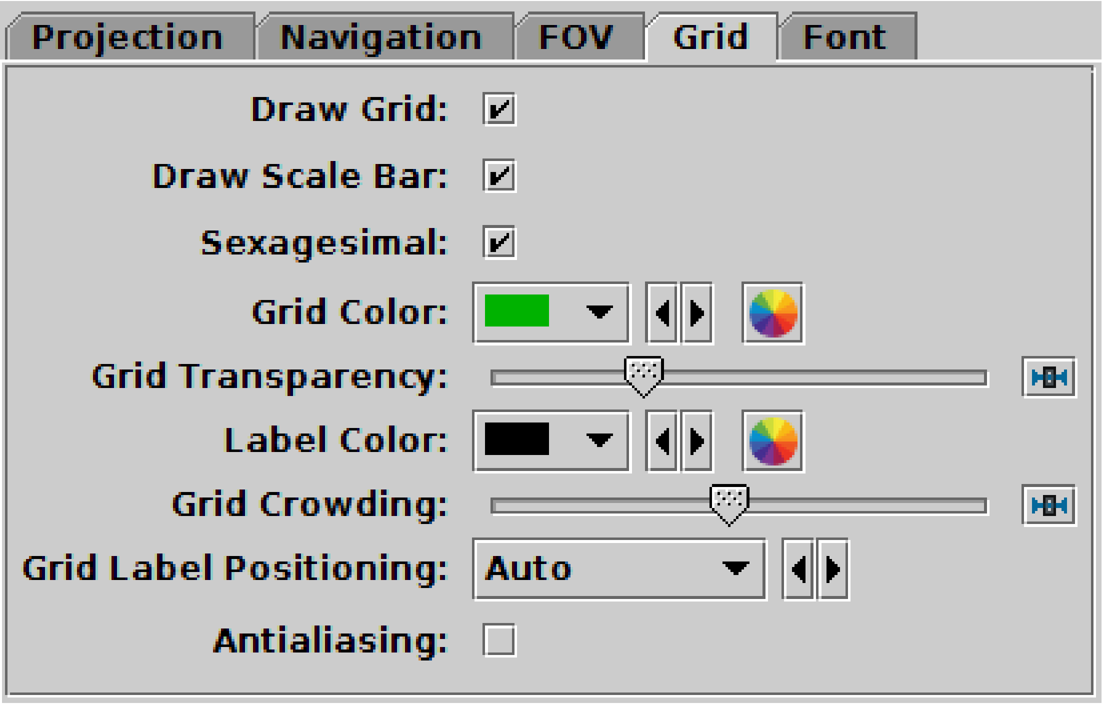
RA/Dec
J2000 positions of the required field centre. These values can either be filled in by the object

name resolution as described above, or by hand.

Radius
Gives the radius of the desired field of view.

Clear
Clears the fields in this tab.

Grid Tab



Draw Grid: ☑

Draw Scale Bar: ☑

Sexagesimal: ☑

Grid Color: 🟩

Grid Transparency:

Label Color: ⬛

Grid Crowding:

Grid Label Positioning: Auto

Antialiasing: □

Grid tab of the sky Axes control

The **Grid** tab controls how the sky coordinate axes appear.

Draw Grid
If selected, grid lines will be drawn on the plot.

Draw Scale Bar
If selected, a small annotated scale bar will usually be drawn at the bottom left of the plot indicating the scale of the image in degrees, minutes or seconds.

Sexagesimal
If selected sky coordinate annotations of the grid will be in sexagesimal format, otherwise in decimal degrees.

Grid Colour
Selects the colour with which grid lines will be drawn.

Grid Transparency
Controls the transparency of the grid lines, which are drawn over the plot content.

Label Colour
Selects the colour in which axis label text will be written.

Grid Crowding
Use the slider to control how closely packed grid lines are on the axes. If you want to control the crowding separately on the two axes, you can use the SkyGrid layer control instead.

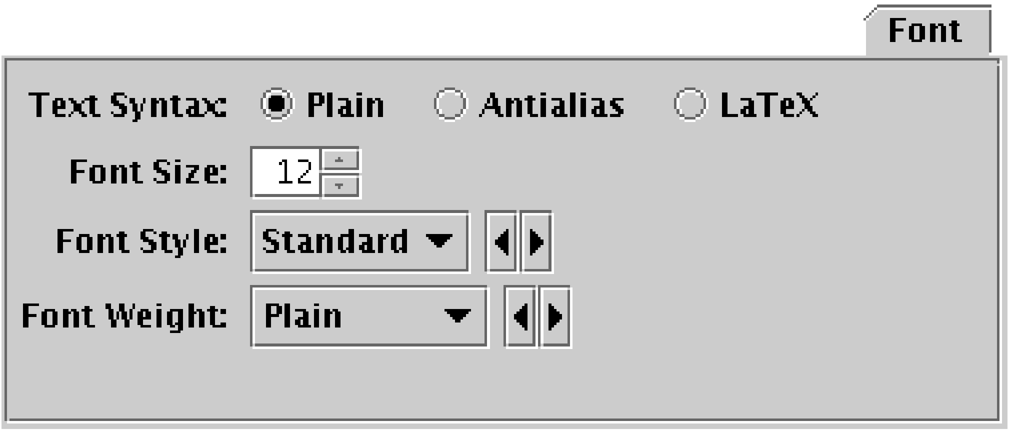
Label Positioning

Controls whether and where the numeric annotations of the lon/lat axes are put. The default option **Auto** usually does the sensible thing, but other options exist to force labelling internally or externally to the plot region, to further annotate the axes with **View Sky System** coordinate names, or to remove numeric labels altogether.

Antialiasing

Controls whether grid lines will be drawn antialiased (smoothed) or not. This option does not affect exported plots.

Font Tab



Text Syntax: ● Plain ○ Antialias ○ LaTeX

Font Size: 12

Font Style: Standard

Font Weight: Plain

Font tab

The **Font** tab configures the font used for axis annotation. It also affects some other things like the legend.

Text Syntax

How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. At time of writing, on MacOS antialiased text seems to be required to stop the writing coming out upside-down for non-horizontal text. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size

Size of the font in points.

Font Style

Style of the font.

Font Weight

Whether the font is plain, bold or italic.

A.4.11 Cube Plot Window

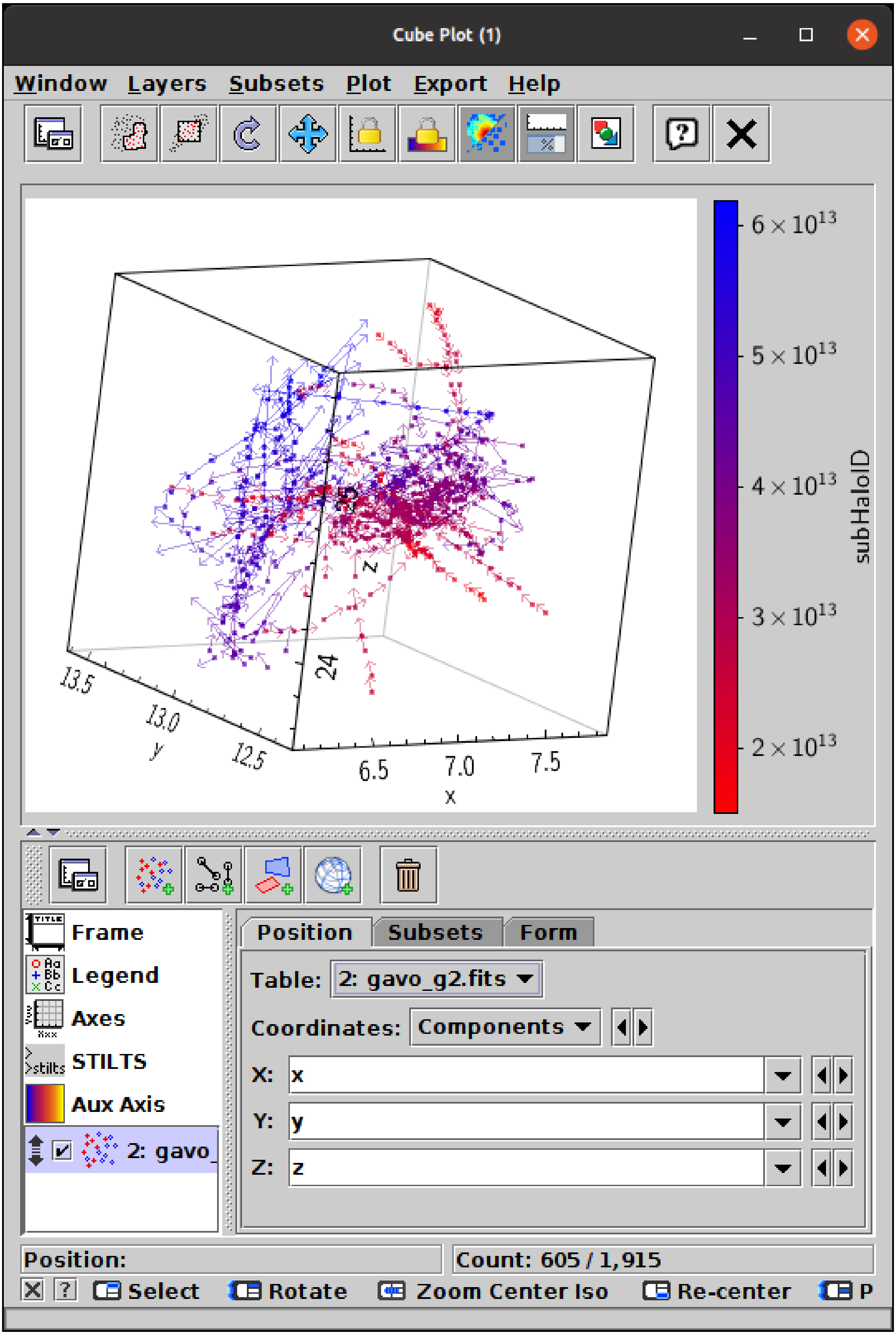


Table: 2: gavo_g2.fits

Coordinates: Components

X: x

Y: y

Z: z

Position:

Count: 605 / 1,915

Cube Plot Window

The **Cube Plot** () plots 3-dimensional Cartesian positions in a 3-d space.

By default the positional coordinates are **X**, **Y** and **Z**, but the **Coordinates** selector lets you specify them in different ways:

Components:
Coordinates **X**, **Y** and **Z**, giving the Cartesian components of the position.

Vector:
A single coordinate **XYZ** which must be a 3-element array giving the three Cartesian components of the position. If the array has more than three elements, later ones are ignored.

Polar:
Coordinates **Lon**, **Lat** and **Radius**, giving the spherical polar coordinates of the position.

To control the direction and linear/logarithmic scaling of the axes, see the **Coords** tab of the Axes control.

The cube plot offers the following plot controls:

- Position Layer Control, with these form options:
  - Mark
  - Size
  - SizeXY
  - Vector
  - Error Bars
  - Polygon
  - Label
  - Line3d
  - Contour
- Pair Layer Control, with the following plot controls:
  - Link2
  - Mark2
- Quad Layer Control, with the following plot controls:
  - Poly4
  -

Mark4

- SphereGrid Layer Control

As well as the standard actions, this window additionally provides the following toolbar button:

- Measure Distance

and **Subsets** menu item:

- Algebraic Subset From Visible

Note that use of the Auto, Density and Weighted shading modes can be confusing in 3 dimensions with multiple datasets. This is because pixels based on density along a line of sight are not located at any point on that line, so shaded pixels can't appear at the "right" place in the 3-d space. The same applies to a lesser extent with contours. They work fine with a single dataset though.

See the Window Overview (Appendix A.4.2) for features common to all plotting windows. The following subsections describe navigation and axis configuration.

A.4.11.1 Cube Navigation

For general comments on plot navigation, see Appendix A.4.2.1. Additional configuration options are available in the **Navigation** tab of the Axes control.

Navigation in three dimensions with a two-dimensional screen and mouse is a bit of a challenge. However, the actions listed below should make it fairly straightforward to navigate around points in 3d space to zoom in on the regions that you are interested in.

Left drag
Rotate. Rotates the view cube about its center.

Wheel
Isotropic zoom. Spinning the mouse wheel forwards/backwards zooms the space in the view cube in/out around the cube center. The mouse position does not affect it.

Right drag (CTRL-drag)
2d stretch zoom. Dragging with the right button does a stretch zoom in the two dimensions best aligned with the plane of the screen. There is no movement in the third (mostly perpendicular to the screen) direction. The zoom is around a line defined by the position of the mouse at the start of the zoom, pointing along the third dimension. You can stretch/squash in both directions by dragging the mouse up/down/left/right - it's easier to try it than to explain it.

Center drag (SHIFT-drag)
2d pan. Dragging with the center button pans the 3d space in the two dimensions best alighned with the plane of the screen. There is no movement in the third (mostly perpendicular to the screen) direction.

Left click
Select. If there are plotted points near the cursor it will identify one, plot a marker on it, and activate (Section 8) it.

In 3d, it's not obvious which is the nearest point to a 2d cursor, since a screen position represents a line not a point. To break the degeneracy, the point used is the one nearest the center of mass of plotted points along the line of sight represented by the cursor position. The upshot of this is that if you click on an isolated point, you'll pick that point, and if you click on a dense cluster, you'll get a point near the center (of mass) of that cluster.

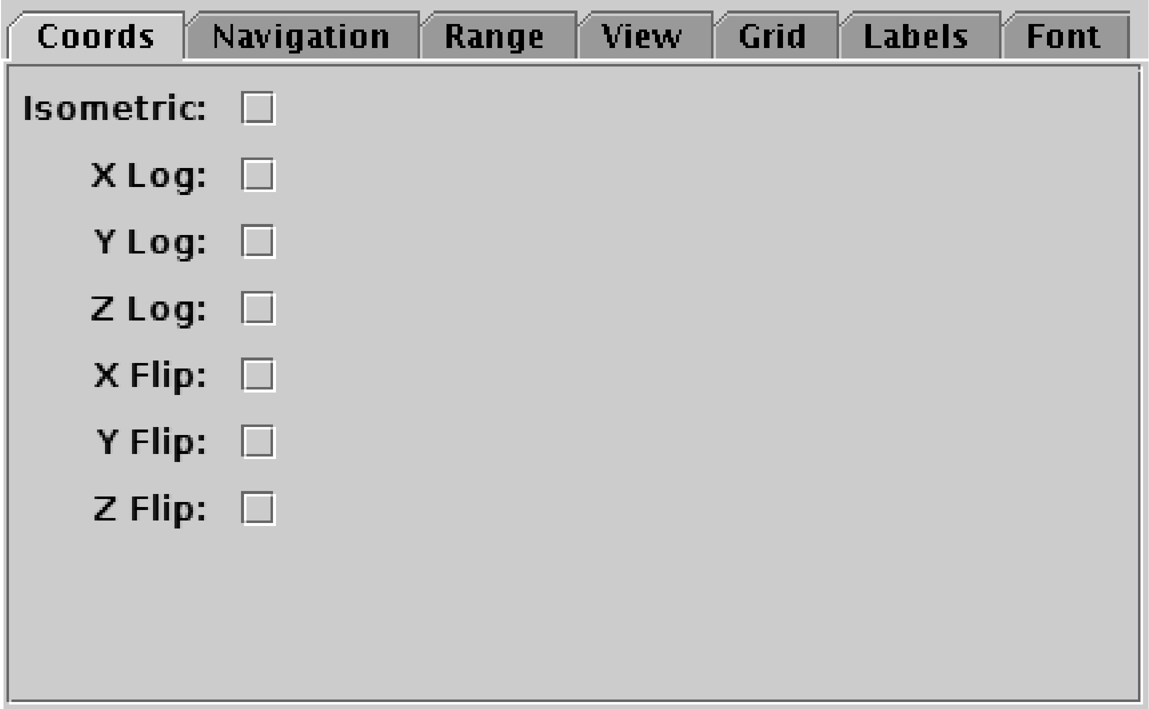
Right click (CTRL-click)
Re-center. Right-clicking identifies a position in a similar way to the left-click Select action, and then translates the plot so that point is at the center of the view cube. This means that clicking on a cluster will put that cluster in the center of the cube. If you click on an empty region, a position half way between front and back of the cube at that X/Y position will be used.

You can also manually fix the plot bounds using the **Range** tab of the Axes control.

A.4.11.2 Cube Axes Control

The **Axes** control ( ) for the cube plot window has the following tabs:

Coords Tab



Isometric: □
X Log: □
Y Log: □
Z Log: □
X Flip: □
Y Flip: □
Z Flip: □

Coords tab of the cube Axes control

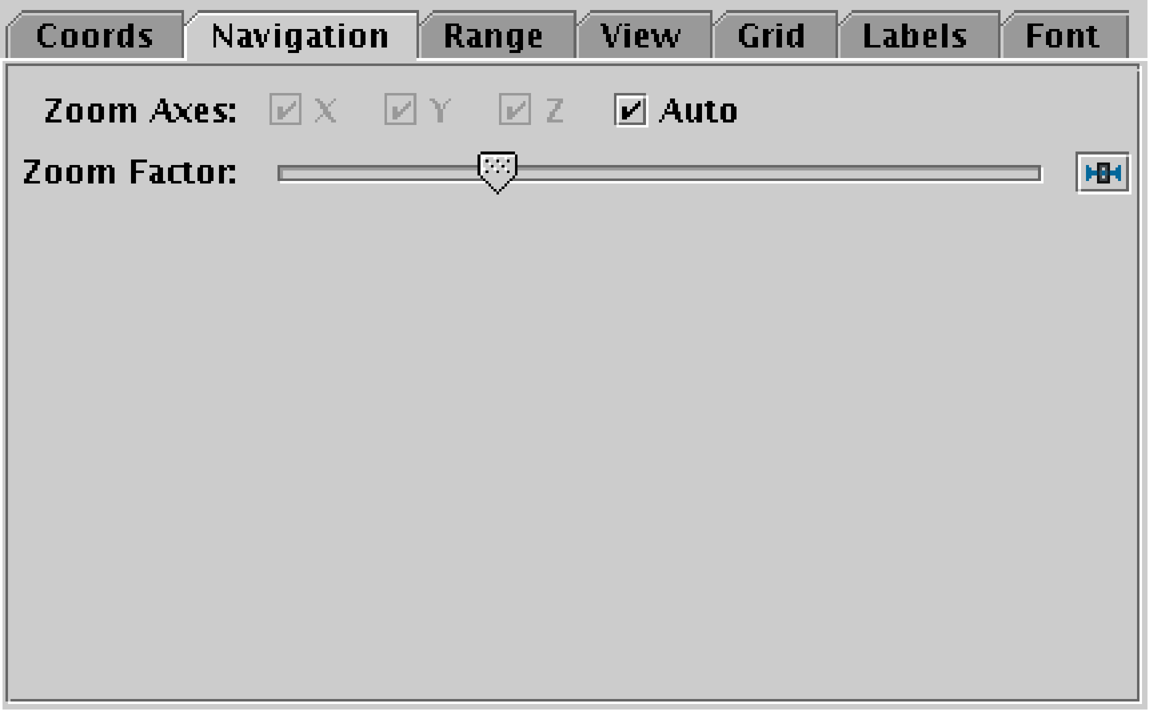
The **Coords** tab controls the axis coordinates. It has the following options:

Isometric
If selected, the scaling will be the same on the X, Y and Z axes, so that positions will retain their natural position in 3-d Cartesian space. When unselected, the three axes will be scaled independently, so that the positions may be squashed in some directions. This option is ignored if there is a mix of linear and logarithmic axes.

X/Y/Z Log
If selected, X/Y/Z axis coordinates are logarithmic, otherwise they are linear.

X/Y/Z Flip
If selected, X/Y/Z axis coordinate axes run in the opposite direction to normal.

Navigation Tab



Coords | Navigation | Range | View | Grid | Labels | Font

Zoom Axes: ☑ X ☑ Y ☑ Z ☑ Auto

Zoom Factor:

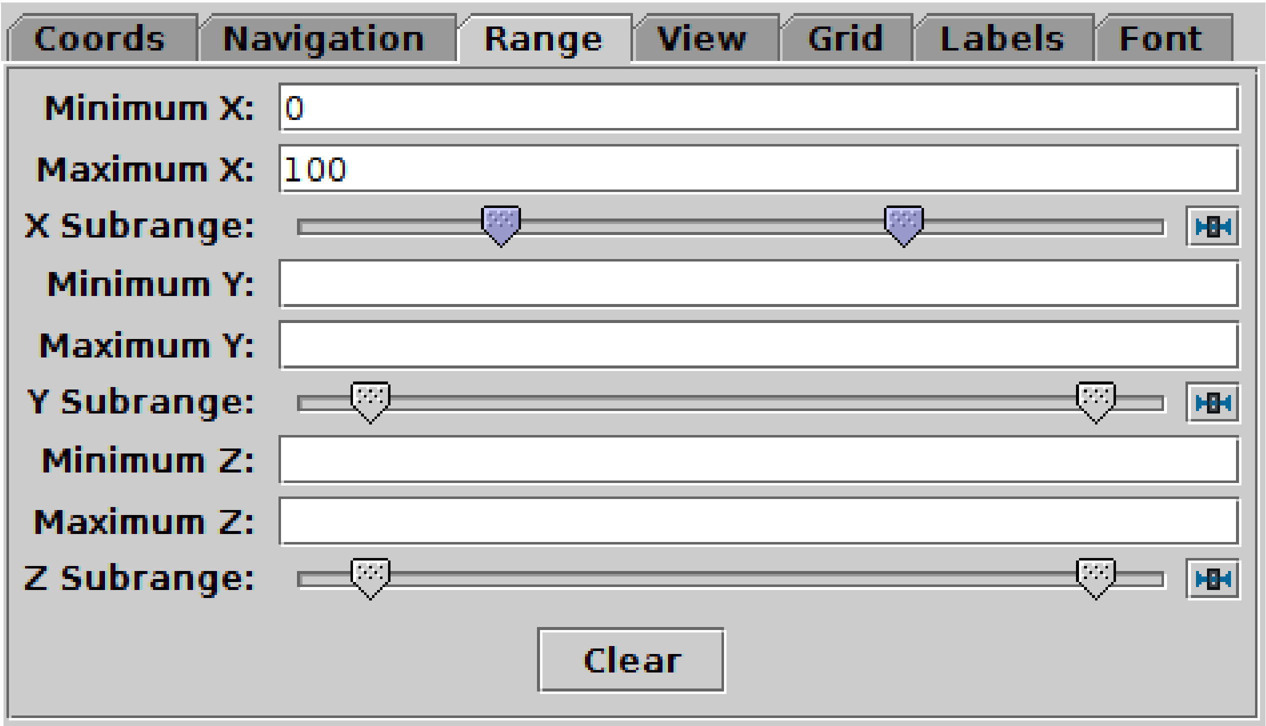
Navigation tab of the cube Axes control

The **Navigation** tab controls details of how the navigation works. It has the following options:

Zoom Axes
By default the **Auto** setting is in effect. This means that the mouse wheel zooms around the center of the cube, and right-button drag zooms in the two dimensions most closely aligned with the plane of the screen, with the reference position set by the initial position of the mouse. If **Auto** is unset, then all zoom operations are around the cube center, and affect the axes indicated by the **X/Y/Z** checkboxes.

Zoom Factor
Controls the factor by which each zoom action zooms the plot. Moving this slider to the left/right makes the mouse more/less sensitive (one wheel click or dragging a fixed distance has more/less zoom effect).

Range Tab



Minimum X: 0

Maximum X: 100

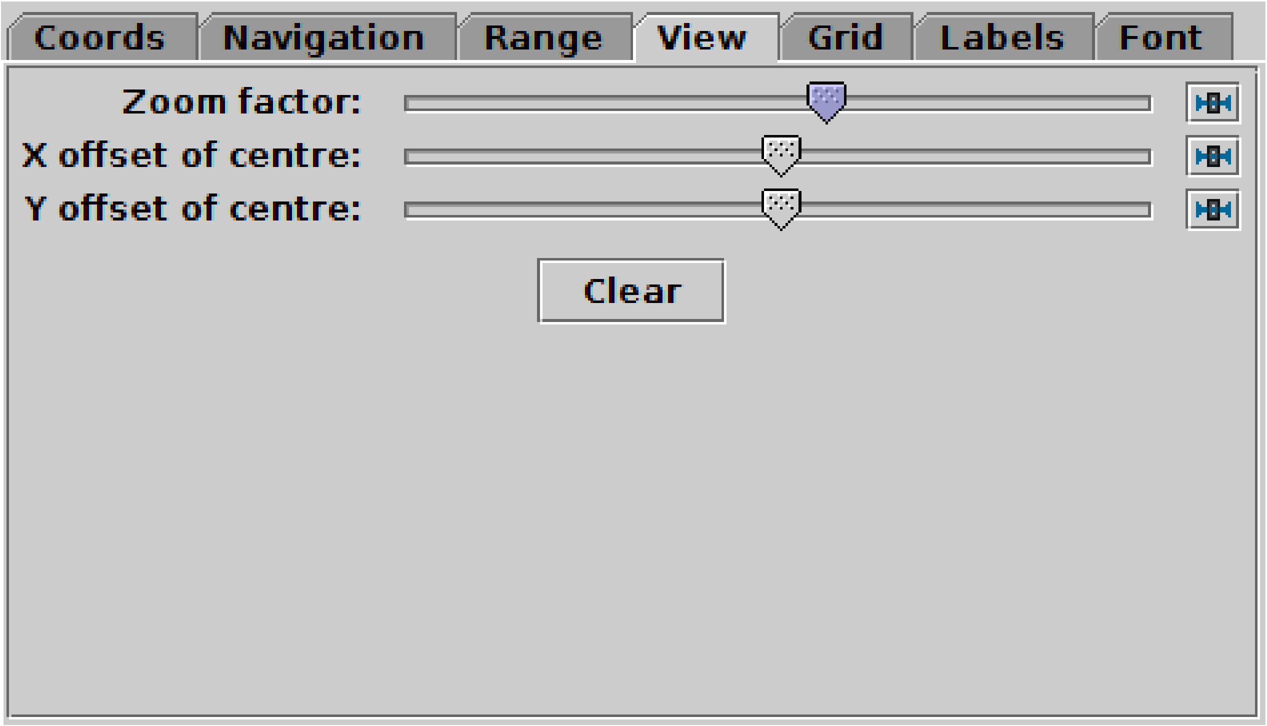
Range tab of the cube Axes control

The **Range** tab provides manual configuration of the visible range of the plot. Making changes to this tab will reset the visible plot range, but not vice versa - zooming and panning in the usual way will not change the settings of this panel.

Filling in the **Minimum**/**Maximum** fields for one or more axes will constrain the corresponding range of the visible data. The limits corresponding to any of those fields that are left blank will initially be worked out from the data. The **Subrange** double-sliders restrict the ranges within the (explicit or automatic) min/max ranges. Note you can move both sliders at once by grabbing a position between the two.

The **Clear** button resets all the fields.

View Tab



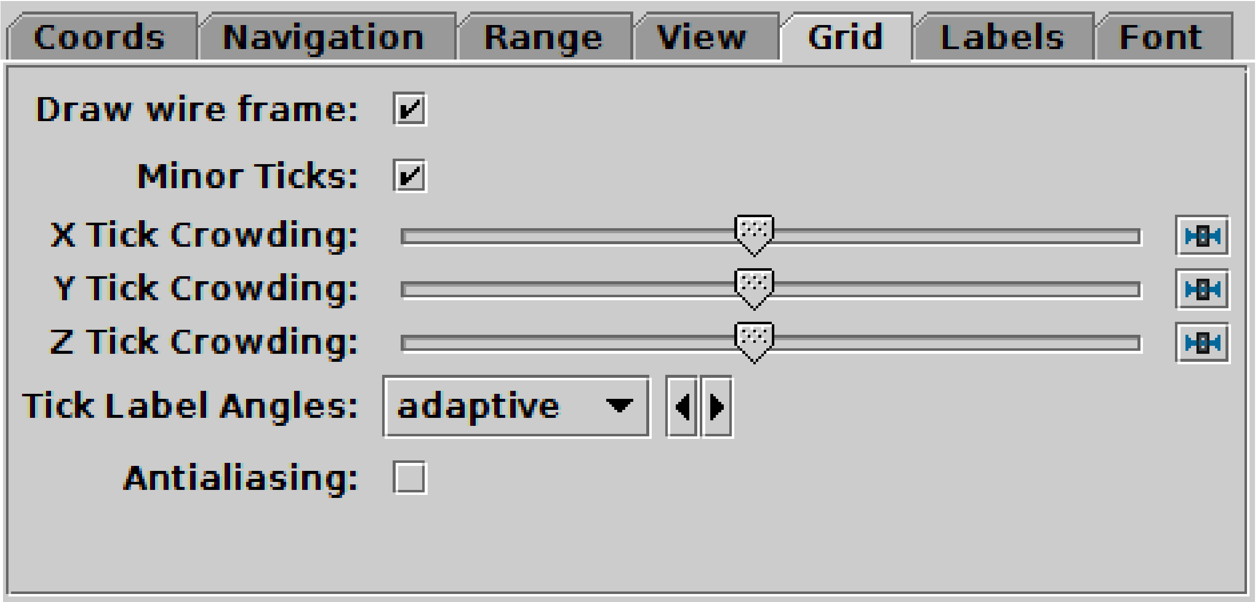
View tab of the cube Axes control

The **View** tab can configure how the cube containing the data is viewed in the plot window, though it does not control the content of the cube.

Zoom factor
Sets the magnification of the cube wireframe itself, without affecting the data volume it contains. This cannot be done with the mouse.

X/Y offset of centre
Controls where on the screen the cube wireframe is centred. This cannot be done with the mouse.

Grid Tab



Draw wire frame: ☑

Minor Ticks: ☑

Tick Label Angles: adaptive

Antialiasing: ☐

Grid tab of the cube Axes control

The **Grid** tab configures the appearance of the cube wire frame enclosing the data volume.

Draw wire frame
Whether the enclosing cube is drawn at all.

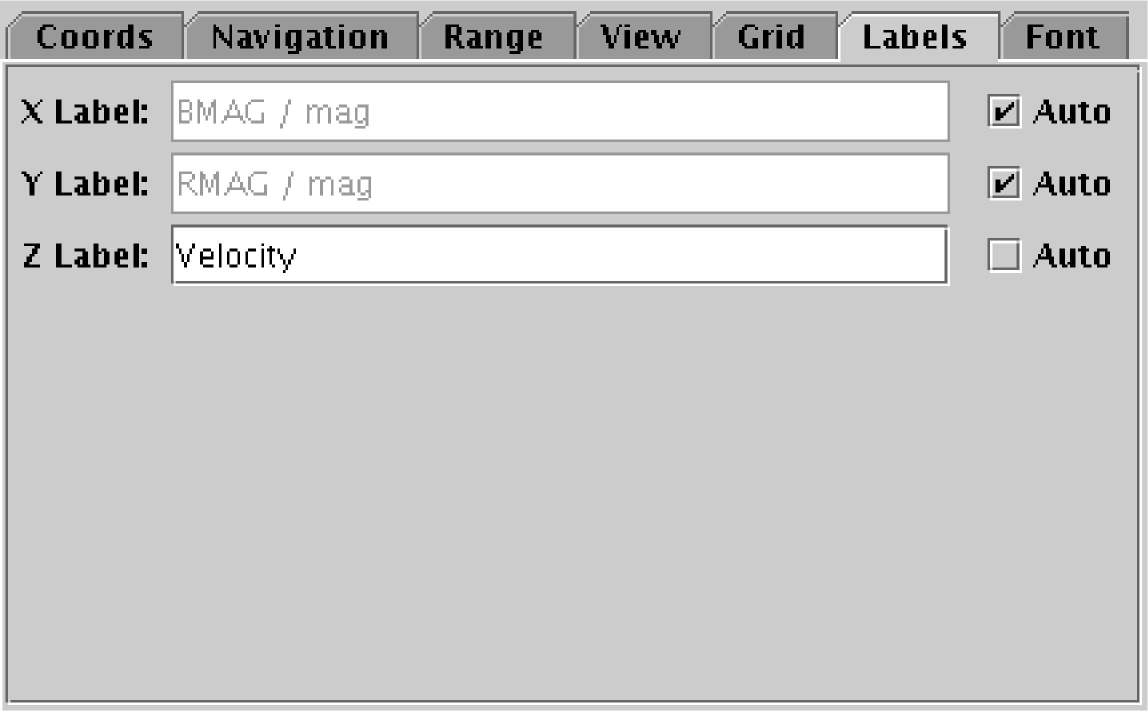
Minor Ticks
If set, minor (unlabelled) tick marks will be drawn between the major (labelled) ones.

X/Y/Z Tick Crowding
Use the slider to influence how many tick marks are draw on each axis.

Tick Label Angles
Controls orientation of numeric labels on the axes. By default they are drawn **adaptive**ly: horizontally where possible, but may be angled to accommodate more labels if crowding is high. But you can choose to fix the orientation **horizontal** or **angled** instead.

Antialiasing
Controls whether grid lines will be drawn antialiased (smoothed) or not. This option does not affect exported plots.

<u>**Labels Tab**</u>



X Label: BMAG / mag ☑ Auto

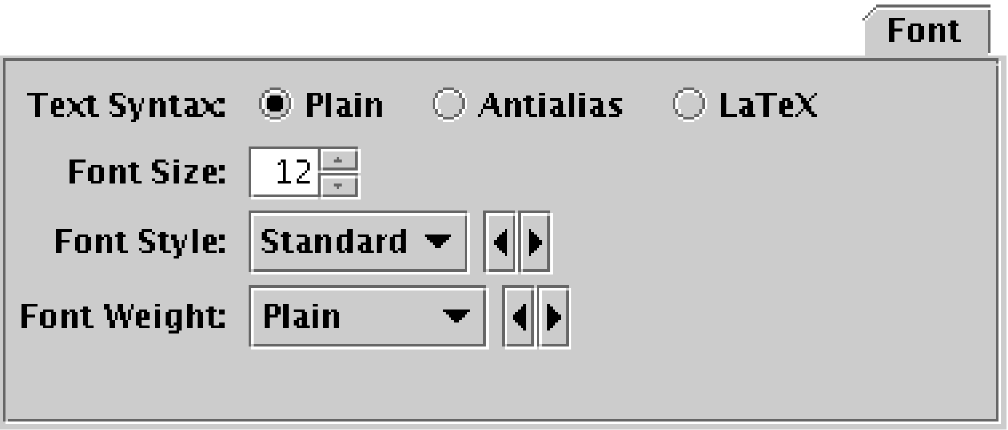
Y Label: RMAG / mag ☑ Auto

Z Label: Velocity □ Auto

Labels tab of the cube Axes control

The **Labels** tab controls the text labels on the axes. If the **Auto** checkbox is set, the text will be taken from one of the data coordinates being plotted on that axis. To override those with your own axis labels, unset Auto and type text in to the **Label** fields.

<u>**Font Tab**</u>



Text Syntax: ● Plain ○ Antialias ○ LaTeX

Font Size: 12

Font Style: Standard

Font Weight: Plain

Font tab

The **Font** tab configures the font used for axis annotation. It also affects some other things like the legend.

Text Syntax
How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. At time of writing, on MacOS antialiased text seems to be required to stop the writing coming out upside-down for non-horizontal text. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size
Size of the font in points.

Font Style
Style of the font.

Font Weight
Whether the font is plain, bold or italic.

A.4.12 Sphere Plot Window

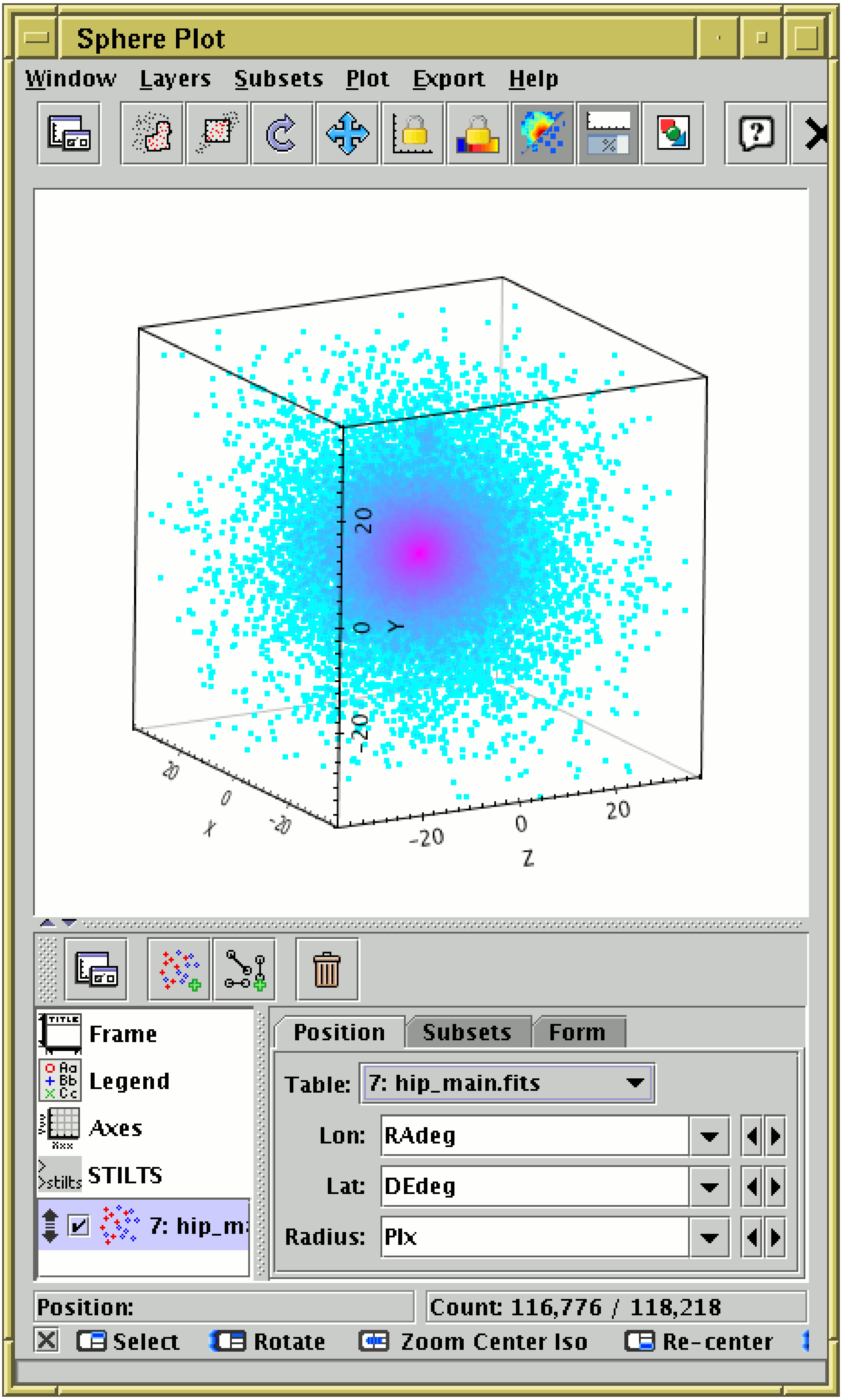


Table: 7: hip_main.fits

Lon: RAdeg

Lat: DEdeg

Radius: Plx

Position:

Count: 116,776 / 118,218

Sphere Plot Window

The **Sphere Plot** ( ) plots spherical polar coordinates in an isotropic 3-dimensional space.

Although the supplied coordinates are spherical polar, the visible space is not necessarily centred on the coordinate origin, and the visible axes are Cartesian. In many respects this works like the Cube Plot Window

The positional coordinates are **Longitude** and **Latitude** (in degrees) and **Radius**.

The sphere plot offers the following plot controls:

- Position Layer Control, with these form options:
  - Mark
  - Size
  - SizeXY
  - Polygon
  - Label
  - Line3d
  - Contour
- Pair Layer Control, with the following plot controls:
  - Link2
  - Mark2
- Area Layer Control, with the following plot control:
  - Area
  - Central
  - Area Label
- Quad Layer Control, with the following plot controls:
  - Poly4
  - Mark4
- SphereGrid Layer Control

Note that use of the Auto, Density and Weighted shading modes can be confusing in 3 dimensions with multiple datasets. This is because pixels based on density along a line of sight are not located at any point on that line, so shaded pixels can't appear at the "right" place in the 3-d space. The same applies to a lesser extent with contours. They work fine with a single dataset though.

See the Window Overview (Appendix A.4.2) for features common to all plotting windows. The following subsections describe navigation and axis configuration.

A.4.12.1 Sphere Navigation

For general comments on plot navigation, see Appendix A.4.2.1. Additional configuration options are available in the **Navigation** tab of the Axes control.

Navigation in three dimensions with a two-dimensional screen and mouse is a bit of a challenge. However, the actions listed below should make it fairly straightforward to navigate around points in 3d space to zoom in on the regions that you are interested in. The actions are quite like for the Cube Window, but all zooming is isotropic.

Left drag
Rotate. Rotates the view cube about its center.

Wheel
Zoom. Spinning the mouse wheel forwards/backwards zooms the space in the view cube in/out around the cube center. The mouse position does not affect it.

Right drag (CTRL-drag)
Zoom. Zooming up-and-right/down-and-left does just the same as spinning the mouse wheel forwards/backwards.

Center drag (SHIFT-drag)
2d pan. Dragging with the center button pans the 3d space in the two dimensions best alighned with the plane of the screen. There is no movement in the third (mostly perpendicular to the screen) direction.

Left click
Select. If there are plotted points near the cursor it will identify one, plot a marker on it, and activate (Section 8) it.

In 3d, it's not obvious which is the nearest point to a 2d cursor, since a screen position represents a line not a point. To break the degeneracy, the point used is the one nearest the center of mass of plotted points along the line of sight represented by the cursor position. The upshot of this is that if you click on an isolated point, you'll pick that point, and if you click on a dense cluster, you'll get a point near the center (of mass) of that cluster.

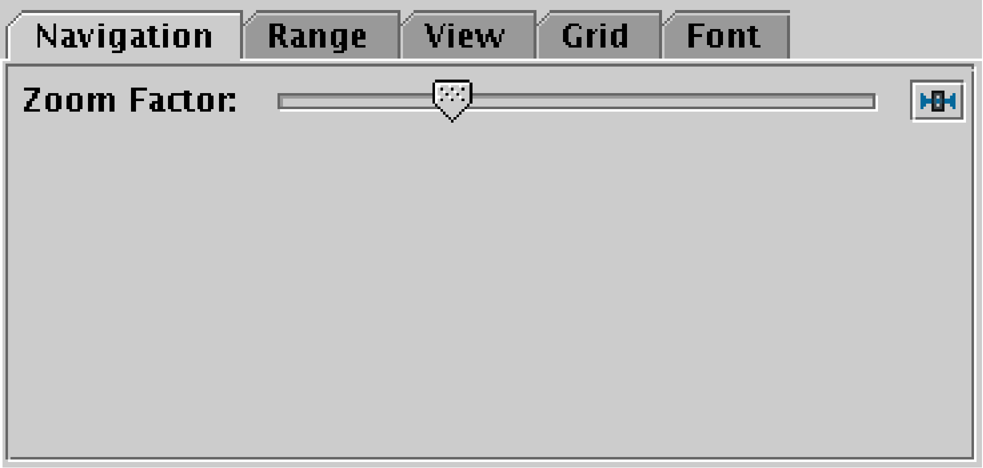
Right click (CTRL-click)
Re-center. Right-clicking identifies a position in a similar way to the left-click Select action, and then translates the plot so that point is at the center of the view cube. This means that clicking on a cluster will put that cluster in the center of the cube. If you click on an empty region, a position half way between front and back of the cube at that X/Y position will be used.

You can also manually fix the plot bounds using the **Range** tab of the Axes control.

A.4.12.2 Sphere Axes Control

The **Axes** control () for the sphere plot window has the following tabs:

Navigation Tab

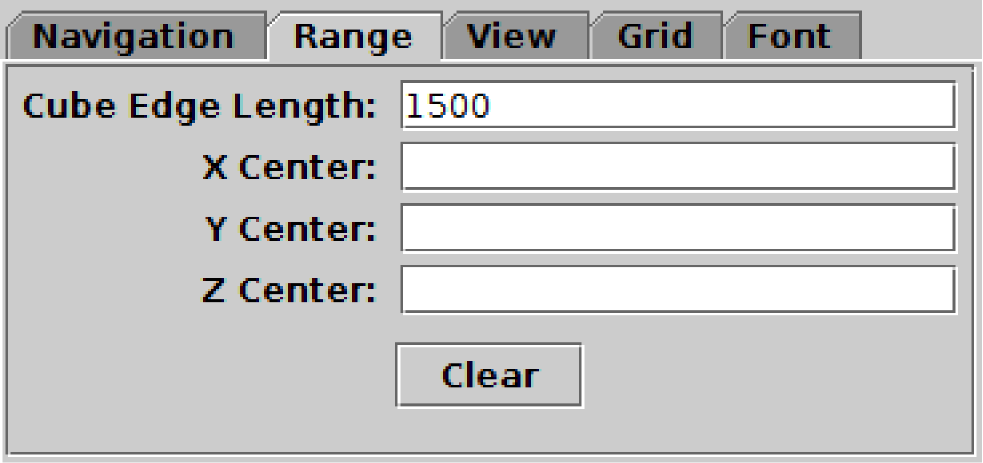


Navigation tab of the sphere Axes control

The **Navigation** tab controls details of how the navigation works. It has the following option:

Zoom Factor
Controls the factor by which each zoom action zooms the plot. Moving this slider to the left/right makes the mouse more/less sensitive (one wheel click or dragging a fixed distance has more/less zoom effect).

Range Tab



Cube Edge Length: 1500
X Center:
Y Center:
Z Center:

Range tab of the sphere Axes control

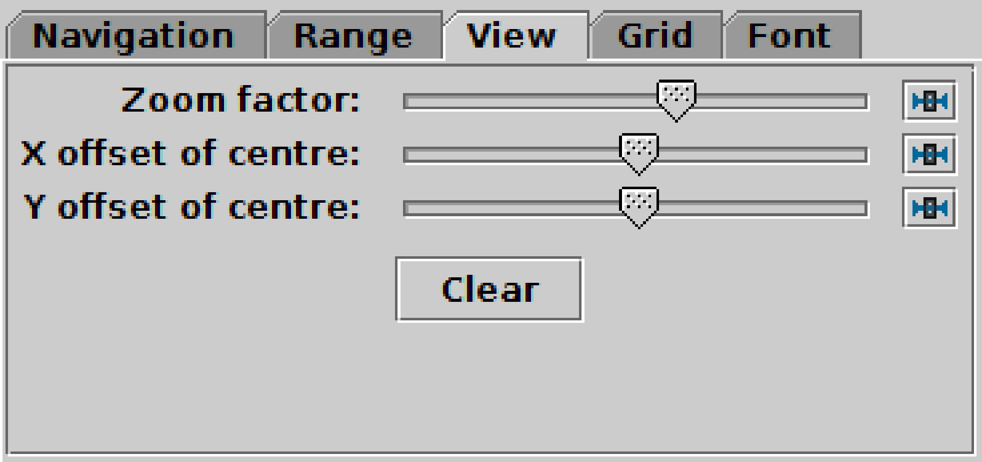
The **Range** tab provides manual configuration of the data range of the plot. Making changes to this tab will reset the visible plot range, but not vice versa - zooming and panning in the usual way will not change the settings of this panel. Any values not filled in will be determined from the data. The fields are:

Cube Edge Length
Specifies the dimension along each side of the view cube in data units.

X/Y/Z Center
Gives the position of the center of the view cube in data coordinates.

The **Clear** button resets all the fields.

View Tab



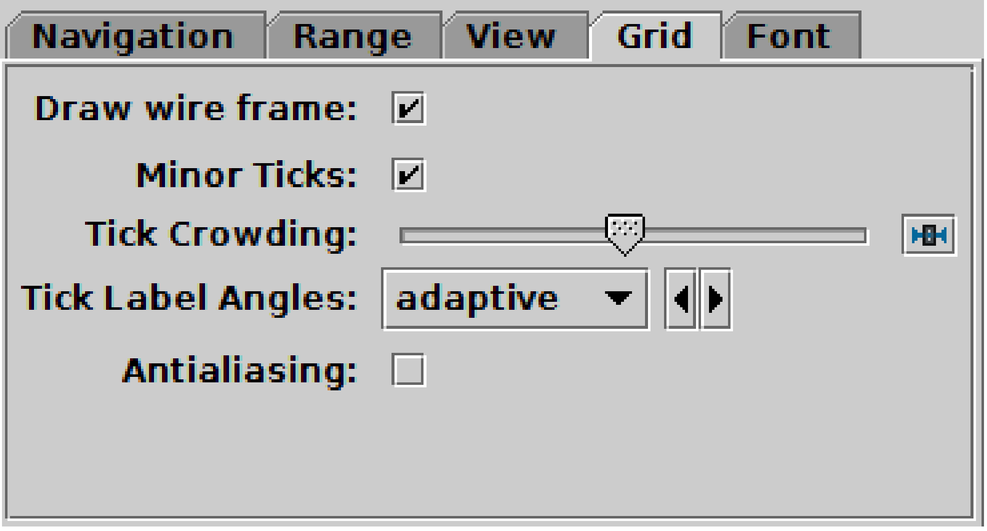
View tab of the sphere Axes control

The **View** tab can configure how the cube containing the data is viewed in the plot window, though it does not control the content of the cube.

Zoom factor
Sets the magnification of the cube wireframe itself, without affecting the data volume it contains. This cannot be done with the mouse.

X/Y offset of centre
Controls where on the screen the cube wireframe is centred. This cannot be done with the mouse.

Grid Tab



Draw wire frame: ☑

Minor Ticks: ☑

Tick Label Angles: adaptive

Antialiasing: □

Grid tab of the sphere Axes control

The **Grid** tab configures the appearance of the cube wire frame enclosing the data volume.

Draw wire frame
Whether the enclosing cube is drawn at all.

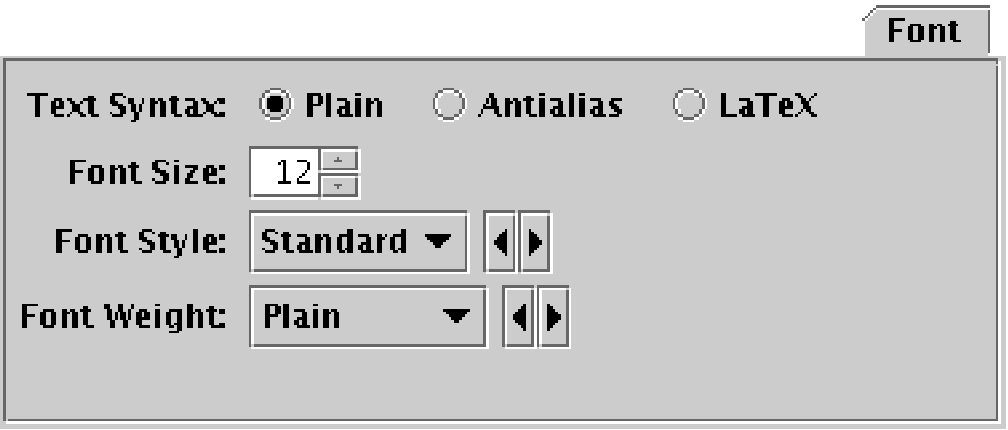
Minor Ticks
If set, minor (unlabelled) tick marks will be drawn between the major (labelled) ones.

Tick Crowding
Use the slider to influence how many tick marks are draw on the axes.

Tick Label Angles
Controls orientation of numeric labels on the axes. By default they are drawn **adaptive**ly: horizontally where possible, but may be angled to accommodate more labels if crowding is high. But you can choose to fix the orientation **horizontal** or **angled** instead.

Antialiasing
Controls whether grid lines will be drawn antialiased (smoothed) or not. This option does not affect plots exported to vector formats.

Font Tab



Text Syntax: ● Plain ○ Antialias ○ LaTeX

Font Size: 12

Font Style: Standard

Font Weight: Plain

Font tab

The **Font** tab configures the font used for axis annotation. It also affects some other things like the legend.

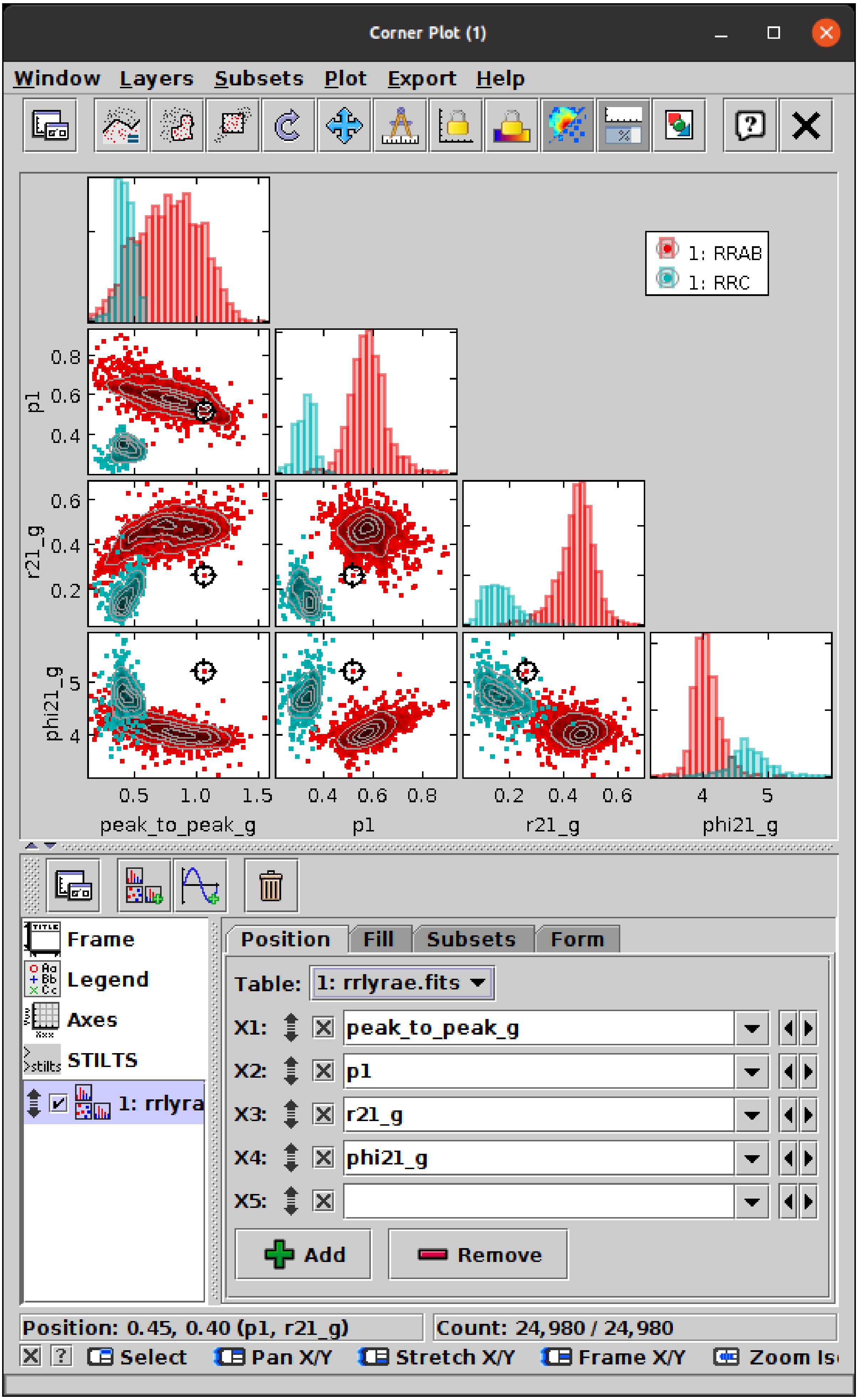
Text Syntax
How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. At time of writing, on MacOS antialiased text seems to be required to stop the writing coming out upside-down for non-horizontal text. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size
Size of the font in points.

Font Style
Style of the font.

Font Weight
Whether the font is plain, bold or italic.

A.4.13 Corner Plot Window



☑ 1: rrlyra

Table: 1: rrlyrae.fits

X1: ☒ peak_to_peak_g

X2: ☒ p1

X3: ☒ r21_g

X4: ☒ phi21_g

X5: ☒

Position: 0.45, 0.40 (p1, r21_g)

Count: 24,980 / 24,980

Corner Plot Window

The **Corner Plot** ( ) represents the relationships between multiple quantities by drawing a scatter-like plot of every pair of coordinates, and/or a histogram-like plot of every single coordinate, and placing these on (half or all of) a square grid. The horizontal coordinates of all the plots on each column, and the vertical coordinates of all the plots on each row, are aligned. The plots are all linked, so you can make graphical selections or click on a point to activate it in one of the panels, and the other panels will immediately reflect that action. Single-coordinate (histogram-like) plots appear on the diagonal, and coordinate-pair (scatter plot-like) plots appear off diagonal. By default only the diagonal and sub-diagonal part of the resulting plot matrix is shown, since the plots above the diagonal are equivalent to those below it, but this is configurable. This representation is variously known as a **corner plot**, **scatter plot matrix**, **splom** or **pairs plot**.

In principle any number of quantities can be simultaneously compared in this way, but attempting to use too many will make the individual plots too small to be useful. Depending on the size of your screen, you may find about 20 is a reasonable upper limit, though more detail will be visible with fewer. Note that the **Float Controls** ( ) button (or equivalent action in the **Window** menu) can be used to detach the control panel from the plot display and so free up more space for the visualisation itself.

By default the linear size of the plot grid will be defined by the highest-numbered non-blank coordinate that is visible in any of the active Position layer controls, but it can be adjusted directly in the **Axes** control ( ).

The configuration of the plots is done in the same way as for the Plane Plot, and many of the plot types that are available there can be used in the grid cells of this plot. The Corner Plot currently offers the following plot controls:

- Matrix Layer Control, containing these scatter-plot like actions:
  - Mark
  - Line
  - Linear Fit
  - Label
  - Contour
  - Grid
  - Fill
  - Quantile

  and these histogram-like actions:
  - Histogram
  - KDE

    - KNN
    - Densogram
    - Gaussian

    The scatter-plot-like options will generate plots in the off-diagonal positions of the grid, and the histogram-like ones will generate plots on the leading diagonal. By default the **Mark** and **Histogram** options are both selected, but you can hide these as well as adding others.

- Function Layer Control; at present this draws the same function in all visible grid positions, both off- and on-diagonal.

As well as the standard actions, this window additionally provides the following toolbar buttons:

- Measure Distance
- Draw Algebraic Subset

See the Window Overview (Appendix A.4.2) for features common to all plotting windows. The following subsections describe navigation and axis configuration.

Note: the Corner Plot Window is new at TOPCAT version 4.9. Some of the features are experimental and may be changed in future releases.

A.4.13.1 Corner Navigation

Navigation actions in the corner plot are the same as for the Plane Plot; you can pan and zoom on each panel in the grid of plots using the mouse. However in this case the action will affect not only the panel that you are pointing at, but all the other panels in the same row and the other panels in the same column, so that the horizontal and vertical axes stay consistent per row and per column.

Using the mouse to drag and zoom on one axis only, rather than both at once, by positioning it just outside the bounds of one of the panels (e.g. between panels) as described below, can be particularly useful in this plot.

The navigation actions are:

Left drag
: *Pan.* On the body of the plot this moves it around up/down/left/right. To move it only vertically, drag on the left of the Y axis or between panels in a row. To move it only horizontally, drag below the X axis or between panels in a column. You can configure dragging on the body of the plot to be only vertical or horizontal by setting the **Pan/Zoom Axes** in the **Navigation** axis configuration tab.

Right drag (CTRL-drag)
: *Stretch zoom.* On the body of the plot, dragging up/down stretches/squashes the plot vertically, and dragging left/right stretches/squashes it horizontally. Zooming is around the mouse position at the start of the drag. To zoom only vertically, drag on the left of the Y axis or between panels in a row. To zoom only horizontally, drag below the X axis or between panels in a column.

Center drag (SHIFT-drag)
: *Frame zoom.* On the body of the plot, dragging right-and-down or right-and-up drags out a frame; when the button is released, the plot will be zoomed in to cover the area enclosed by the

frame. Dragging left (and up or down) does something like the opposite, you can zoom out using a similar (though not quite the same) mechanism. To zoom in/out only horizontally, drag right/left below the X axis or between panels in a column. To zoom in/out only vertically, drag up/down on the left of the Y axis or between panels in a row.

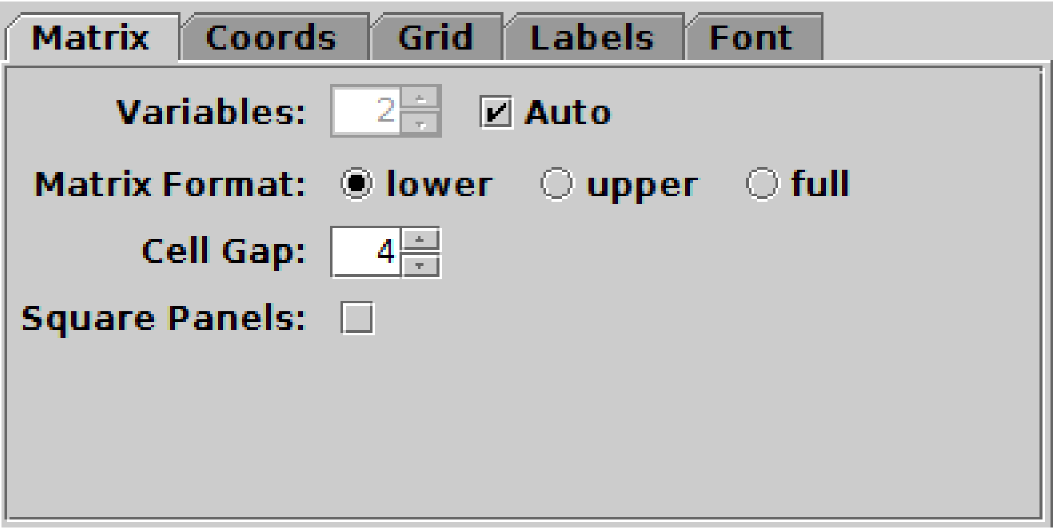
Wheel
Isotropic zoom. Spinning the mouse wheel forwards/backwards will zoom in/out around the mouse position, just like dragging with the right button up-and-right or down-and-left.

Left click
Select. If there is a plotted point near the cursor, it will plot a marker on it and activate (Section 8) it. This includes causing the marker to show up in all the other visible panels of the plot grid, not just the one you clicked on.

A.4.13.2 Corner Axes Control

The **Axes** control () for the corner plot window has the following tabs:

Matrix Tab



Variables: 2 ☑ Auto

Matrix Format: ● lower ○ upper ○ full

Cell Gap: 4

Square Panels: □

Matrix tab of corner Axes control

The **Matrix** tab configures the grid on which the individual plots are placed. It has the following options:

Variables
Sets the number of variables to be compared; that corresponds to the linear size of the plot grid (or size+1 if no histogram-like plots are included). If the **Auto** box is checked then this value will be determined automatically from the values that are filled in on the **Position** controllers.

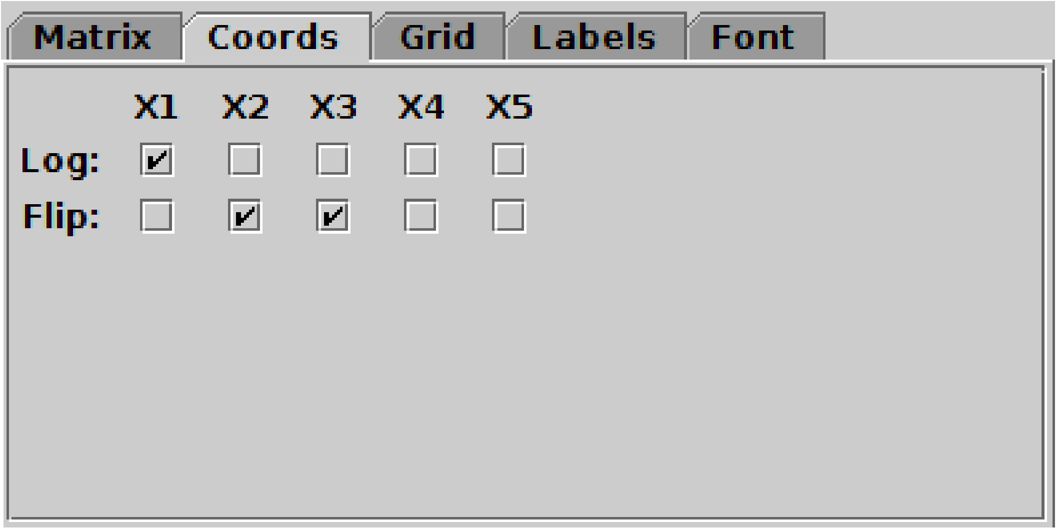
Matrix Format
Configures which cells of the matrix grid will be filled in. Below-diagonal, above-diagonal, or the full matrix can be chosen. Given grid cells will only appear if there are appropriate plot layers specified, i.e. 2-coordinate (scatter-plot-like) plots for the off-diagonal cells and 1-coordinate (histogram-like) plots for the diagonal cells. The available options are:

- **lower**: only the lower diagonal part of the matrix is populated, as well as the diagonal if histogram-like elements are present
- **upper**: only the upper diagonal part of the matrix is populated, as well as the diagonal if histogram-like elements are present
- **full**: all cells of the matrix are populated where present

Cell Gap
Sets the number of pixels between cells in the displayed matrix of plots.

Square Panels
If checked, each of the plotted panels in the matrix will have the same vertical and horizontal dimension. If unchecked, the shape of each panel will be determined by the shape of the overall plotting area.

Coords Tab



	X1	X2	X3	X4	X5
Log:	☑	□	□	□	□
Flip:	□	☑	☑	□	□

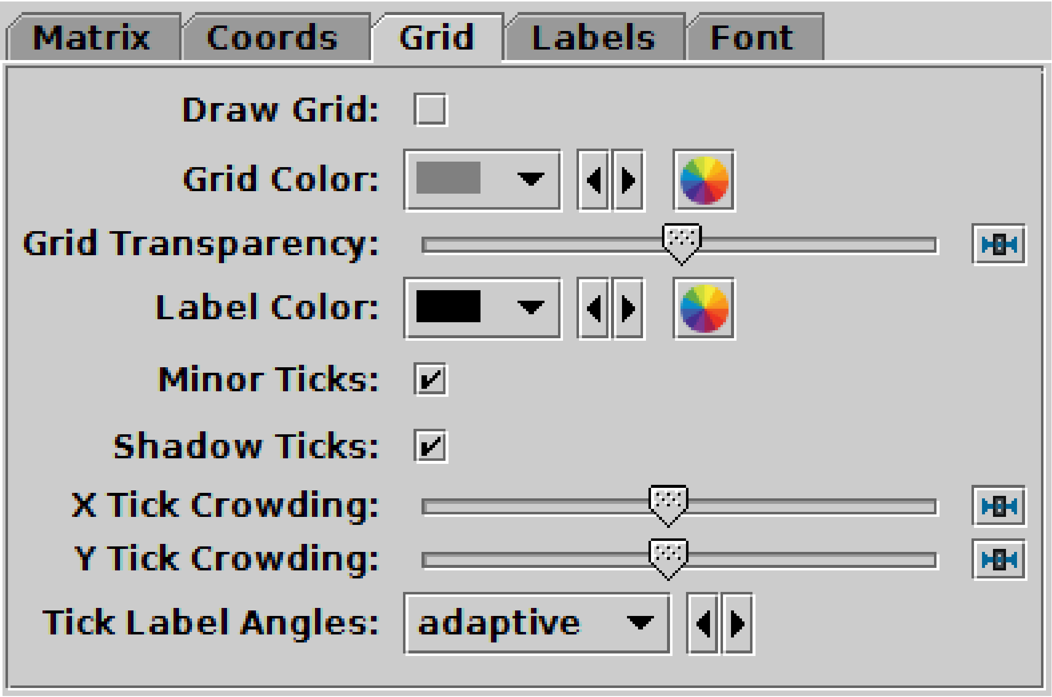
Coords tab of corner Axes control

The **Coords** tab controls the axis coordinates. For each currently visible axis there are two checkboxes:

Log
If selected, the coordinates corresponding to this axis are logarithmic, otherwise they are linear.

Flip
If selected, the coordinates corresponding to this axis run in the opposite direction to normal.

Grid Tab



Draw Grid: □

Grid Color: ▨

Grid Transparency:

Label Color: ■

Minor Ticks: ☑

Shadow Ticks: ☑

X Tick Crowding:

Y Tick Crowding:

Tick Label Angles: adaptive

Grid tab of corner Axes control

The **Grid** tab configures the appearance of the axis grid in each panel. It has the following options:

Draw Grid
: If true, grid lines will be drawn across the plot for every tick mark.

Grid Colour
: Selects the colour with which grid lines will be drawn.

Grid Transparency
: Controls the transparency of the grid lines, which are drawn over the plot content.

Label Colour
: Selects the colour in which axis label text will be written.

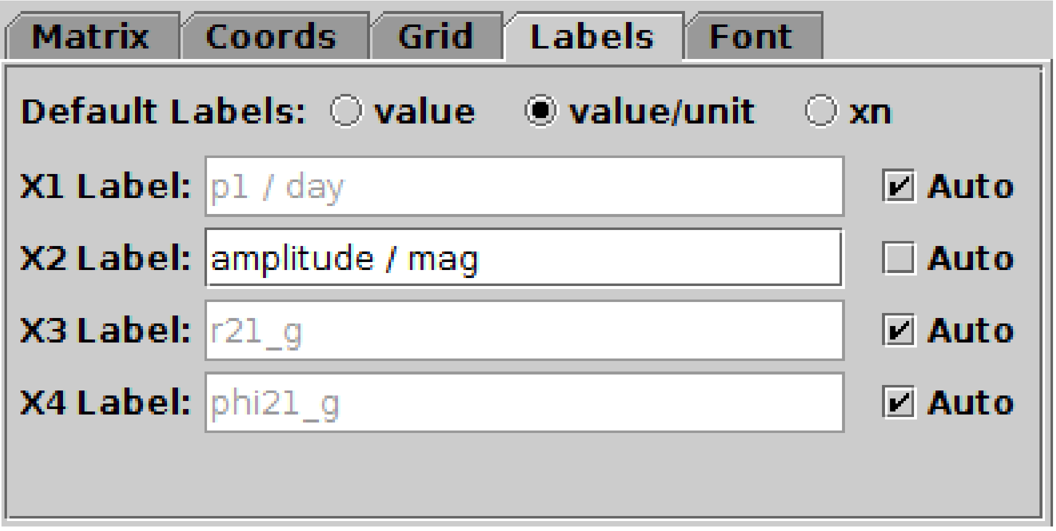
Minor Ticks
: If set, minor (unlabelled) tick marks will be drawn between the major (labelled) ones.

Shadow Ticks
: If set then tick marks without numeric labels are painted along the axis opposite to the primary axis, so that tick marks are visible along all edges not just the ones with numeric labels.

X/Y Tick Crowding
: Use the slider to influence how many tick marks are draw on each axis.

Tick Label Angles
: Controls orientation of numeric labels on the axes. By default they are drawn **angled** to accommodate more labels on crowded axes, but you can also choose **horizontal**. If **adaptive** is selected, they will be horizontal where possible, but may be angled to accommodate more labels if crowding is high; note this option is currently not perfect and can result in suboptimal border placement.

Labels Tab



Default Labels: ○ value ● value/unit ○ xn

X1 Label: p1 / day ☑ Auto

X2 Label: amplitude / mag □ Auto

X3 Label: r21_g ☑ Auto

X4 Label: phi21_g ☑ Auto

Labels tab of corner Axes control

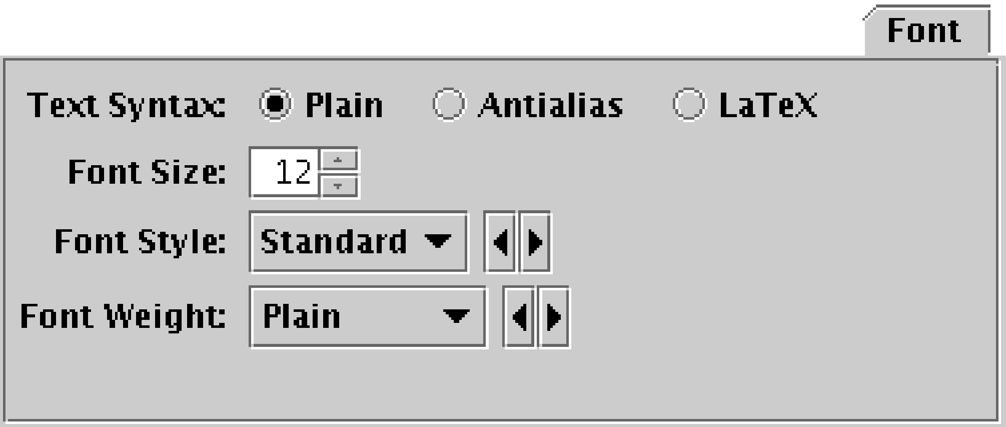
The **Labels** tab controls the text labels on the axes. The axes are labelled according to the text listed for each coordinate **X1**, **X2**, ... By default these labels are assigned automatically, but you can override this by unchecking the **Auto** checkbox for the coordinate in question to enter your own text.

The automatically assigned values are generally taken by looking at the values entered into the coordinate selector fields of the Position tabs controlling the plot, but the details are determined by the **Default Labels** selector. This has three options:

- **value**: The quantity entered in the coordinate selector is used.
- **value/unit**: Like **value** above, but if the unit of the quantity is known it is appended after a "/" symbol.
- **xn** The axis label is used (**X1**, **X2**, ...).

The other plot windows effectively use the **value/unit** policy, but annotations can get rather crowded in the corner plot, so the default here is **value**, but it can be made even more compact by selecting **xn** if required.

Font Tab



Text Syntax: ● Plain ○ Antialias ○ LaTeX

Font Size: 12

Font Style: Standard

Font Weight: Plain

Font tab

The **Font** tab configures the font used for axis annotation. It also affects some other things like the legend.

Text Syntax

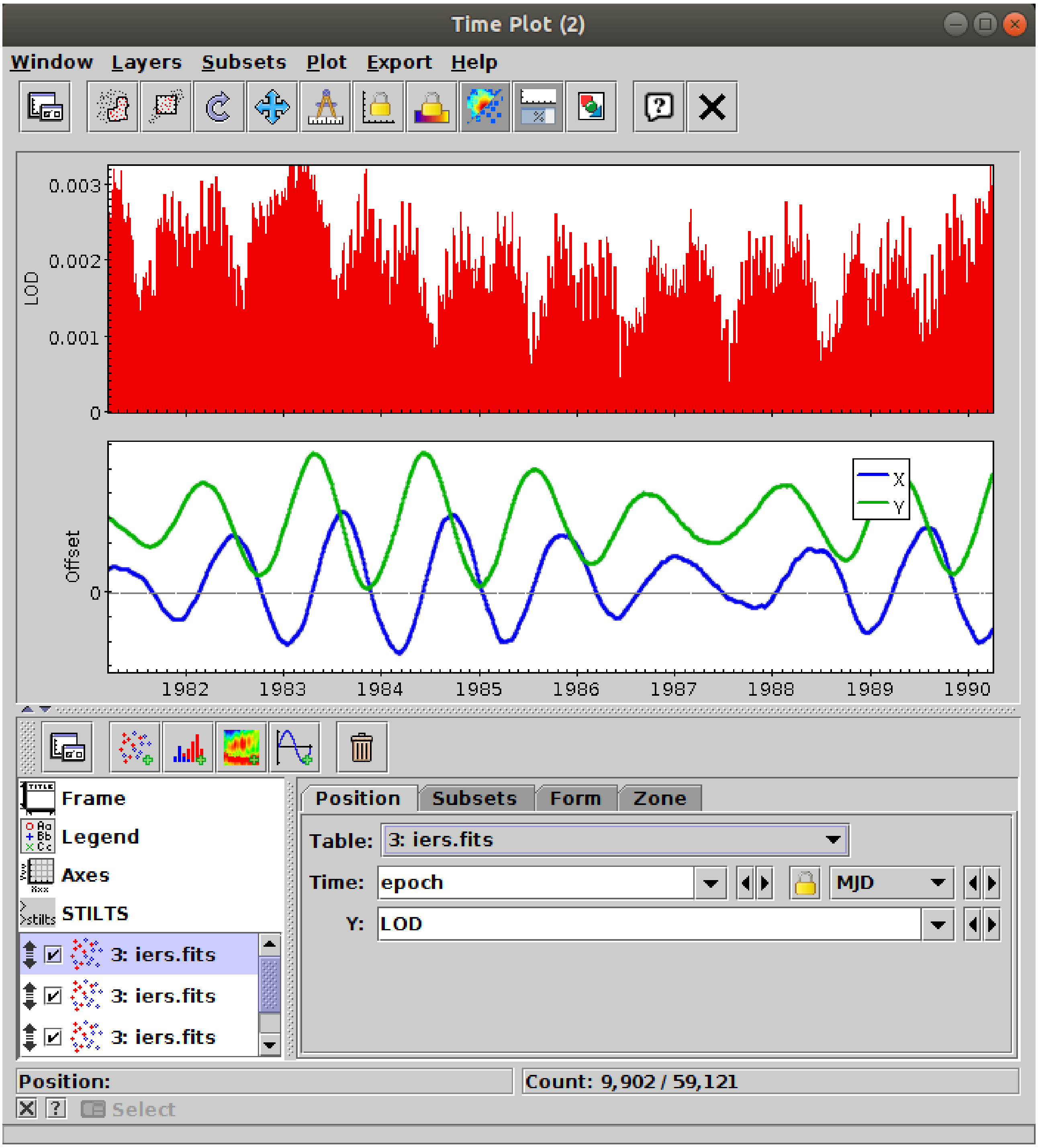
How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size
Size of the font in points.

Font Style
Style of the font.

Font Weight
Whether the font is plain, bold or italic.

A.4.14 Time Plot Window



☑ 3: iers.fits
☑ 3: iers.fits
☑ 3: iers.fits

Table: 3: iers.fits
Time: epoch MJD
Y: LOD

Position:
Count: 9,902 / 59,121

Time Plot Window

The **Time Plot** () is intended for plotting time series data.

The horizontal axis represents time, and can be labelled accordingly (for instance in minutes, hours, days, months and years), and the window can display appropriate types of plot including spectrograms.

To define the time coordinate, use the **Time** selector in position tabs as usual. Note however that this contains a **time format** selector on the right hand side that indicates how the selected quantity will be interpreted as a time value. The options are:

- **DecYear**: Years since 0AD
- **MJD**: Modified Julian Date (days since 17 Nov 1858)

- **JD**: Julian Day (days since 1 Jan 4713 BC)
- **Unix**: Seconds since midnight 1 Jan 1970
- **Iso8601**: ISO 8601 date-time string

If the input table contains suitable metadata, for instance in a CDF file or a TIMESYS-bearing VOTable 1.4, additional options may be available that are taken from information in the table. TOPCAT will make a guess at the correct format to use, but if it gets it wrong you can set the format selector manually. The format selector is updated to the best guess when you choose a new Time value; if you want to stop it doing that, use the Lock button (🔒).

Unlike most of the other plot windows the Time plot can display different data plots in different plot Zones (Appendix A.4.14.1) stacked vertically on top of each other, so that different plots share a time axis but have their own Y axis. The vertical spacing between the stacked plots can be configured using the **Cell Gap** configuration option in in the **Spacing** tab of the Frame Control.

It is possible to abuse this plot type to stack plots that do not actually have time on the horizontal axis. To do that, select **mjd** as the **Time Format** in the Grid tab of the **Axes Control**, and make sure that any **time format** selectors (described above) are similarly set to **MJD**. You may need to check that the right Zone is chosen (usually the bottom one) when you configure the axes.

The Time Plot offers the following plot controls:

- Position Layer Control, with these form options:
  - Line
  - Linear Fit
  - Mark
  - Error Bars (vertical bars only)
  - Label
  - Fill
  - Quantile
  - Grid
- Histogram Layer Control with these form options:
  - Histogram
  - KDE
  - KNN
  - Densogram
  - Gaussian

  Note that some of these histogram-like controls offer the **Combine** option that lets you define how values are combined together. If you choose the **sum-per-unit** or **count-per-unit** options, you can also use the **Per Unit** selector to determine whether values are represented as sums/counts per second, hour, day, year, etc.

- Spectrogram Layer Control
- Function Layer Control

As well as the standard actions, this window additionally provides the following toolbar button:

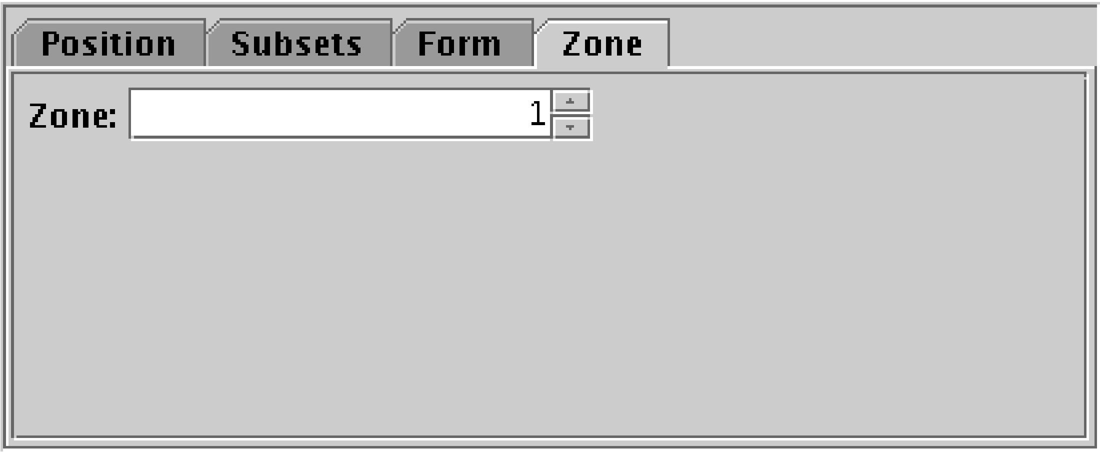
- Measure Distance

See the Window Overview (Appendix A.4.2) for features common to all plotting windows. The following subsections describe zones, navigation and axis configuration.

A.4.14.1 Plot Zones

Unlike the other plot types, the Time plot can display data in multiple panels, known as **Zones**, stacked vertically. All plots therefore share the same time axis, but can have different Y axes.

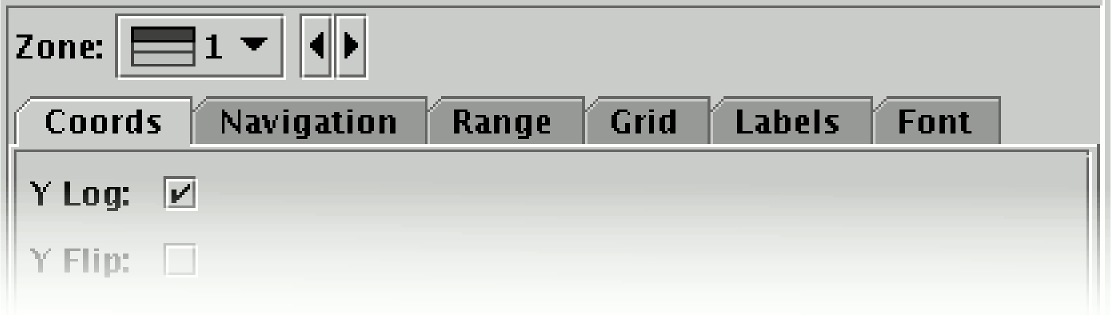
By default, each new plot added is displayed in a new zone stacked beneath the existing ones. By using the **Zone** tab for each plot however, you can control which plots appear in which zones. Each zone has a numeric identifier, which by default increments by one for each new plot, but if you select the same identifier for several plots, they will all appear in the same zone.



Zone: 1

Zone selection tab

Some of the fixed controls (Axes and Aux) operate on a per-zone basis. If multiple zones are visible, then a zone selector is displayed above the tabs:



Zone: 1

Y Log: ☑

Y Flip: □

Zone selector for Axes Control

In this case you should select the zone you want to control and adjust the configuration for that zone only. The selector shows an icon indicating the position of the zone in question. Each zone has to be configured separately. In a future release a global zone configuration option may be introduced as well.

This multi-zone feature is currently available only for the Time plot, but it may be added to other plot types at some point in the future.

A.4.14.2 Time Navigation

For general comments on plot navigation, see Appendix A.4.2.1. Additional configuration options are available in the **Navigation** tab of the Axes control.

Horizontal navigation (panning and zooming left and right along the shared time axis) affects all the stacked zones, but vertical navigation (panning and zooming along the Y axis of each zone) is done independently for each zone.

The navigation actions for this window are:

Left drag
Pan. On the body of the plot this usually moves it around left/right. By default, vertical movement is inhibited since you don't normally want it for a time plot. To pan vertically, drag on the left of the Y axis for the zone you want to affect. It is also possible to configure panning on the body of the plot to be in both directions using the **Pan/Zoom Axes** in the **Navigation** axis configuration tab.

Right drag (CTRL-drag)
Stretch zoom. On the body of the plot this usually stretches/squashes the plot centered on the horizontal position of the mouse. By default, vertical zooming is inhibited since you don't normally want it for a time plot. To zoom vertically, drag on the left of the Y axis for the zone you want to affect. It is also possible to configure zooming on the body of the plot to be in both directions using the **Pan/Zoom Axes** in the **Navigation** axis configuration tab.

Center drag (SHIFT-drag)
Frame zoom. On the body of the plot, dragging right usually drags out a frame; when the button is released, the plot will be zoomed in to cover the area enclosed by the frame. Dragging left does something like the opposite, you can zoom out using a similar (though not quite the same) mechanism. By default, vertical zooming is inhibited since you don't normally want it for a time plot. To zoom in/out vertically, drag up/down on the left of the Y axis for the zone you want to affect. It is also possible to configure zooming on the body of the plot to be in both directions using the **Pan/Zoom Axes** in the **Navigation** axis configuration tab.

Wheel
Zoom. Spinning the mouse wheel forwards/backwards will zoom horizontally in/out around the mouse position, just like dragging with the right button up-and-right or down-and-left.

Left click
Select. If there is a plotted point near the cursor, it will plot a marker on it and activate (Section 8) it.

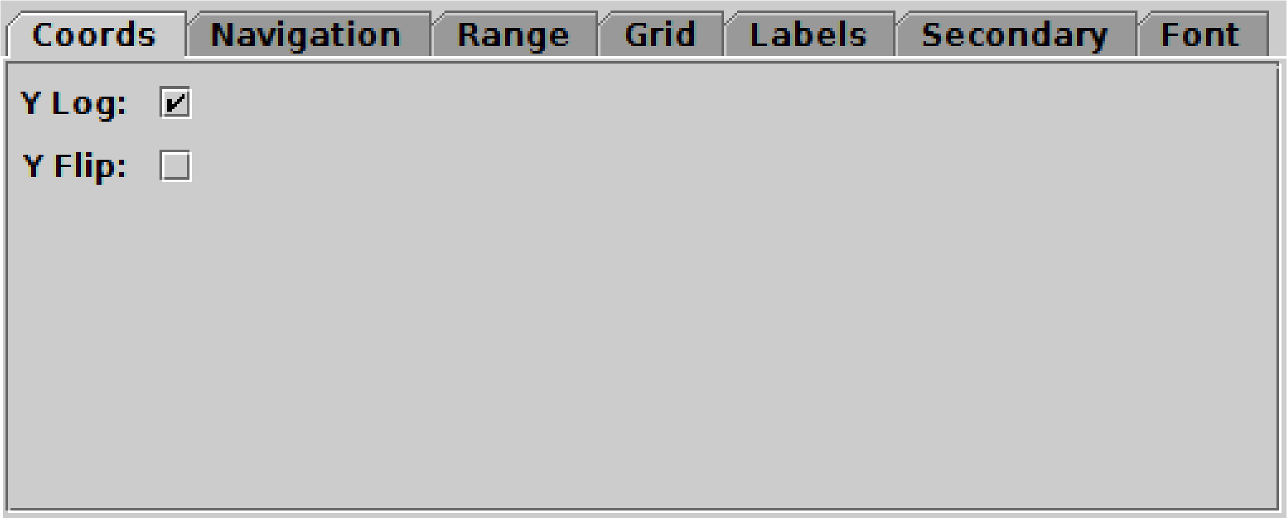
You can also manually fix the plot bounds using the **Range** tab of the Axes control.

A.4.14.3 Time Axes Control

Note: the Time Plot has multiple plot Zones (Appendix A.4.14.1), and the axes are configured individually for each zone.

The **Axes** () control for the time plot window has the following tabs:

Coords Tab



Y Log: ☑

Y Flip: □

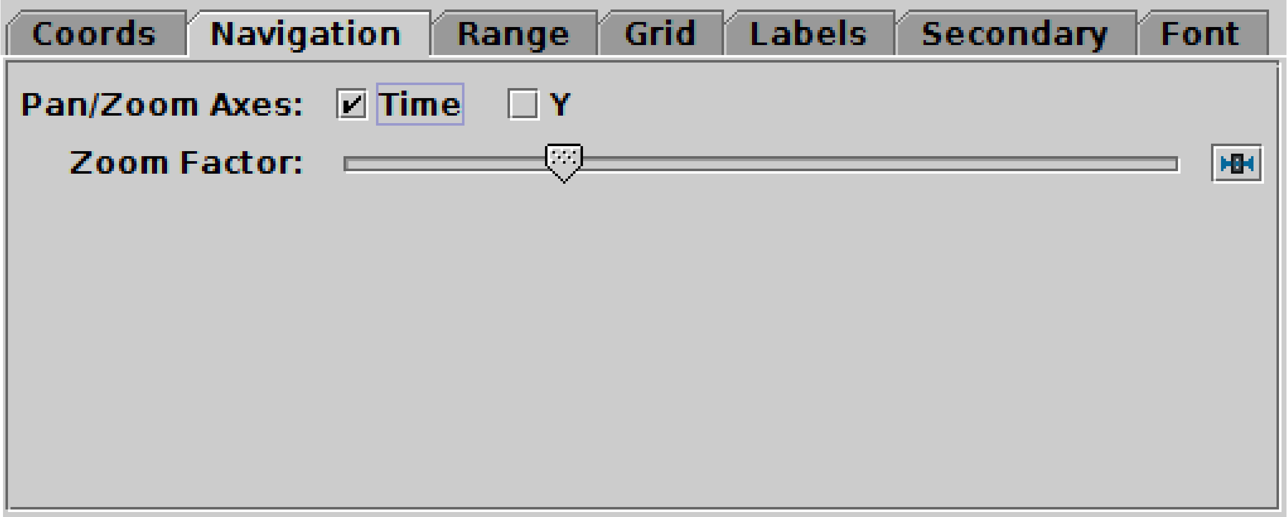
Coords tab of time Axes control

The **Coords** tab controls the vertical axis coordinates (you can't flip or rescale the time axis). It has the following options:

Y Log
If selected, vertical axis coordinates are logarithmic, otherwise they are linear.

Y Flip
If selected, vertical axis coordinate axes increase down rather than up.

Navigation Tab



Pan/Zoom Axes: ☑ Time □ Y

Zoom Factor:

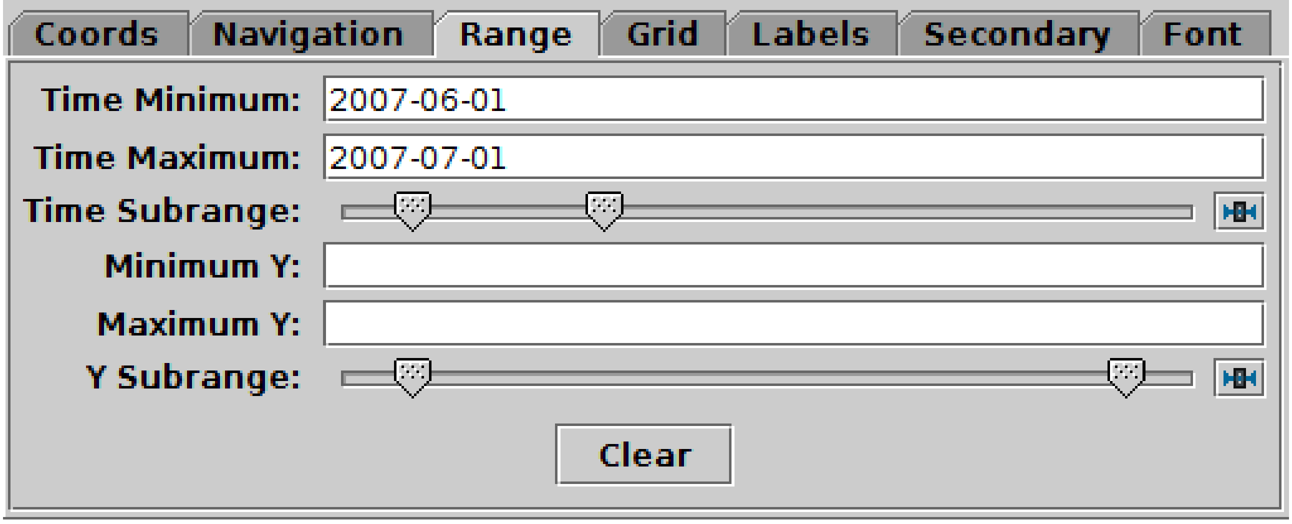
Navigation tab of time Axes control

The **Navigation** tab controls details of how the navigation works. It has the following options:

Pan/Zoom Axes
By default, dragging with the mouse or using the mouse wheel on the body of the plot will pan or zoom in the horizontal (time) direction only, which is usually what you want for a time series. Using this control you can make it work in the vertical direction as well.

Zoom Factor
Controls the factor by which each zoom action zooms the plot. Moving this slider to the left/right makes the mouse more/less sensitive (one wheel click or dragging a fixed distance has more/less zoom effect).

Range Tab



Time Minimum: 2007-06-01

Time Maximum: 2007-07-01

Minimum Y:

Maximum Y:

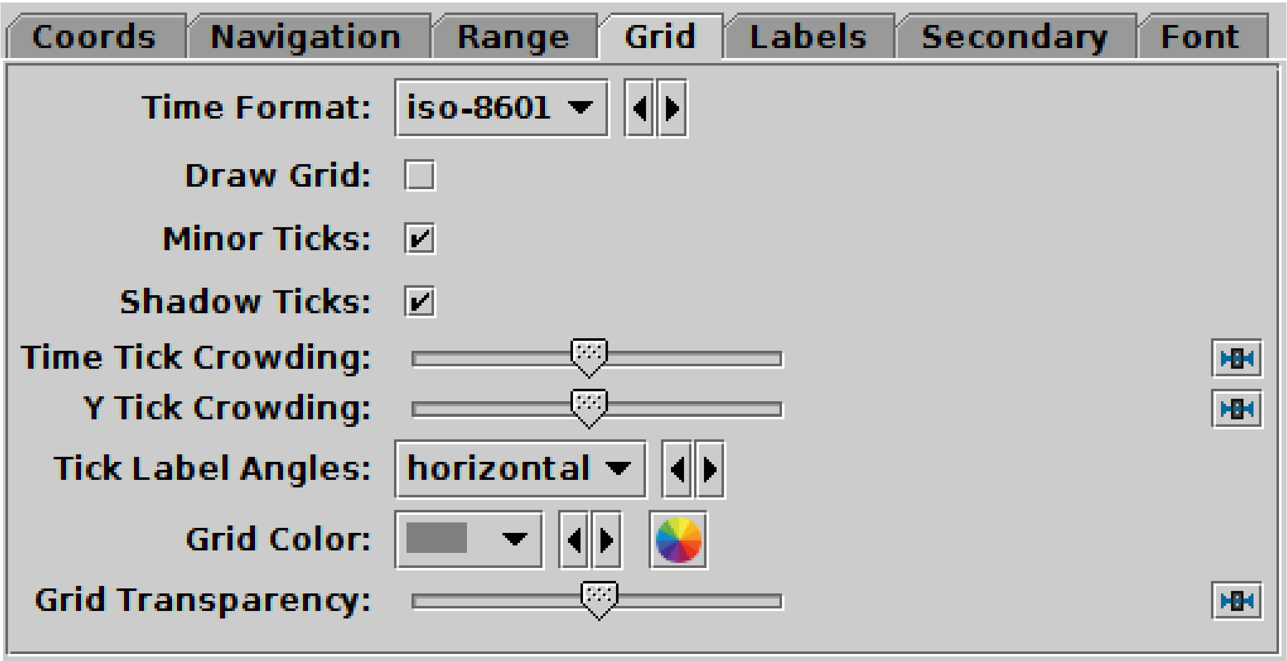
Range tab of time Axes control

The **Range** tab provides manual configuration of the visible range of the plot. Making changes to this tab will reset the visible plot range, but not vice versa - zooming and panning in the usual way will not change the settings of this panel.

Filling in the **Minimum**/**Maximum** fields for either or both axes will constrain the corresponding range of the visible data. The limits corresponding to any of those fields that are left blank will initially be worked out from the data. The **Subrange** double-sliders restrict the ranges within the (explicit or automatic) min/max ranges. Note you can move both sliders at once by grabbing a position between the two. For the time axis, the range may be entered as an ISO-8601 date/time value.

The **Clear** button resets all the fields.

Grid Tab



Time Format: iso-8601

Draw Grid: □

Minor Ticks: ☑

Shadow Ticks: ☑

Time Tick Crowding:

Y Tick Crowding:

Tick Label Angles: horizontal

Grid Color: ▨

Grid Transparency:

Grid tab of time Axes control

The **Grid** tab configures the appearance of the axis grid. It has the following options:

Time Format
: Selects the representation for time/date in which the horizontal axis is labelled. Options are ISO-8601, decimal year, Modified Julian Day, and Unix (seconds since midnight on 1 Jan 1970).

Draw Grid
: If true, grid lines will be drawn across the plot for every tick mark.

Minor Ticks
: If set, minor (unlabelled) tick marks will be drawn between the major (labelled) ones.

Shadow Ticks
: If set and no secondary axis is in use, then tick marks without numeric labels are painted along the axis opposite to the primary axis, so that tick marks are visible along all edges not just the ones with numeric labels. If a secondary axis is in use, this setting is ignored.

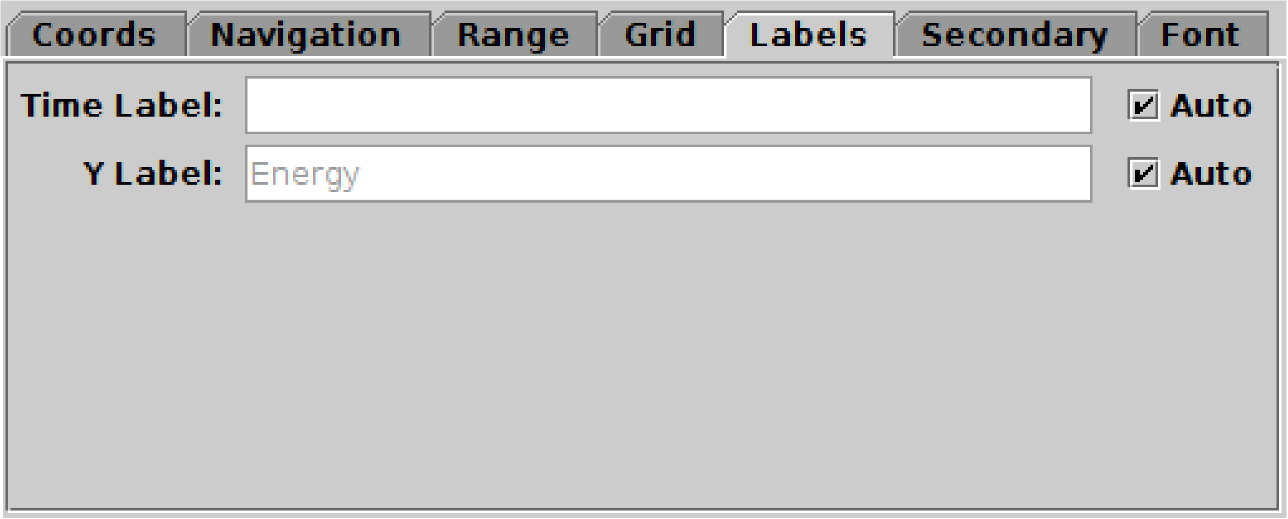
Time/Y Tick Crowding
: Use the slider to influence how many tick marks are drawn on each axis.

Tick Label Angles
: Controls orientation of numeric labels on the axes. By default they are drawn with **horizontal** alignment, but you can also choose **angled**. If **adaptive** is selected, they will be horizontal where possible, but may be angled to accommodate more labels if crowding is high; note this option is currently not perfect and can result in suboptimal border placement.

Grid Colour
: Selects the colour with which grid lines will be drawn.

Grid Transparency
: Controls the transparency of the grid lines, which are drawn over the plot content.

Labels Tab



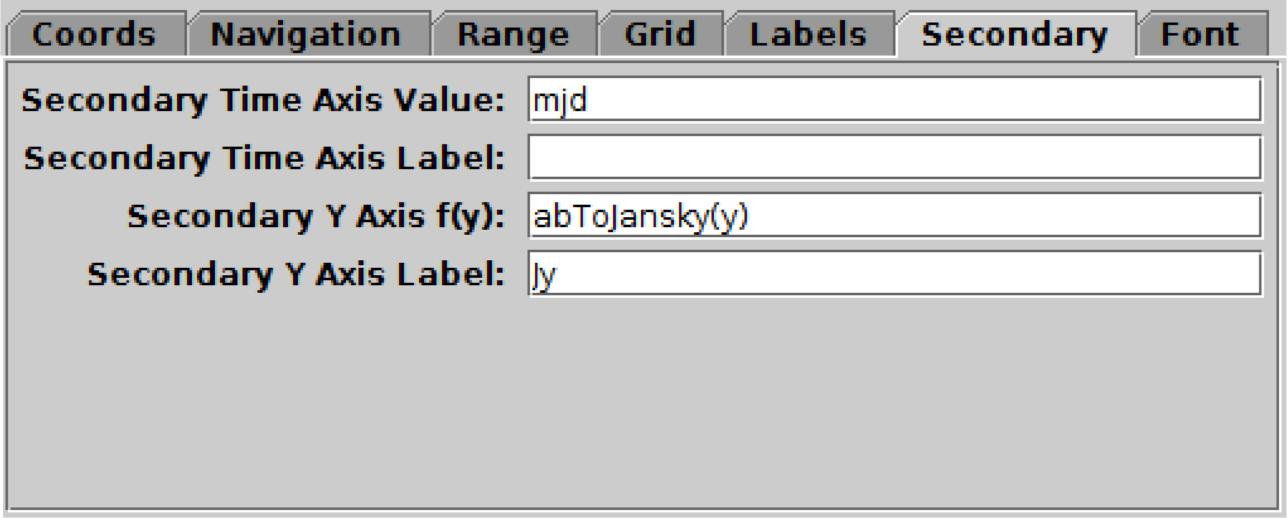
Time Label: ☑ Auto

Y Label: Energy ☑ Auto

Labels tab of time Axes control

The **Labels** tab controls the text labels on the horizontal and vertical axes. If the **Auto** checkboxes are set, the Time axis will be unlabelled, and the Y axis label will be taken from one of the data coordinates being plotted on the Y axis. To override those with your own axis labels, unset Auto and type text in to the field.

Secondary Axes Tab



Secondary Time Axis Value: mjd

Secondary Time Axis Label:

Secondary Y Axis f(y): abToJansky(y)

Secondary Y Axis Label: Jy

Secondary tab of time Axes control

The **Secondary** tab controls optional secondary Time and Y axes at the top and right edges of the plot, to go with the standard (primary) Time and Y axes at the bottom and left edges, so for instance you can annotate the vertical dimension with both magnitudes and flux, or the horizontal dimension with both decimal year and MJD.

Secondary Time Axis Value

Defines the secondary axis in relation to the time value displayed on the primary one. The value you enter is an algebraic expression using one of the following variables:

- `mjd`: Modified Julian Date
- `jd`: Julian Day
- `decYear`: decimal year CE
- `unixSec`: seconds since 1970-01-01T00:00:00

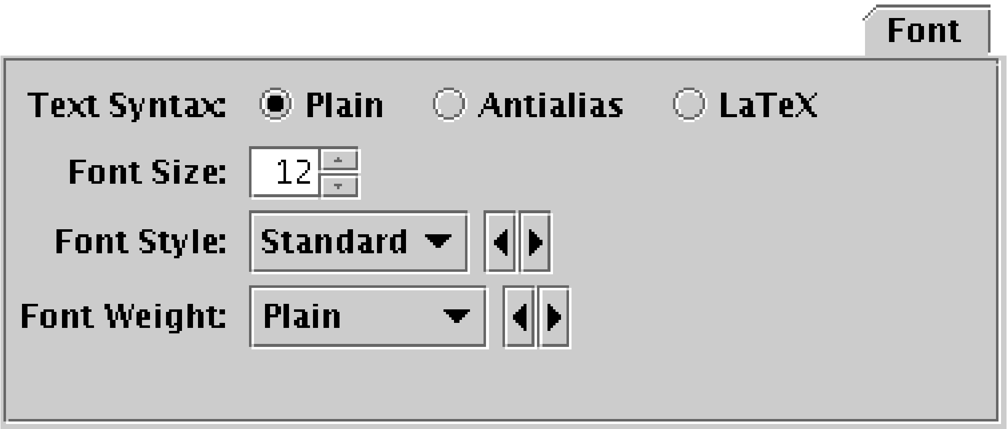
In most cases, you will just use one of these strings, e.g. "mjd" to label using MJD, but you can apply operations to these values in the usual way if required, for instance to provide a differently offset date scale. The function supplied should be monotonic and reasonably well-behaved, otherwise the secondary axis annotation may not work well. Tick marks will always be applied on a linear scale. Currently there is no way to annotate the secondary axis with ISO-8601 dates or other non-numeric labels.

Secondary Time Axis Label
Provides a textual annotation near the secondary Time axis (at the top). This can be supplied whether or not the Time axis mapping function is actually present.

Secondary Y Axis f(y)
Defines the secondary Y axis in relation to the primary one by means of a supplied function of the variable **y** that maps primary to secondary axis values, written using TOPCAT's expression language. The function supplied should be monotonic and reasonably well-behaved, otherwise the secondary axis annotation may not work well. TOPCAT will attempt to make a sensible decision about whether to use linear or logarithmic tick marks.

Secondary Y Axis Label
Provides a textual annotation near the secondary Y axis (on the right). This can be supplied whether or not the Y axis mapping function is actually present.

Font Tab



Font

Text Syntax: ● Plain ○ Antialias ○ LaTeX

Font Size: 12

Font Style: Standard

Font Weight: Plain

Font tab

The **Font** tab configures the font used for axis annotation. It also affects some other things like the legend.

Text Syntax
How to turn the text into characters on the screen. **Plain** and **Antialias** both take the text at face value, but Antialias smooths the characters. Antialiased text usually looks nicer, but can be perceptibly slower to plot. At time of writing, on MacOS antialiased text seems to be required to stop the writing coming out upside-down for non-horizontal text. **LaTeX** interprets the text as LaTeX source code and typesets it accordingly.

Font Size
Size of the font in points.

Font Style
Style of the font.

Font Weight
Whether the font is plain, bold or italic.

A.5 Old-Style Plot Windows

This section describes the old-style plotting windows used as standard in TOPCAT versions 2 and 3. Since version 4 (2013), the visualisation has been rewritten, and the new standard plotting windows are described in Appendix A.4. The windows described in this section are mildly deprecated, and will not be developed further. Most of their capabilities are handled better by the new-style plotting windows. However, these ones are still functional and if you want to use them you can find them in the ***Graphics*** *menu of the Control Window, below the new-style plot options.*

Common features of the old-style plotting windows are described in the first subsection below; the specific windows themselves are described in the later subsections.

A.5.1 Common Features

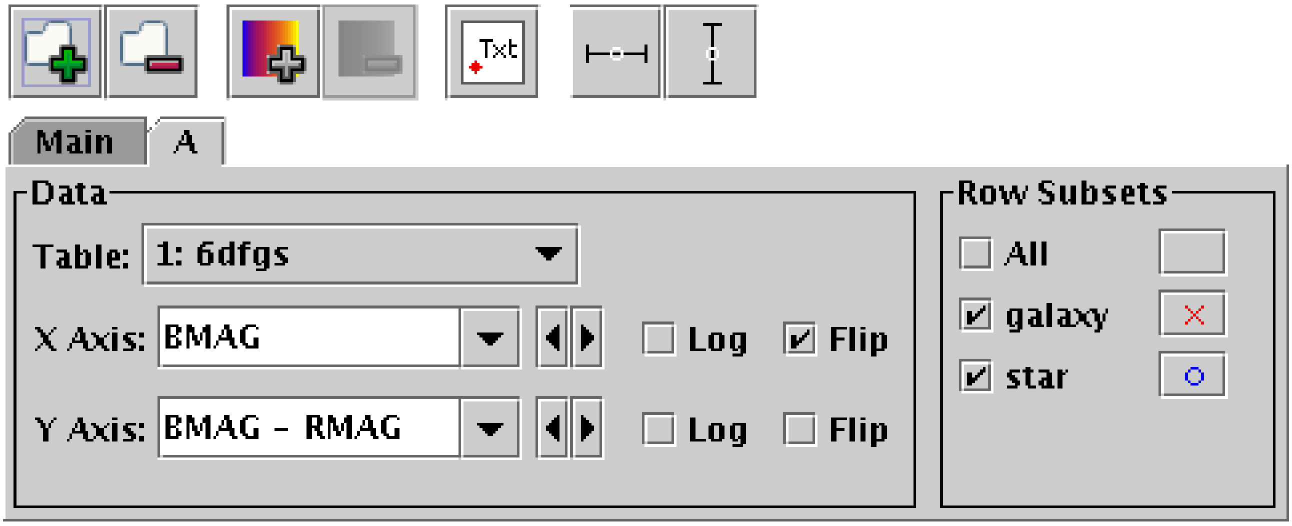
The various types of old-style plotting windows have different characteristics to fulfil their different functions, but they share a common way of doing things. Each window contains a number of controls including toolbar buttons, menu items, column selectors and others. In general any change that you make to any of the controls will cause the plot to be redrawn to take account of that change. If this requires a read or re-read of the data, a progress bar at the bottom of the window may show this progressing - except for very large tables it is usually pretty fast.

Each of the graphics windows is displayed by clicking its menu item in the **Graphics** menu of the Control Window. If one of the tables in the list is selected at the time (the Current Table) the new plot window will initially be displayed showing data from some of its columns (generally the first few numeric columns) by way of illustration. You will usually want to change the controls so it displays the quantities you are interested in.

The following subsections describe some of the features which work the same for most or all of the old-style graphics windows.

A.5.1.1 Dataset Selectors

All the old-style graphics windows provide one or more axes on which to plot - the histogram has 1, the 2d scatter and density plots have 2, the 3d scatter plot has 3 and the spherical plot has 2 or 3. In each case you select one or more dataset to plot on these axes, and select what plotting style to use for each set. A dataset is typically a number of columns from a table (the number matching the dimensionality of the plot) and a selection of row subsets associated with that table. You select this and the plotting style(s) using the panel at the bottom of each plot window. Here is dataset selector for the 2d scatter plot:



Data

Table: 1: 6dfgs

X Axis: BMAG □ Log ☑ Flip

Y Axis: BMAG - RMAG □ Log □ Flip

Row Subsets

□ All

☑ galaxy ×

☑ star ○

Default dataset selector from 2d scatter plot window

The different parts of this control work as follows:

Data panel

The **Table** selector gives the identifier of the table (one of the ones loaded into TOPCAT) that the data comes from.

The **Axis** selectors (here **X Axis** and **Y Axis**) give the quantities to be plotted. If you click the little down arrow at the right of each selector you get a list of all the numeric columns in the chosen table, from which you can select one. If you click the little left and right arrows to the right of the selector it will cycle through all the columns in the table. However, if you prefer you can type in an expression to use here. This may be more convenient if there's a very long list of columns (another way to deal with this is to hide most of the columns using the Column Window). However, what you type in doesn't have to be a column name, it can be an algebraic expression based on columns in the table, or even a constant. In the example, the X axis is a straight column name, and the Y axis is an expression. The expression language syntax is explained in Section 7.

The **Log** checkbox for each axis is used to select whether the scale should be logarithmic or linear.

The **Flip** checkbox for each axis is used to select whether the axis values increase in the conventional direction (left to right, bottom to top) or its opposite.

Some of the buttons in the toolbar shown will modify what is visible in this panel, for instance inserting new selectors to allow selection of error values. All the selectors work in the same way however.

Row Subsets panel

This defines which row subsets for will be plotted in this window, and what plotting style should be used for them. In this case there are three defined subsets, All, galaxy and star. The checkboxes on the left indicate which ones will be displayed - here, only the latter two. Sets of points are generally plotted in the order they are selected for viewing; since points plotted afterwards can obscure ones plotted before ("underneath") this makes a difference. If you want to see a set of points without it being obscured by other ones in the plot, then deselect it and reselect it again (clicking twice in the corresponding checkbox), and this will ensure that its points are plotted on top.

The buttons to the right of each subset name show the symbol that is used in the plot to display the data from that subset, in this case a red cross and a blue circle. These are selected automatically when the subset is first selected for viewing (the initial default style set depends mainly on how many rows there are in the selected table - many rows gives small dots, few gives big ones). However, you have a lot of freedom to configure how they appear. If you click the button with the symbol on it a dialogue will pop up which allows you to select colour, shape, transparency and so on, as well as error bar style if appropriate and things like whether fitted lines will be plotted for that subset. The options available differ according to the kind of plot, and are described along with the different graphics windows in the following subsections. The style window stays visible until you dismiss it, but if you click on another of the buttons in the Row Subsets panel its contents will change to allow you to select values for the corresponding subset. Most graphics windows have a **Marker Style** menu. This allows you to change all the styles in the plot at once to be members of a uniform set, for instance different coloured pixels, or black open shapes. If you select one of these it will overwrite the existing style selections, but they can be edited individually afterwards.

Dataset Tool Bar

The toolbar shown above the data panel in the figure contains buttons which affect the dataset selector itself. The first two buttons add and remove dataset **Tabs** (see below) and are present for all plots. The other items configure optional selectors appearing in the **Data Panel** - the ones shown here are concerned with Auxiliary Axes (Appendix A.5.1.5), Point Labels (Appendix A.5.1.4) and Error Bars (Appendix A.5.1.3), but not all the types of plot have exactly the same ones.

Tabs

The example shows two tabs: **Main** and **A**; the currently visible one is A. You can select a tab by clicking on its name. In each tab you select a table and a set of columns/expressions, and if they are all filled in it will contribute points (or bars, or whatever) to the plot. The Main dataset determines the initial values for the axis labels, but the data comes equally from all of them. The numerical values of the coordinates are treated the same for all the datasets, but their meanings might be different, for instance one dataset may be plotting V magnitude against ellipticity and another plotting B magnitude against ellipticity.

The **Add Dataset** () and **Remove Dataset** () buttons in the toolbar add a new tab or remove the selected one respectively. Initially only the Main tab is present, and this one cannot be removed.

Sometimes (high-dimensional plots, auxiliary axes, error bars) a lot of information needs to be entered into the data panel, and the bottom part of the window can get quite large. Normally, the plot in the upper part of the window shrinks to accommodate it. You can of course resize the window to gain more space, but if your screen is small you may still end up with an uncomfortably small plot. If this happens, you can use the following button from the main toolbar:

Split Window

When this toggle button is on, the dataset selector can be resized by dragging the bar between it and the plot itself up or down. If there is insufficient space for all the components in the selector, a scrollbar will appear. When it is off (the default), the full height of the selector will be visible, and the plot will shrink to accommodate it.

A.5.1.2 Axis Configuration and Zooming

In general terms the axes on which the graphics are plotted are defined by the datasets you have selected. The axis labels are set from the column names or expressions chosen for the **Main** dataset, and the ranges are determined so that all the data from the chosen datasets can be seen. However, these things can be adjusted manually.

The following features are available directly from the window for configuring axis range:

X-Y Zoom

In some of the windows (2d scatter plot, histogram and density map), you can change both axis ranges by zooming in or out with the mouse on the plot surface itself. To zoom in, place the mouse at the top left of the region you want to examine, press the button, drag it to the bottom right corner, and release the button. To zoom out, drag up and left instead. A box is drawn as you drag so you can see what you're doing.

Centre Zoom

The 3d and spherical plots allow you to zoom in on the central part of the window. The 'active region' for dragging is to the left or right of the plot (the region on the right is rather thin, and does not include the width used by the legend). When the pointer is in these regions, the mouse cursor symbol should change to indicate that zooming can be done. Drag down to zoom in and up to zoom out.

An easier alternative for zooming in the 3D windows is to use the mouse wheel, if you have one: wheel forward to zoom in and backward to zoom out.

Axis Zoom

In some of the windows (2d scatter plot, histogram, density map, stacked lines) you can modify the range on each axis independently by dragging the mouse over where the axis is drawn. The 'active region' for dragging is just below the X axis and just to the left of the Y axis, in the region where the numeric and text labels are written. When the pointer is in one of these regions, the mouse cursor symbol should change to indicate that zooming can be done. As for the X-Y Zoom, drag left-to-right or up-to-down to zoom in and right-to-left or down-to-up to zoom out.

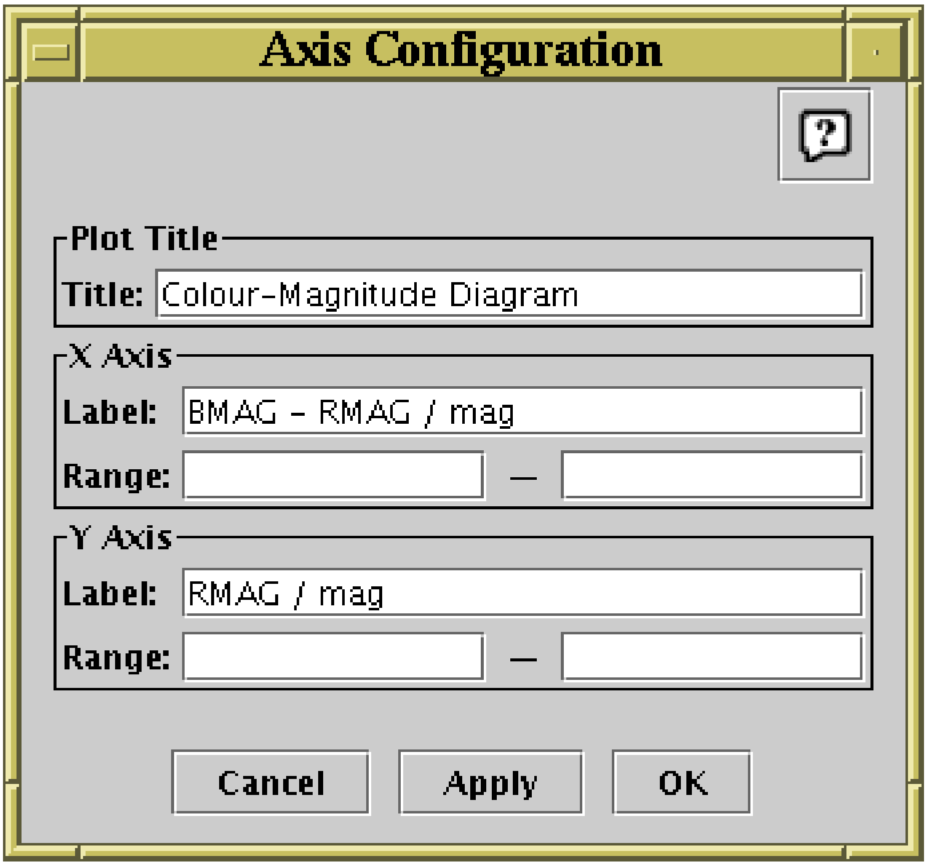
Auxiliary (Colour) Axis Zoom

When Auxilary Axes are in use, you can zoom in and out of them by dragging up and down on the colour bar to the right of the plot, in the same way as for a normal Axis Zoom above.

Rescale

If you find you're zoomed to a region you don't want to be in, you can use the Rescale toolbar button to return to the default scale (full coverage). Note this affects any auxiliary axes as well as the spatial ones. Some windows may have per-axis rescale buttons too (↔, ↕).

For more control over axis range and labelling, use the **Configure Axes and Title** (TITLE) toolbar button, which will pop up a dialogue like the following:



Plot Title

Title: Colour-Magnitude Diagram

X Axis

Label: BMAG - RMAG / mag

Range: —

Y Axis

Label: RMAG / mag

Range: —

Axis Configuration Dialogue for 2-d axes

You can fill in these values for each axis as follows:

Label

For each axis the label box contains the text used to annotate the axis in the plot. By default this is the same as the text in the **Main** dataset column selector (usually a column name), followed by the units if known. However, you can change it by typing whatever text you like.

Range
The range boxes allow you to specify the lower and upper limits of each axis. By default these are blank, meaning that the plot will size its axes so that all the data can be seen. However, if you fill in one or both of the boxes with a suitable numeric value, the lower/upper bound will be fixed at that. Note that the lower bound (left box) must be numerically less than the upper bound (right box).

Both values are reset if the plot's axis is changed (a new column or expression is selected for the Main dataset), or if the range is reset in some other way (e.g. by zooming).

The plot title may also be set in the **Plot Title** panel of this window:

Title
Any text entered here will be displayed at the top of the plot to provide a title.

A.5.1.3 Error Bars

TOPCAT provides quite flexible graphical representation of symmetric or asymmetric errors in 1, 2 and 3 dimensions. The plots with error bar support are the 2D, 3D and spherical scatter plots and the stacked lines plot.

By default, error bar drawing is switched off. The simplest way to activate it is to use the relevant error bar button(s) in the data selector tool bar (the one below the plot). For the Cartesian (2D, 3D, lines) plots, some or all of the following buttons are present:

◘ **X symmetric errors**

◘ **Y symmetric errors**

◘ **Z symmetric errors**

Any combination of them can be active at once. Clicking one of these buttons toggles symmetric error bar drawing for that axis on or off. When it is on, an additional column selector will appear to the right of the main column selector for the axis in question. If you fill this in with a column name or expression which gives the error for that axis, then the error bars will be plotted. TOPCAT may make a guess based on column names and UCDs about which columns provide error values for which other columns, so the error selector may get filled in automatically. However, in most cases you will need to provide the error values by selecting a column yourself, and occasionally you may need to correct TOPCAT's guesses.

Here is a 2D plot in which symmetric X and asymmetric Y errors are being used:

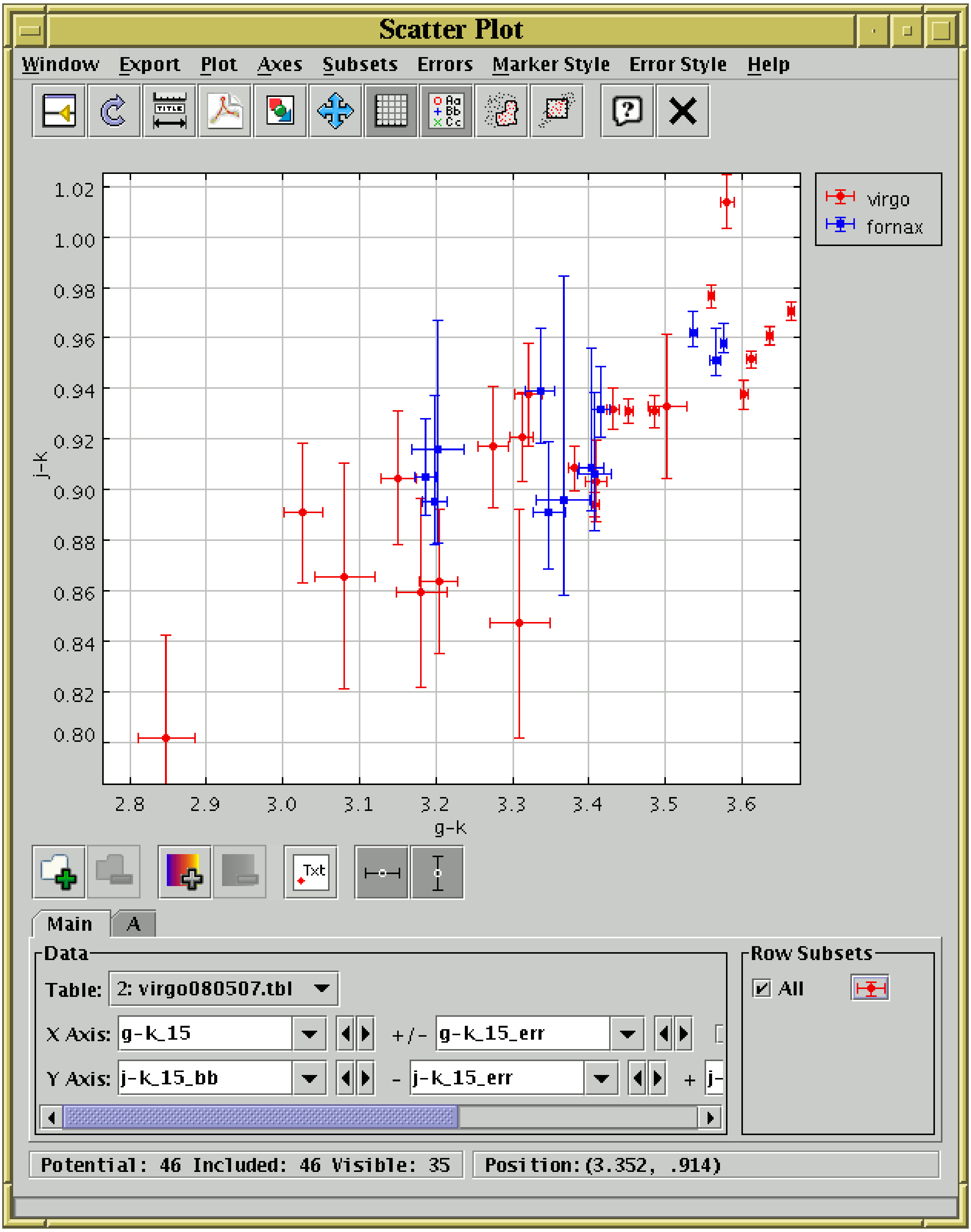


Table: 2: virgo080507.tbl

X Axis: g-k_15 +/- g-k_15_err

Y Axis: j-k_15_bb - j-k_15_err + j-

☑ All

Potential: 46 Included: 46 Visible: 35

Position: (3.352, .914)

Plot window with symmetric X and asymmetric Y errors

You can see that with the error column selector, the panel has become too wide for the window so a scrollbar has appeared at the bottom - you can scroll this left and right or enlarge the window to see the parts that you need to.

For the spherical plot the following error toggle buttons are present:

- **tangential isotropic errors**
- **radial symmetric errors**

These work in a similar way to the Cartesian erors above, except that the tangential one adds a single column selector, with an associated unit selector, near the latitude and longitude selectors to determine the isotropic angular size of error small circles.

If you want to use **asymmetric** or one-sided errors, use the options in the **Errors** menu instead of the toolbar buttons. For instance the options for X axis error bars in the 2D scatter plot are:

- **None**
- **Symmetric**
- **Lower Only**
- **Upper Only**
- **Lower & Upper**

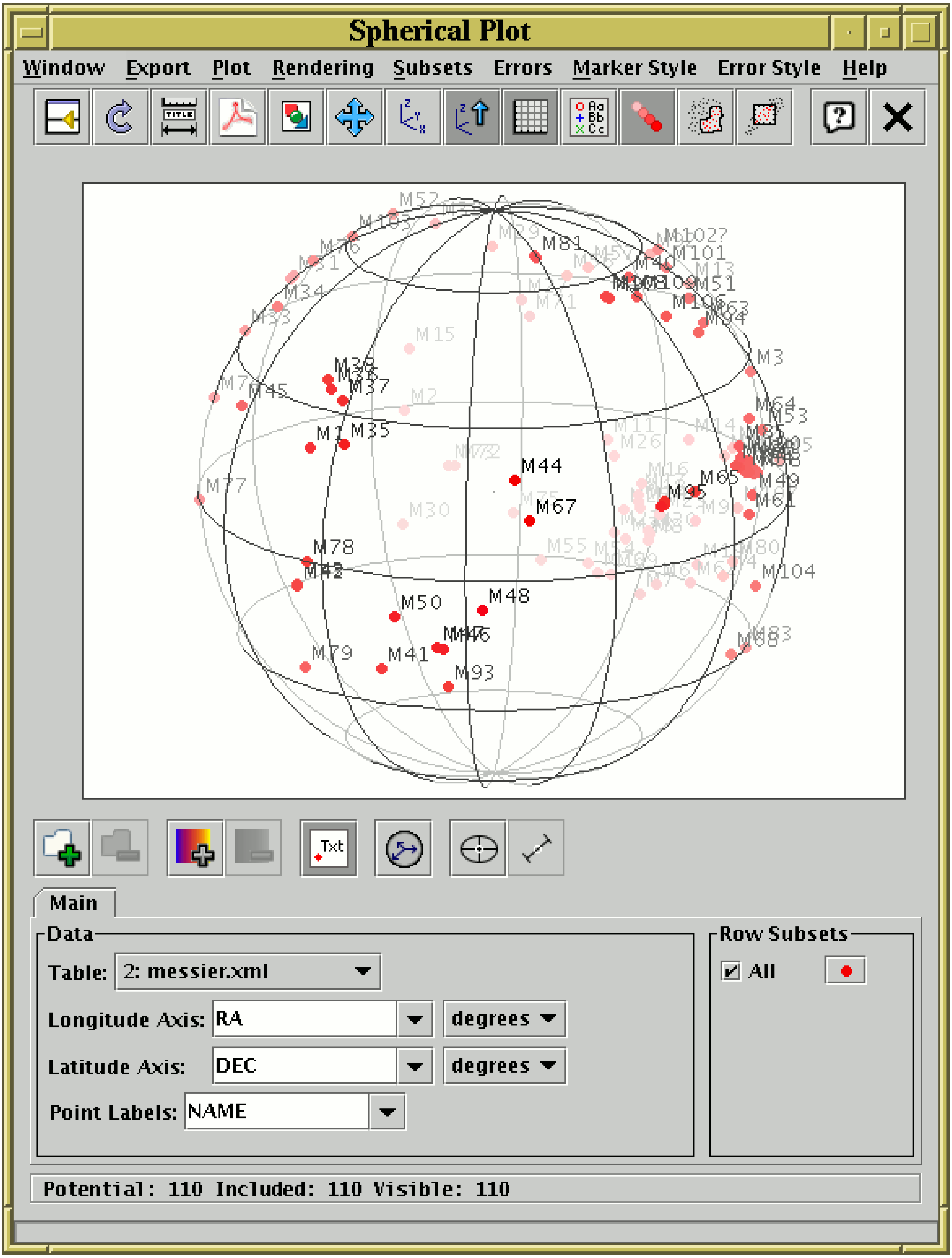
These give you different column selector boxes, but work in much the same way as the symmetric ones.

There are many options for the plotting style of one, two and three dimensional error bars, including capped and uncapped bars, crosshairs, ellipses and rectangles. This plotting style is controlled from the plot window's **Style Editor** window (see e.g. Appendix A.5.3.1), which can be viewed by clicking on the marker icon in the Row Subsets panel at the bottom right of the window. The available error bar styles will depend on which axes currently have errors; if none do, then the error bar selector will be disabled. You can also use the **Error Style** menu to change the error style for all the visible datasets at once.

A.5.1.4 Point Labels

On the 2-d and 3-d scatter plots you can write text labels adjacent to plotted points. To do this click the **Draw Labels** (Txt) button in the dataset toolbar (below the plotting area in the plot window).

This will reveal a new **Point Labels** selector below the existing spatial ones. Using this you can select any of the table columns (not just the numeric ones as for the other selectors), or give a string or numeric expression involving them. When this selector is filled in, every point in the dataset which has a non-blank value for this quantity will have it written next to the point on the display.



Data

Table: 2: messier.xml

Longitude Axis: RA degrees

Latitude Axis: DEC degrees

Point Labels: NAME

Row Subsets

☑ All ●

Potential: 110 Included: 110 Visible: 110

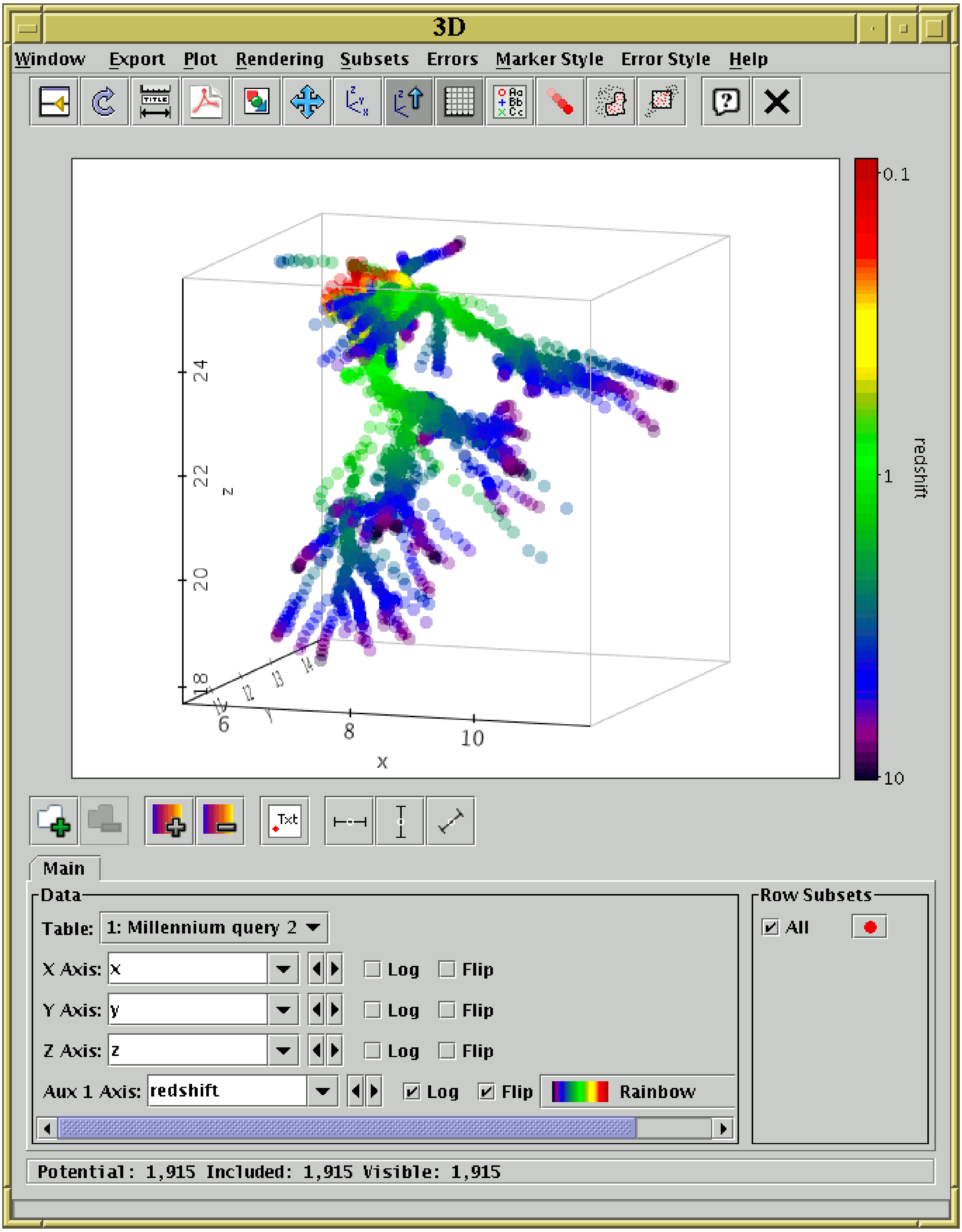
Point Labelling for Messier objects in the spherical plot

In this example the NAME column has been selected, so that each point plotted (in this case all the Messier objects) is labelled with its name. As you can see, where many labels are plotted near to each other they can get in each others' way. In some cases TOPCAT will omit plotting labels in crowded regions, in others not - but in any case if you have labels too tightly grouped they are unlikely to be legible.

A.5.1.5 Auxiliary Axes

TOPCAT can plot data in one, two or three spatial dimensions, but sometimes the the data which you need to visualise is of higher dimensionality. For this purpose, some of the plotting windows (2D and 3D scatter plots) allow you to control the colouring of plotted points according to values from one or more additional columns (or calculated expressions), which gives you more visual information about the data you are examining.

To use this facility, click the **Add auxiliary axis** ( ) button in the dataset toolbar (below the plot area in a plot window). A new axis selector will appear below the existing spatial ones, labelled **Aux 1 Axis**. It has log and flip checkboxes like the spatial axes, and to the right (you may need to widen the window or use the scrollbar at the bottom to see it) is a selector depicting a number of colourmaps to choose from - the default one resembling a rainbow is usually quite suitable, but you can pick others. If you enter a column name or expression into the selector, each plotted point will be coloured according to the value of that quantity in the corresponding row of data. If that quantity is null for a row, the corresponding point will not be plotted. A scale on the right of the plot indicates how the colour map corresponds to numeric values. To remove the auxiliary axis and go back to normally-coloured points, simply click the **Remove auxiliary axis** ( ) button.



Data

Table: 1: Millennium query 2

X Axis: x □ Log □ Flip

Y Axis: y □ Log □ Flip

Z Axis: z □ Log □ Flip

Aux 1 Axis: redshift ☑ Log ☑ Flip Rainbow

Row Subsets

☑ All ●

Potential: 1,915 Included: 1,915 Visible: 1,915

3D plot of simulation data showing X, Y, Z spatial position with the auxiliary axis indicating timestep.

There are two types of colour maps you can choose from: **colour fixing** and **colour modifying**. The fixing ones are easiest to understand: the original colour of the point (as drawn in the legend) is ignored, and it is coloured according to the relevant value on the selected auxiliary axis. The colour modifying maps take the original colour and affect it somehow, for instance by changing its transparency or its blue component. These are marked with an asterisk ("*") in the colour map selector. They can be used to convey more information but are often harder to interpret visually - for one thing the shading of the colour bar in the legend will not correspond exactly to the colours of the plotted points.

By using modifying colour maps it is possible to perform plots with more than one auxiliary axis - typically the first one will be a fixing map and subsequent ones will be modifying. So the first auxiliary axis could have the (fixing) Rainbow map, and the second could have the (modifying) Transparency map. The colour alterations are applied in order. It is possible, but pointless, to have multiple fixing maps applied to the same points - the last-numbered one will determine the colour and earlier ones will get ignored. Multiple aux axes can be obtained by clicking the **Add auxiliary axis** button more than once. When combining several maps some thought has to be given to which ones to use - some good combinations are the three RGB ones or the three YUV ones.

A fairly wide range of colour maps of both kinds is provided by default. If these do not suit your needs, it is possible to provide your own custom colour fixing maps using the **lut.files** system property - see Section 10.2.3.

It is easy to generate attractive screenshots using auxiliary axes. Making visual sense of the results is a different matter. One visualisation expert tried to dissuade their introduction in TOPCAT on the grounds that the graphics they produce are too hard for humans to interpret - I hope that these plots can assist with some analysis, but it is a somewhat experimental feature which may or may not end up being widely useful. The maximum number of auxiliary axes which can be used together is currently three. This could be increased on request, but if you feel you can generate an intelligible plot using more than this then you're considerably smarter than me.

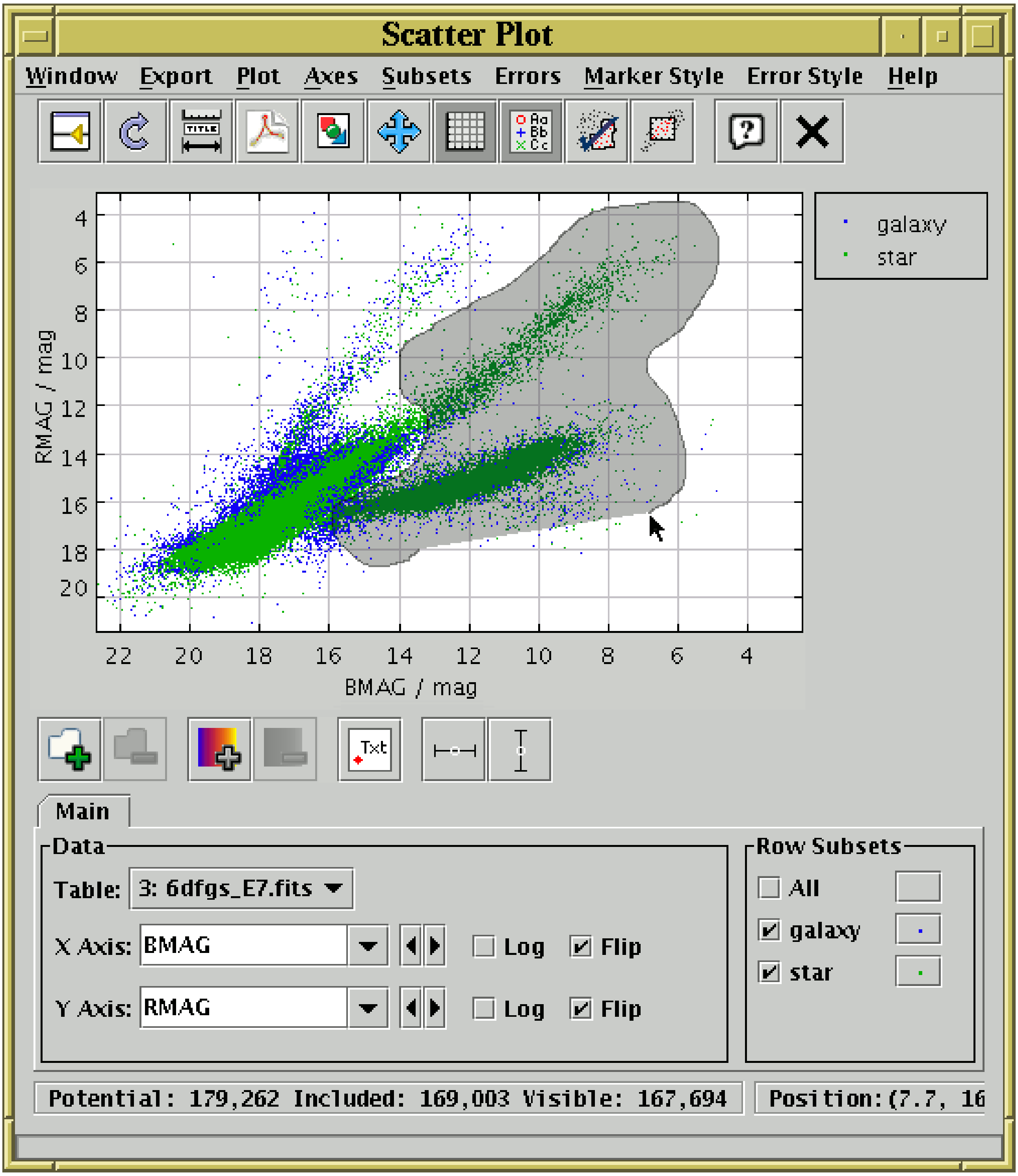
A.5.1.6 Defining Subsets by Region

When quantities are plotted in one of the graphics windows it becomes easy to see groupings of the data which might not otherwise be apparent; a cluster of (X,Y) points representing a group of rows may correspond to a physically meaningful grouping of objects which you would like to treat separately elsewhere in the program, for instance by calculating statistics on just these rows, writing them out to a new table, or plotting them in a different colour on graphs with different coordinates. This is easily accomplished by creating a new Row Subset containing the grouped points, and the graphics windows provide ways of doing this.

In some of the plots (Histogram 2d Scatter plot Density map and Spherical plot) you can set the axis ranges (either manually or by zooming with the mouse - see Appendix A.5.1.2) so that only the points you want to identify are visible, and then click the **Subset From Visible** toolbar button (the icon is , or depending on the plot type). This defines a subset consisting of all the points that are visible on the current plot. This is only useful if the group you are interested in corresponds to a rectangular region in the plotting space.

A more flexible way is to draw a region or regions on the plot which identify the points you are interested in. To do this, hit the **Draw Subset Region** () toolbar button. Having done this, you can drag the mouse around on the plot (keep the left mouse button down while you move) to encircle the points that you're interested in. As you do so, a translucent grey blob will be left behind - anything inside the blob will end up in the subset. You can draw one or many blobs, which may be overlapping or not. If you make a mistake while drawing a sequence of blobs, you can click the right mouse button, and the most recently added blob will disappear. When you're in this region-drawing mode, you can't zoom or resize the window or change the characteristics of the plot, and the **Draw Subset Region** button appears with a tick over it () to remind you you're in it.

Here's what the plot looks like while you're drawing:



Data

Table: 3: 6dfgs_E7.fits

X Axis: BMAG ☐ Log ☑ Flip

Y Axis: RMAG ☐ Log ☑ Flip

Row Subsets

☐ All

☑ galaxy

☑ star

Potential: 179,262 Included: 169,003 Visible: 167,694

Position:(7.7, 16

Region-Drawing Mode

When you're happy with the region you've defined, click the toolbar button again.

In either case, when you have indicated that you want to define a new row subset, a dialogue box will pop up to ask you its name. As described in Section 3.1.1, it's a good idea to use a name which is just composed of letters, numbers and underscores. You can optionally select a subset name which has been used before from the list, which will overwrite the former contents of that subset. When you enter a name and hit the **OK** button, the new subset will be created and the points in it will be shown straight away on the plot using a new symbol. As usual, you can toggle whether the points in this subset are displayed using the **Row Subsets** box at the bottom of the Plot Window.

A.5.1.7 Exporting Graphics

All the old-style graphics windows have the following export options in the toolbar:

Export as PDF

Export as GIF

and additionally, the **Export** menu contains:

Export as Encapsulated PostScript

Export as Gzipped Encapsulated PostScript

Export as JPEG

Export as PNG

These can be used to export the currently visible plot to an external graphics format for later use.

Exporting to the pixel-based formats (GIF, JPEG, PNG) is fairly straightforward: each pixel on the screen appears as one pixel in the output file. PNG is the most capable, but it is not supported by all image viewers. GIF works well in most cases, but if there are more than 255 colours some of the colour resolution will be lost. JPEG can preserve a wide range of colours, but does not support transparency and is lossy, so close inspection of image features will reveal blurring.

When exporting to Portable Document Format (PDF), Scalable Vector Graphics (SVG) or Encapsulated PostScript (EPS), which are vector graphics formats, there are a few things to consider:

Positional Quantisation

The output file will render text, lines and markers properly, with smooth edges rather than steps where pixel boundaries would be on the screen. However, the *positional* resolution is the same as it would be on the screen, so if you have a 400-pixel high plot for instance, there are only 400 possible Y coordinates at which a marker can be plotted. In general this is not obvious by looking at the output plot, but you may find it helpful to increase the size of the plot on the screen by resizing the window before performing an export to EPS. This reduces the effect of the positional quantisation, but note it will also have the effect of making the text labels proportionally smaller to the graphics.

Transparency

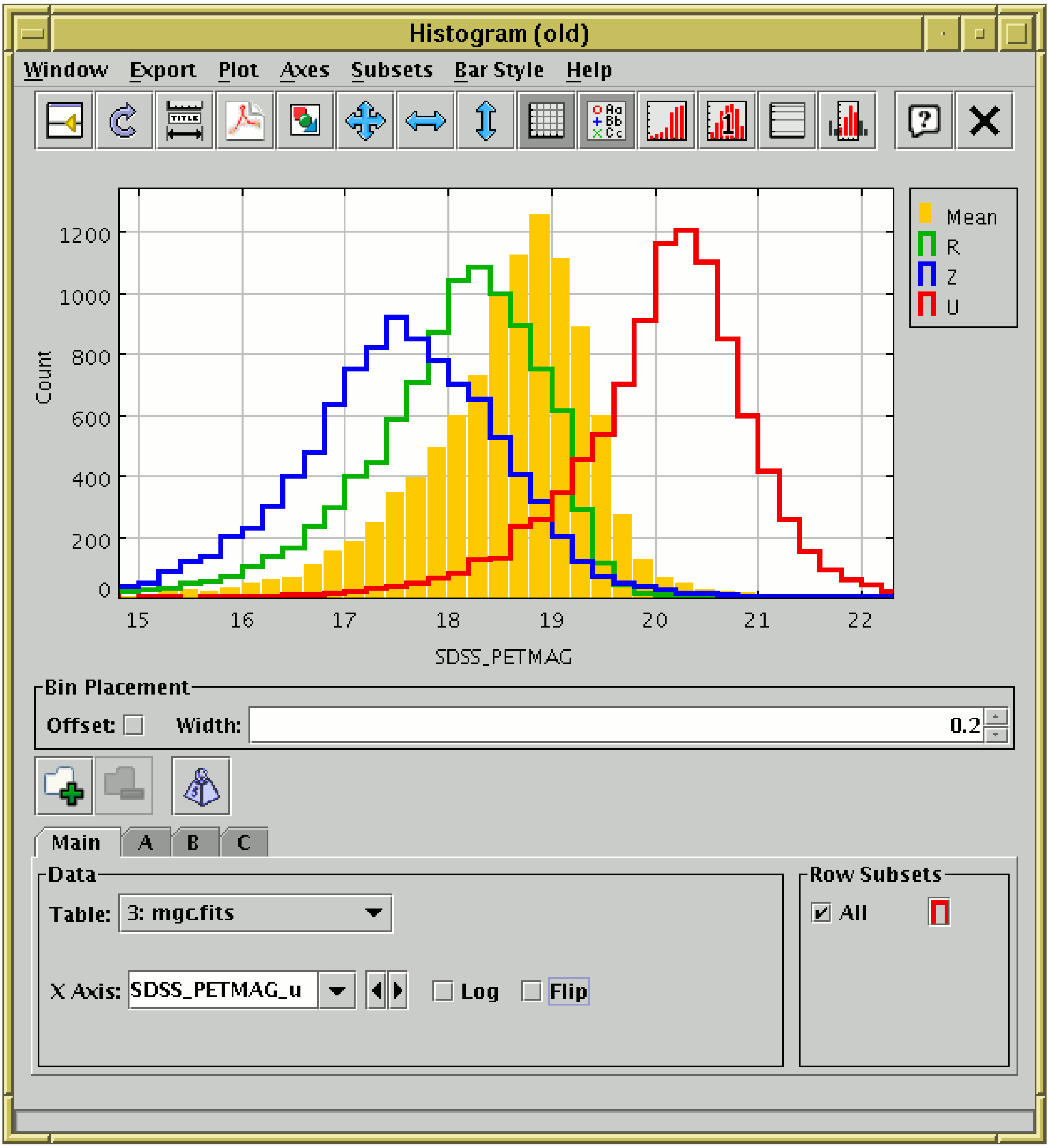
For technical reasons transparent markers cannot easily be rendered when a plot is exported to PostScript. In some cases the plot is done using a bitmap in the PostScript output to permit transparency and in some cases the points are just plotted opaque. Try it and see; if the points come out opaque instead of transparent you may need to export to GIF instead. Better workarounds may be provided in future releases.

File Size

In some cases (2D and 3D scatter plots with many thousands of points) output EPS files can get extremely large; the size scales with the number of points drawn, currently with a factor of a few hundred bytes per point. In some cases you can work round this by plotting some points as transparent so that the plot is rendered as a bitmap (see the discussion of transparency above) which scales as the number of pixels rather than the number of points. The Gzipped EPS format helps somewhat (though can be slow); PDF output is better still. Even PDF files may be unmanageably large for very many points however.

A.5.2 Histogram (old-style)

This section describes an old-style plotting window. The standard plotting windows are described in Appendix A.4.



Offset: □ Width: 0.2

Table: 3: mgcfits

X Axis: SDSS_PETMAG_u □ Log □ Flip

☑ All

Histogram Window

The histogram window lets you plot histograms of one or more columns or derived quantities. You can display it using the **Histogram** ( ) item in the Control Window's **Graphics** menu.

You select the quantity or quantities to plot using the dataset selector (Appendix A.5.1.1) at the bottom of the window. You can configure the axes, including **zooming** in and out, with the mouse (drag on the plot or the axes) or manually as described in Appendix A.5.1.2.

The **Bin Placement** box below the main plot controls where the bars are drawn. Select the horizontal range of each bar using the **Width** entry box - either type in the value you want or use the tiny up/down arrows at the right to increase/decrease the bin size. The **Offset** checkbox on the left determines where the zero point on the horizontal axis falls in relation to the bins; if the box is checked then zero (or one for logarithmic X axis) is in the centre of a bin, and if it's unchecked then zero (or one) is on a bin boundary.

The following buttons are available on the toolbar:

Split Window

Allows the dataset selector to be resized by dragging a separator between it and the plot area. Good for small screens.

Replot

Redraws the current plot. It is usually not necessary to use this button, since if you change any of the plot characteristics with the controls in this window the plot will be redrawn automatically. However if you have changed the data, e.g. by editing cells in the Data Window, the plot is not automatically redrawn (since this is potentially an expensive operation and you may not require it). Clicking this button redraws the plot taking account of any changes to the table data.

Configure Axes and Title

Pops up a dialogue to allow manual configuration of axis ranges, axis labels and plot title - see Appendix A.5.1.2.

Export as PDF

Pops up a dialogue which will write the current plot as a PDF file.

Export as GIF

Pops up a dialogue which will output the current plot to a GIF file. The output file is just the same as the plotted image that you see. Resize the plotting window before the export to control the size of the output GIF.

Rescale

Rescales the axes so that the width and height are sufficient to accommodate all the non-zero bars in the histogram for all the currently selected datasets. By default the plot will be scaled like this, but it it may have changed because of changes in the subset selection or from zooming in or out.

Rescale X

Rescales the horizontal axis to accommodate all the currently plotted bars. The vertical axis scaling is not changed.

Rescale Y

Rescales the vertical axis to accommodate all the currently plotted bars. The horizontal axis scaling is not changed.

Grid

Toggles whether a grid is drawn over the plotting surface or not.

Show Legend

Toggles whether a legend showing how each data set is represented is visible to the right of the plot. Initially the legend is shown only if more than one data set is being shown at once.

Cumulative Plot

Toggles whether the histogram should be normal or cumulative. Normally the height of each bar is determined by counting the number of points which fall into the range on the X axis that it covers. For a cumulative plot, the height is determined by counting all the points between negative infinity and the upper bound of the range on the X axis that it covers.

Normalisation

Toggles whether histograms are normalised. When selected, each dataset will be normalised so that the sum of the counts of all its bars over the whole range of data is equal to one.

Log Y Axis

Toggles whether the Y axis is scaled logarithmically or not.

Subset From Visible

Defines a new **Row Subset** consisting of only the data in the bars which are visible in the current plot. See Appendix A.5.1.6 for more explanation.

The **Dataset Toolbar** contains the following options:

/ **Add/Remove dataset**

Adds/removes tabs in which the data for extra datasets can be entered. See Appendix A.5.1.1.

Weight Counts

If this toggle button is on, an additional **Weight Axis** selector appears below the **X Axis** selector. If this is filled in with a column name or expression, then instead of simply accumulating the number of points per bin, the Y axis will show the sum over the weighting quantity for points in each bin. Not having a weight axis is equivalent to filling in its value with the quantity 1. Note that with weighting, the figure drawn is no longer strictly speaking a histogram.

When weighted, bars can be of negative height. An anomaly of the plot as currently implemented is that the Y axis never descends below zero, so any such bars are currently invisible. This may be amended in a future release (contact the author to vote for such an amendment).

The **Export** menu contains additional image export options and the following options specific to the histogram:

Save as Table

The bin counts/sums corresponding to the currently plotted histogram will be written to disk in tabular form. The first two columns give the lower and upper bounds of each bin, and the subsequent columns give the occupancies of each bin for each plotted data set. If only one dataset is plotted, there will only be three columns.

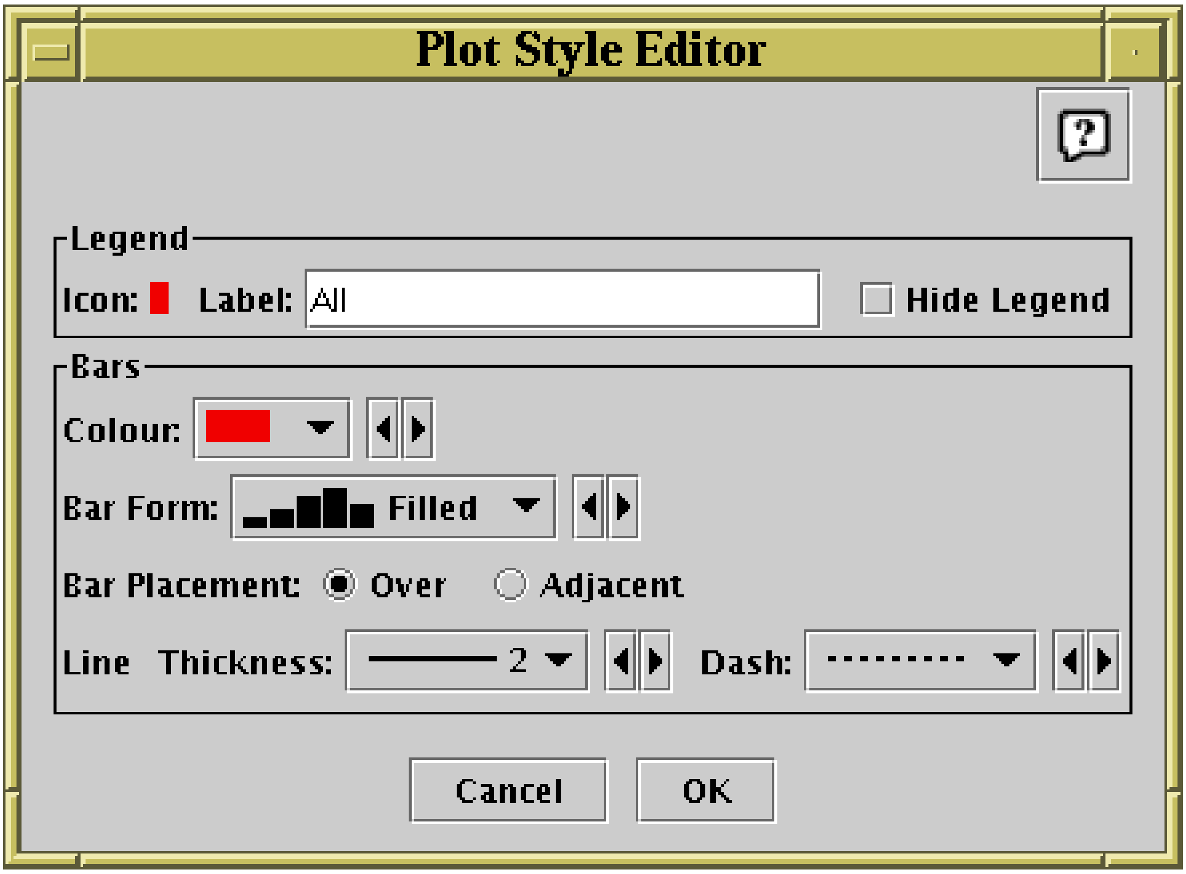
Import as Table

Assembles a table as per the **Save** option above, but rather than writing it to disk imports it directly into TOPCAT, where it can be manipulated in all the usual ways.

You have considerable freedom to configure how the bars are drawn; controlling this is described in the following subsection.

A.5.2.1 Histogram Style Editor

The bins in a histogram can be represented in many different ways. A representation of how a bar will be displayed is shown on a button to the right of the name of each visible subset, at the bottom right of the histogram window. If you click this button the following dialogue will pop up which enables you to change the appearance.



Legend

Icon: 🟥 Label: All □ Hide Legend

Bars

Colour: 🟥

Bar Form: Filled

Bar Placement: ● Over ○ Adjacent

Line Thickness: 2 Dash: ---------

Style editor dialogue for histogram bars

The **Legend** box defines how the selected set will be identified in the legend which appears alongside the plot (though the legend will only be visible if **Show Legend** (○ Aa + Bb × Cc) is on):

Icon
: Displays the icon which will be shown to identify the points in the selected set. Its appearance depends on the selections you make in the rest of this dialogue window.

Label
: Gives the name written in the legend to label the subset. By default this is derived from the Row Subset's name and, if it's not part of the main dataset, the name of the dataset's tab. You can type in a new value to change what is written in the legend.

Hide Legend
: If this checkbox is selected, then no entry for the selected set will appear in the legend.

The **Bins** panel describes the form of the bars to be plotted for each data set.

Colour
Selects the colour for the bar or line which will represent this set.

Bar Form
Selects the style of bar which will be plotted. Available styles are filled rectangles, open rectangles, stepped lines and spikes.

Bar Placement
Selects where each bar will be placed on the X axis within the ordinate region which it represents. There are currently two options: **Over** means that it covers the whole of its range, and **Adjacent** means that it covers only a proportion of its range, so that multiple datasets plotted on the same graph don't obscure each other (if 2 sets are plotted, the first one will take the left half and the second will take the right half of each bin region, and so on). In the case that there is only a single data set plotted, it doesn't matter which of these is chosen.

Line Thickness
Determines the thickness in pixels of any lines which are drawn. Only applies to those **Bar Form**s which use lines rather than solid blocks.

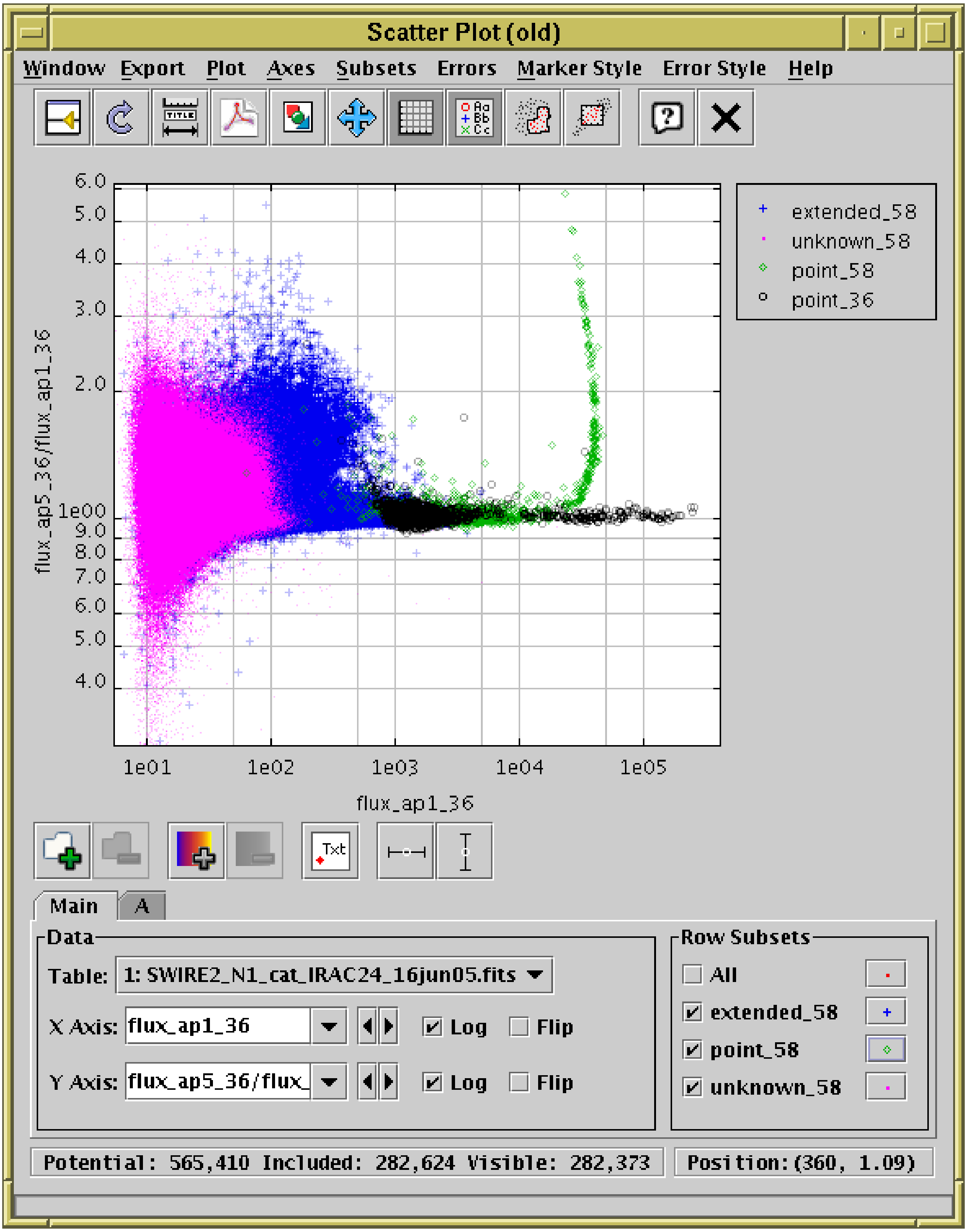
Dash
Determines the dash style (solid, dotted, dashed or dot-dashed) for any lines which are drawn. Only applies to those **Bar Form**s which use lines rather than solid blocks.

Any changes you make in this window are reflected in the plot straight away. If you click the **OK** button at the bottom, the window will disappear and the changes remain. If you click **Cancel** the window will disappear and any changes you made will be discarded.

You can also change all the plotting styles at once by using the **Bar Style** menu in the histogram window. Here you can select a standard group of styles (e.g. all filled adjacent bars with different colours) for the plotted sets.

A.5.3 2D Plot (old-style)

This section describes an old-style plotting window. The standard plotting windows are described in Appendix A.4.



Data

Table: 1: SWIRE2_N1_cat_IRAC24_16jun05.fits

X Axis: flux_ap1_36 ☑ Log □ Flip

Y Axis: flux_ap5_36/flux_ ☑ Log □ Flip

Row Subsets

□ All

☑ extended_58

☑ point_58

☑ unknown_58

Potential: 565,410 Included: 282,624 Visible: 282,373

Position: (360, 1.09)

Plot Window

The plot window allows you to do 2-dimensional scatter plots of one or more pair of table columns (or derived quantities). You can display it using the **Plot** () item in the Control Window's **Graphics** menu.

On the plotting surface a marker is plotted for each row in the selected dataset(s) at a position determined by the values in the table columns selected to provide the X and Y values. A marker will only be plotted if both the X and Y values are not blank. Select the quantities to plot and the plotting symbols with the dataset selector (Appendix A.5.1.1) at the bottom. You can configure the axes, including **zooming** in and out, with the mouse (drag on the plot or the axes) or manually as described in Appendix A.5.1.2.

Clicking on any of the plotted points will **activate** it - see Section 8.

The following buttons are available on the toolbar:

Split Window

Allows the dataset selector to be resized by dragging a separator between it and the plot area. Good for small screens.

Replot

Redraws the current plot. It is usually not necessary to use this button, since if you change any of the plot characteristics with the controls in this window the plot will be redrawn automatically. However if you have changed the data, e.g. by editing cells in the Data Window, the plot is not automatically redrawn (since this is potentially an expensive operation and you may not require it). Clicking this button redraws the plot taking account of any changes to the table data.

Configure Axes and Title

Pops up a dialogue to allow manual configuration of axis ranges, axis labels and plot title - see Appendix A.5.1.2.

Export as PDF

Pops up a dialogue which will write the current plot as a PDF file. In general this is a faithful and high quality rendering of what is displayed in the plot window. However, if plotting is being done using the transparent markers, the markers will be rendered as if they were opaque. Plots with very many points can result in rather large output PDFs.

Export as GIF

Pops up a dialogue which will output the current plot to a GIF file. The output file is just the same as the plotted image that you see. Resize the plotting window before the export to control the size of the output GIF.

Rescale

Rescales the axes of the current plot so that it contains all the data points in the currently selected subsets. By default the plot will be scaled like this, but it it may have changed because of changes in the subset selection or from zooming in or out.

Grid

Toggles whether a grid is drawn over the plotting surface or not.

Show Legend

Toggles whether a legend showing how each data set is represented is visible to the right of the plot. Initially the legend is shown only if more than one data set is being shown at once.

Draw Subset Region

Allows you to draw a region on the screen defining a new Row Subset. When you have finished drawing it, click this button again to indicate you're done. See Appendix A.5.1.6 for more details.

Subset From Visible

Defines a new **Row Subset** consisting of only the points which are currently visible on the plotting surface. See Appendix A.5.1.6 for more explanation.

The **Dataset Toolbar** contains the following options:

/ **Add/Remove dataset**

Adds/removes tabs in which the data for extra datasets can be entered. See Appendix A.5.1.1.

/ **Add/Remove auxiliary axis**

Adds/removes a selector for entering an auxiliary axis to modify point colours etc. See Appendix A.5.1.5.

Toggle point labelling

Allows text labels to be drawn near plotted points. See Appendix A.5.1.4.

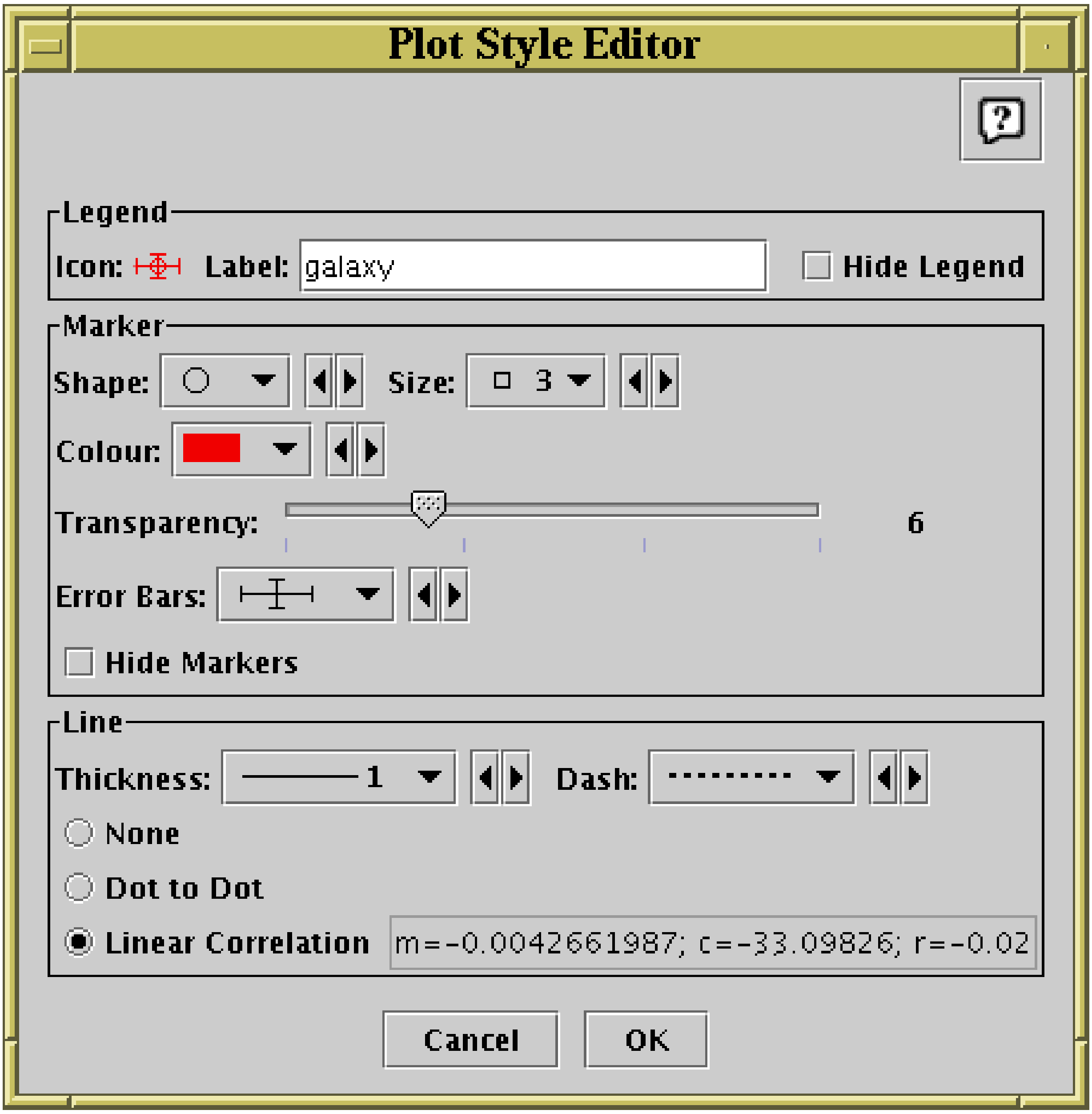
/ **Toggle X/Y error bars**

Switches between drawing symmetric error bars and no error bars in the X and Y directions respectively. Other options are available in the **Errors** menu. See Appendix A.5.1.3.

You have considerable freedom to configure how the points are plotted including the shape, colour and transparency of symbols, the type of lines which join them if any, and the representation of error bars if active. These options are described in the following subsection.

A.5.3.1 Plot Style Editor

When plotting points in a scatter plot there are many different ways that each point can be displayed. By default, TOPCAT chooses a set of markers on the basis of how many points there are in the table and uses a different one for each plotted set. The marker for each set is displayed in a button to the right of its name in the dataset selector panel at the bottom of the plot window. If you click this button the following dialogue will pop up which enables you to change the appearance.



Plot Style Editor

Legend

Icon: Label: galaxy □ Hide Legend

Marker

Shape: ○ Size: □ 3

Colour: 🟥

Transparency: 6

Error Bars:

□ Hide Markers

Line

Thickness: 1 Dash: ---------

○ None

○ Dot to Dot

● Linear Correlation m=-0.0042661987; c=-33.09826; r=-0.02

Style editor dialogue for 2d scatter plot

The **Legend** box defines how the selected set will be identified in the legend which appears alongside the plot (though the legend will only be visible if **Show Legend** ( ) is on):

Icon
: Displays the icon which will be shown to identify the points in the selected set. Its appearance depends on the selections you make in the rest of this dialogue window.

Label
: Gives the name written in the legend to label the subset. By default this is derived from the Row Subset's name and, if it's not part of the main dataset, the name of the dataset's tab. You can type in a new value to change what is written in the legend.

Hide Legend
: If this checkbox is selected, then no entry for the selected set will appear in the legend.

The **Marker** box defines how the markers plotted for each data point will appear:

Shape

Choose from a variety of shapes such as open or filled circles, squares, crosses etc.

Size
Choose the size of the marker; the value given is approximate radius in pixels. If a size of zero is chosen, then the shape doesn't matter, the points will be plotted as single pixels.

Colour
Choose the colour in which the markers, and any line if one is drawn, will be plotted.

Transparency
Choose transparency of the plotted symbols. The scale on the slider is logarithmic, with 1 at the left hand end. The actual value chosen is an integer written at the right of the slider. This number gives the number of markers for this set which need to be plotted in the same position to result in fully opaque pixels - any fewer and the background, or other markers plotted underneath, will show through to some extent. Setting this to some value greater than 1 is very useful if you have a very large number of points being plotted (especially if it's comparable with the number of the pixels on the screen), since it enables to you distinguish regions where there are lots of points on top of each other from those where there are only a few.

Error Bars
If error bars are active for this plot, allows you to select the way they will appear. The options which can be selected here will depend on whether X and/or Y errors are in use.

Hide Markers
This check box is only enabled if a line and/or error bars are being plotted; it allows the markers to be invisible, so that only the line/error bars are seen.

The **Line** box determines if any lines are drawn associated with the current set and if so what their appearance will be.

Thickness
Selects the line thickness in pixels.

Dash
Selects a dash pattern (solid, dotted, dashed or dot-dashed) for the line.

Type
The other radio buttons determine what kind of line, if any, will be plotted for these points. There are three options:

None
No line is drawn - this is the default.

Dot to Dot
A straight line segment is drawn between each of the points. If the points effectively form an ordered set of samples of a function, this will result in a more-or-less smooth drawing of that function on the plot. Note that the lines are drawn in the order that the points appear in the basic table, and if this doesn't match the 'ordinate' order the result will be a mess. Really, the drawing order ought to be the table's current sort order - that it is not is a misfeature which may be corrected at some point. Note also that if you try this with a huge table you're likely to get a result which (a) is messy and (b) takes a very long time to draw.

Linear Correlation
If you select this option then a linear regression line will be plotted. The correlation coefficients will also be displayed to the right of the radio button (you may need to resize the window to see them all). The values cited are *m* (gradient), *c* (intercept) and *r* (product moment correlation coefficient). You can cut and paste from this text.

Note that for both the plotted line and the quoted coefficients the data is taken only from the points which are currently visible - that means that if you've zoomed the axes to exclude some of the data points, they will not be contributing to the calculated statistics.

Any changes you make in this window are reflected in the plot straight away. If you click the **OK** button at the bottom, the window will disappear and the changes remain. If you click **Cancel** the window will disappear and any changes you made will be discarded.

You can also change all the plotting styles at once by using the **Marker Style** menu in the plot window. Here you can select a standard group of styles (e.g. all open 2-pixel markers with different colours and shapes) for the plotted sets. Similarly, error styles can be changed all at once using the **Error Style** menu.

A.5.4 Stacked Line Plot (old-style)

This section describes an old-style plotting window. The standard plotting windows are described in Appendix A.4, though note that at time of writing (v4.1) no equivalent new-style plotting window is available for plotting stacked lines.

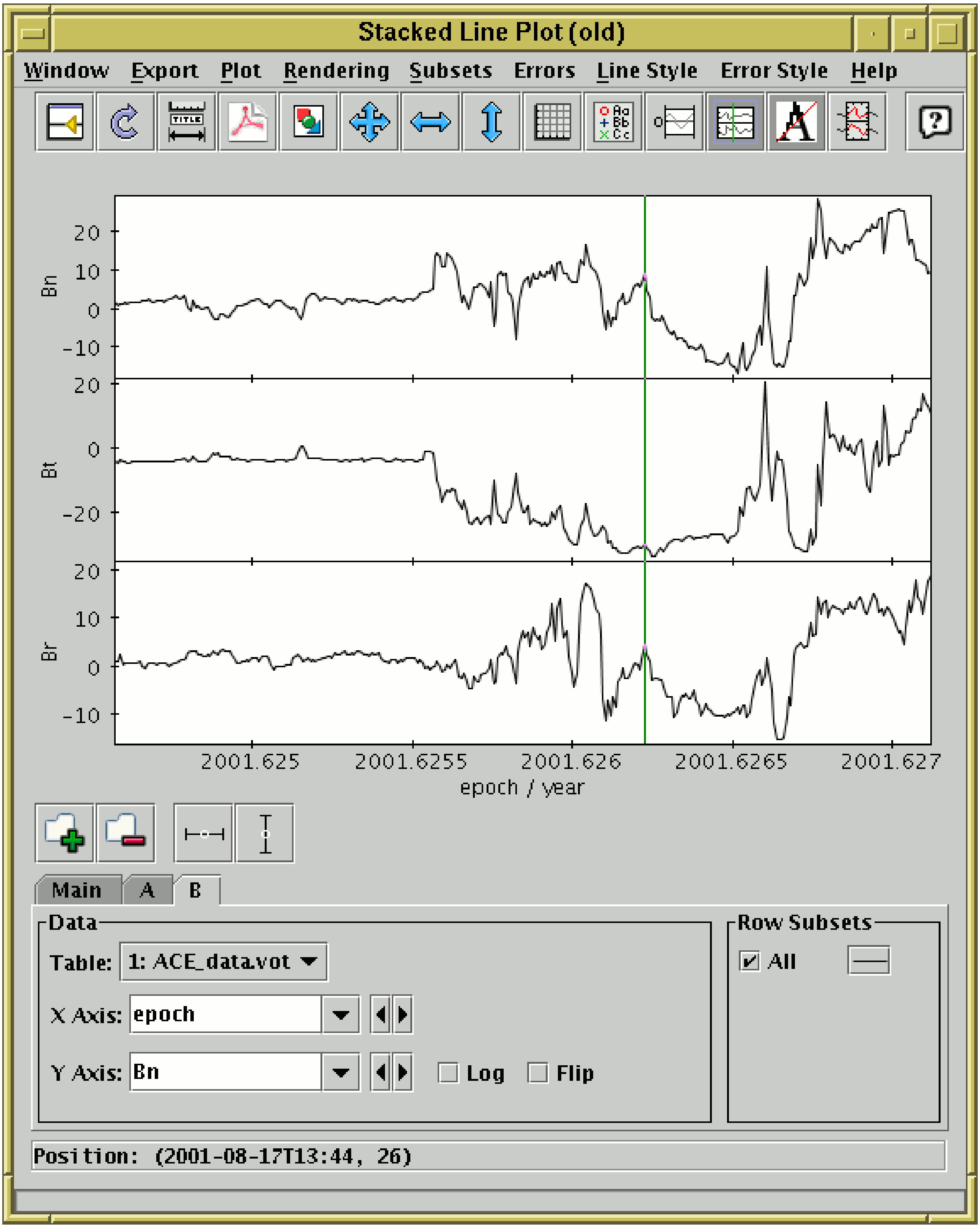


Table: 1: ACE_data.vot

X Axis: epoch

Y Axis: Bn □ Log □ Flip

☑ All

Position: (2001-08-17T13:44, 26)

Stacked Lines Window

The stacked line plot window allows you to plot one or more ordinate (Y) quantities against a monotonic abscissa (X) quantity. For clarity, the different plots are displayed on vertically displaced graphs which share the same X axis. You can display this window using the **Lines** () item in the Control Window's **Graphics** menu.

The display initially holds a single X-Y graph, usually with lines connecting adjacent points. The points will be reordered before drawing if necessary so that the line is displayed as a function of X, rather than of an invisible third independent variable (in the Scatter Plot this isn't done which can lead to lines being scribbled all over the plot). If one of the columns in the table appears to represent a time value, this will be selected as the default X axis. Otherwise, the 'magic' **index** variable will be used, which represents the row number. Of course, these can be changed from their default values using the selectors in the usual way.

To add a new graph with a different Y axis, use the **Add Dataset** ( ) button in the **Dataset Toolbar** at the bottom of the window. This has a slightly different effect from what it does in the other plot windows, in that it inserts a new plotting region with its own Y axis at the top of the plot on which the specified data is drawn, rather than only causing a new set of points to be plotted on the existing plot region. Thus all the datasets appear in their own graphs with their own Y axes (though if you have multiple row subsets plotted for the same dataset they will appear on the same part of the plot as usual). To remove one of the graphs, select its tab and use the **Remove Dataset** ( ) button as usual.

Zooming can only be done on one axis at a time rather than dragging out an X-Y region on the plot surface, since there isn't a single Y axis to zoom on. To zoom the X axis in/out, position the mouse just below the X axis at the bottom of the plot and drag right/left. To zoom one of the Y axes in or out, position the mouse just to the left of the Y axis you're interested in and drag down/up. To set the ranges manually, use the **Configure Axes and Title** ( ) button as usual, but note that there is one label/range setting box for each of the Y axes. These things work largely as described in Appendix A.5.1.2, as long as you bear in mind that the range of each of the Y axes is treated independently of the others.

Clicking on any of the points will **activate** it - see Section 8.

The following buttons are available on the toolbar:

Split Window

Allows the dataset selector to be resized by dragging a separator between it and the plot area. Good for small screens.

Redraws the current plot. It is usually not necessary to use this button, since if you change any of the plot characteristics with the controls in this window the plot will be redrawn automatically. However if you have changed the data, e.g. by editing cells in the Data Window, the plot is not automatically redrawn (since this is potentially an expensive operation and you may not require it). Clicking this button redraws the plot taking account of any changes to the table data.

Configure Axes and Title

Pops up a dialogue to allow manual configuration of the range and label for the X axis and each of the Y axes, as well as a plot title - see Appendix A.5.1.2.

Export as PDF

Pops up a dialogue which will write the current plot as a PDF file.

Export as GIF

Pops up a dialogue which will output the current plot to a GIF file. The output file is just the same as the plotted image that you see. Resize the plotting window before the export to control the size of the output GIF.

Rescale

Rescales all the plots so that all points in the plotted datasets can be seen. The X axis and all the Y axes are rescaled to fit the data.

Rescale X Axis

Rescales the X axis only. The X axis is rescaled to cover the lowest and highest values on any of the plotted datasets, but the Y ranges are left as they are.

Rescale Y Axes

Rescales the Y axes only. Each of the plotted Y axes are independently rescaled so that they cover the lowest and highest values within the currently visible X range.

Full Grid

Toggles whether X and Y grid lines are drawn over the plots or not. If this is selected, the single y=0 grid line (see next item) will automatically be deselected.

Show Legend

Toggles whether a legend showing how each data set is represented is visible to the right of the plot. Initially the legend is shown only if more than one data set is being shown at once.

y=0 Grid Lines

Toggles whether a single horizontal line at y=0 is drawn. If this is selected, the full grid (see previous item) will automatically be deselected.

Show Vertical Crosshair

Toggles whether a vertical line follows the mouse when it is positioned over the plot. This can be useful to compare features in different graphs at the same X coordinate position.

Antialias

Toggles whether lines are drawn with antialiasing. Antialiasing means smoothing lines so that they appear less pixelised, and generally improves the aesthetic appearance of the plot, but in some circumstances it might look better not antialiased. The state of this button does not affect images exported to postscript.

Subset From X Range

Defines a new **Row Subset** in each of the plotted tables consisting of only the points in the currently visible range on the horizontal axis. Points will be included even if they are outside the current ranges of the Y axes.

The **Dataset Toolbar** contains the following options:

/ **Add/Remove dataset**

Adds/removes a dataset for plotting, which both adds/removes a tab from the dataset selector and adds/removes a plot in the currently visible stack in the plotting area. See above for more explanation.

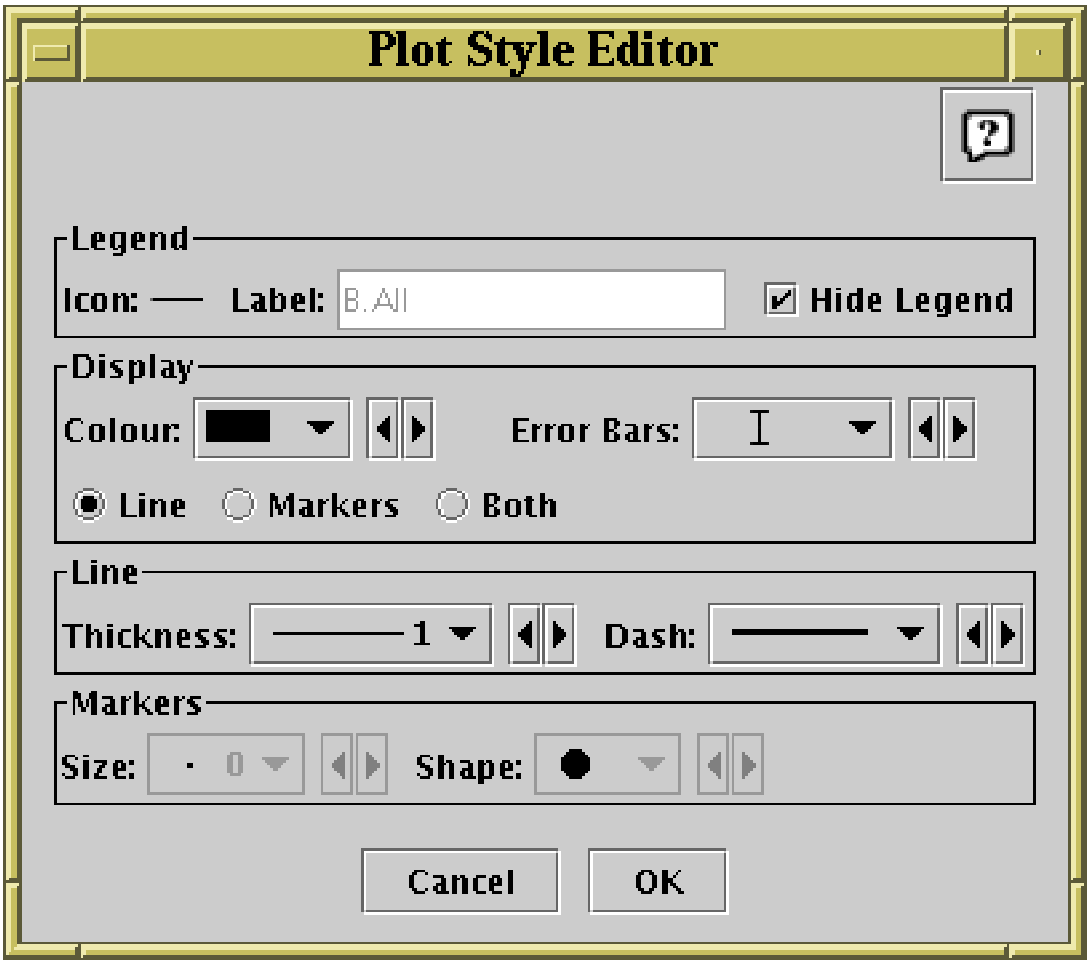
◘ / ◘ **Toggle X/Y error bars**

Switches between drawing symmetric error bars and no error bars in the X and Y directions respectively. Other options are available in the **Errors** menu. See Appendix A.5.1.3.

You can determine how the data are plotted using lines and/or markers as described in the following subsection.

A.5.4.1 Lines Style Editor

The default plotting style for the stacked lines plot is a simple black line for each graph. Since the plots typically do not overlap each other, this is in many cases suitable as it stands. However, you can configure the plotting style so that the points are plotted with markers as well as or instead of lines, and change the colours, marker shapes, line styles etc. The style for each row subset is displayed in a button to the right of its name in the bottom right of the plotting window. If you click this button the following dialogue will pop up which entables you to configure the plotting style.



Plot Style Editor

Legend

Icon: —— Label: B.All ☑ Hide Legend

Display

Colour: ■ Error Bars: I

● Line ○ Markers ○ Both

Line

Thickness: 1 Dash: ——

Markers

Size: 0 Shape: ●

Stacked Line Plot Style Editor

The **Legend** box defines how the selected set will be identified in the legend which appears alongside the plot (though the legend will only be visible if **Show Legend** (☷) is on):

Icon

Displays the icon which will be shown to identify the points in the selected set. Its appearance depends on the selections you make in the rest of this dialogue window.

Label

Gives the name written in the legend to label the subset. By default this is derived from the Row Subset's name and, if it's not part of the main dataset, the name of the dataset's tab. You can type in a new value to change what is written in the legend.

Hide Legend
If this checkbox is selected, then no entry for the selected set will appear in the legend.

The **Display** box defines how the markers plotted for each data point will appear:

Colour
Choose the colour in which the lines and/or markers will be plotted.

Line/Markers
Select from the radio buttons whether you want just lines between the data points, or markers at each point, or both.

Error Bars
If error bars are active for this plot, allows you to select the way they will appear. The options which can be selected here will depend on whether X and/or Y erors are in use.

The **Line** box defines how the lines joining the points will look. These controls will only be active if the Display selection is **Line** or **Both**.

Thickness
Determines line thickness.

Dash
Determines line dash pattern.

The **Markers** box defines how markers at the data points will look. These controls will only be active if the Display selection is **Markers** or **Both**.

Size
Determines the size of the markers in pixels. If a size of zero is chosen then the shape doesn't matter, the points will be plotted as single pixels.

Shape
Determines the shape of the markers from a selection such as open or filled circles, squares, crosses etc.

Any changes you make in this window are reflected in the plot straight away. If you click the **OK** button at the bottom, the window will disappear and the changes remain. If you click **Cancel** the window will disappear and any changes you made will be discarded.

You can also change all the plotting styles at once by using the **Line Style** menu in the stacked lines plot window. Here you can select a standard group of styles (e.g. dashed lines, coloured lines) for the plotted sets. Similarly, error styles can be changed all at once using the **Error Style** menu.

A.5.5 3D Plot (old-style)

This section describes an old-style plotting window. The standard plotting windows are described in Appendix A.4.

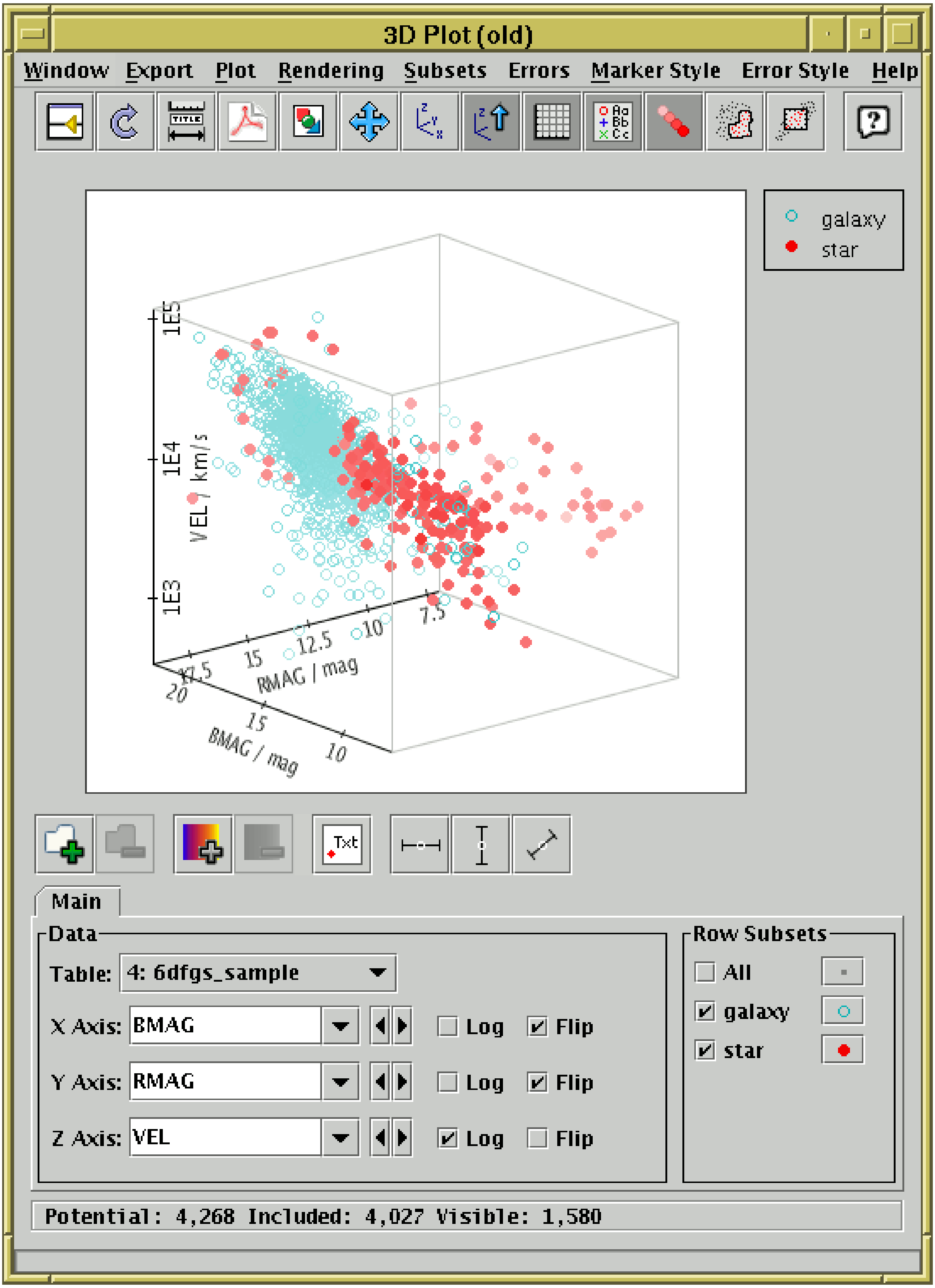


Table: 4: 6dfgs_sample

X Axis: BMAG □ Log ☑ Flip

Y Axis: RMAG □ Log ☑ Flip

Z Axis: VEL ☑ Log □ Flip

□ All

☑ galaxy ○

☑ star ●

```
Potential: 4,268 Included: 4,027 Visible: 1,580
```

3D scatter plot window

The 3D plot window draws 3-dimensional scatter plots of one or more triples of table columns (or derived quantities) on Cartesian axes. You can display it using the **3D** ( ) item in the Control

Window's **Graphics** menu.

On the display a marker is plotted for each row in the selected dataset(s) at a position determined by the values in the table columns selected to provide the X, Y and Z values. A marker will only be plotted if none of the X, Y and Z values are blank. Select the quantities to plot and the plotting symbols with the dataset selector (Appendix A.5.1.1) at the bottom.

The 3D space can be rotated by dragging the mouse around on the surface - it will rotate around the point in the centre of the plotted cube. The axis labels try to display themselves the right way up and in a way which is readable from the viewing point if possible, which means they move around while the rotation is happening. By default the points are rendered as though the 3D space is filled with a 'fog', so that more distant points appear more washed out - this provides a visual cue which can help to distinguish the depth of plotted points. However, you can turn this off if you want. If there are many points, then you may find that they're not all plotted while a drag-to-rotate gesture is in progress. This is done to cut down on rendering time so that GUI response stays fast. When the drag is finished (i.e. when you release the mouse button) all the points will come back again.

Zooming is also possible. You can zoom in around the centre of the plot so that the viewing window only covers the middle. The easiest way to do this is to use the mouse wheel if you have one - wheel forward to zoom in and backward to zoom out. Alternatively you can do it by dragging on the region to the left of the plot, like the **Axis Zoom** in some of the 2-d plots. Drag the mouse down to zoom in or up to zoom out on this part of the window. Currently it is only possible to zoom in/out around the centre of the plot. When zoomed you can use the **Subset From Visible** ( ) toolbar button to define a new **Row Subset** consisting only of the points which are currently visible. See Appendix A.5.1.6 for more explanation.

Clicking on any of the plotted points will **activate** it - see Section 8.

The following buttons are available on the toolbar:

Split Window

Allows the dataset selector to be resized by dragging a separator between it and the plot area. Good for small screens.

Replot

Redraws the current plot. It is usually not necessary to use this button, since if you change any of the plot characteristics with the controls in this window the plot will be redrawn automatically. However if you have changed the data, e.g. by editing cells in the Data Window, the plot is not automatically redrawn (since this is potentially an expensive operation and you may not require it). Clicking this button redraws the plot taking account of any changes to the table data.

Configure Axes and Title

Pops up a dialogue to allow manual configuration of axis ranges, axis labels and plot title - see Appendix A.5.1.2.

Export as PDF

Pops up a dialogue which will write the current plot as a PDF file. In general this is a faithful and high quality rendering of what is displayed in the plot window. However, if plotting is being done using the transparent markers, the markers will be rendered as if they were opaque. Plots with very many points can result in rather large output PDFs.

Export as GIF

Pops up a dialogue which will output the current plot to a GIF file. The output file is just the same as the plotted image that you see. Resize the plotting window before the export to control the size of the output GIF.

Rescale

Rescales the axes of the current plot so that it contains all the data points in the currently selected subsets. By default the plot will be scaled like this, but it it may have changed because of changes in the subset selection.

Reorient

Reorients the axes of the current plot to their default position. This can be useful if you've lost track of where you've rotated the plot to with the mouse. This also resets the zoom level to normal if you've changed it.

Stay Upright

Toggle button which when selected ensures that the Z axis is oriented vertically on the screen at all times. By default this is off which means you can drag the axes round to any orientation, but it can be convenient to switch it on to keep the plot pointing in a sensible direction.

Grid

Toggles whether the cubic frame bounding the plot is drawn or not.

Show Legend

Toggles whether a legend showing how each data set is represented is visible to the right of the plot. Initially the legend is shown only if more than one data set is being shown at once.

Fog

Toggles whether rendering is done as if the space is filled with fog. If this option is selected, distant points will appear more washed out than near ones.

Draw Subset Region

Allows you to draw a region on the screen defining a new Row Subset. When you have finished drawing it, click this button again to indicate you're done. The subset will include points at all depths in the viewing direction which fall in the region you have drawn. See Appendix A.5.1.6 for more details.

Subset From Visible

Defines a new **Row Subset** consisting of only the points which are currently visible on the plotting surface. See Appendix A.5.1.6 for more explanation.

The following additional item is available as a menu item only:

Antialias

Toggles whether the axes and their annotations are drawn antialiased. Antialiased lines are smoother and generally look more pleasing, especially for text at a sharp angle, but it can slow the rendering down a bit.

The **Dataset Toolbar** contains the following options:

- / **Add/Remove dataset**

  Adds/removes tabs in which the data for extra datasets can be entered. See Appendix A.5.1.1.

- / **Add/Remove auxiliary axis**

  Adds/removes a selector for entering an auxiliary axis to modify point colours etc. See Appendix A.5.1.5.

- Txt **Toggle point labelling**

  Allows text labels to be drawn near plotted points. See Appendix A.5.1.4.

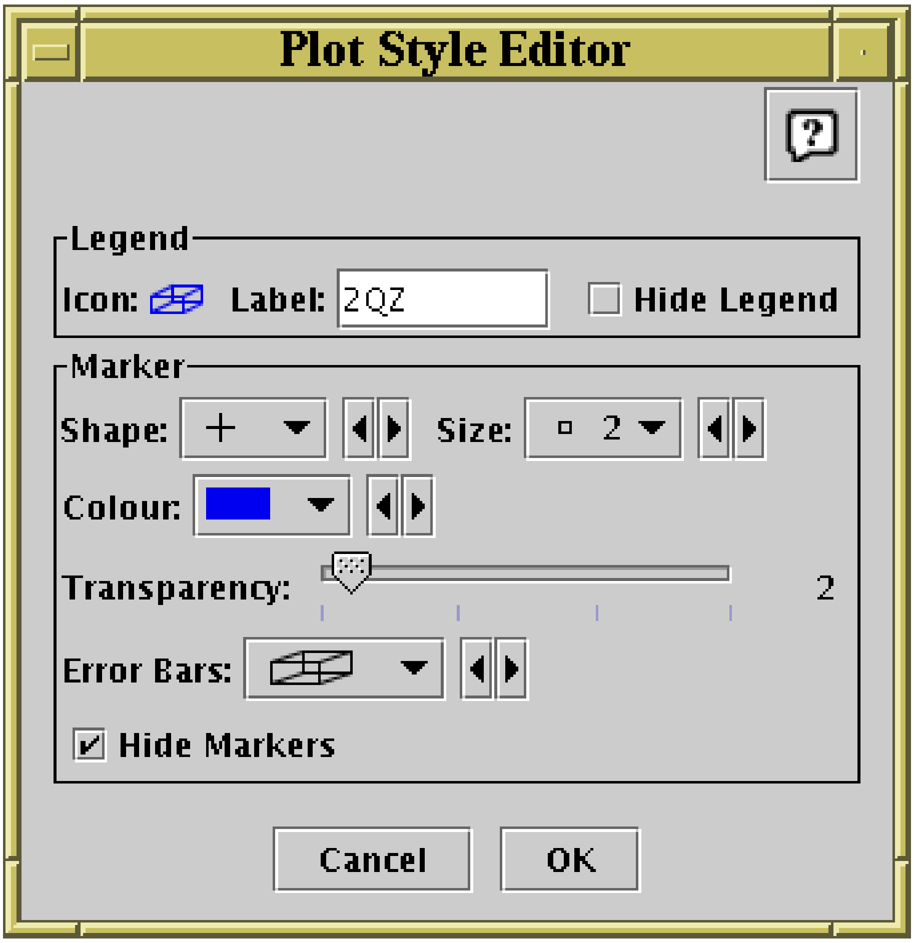
- / / **Toggle X/Y/Z error bars**

  Switches between drawing symmetric error bars and no error bars in the X, Y and Z directions respectively. Other options are available in the **Errors** menu. See Appendix A.5.1.3.

You have considerable freedom to configure how the points are plotted including the shape, colour and transparency of symbols and the representation of error bars if used. These options are described in the following subsection.

A.5.5.1 3D Plot Style Editor

When plotting points in a 3D plot there are many different ways that each point can be displayed. By default, TOPCAT chooses a set of markers on the basis of how many points there are in the table and uses a different one for each plotted set. The marker for each set is displayed in a button to the right of its name in the dataset selector panel at the bottom of the plot window. If you click this button the following dialogue will pop up which enables you to change the appearance.



Icon:

Label: 2QZ

☐ Hide Legend

Shape: +

Size: 2

Colour: 🟦

Transparency: 2

Error Bars:

☑ Hide Markers

Style editor dialogue for 3d plots

The **Legend** box defines how the selected set will be identified in the legend which appears alongside the plot (though the legend will only be visible if **Show Legend** (Aa Bb Cc) is on):

Icon
: Displays the icon which will be shown to identify the points in the selected set. Its appearance depends on the selections you make in the rest of this dialogue window.

Label
: Gives the name written in the legend to label the subset. By default this is derived from the Row Subset's name and, if it's not part of the main dataset, the name of the dataset's tab. You can type in a new value to change what is written in the legend.

Hide Legend
: If this checkbox is selected, then no entry for the selected set will appear in the legend.

The **Marker** box defines how the markers plotted for each data point will appear:

Shape
: Choose from a variety of shapes such as open or filled circles, squares, crosses etc.

Size
: Choose the size of the marker; the value given is approximate radius in pixels. If a size of zero is chosen, then the shape doesn't matter, the points will be plotted as single pixels.

Colour
: Choose the colour in which the markers will be plotted.

Transparency

Choose transparency of the plotted symbols. The scale on the slider is logarithmic, with 1 at the left hand end. The actual value chosen is an integer written at the right of the slider. This number gives the number of markers for this set which need to be plotted in the same position to result in fully opaque pixels - any fewer and the background, or other markers plotted underneath, will show through to some extent. Setting this to some value greater than 1 is very useful if you have a very large number of points being plotted (especially if it's comparable with the number of the pixels on the screen), since it enables to you distinguish regions where there are lots of points on top of each other from those where there are only a few. If a finite transparency is set, you may find it useful to turn off fogging (see above).

Error Bars

If error bars are active for this plot, allows you to select the way they will appear. The options which can be selected here will depend on whether X, Y and/or Z errors are in use.

Hide Markers

This check box is only enabled if error bars are being plotted; it allows the markers to be invisible, so that only the error bars are seen.

Any changes you make in this window are reflected in the plot straight away. If you click the **OK** button at the bottom, the window will disappear and the changes remain. If you click **Cancel** the window will disappear and any changes you made will be discarded.

You can also change all the plotting styles at once by using the **Marker Style** menu in the plot window. Here you can select a standard group of styles (e.g. all open 2-pixel markers with different colours and shapes) for the plotted sets. Similarly, error styles can be changed all at once using the **Error Style** menu.

A.5.6 Spherical Plot (old-style)

This section describes an old-style plotting window. The standard plotting windows are described in Appendix A.4.

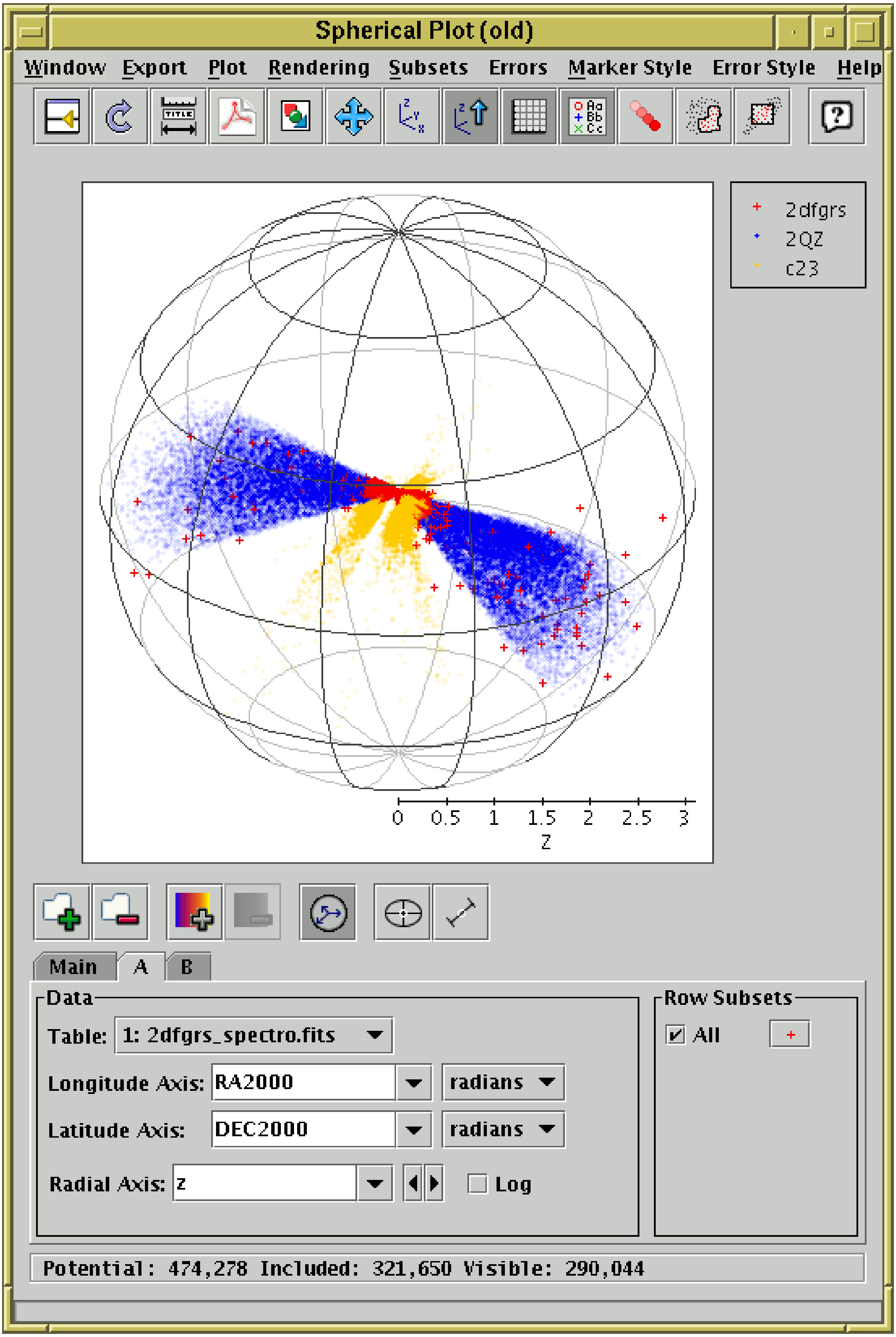


Table: 1: 2dfgrs_spectro.fits

Longitude Axis: RA2000 radians

Latitude Axis: DEC2000 radians

Radial Axis: z □ Log

☑ All

Potential: 474,278 Included: 321,650 Visible: 290,044

Spherical plot window

The spherical plot window draws 3-dimensional scatter plots of datasets from one or more tables on spherical polar axes, so it's suitable for displaying the position of coordinates on the sky or some other spherical coordinate system, such as the surface of a planet or the sun. You can display it using the **Sphere** () item in the Control Window's **Graphics** menu.

In most respects this window works like the 3D Plot window, but it uses spherical polar axes rather than Cartesian ones, You have to fill in the dataset selector (Appendix A.5.1.1) at the bottom with longitude- and latitude-type coordinates from the table. Selectors are included to indicate the units of those coordinates. If TOPCAT can locate columns in the table which appear to represent Right Ascension and Declination, these will be filled in automatically. If only these two are filled in, then the points will be plotted on the surface of the unit sphere - this is suitable if you just want to inspect the positions of a set of objects in the sky.

If the **Radial Coordinates** () button is activated, you can optionally fill in a value in the **Radial Axis** selector as well. In this case points will be plotted in the interior of the sphere, at a distance from the centre given by the value of the radial coordinate.

The 3D space can be rotated by dragging the mouse around on the surface - it will rotate around the centre of the sphere. By default the points are rendered as though the 3D space is filled with a 'fog', so that more distant points appear more washed out - this provides a visual cue which can help to distinguish the depth of plotted points. However, you can turn this off if you want. If there are many points, then you may find that they're not all plotted while a drag-to-rotate gesture is in progress. This is done to cut down on rendering time so that GUI response stays fast. When the drag is finished (i.e. when you release the mouse button) all the points will come back again.

Zooming is also possible. You can zoom in around the centre of the plot so that the viewing window only covers the middle. The easiest way to do this is to use the mouse wheel if you have one - wheel forward to zoom in and backward to zoom out. Alternatively you can do it by dragging on the region to the left of the plot, like the **Axis Zoom** in some of the 2-d plots. Drag the mouse down to zoom in or up to zoom out on this part of the window. Currently it is only possible to zoom in/out around the centre of the plot. When zoomed you can use the **Subset From Visible** () toolbar button to define a new **Row Subset** consisting only of the points which are currently visible. See Appendix A.5.1.6 for more explanation.

Clicking on any of the plotted points will **activate** it - see Section 8.

The following buttons are available on the toolbar:

Split Window

Allows the dataset selector to be resized by dragging a separator between it and the plot area. Good for small screens.

Replot

Redraws the current plot. It is usually not necessary to use this button, since if you change any of the plot characteristics with the controls in this window the plot will be redrawn automatically. However if you have changed the data, e.g. by editing cells in the Data Window, the plot is not automatically redrawn (since this is potentially an expensive operation and you may not require it). Clicking this button redraws the plot taking account of any changes to the table data.

Configure Axes and Title

Pops up a dialogue to allow manual configuration of axis ranges, axis labels and plot title - see Appendix A.5.1.2. The only configurable axis range is the upper limit of the radial axis.

Export as PDF

Pops up a dialogue which will write the current plot as a PDF file. In general this is a faithful and high quality rendering of what is displayed in the plot window. However, if plotting is being done using the transparent markers, the markers will be rendered as if they were opaque. Plots with very many points can result in rather large output PDFs.

Export as GIF

Pops up a dialogue which will output the current plot to a GIF file. The output file is just the same as the plotted image that you see. Resize the plotting window before the export to control the size of the output GIF.

Rescale

Rescales the axes of the current plot so that it contains all the data points in the currently selected subsets. By default the plot will be scaled like this, but it it may have changed because of changes in the subset selection.

Reorient

Reorients the axes of the current plot to their default position. This can be useful if you've lost track of where you've rotated the plot to with the mouse. This also resets the zoom level to normal if you've changed it.

Stay Upright

Toggle button which when selected ensures that the north pole (latitude = +90 degrees) is oriented vertically on the screen at all times. By default this is on.

Grid

Toggles whether the spherical wire frame bounding the plot is drawn or not.

Show Legend

Toggles whether a legend showing how each data set is represented is visible to the right of the plot. Initially the legend is shown only if more than one data set is being shown at once.

Fog

Toggles whether rendering is done as if the space is filled with fog. If this option is selected, distant points will appear more washed out than near ones.

Draw Subset Region

Allows you to draw a region on the screen defining a new Row Subset. When you have finished drawing it, click this button again to indicate you're done. The subset will include points at all depths in the viewing direction which fall in the region you have drawn. See Appendix A.5.1.6 for more details.

Subset From Visible

Defines a new **Row Subset** consisting of only the points which are currently visible on the plotting surface. See Appendix A.5.1.6 for more explanation.

The following additional item is available as a menu item only:

Antialias

Toggles whether the axes and their annotations are drawn antialiased. Antialiased lines are smoother and generally look more pleasing, especially for text at a sharp angle, but it can slow the rendering down a bit.

The **Dataset Toolbar** contains the following options:

/ **Add/Remove dataset**

Adds/removes tabs in which the data for extra datasets can be entered. See Appendix A.5.1.1.

/ **Add/Remove auxiliary axis**

Adds/removes a selector for entering an auxiliary axis to modify point colours etc. See Appendix A.5.1.5.

Toggle point labelling

Allows text labels to be drawn near plotted points. See Appendix A.5.1.4.

Toggle Radial Coordinates

When activated, a column selector labelled **Radial Axis** will be visible below the Longitude and Latitude Axis selectors. This enables you to enter a value for the radial coordinate of each point. If this is disabled, or if it has a blank value, then all the points will have an assumed radial coordinate of unity and will be plotted on the surface of the sphere.

Toggle tangential error bars

When activated, an additional column selector appears in the dataset panel to the right of the Longitude and Latitude selectors, along with its own unit selector. You can fill this in with an isotropic error value representing the radius of a small circle associated with the selected points, in units of arcsec, arcmin, degrees or radians. This will cause 2-d error bars to be plotted. You can configure their appearance (e.g. crosshairs, ellipses, rectangles, ...) using the style editor in the usual way. See Appendix A.5.1.3 for more information.

Toggle radial error bars

Switches radial error bars on and off. See Appendix A.5.1.3. This button will only be enabled if the Radial Coordinates are in use.

You have considerable freedom to configure how points are plotted including the shape, colour and transparency of symbols and the representation of errors if used. This works exactly as for the Cartesian 3D plot as described in Appendix A.5.5.1.

A.5.7 Density Map (old-style)

This section describes an old-style plotting window. The standard plotting windows are described in Appendix A.4.

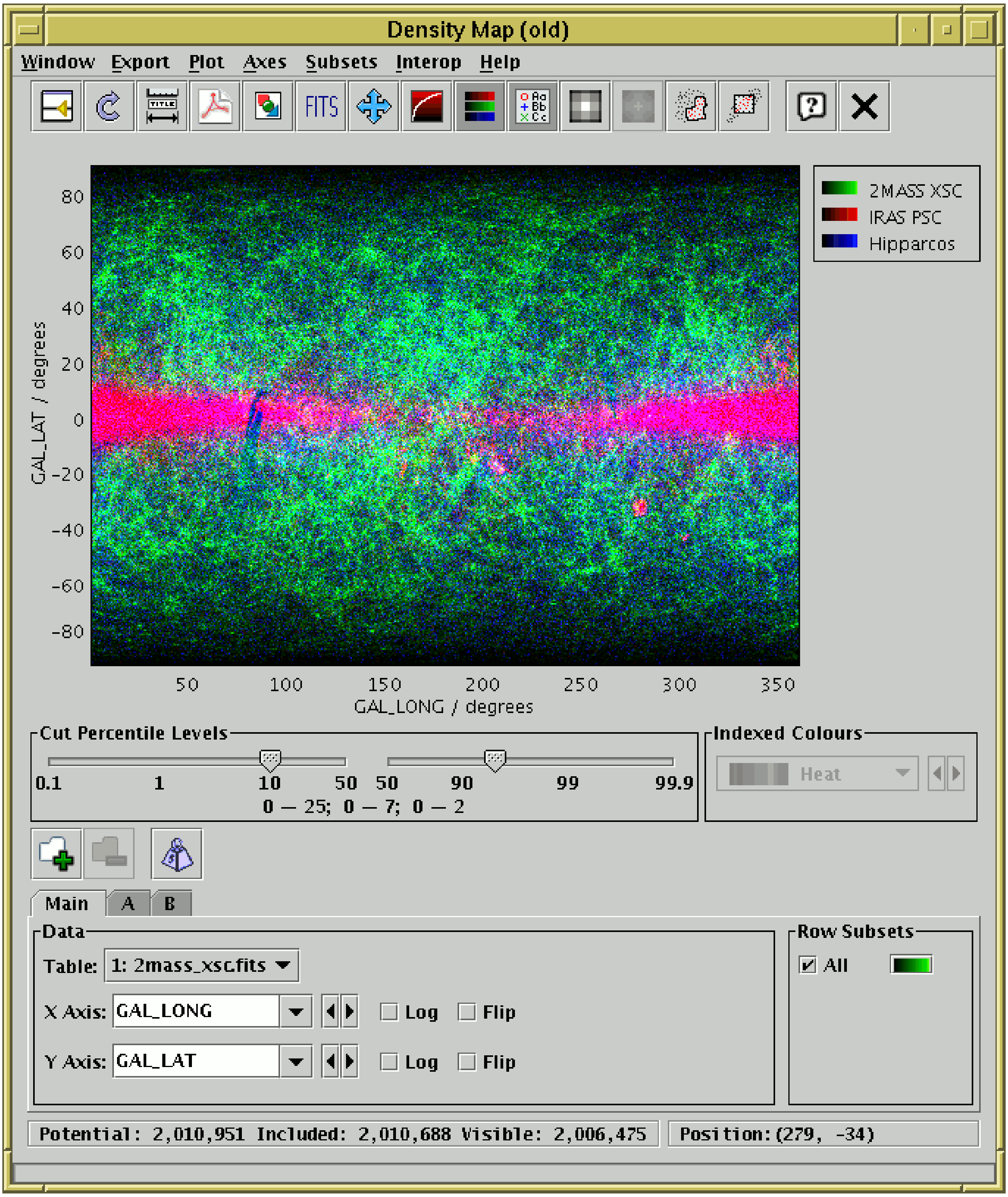


Table: 1: 2mass_xsc.fits

X Axis: GAL_LONG □ Log □ Flip

Y Axis: GAL_LAT □ Log □ Flip

☑ All

Potential: 2,010,951 Included: 2,010,688 Visible: 2,006,475 Position:(279, -34)

Density map window in RGB mode

The density map window plots a 2-dimensional density map of one or more pairs of table columns (or derived quantities); the colour of each pixel displayed is determined by the number of points in the data set which fall within its bounds. Another way to think of this is as a histogram on a 2-dimensional grid, rather than a 1-dimensional one as in the Histogram Window. You can optionally weight these binned counts with another value from the table.

Density maps are suitable when you have a very large number of points to plot, since in this case it's important to be able to see not just whether there is a point at a given pixel, but how many points fall on that pixel. To a large extent, the transparency features of the other 2d and 3d plotting windows address this issue, but the density map gives you a bit more control. It can also export the result as a FITS image, which can then be processed or viewed using image-specific software such as GAIA or Aladin.

This window can be operated in two modes:

Indexed Mode
Each pixel represents a single scalar value, corresponding to a count or sum as indicated by the selected dataset(s). If multiple datasets are being plotted at once, the values from each will be summed to give the result at each point. The mapping from numeric value to pixel colour at each point on the plot is determined by the colour map selected in the **Indexed Colours** selector below the plot. In this case the style editor colour selectors have no effect and are disabled. A fairly wide range of colour maps is provided by default. If these do not suit your needs, it is possible to provide your own custom colour maps using the **lut.files** system property - see Section 10.2.3.

RGB Mode
Each pixel has up to three independent contributions, its intensity in Red, Green and Blue channels. These can come from different datasets, as configured in the style editor. If more than one dataset is assigned the same colour, the contributions are summed for that channel. In this case the Indexed Colours colour map selector has no effect and is disabled.

Switch between the modes using the **RGB** ( ) button.

You can configure the axes, including **zooming** in and out, with the mouse (drag on the plot or the axes) or manually as described in Appendix A.5.1.2.

Two controls specific to this window are shown below the plot itself:

Cut Percentile Levels
This controls how the number of counts in each pixel maps to a brightness. There are two sliders, one for the lower bound and one for the upper bound. They are labelled (logarithmically) with percentile values. If the upper one is set to 90, it means that any pixel above the 90th percentile of the pixels in the image in terms of count level will be shown with maximum brightness, and similarly for the lower one. These values apply independently to each colour channel if more than one is in use. Immediately below the sliders, the pixel values which correspond to minimum and maximum brightness are displayed. In indexed mode there is one range, and in RGB mode there may be up to three. If the image is not fairly completely covered, this control doesn't give you as much freedom as you might like - the user interface may be improved in future releases.

Indexed Colours
When in indexed (non-RGB) mode only, this allows you to select a colour map which determines how pixel values (counts or sums per bin) are turned into colours on the screen. The lowest value corresponds to the colour at the left side of the icon and the highest value to the right side. In RGB mode this is disabled.

The following buttons are available on the toolbar:

Split Window
Allows the dataset selector to be resized by dragging a separator between it and the plot area. Good for small screens.

Replot

Redraws the current plot. It is usually not necessary to use this button, since if you change any of the plot characteristics with the controls in this window the plot will be redrawn automatically. However if you have changed the data, e.g. by editing cells in the Data Window, the plot is not automatically redrawn (since this is potentially an expensive operation and you may not require it). Clicking this button redraws the plot taking account of any changes to the table data.

Configure Axes and Title

Pops up a dialogue to allow manual configuration of axis ranges, axis labels and plot title - see Appendix A.5.1.2.

Export as PDF

Pops up a dialogue which will write the current plot as a PDF file.

Export as GIF

Pops up a dialogue which will output the current plot to a GIF file. The output file is just the same as the plotted image that you see. Resize the plotting window before the export to control the size of the output GIF.

Export as FITS

Pops up a dialogue which will output the plotted map as a FITS array. If only one channel is visible (either one colour channel or indexed mode) then the output FITS file will be a 2d array with dimensions the same as the displayed image. If there are multiple RGB channels then the output array will be 3d with the third dimension having an extent of 2 or 3, depending on the number of colour channels visible. In either case the FITS file will have a single (primary) HDU. Basic coordinate system information, as well as DATAMIN and DATAMAX cards, will be written to the header. The type of the output array will be double precision for weighted values, or some integer type of sufficient length for unweighted ones.

Rescale

Rescales the axes of the current plot so that it contains all the data points in the currently selected subsets. By default the plot will be scaled like this, but it it may have changed because of changes in the subset selection or from zooming in or out.

Log Intensity

Toggles between linear and logarithmic mapping for colour intensity as a function of number of counts.

Colour

Toggles between indexed and RGB modes (see the explanation above).

Show Legend

Toggles whether a legend showing how each data set is represented is visible to the right of the plot. Initially the legend is shown only if more than one data set is being shown at once.

Bigger Pixels

Increments the size of screen pixel corresponding to one density map bin.

Smaller Pixels

Decrements the size of screen pixel corresponding to one density map bin.

Draw Subset Region

Allows you to draw a region on the screen defining a new Row Subset. When you have finished drawing it, click this button again to indicate you're done. See Appendix A.5.1.6 for more details.

Subset From Visible

Defines a new **Row Subset** consisting of only the points which are currently visible on the plotting surface. See Appendix A.5.1.6 for more explanation.

The **Dataset Toolbar** contains the following options:

/ **Add/Remove dataset**

Adds/removes tabs in which the data for extra datasets can be entered. See Appendix A.5.1.1.

Weight Counts

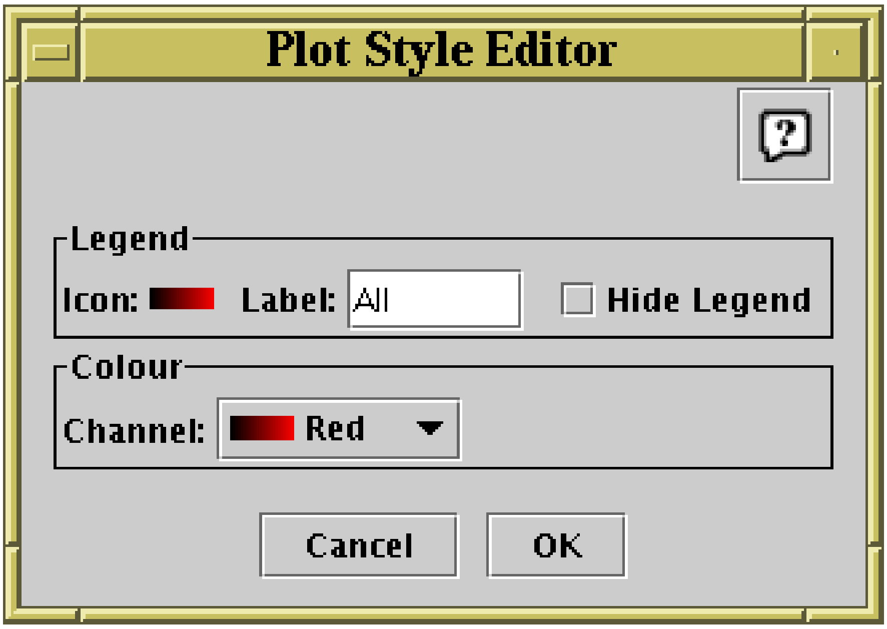
If this toggle button is on, an additional **Weight Axis** selector appears below the **X Axis** selector. If this is filled in with a column name or expression, then instead of simply accumulating the number of points per bin, each pixel will represent the sum over the weighting quantity for points in each bin. Not having a weight axis is equivalent to filling in its value with the quantity 1. Note that with weighting, the figure drawn is no longer strictly speaking a histogram or density map.

The **Export** menu provides a number of ways to export the displayed image for external viewing or analysis. As well as options to export as GIF, JPEG, EPS and FITS, there is also the option to transmit the FITS image to one or all applications listening using the SAMP or PLASTIC tool interoperability protocol which will receive images. In this way you can transmit the image directly to SAMP/PLASTIC-aware image manipulation tools such as GAIA or Aladin. See Section 9 for more information about tool interoperability.

How to set the colour channel corresponding to each dataset is explained in the following subsection.

A.5.7.1 Density Style Editor

For a density map in RGB mode, each dataset is assigned a colour channel to which it contributes. A representation of this is displayed in a button to the right of its name in the dataset selector panel at the bottom of the density map window. If you click this button the following dialogue will pop up which enables you to change the colour channel.



Label: All

□ Hide Legend

Channel: Red

Style editor dialogue for density map

The **Legend** box defines how the selected set will be identified in the legend which appears alongside the plot (though the legend will only be visible if **Show Legend** (○Aa +Bb ×Cc) is on):

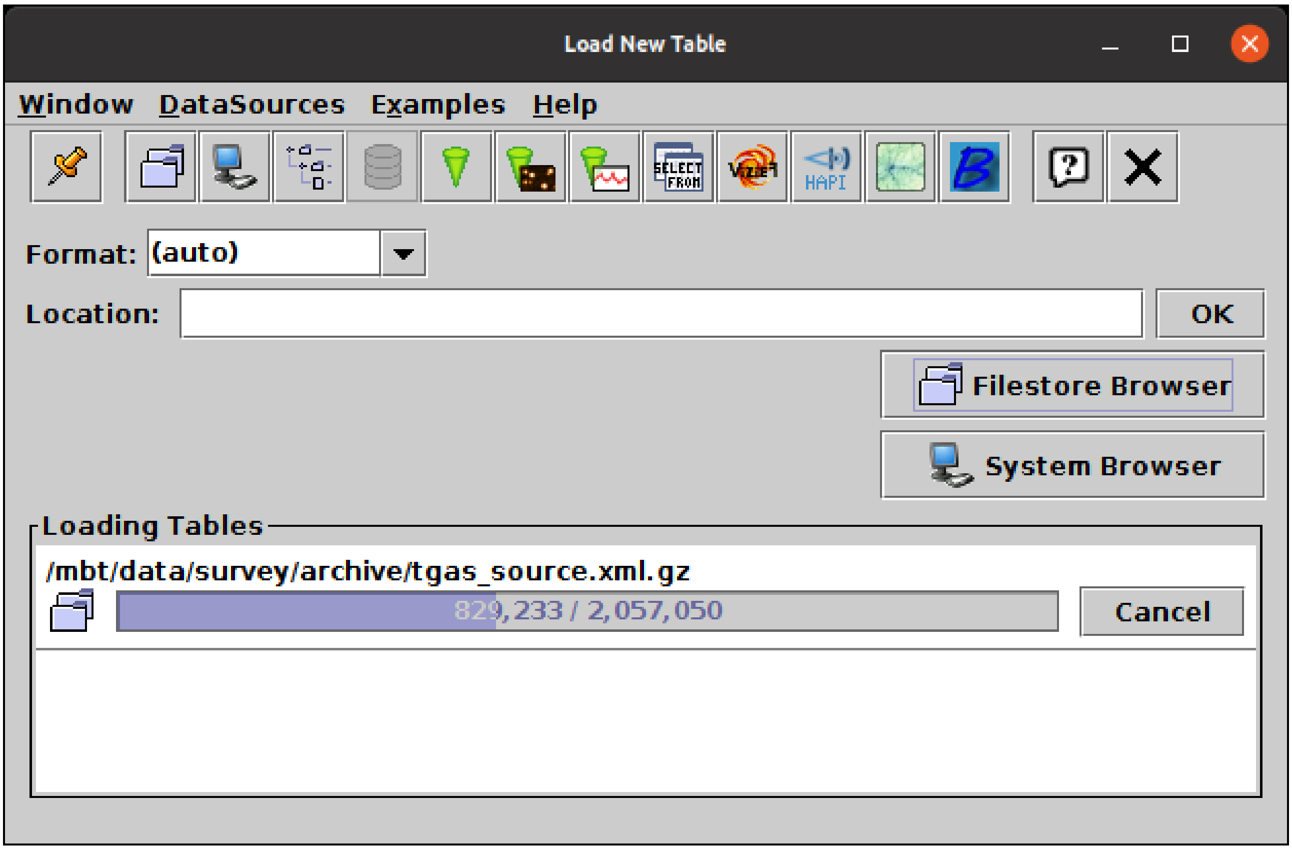
Icon
: Displays the icon which will be shown to identify the points in the selected set. Its appearance depends on the selections you make in the rest of this dialogue window.

Label
: Gives the name written in the legend to label the subset. By default this is derived from the Row Subset's name and, if it's not part of the main dataset, the name of the dataset's tab. You can type in a new value to change what is written in the legend.

Hide Legend
: If this checkbox is selected, then no entry for the selected set will appear in the legend.

The **Channel** selector allows you to select either the Red, Green or Blue channel for this dataset to contribute to. Note that this is only enabled in RGB mode; in indexed mode it has no effect and is disabled.

A.6 Load Window



Format: (auto)

Location:

/mbt/data/survey/archive/tgas_source.xml.gz

829,233 / 2,057,050

Load Window

The Load Window is used for loading tables from an external location (e.g. disk or URL) into TOPCAT. It is obtained using the **Load Table** button ( ) in the Control Window toolbar or **File** menu.

This dialogue allows you to specify a new table to open in a number of different ways, described below. If you perform a successful load using any of these options, a new entry or entries will be added into the Table List in the Control Window, which you can then use in the usual ways. If you choose a location which can't be turned into a table (for instance because the file doesn't exist), a window will pop up telling you what went wrong. The panel at the bottom of the window displays progress for tables currently loading; it is used to show progress for tables loaded from other sources too, for instance received via SAMP or specified on the command line.

In the simplest case, you can type a name into the **Location** field and hit return or the **OK** button. This location can be a filename or a URL, possibly followed by a '#' character and a 'fragment identifier' to indicate where in the file or URL the table is located; the details of what such fragment identifiers mean can be found in the relevant subsection within Section 4.1.1. Allowed URL types are described in Section 4.2. You should select the relevant table format from the **Format** selector box - in many cases (file formats that can be recognised by content such as FITS or VOTable, or files named in a conventional way such as CSV files with the "`.csv`" extension) the default **(auto)** setting will work, but in other cases TOPCAT may not be able to guess the file format, and you have to select the right one explicitly (again, see Section 4.1.1 for details).

You can also enter a scheme specification (Section 4.3) into the **Location** field. These have the form "`:<scheme-name>:<scheme-args>`" and can be used for tables that are created in some other way than reading a stream of bytes; for instance "`:skysim:1e6`" generates a simulated sky survey with a million rows.

There are many other ways of loading tables however, described in the following subsections. The

Filestore Browser and **System Browser** buttons are always visible below the location field. Depending on startup options, there may be other buttons here. There are also a number of toolbar buttons which display various load dialogues; the **DataSources** contains all of these along with some lesser-used ones. The following subsections describe most of the available options, though others may be available in your installation.

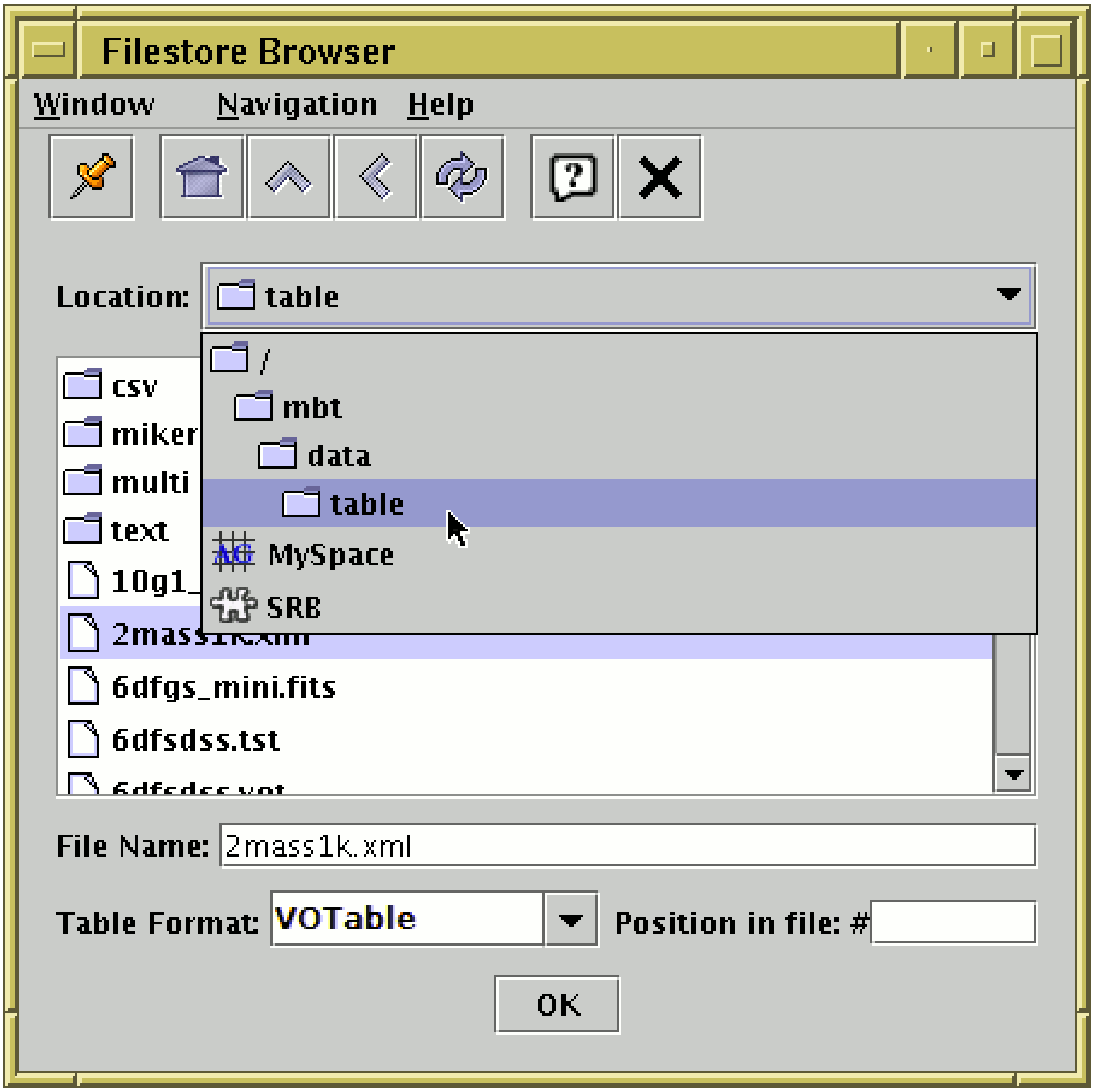
The options available on the toolbar by default are these:

- Filestore Browser
- System Browser
- Hierarchy Browser
- SQL Query
- Cone Search
- Simple Image Access (SIA) Query
- Simple Spectral Access (SSA) Query
- Table Access Protocol (TAP) Query
- VizieR Catalogue Service Query
- HAPI query
- Virgo-Millennium Simulation Query
- BaSTI Theory Database Query

All of these load dialogues have an **OK** button. Once you click it, whatever load you have specified will start. If the load takes more than a few hundredths of a second, a progress bar will appear in the **Loading Tables** panel of the load window, as in the screenshot above. This will report on how many rows have been loaded, and if known, how many there are in total. If you want to interrupt the load for any reason while it is in progress, click the **Cancel** button, and the load will cease. If the load completes without cancellation, a new table will appear in the Table List of the main Control Window, ready for use.

By default, when a table load has completed, both the Load Window itself and whichever specific load dialogue window you used will close. If you don't want this to happen use the **Stay Open** ( ) item in the **Window** menu or toolbar of the Load Window and/or specific load dialogue. If this is selected, the window will not automatically close. This can be very convenient if you want to load many tables from a similar place, for instance several files from the same directory or several cone searches to different services.

A.6.1 Filestore Browser



Location: table

File Name: 2mass1k.xml

Table Format: VOTable

Position in file: #

Filestore Browser window

By clicking the **Filestore Browser** button or toolbar button ( ) in the Load Window, you can obtain a file browser which will display the files in a given directory. The way this window works is almost certainly familiar to you from other applications.

Unlike a standard file browser however, it can also browse files in remote filestores: currently supported are MySpace and SRB. MySpace is a distributed storage system developed for use with the Virtual Observatory by the AstroGrid project, and SRB (Storage Resource Broker) is a similar general purpose system developed at SDSC. To make use of these facilities, select the relevant entry from the selector box at the top of the window as illustrated above; this will show you a **Log In** button which prompts you for username, password etc, and you will then be able to browse the remote filestore as if it were local. The same button can be used to log out when you are finished, but the session will be logged out automatically when TOPCAT ends in any case.

The browser initially displays the current directory, but this can be changed by typing a new directory into the **File Name** field, or moving up the directory hierarchy using the selector box at the top, or navigating the file system by clicking the up-directory button or double-clicking on displayed directories. The initial default directory can be changed by setting the `user.dir` system property (Section 10.2.3).

All files are shown, and there is no indication of which ones represent tables and which do not. To open one of the displayed files as a table, double-click on it or select it by clicking once and click the **Open Table** button. The **Table Format** selector must be set correctly: the "(auto)" setting will automatically detect the format of VOTable or FITS tables, otherwise you will need to select the option describing the format of the file you are attempting to load (see Section 4.1.1). If you pick a file which cannot be converted into a table an error window will pop up.

In most cases, selecting the file name and possibly the format is all you need to do. However, the **Position in file** field allows you to add information about where in the file the table you want is situated. The meaning of this varies according to the file format: for FITS files, it is the index or EXTNAME of the HDU containing the table you're after (see Section 4.1.1.1 for details), and for VOTables it is the index of the TABLE element (the first TABLE encountered is numbered 0). If you leave this blank, you will get all the tables in the file; this is usually just one table, since most file formats cannot accommodate more than one table per file, and even for those which can (FITS and VOTable) most files contain only a single file in any case.

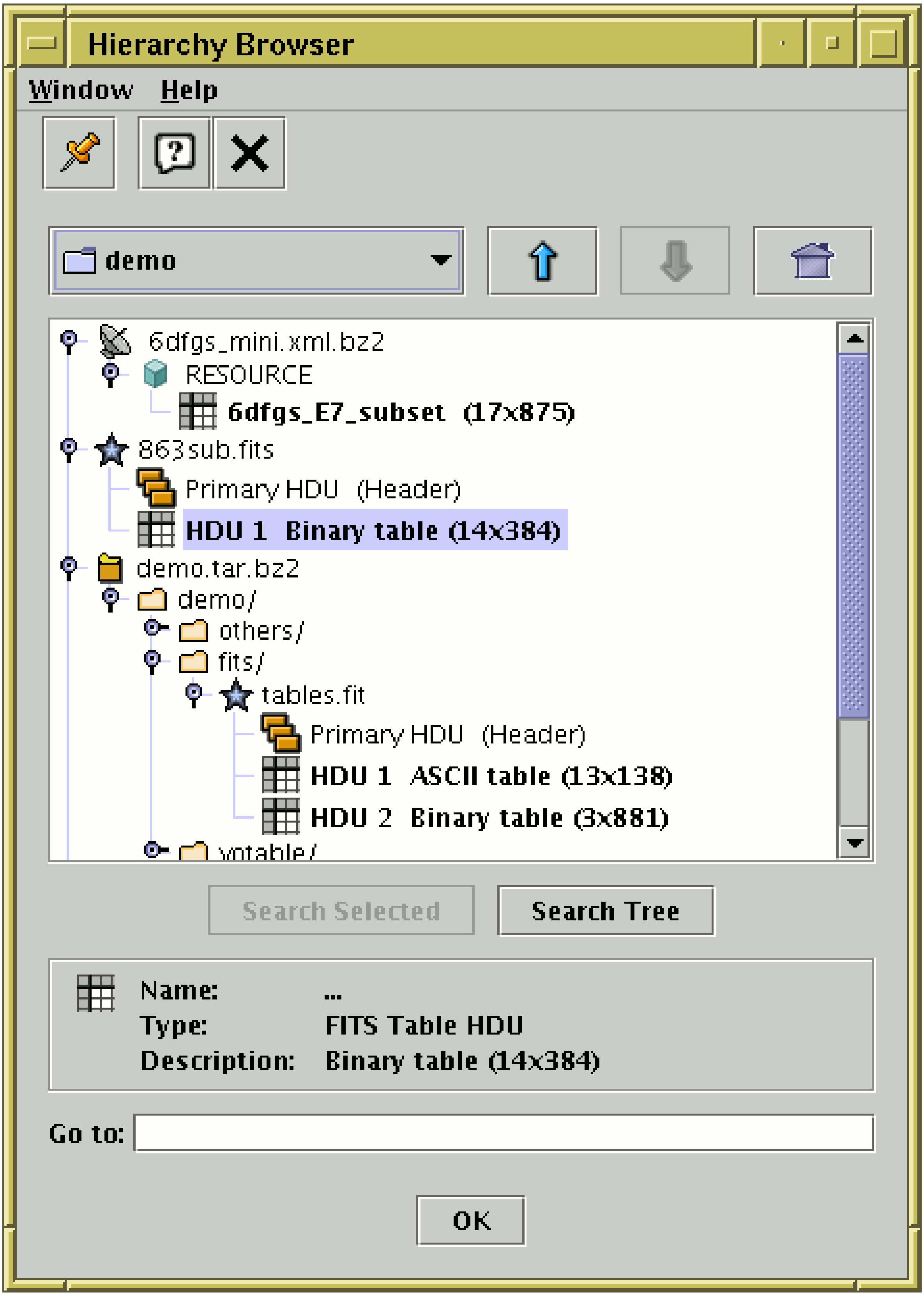
For a more table-aware view of the file system, use the Hierarchy Browser (Appendix A.6.3) instead.

A.6.2 System Browser

By clicking the **System Browser** button or toolbar button (🖥) in the Load Window, you can use your Operating System's native file browser to choose a file to load from. What this looks like is entirely dependent on your OS; it may or may not contain system-specific features like shortcuts to commonly-used directories.

Since TOPCAT has no control over the way this dialogue looks, it cannot contain the **Table Format** selector, and certain other things such as load cancel may not work as well as for the other dialogue types. To select the table format, you will need to use the selector in the main **Load Window**.

A.6.3 Hierarchy Browser



6dfgs_mini.xml.bz2
RESOURCE
6dfgs_E7_subset (17x875)
863sub.fits
Primary HDU (Header)
HDU 1 Binary table (14x384)
demo.tar.bz2
demo/
others/
fits/
tables.fit
Primary HDU (Header)
HDU 1 ASCII table (13x138)
HDU 2 Binary table (3x881)
votable/

Name:	...
Type:	FITS Table HDU
Description:	Binary table (14x384)

File load Hierarchy Browser window

By selecting the **Hierarchy Browser** button ( ) from the Load Window's toolbar or **DataSources** menu, you can obtain a browser which presents a table-aware hierarchical view of the file system. (Note that a freestanding version of this panel with additional functionality is available in the separate Treeview application).

This browser resembles the Filestore Browser in some ways, but with important differences:

- It shows the file system in a 'tree-like' fashion, so that multiple levels of the hierarchy are displayed at once
- It understands which items in the hierarchy represent tables that can be automatically detected and which represent other kinds of object (for instance directories, zip files, or plain text files)
- It can look inside hierarchical files, so for instance it can investigate a Tar or Zip archive which may contain table entries, or display multiple tabular HDUs in a FITS file, or multiple TABLE elements at different levels in a VOTable document

The main part of the window shows a "tree" representation of the hierarchy, initially rooted at the current directory (the initial directory can be changed by setting the `user.dir` system property (Section 10.2.3)). Each line displayed represents a "node" which may be a file or some other type of item (for instance an HDU in a FITS file or an entry in a tar archive). The line contains a little icon which indicates what kind of node it is and a short text string which gives its name and maybe some description. Nodes which represent tables are indicated by the ▦ icon. For nodes which have some internal structure there is also a "handle" which indicates whether they are collapsed (⊸) or expanded (⫯). You can examine remote filespaces (MySpace, SRB) as well as local ones in the same way as with the Filestore Browser.

If you select a node by clicking on it, it will be highlighted and some additional description will appear in the panel below the hierarchy display. The text is in **bold** if the node in question can be opened as a table, and non-bold if it is some non-table item.

Note: an important restriction of this browser is that it will only pick up tables which can be identified automatically - this includes FITS and VOTable files, but does not include text-based formats such as ASCII and Comma-Separated Values. If you want to load one of the latter types of table, you will need to use one of the other load methods and specify table format explicitly.

You can see how this browser works on an example directory of tables as described in Appendix A.6.13.

Note that this window requires certain optional components of the TOPCAT installation, and may not always be available.

A.6.3.1 Navigation

Navigation is a bit different from navigation in the **File Browser** window. To expand a node and see its contents, click on its handle (clicking on the handle when it is expanded will collapse it again). When you have identified the table you want to open, highlight it by clicking on it, and then click the **Open Table** button at the bottom.

To move to a different directory, i.e. to change the root of the tree which is displayed, use one of the buttons above the tree display:

Selector box
Allows you to move straight to any directory higher up than the current one.

⇧ **Up**
Moves to the parent of the current directory.

⇩ **Down**
Moves to the currently selected (highlighted) node.

Home

Moves to the user's home directory.

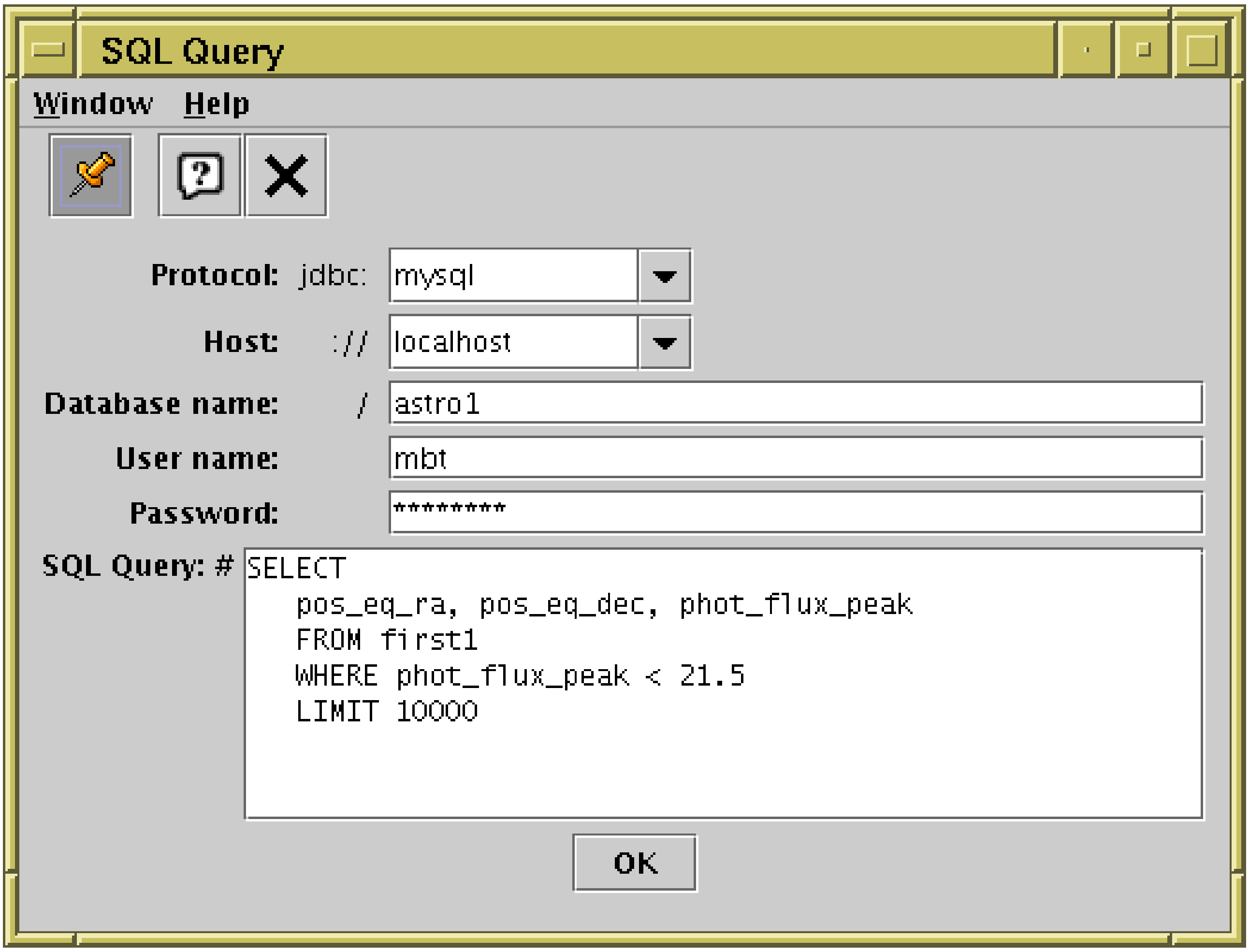
Alternatively, you can type in a new directory in the **Go to** field at the bottom of the window.

(In fact the above navigation options are not restricted to changing the root to a new directory, they can move to any node in the tree, for instance a level in a Tar archive.)

A.6.3.2 Table Searches

There are two more buttons in the browser, **Search Selected** and **Search Tree**. These do a recursive search for tables in all the nodes starting at the currently selected one or the current root respectively. What this means is that the program will investigate the whole hierarchy looking for any items which can be used as tables. If it finds any it will open up the tree so that they are visible (note that this doesn't mean that the only nodes revealed will be tables, ancestors and siblings will be revealed too). This can be useful if you believe there are a few tables buried somewhere in a deep directory structure or Tar archive, but you're not sure where. Note that this may be time-consuming - a busy cursor is displayed while the search is going on. Changing the root of the tree will interrupt the search.

A.6.4 SQL Query



Protocol: jdbc: mysql

Host: :// localhost

Database name: / astro1

User name: mbt

Password: *********

SQL Query: #

```
SELECT
   pos_eq_ra, pos_eq_dec, phot_flux_peak
   FROM first1
   WHERE phot_flux_peak < 21.5
   LIMIT 10000
```

SQL Query Dialogue

If you want to read a table from an SQL database, you can use a specialised dialogue to specify the

) or **DataSources** menu.

This provides you with a list of fields to fill in which make up the query, as follows:

Protocol
The name of the appropriate JDBC sub-protocol. This is defined by the JDBC driver that you are using, and is for instance "`mysql`" for MySQL's Connector/J driver or "`postgresql`" for PostgreSQL's JDBC driver.

Host
The hostname of the machine on which the database resides (may be "`localhost`" if the database is local).

Database name
The name of the database.

User name
The username under which you wish to access the database. This is not strictly necessary if there is no access control for the database in question.

Password
The password for the given username. Again, whether this is necessary depends on the access policy of the database.

SQL Query
The text of the query which will define the resulting table. If you want to look at a table named XXX as it exists in the database, you can write something like "`SELECT * from XXX`". In principle any SQL query on the database can be used here, but the details of what SQL syntax is permitted may be restricted by the JDBC driver you are using.

There are a number of criteria which must be satisfied for SQL access to work within TOPCAT (installation of appropriate drivers and so on) - see Section 10.3. If you don't take these steps, this dialogue may be inaccessible.

The way that this dialogue contacts the database makes some assumptions about how the JDBC driver works which are not always true, so for some databases and drivers it may not be possible to get it to work. It may be improved in the future; if you have particular problems, please contact the author or the mailing list.

A.6.5 Cone Search

See Appendix A.9.2 for details.

A.6.6 SIA Query

See Appendix A.9.3 for details.

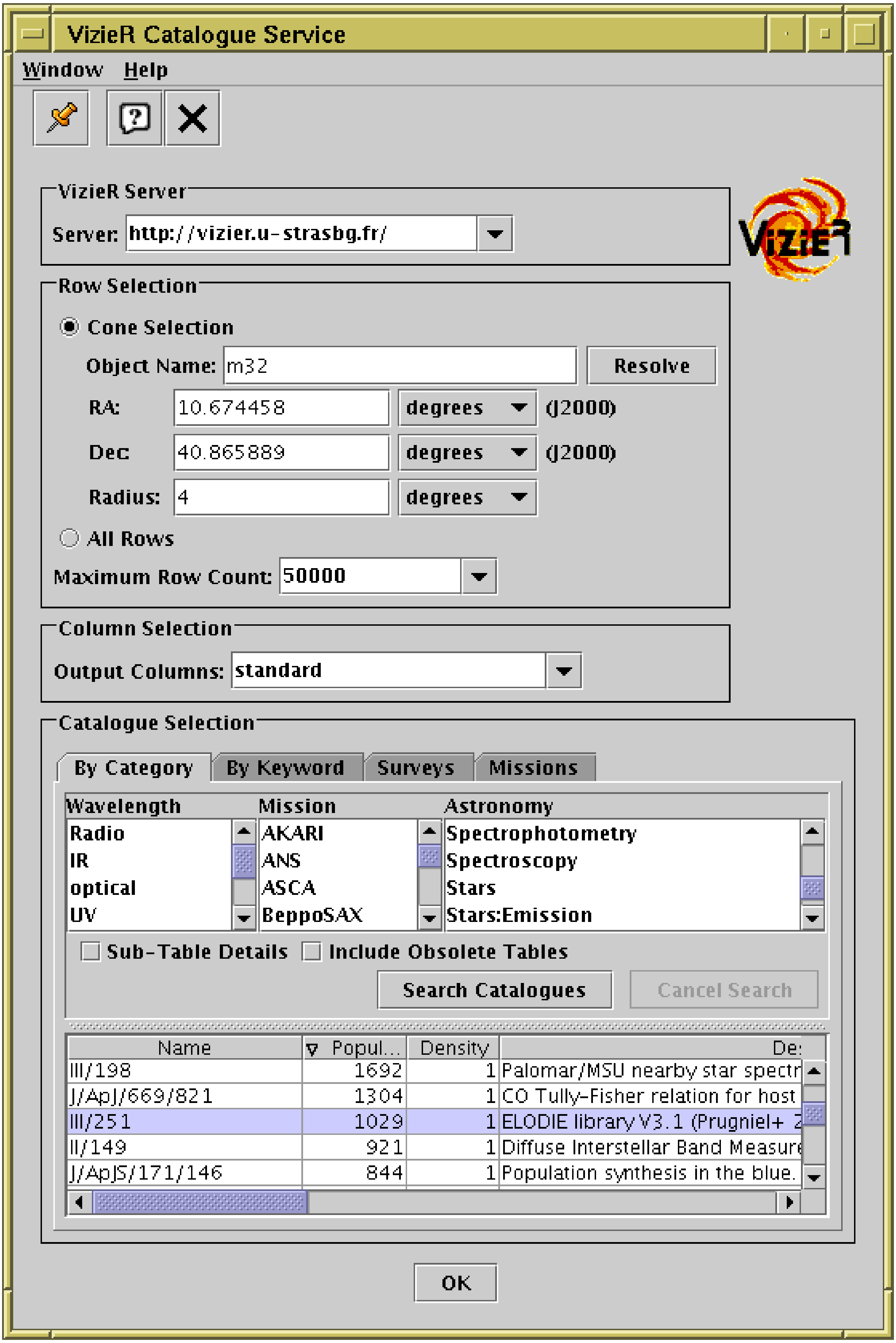
A.6.7 SSA Query

See Appendix A.9.4 for details.

A.6.8 TAP Query

See Appendix A.9.8 for details.

A.6.9 VizieR Catalogue Service Query



VizieR Server

Server: http://vizier.u-strasbg.fr/

Row Selection

● Cone Selection

Object Name: m32

RA: 10.674458 degrees (J2000)

Dec: 40.865889 degrees (J2000)

Radius: 4 degrees

○ All Rows

Maximum Row Count: 50000

Column Selection

Output Columns: standard

Catalogue Selection

By Category | By Keyword | Surveys | Missions

Wavelength	Mission	Astronomy
Radio	AKARI	Spectrophotometry
IR	ANS	Spectroscopy
optical	ASCA	Stars
UV	BeppoSAX	Stars:Emission

□ Sub-Table Details □ Include Obsolete Tables

Name	Popul...	Density	De:
III/198	1692	1	Palomar/MSU nearby star spectr
J/ApJ/669/821	1304	1	CO Tully-Fisher relation for host
III/251	1029	1	ELODIE library V3.1 (Prugniel+ 2
II/149	921	1	Diffuse Interstellar Band Measure
J/ApJS/171/146	844	1	Population synthesis in the blue.

VizieR load dialogue

The VizieR query dialogue can be opened using the **VizieR Catalogue Service** button (VizieR) in the Load Window's toolbar or the Control Window's **VO** menu. It allows you to make queries directly to the VizieR service operated by CDS. VizieR is a comprehensive library of very many published astronomical catalogues. These items can equally be accessed from the web or other interfaces, but this load dialogue makes it convenient to load data directly from VizieR into TOPCAT.

Note that VizieR's idea of a catalogue is more complex than a single table; this means that in some cases querying one of VizieR's catalogues may result in more than one table being loaded into TOPCAT (the **Sub-Table Details** checkbox described below can help to control this).

The dialogue consists of four parts: the **VizieR Server**, **Row Selection**, **Column Selection** and **Catalogue Selection** panels, arranged top to bottom in the window. These are described below.

The **VizieR Server** panel allows you to specify which VizieR server you want to use for data download. By default the server at CDS is used, but there are mirrors elsewhere, whose URLs can be chosen from the selector. If you see a popup window complaining that the server cannot be contacted, you can choose a different one; you might also want to select one that is close to you for performance reasons.

The **Row Selection** panel specifies which rows you want to retrieve from the catalogue that you will select. You can choose one of the two radio buttons according to the kind of query that you want to make:

Cone Selection

In this case you must give a central sky position and the search radius to define a cone-shaped region of interest. Rows within that range will be returned. For the central position you can either fill in the **RA** and **Dec** fields directly, or you can fill in the **Object Name** field and hit the **Resolve** button; in the latter case, a SIMBAD query will be made to determine the coordinates corresponding to the named object.

All Rows

Alternatively, you can choose to download the whole catalogue without spatial restrictions.

In either case, the **Maximum Row Count** selector indicates the largest number of rows which will be returned. If your query requests more rows than the limit given, extra rows will simply be omitted from the returned result (the limit seems to be approximate). It is possible to choose any value for this field, including very large ones or the special value "unlimited"; however consider before doing this whether you want to download a potentially very large data set. The server may in any case time out in the case of a very long connection, so it is probably not possible, even if it were desirable, to download for instance the entire 2MASS point source catalogue.

The **Column Selection** panel gives you some control over which columns will be included in the loaded table. This will include some or all of the columns the table has in the VizieR archive, and perhaps some standard ones added automatically by the service. The options are currently:

standard

Contains a selection of those columns considered most interesting by the service.

default

Contains the 'standard' columns plus numeric "_RAJ2000" and "_DEJ2000" positional columns inserted by the service; if the query is a Cone Selection rather than All Rows, it also contains a column "_r" inserted by the service giving the distance between the selected position and the row's position.

all

Contains all the columns from the archived catalogue.

VizieR experts may fill in custom column requirements here by typing them into the selector box rather than choosing one of the predefined options, for instance `-out.add=_GLON,_GLAT` would add galactic coordinates to the standard set; see http://vizier.u-strasbg.fr/doc/asu-summary.htx for more details on VizieR hacking. (In fact, this trick can be used to add VizieR parameters unrelated to column selection as well).

The **Catalogue Selection** panel offers several different ways to identify which of the catalogues in the VizieR archive you want to query. In all cases you will be presented with a JTable of VizieR catalogues, and you must select one by clicking on the relevant row. You can sort within the displayed table by clicking on the column header. **Note:** for some of these options it is necessary to fill in the Row Selection panel before you can operate them (the controls will be disabled if no row selection has been made). That is because the catalogues listed will depend on the region you are going to query; VizieR is smart enough to know which catalogues have some coverage in the region in question. The options for catalogue selection are as follows:

By Category
You may select one or more terms from one or more of the presented lists of predefined keywords in the categories **Wavelength**, **Mission** and **Astronomy** to restrict the catalogues that you are interested in. How you select multiple entries from the same list is platform-dependent, but CTRL-click may work. When you have made your selections, hit the **Search Catalogues** button, and those catalogues in the categories you have identified, and with coverage in the region defined by the Row Selection panel, will be listed below the category selection panel. Select one of these by clicking on it. The **Sub-Table Details** checkbox controls whether the list displays only top-level VizieR catalogues (each of which may contain multiple tables) or entries for each table within each catalogue as well. The **Include Obsolete Tables** checkbox controls whether just the most current, or all versions of each catalogue are shown.

By Keyword
A **Keywords** text field is shown; you may enter a space-separated list of words which will be matched against catalogue names and descriptions. When you have entered the search terms, hit the **Search Catalogues** button, and those catalogues which match your terms, and with coverage in the region defined by the Row Selection panel, will be listed below the Keywords field. Select one of these by clicking on it. The **Sub-Table Details** checkbox controls whether the list displays only top-level VizieR catalogues (each of which may contain multiple tables) or entries for each table within each catalogue as well. The **Include Obsolete Tables** checkbox controls whether just the most current, or all versions of each catalogue are shown.

Surveys
A (fairly short) list of large surveys held by VizieR is presented in a table. An indication of the size of each, in terms of number of thousands of rows, is given. Select one of these by clicking on it.

Missions
A (fairly short) list of data holdings at VizieR originating from large missions is presented in a table. An indication of the size of each, in terms of number of thousands of rows, is given. Select one of these by clicking on it.

Depending on the type of catalogue search you make, various attributes of the catalogues in question will be listed in the table for selection:

Name
Unique VizieR identifier for the catalogue

Description
Short description of contents

KRows
Approximate number of thousands of rows

Rows
Approximate number of rows

Tables
Number of sub-tables contained within the VizieR catalogue

Popularity
A measure of the number of queries on the catalogue served by VizieR

Density
A measure of the number of sources per unit solid angle on the sky

Wavelengths
Keywords describing wavelength regimes covered

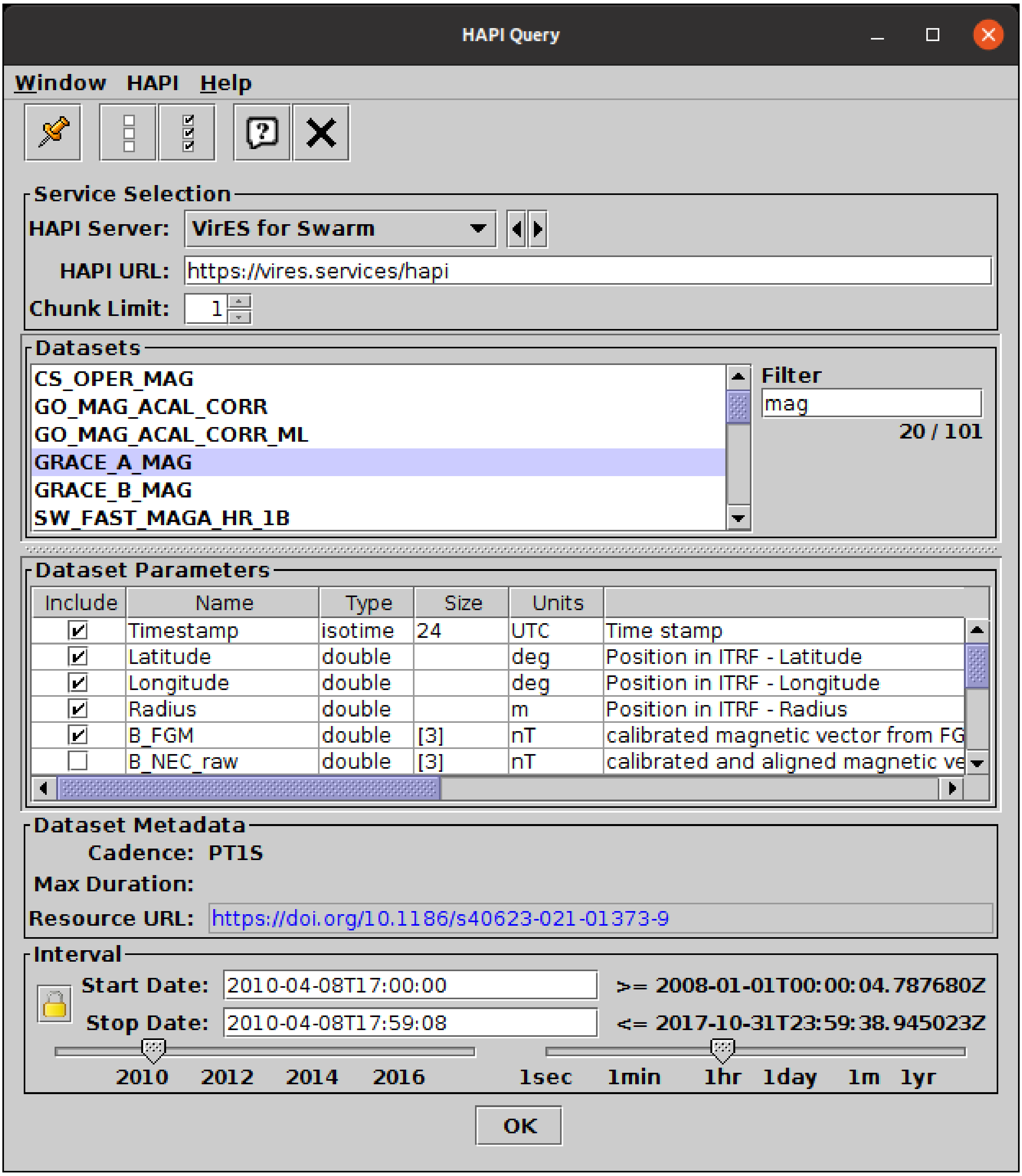
Astronomy
Keywords describing subject domain

When you have made your selection of rows, columns and catalogue you can hit the **OK** button and TOPCAT will attempt to contact the VizieR service to load the resulting table or tables. You can cancel a request in progress with the **Cancel** button.

CDS make the following request:

> *If the access to catalogues with VizieR was helpful for your research work, the following acknowledgment would be appreciated: "This research has made use of the VizieR catalogue access tool, CDS, Strasbourg, France". The original description of the VizieR service was published in A&AS 143, 23 (2000).*

A.6.10 HAPI Query



HAPI Server: VirES for Swarm

HAPI URL: https://vires.services/hapi

Chunk Limit: 1

CS_OPER_MAG
GO_MAG_ACAL_CORR
GO_MAG_ACAL_CORR_ML
GRACE_A_MAG
GRACE_B_MAG
SW_FAST_MAGA_HR_1B

Filter: mag
20 / 101

Include	Name	Type	Size	Units	
☑	Timestamp	isotime	24	UTC	Time stamp
☑	Latitude	double		deg	Position in ITRF - Latitude
☑	Longitude	double		deg	Position in ITRF - Longitude
☑	Radius	double		m	Position in ITRF - Radius
☑	B_FGM	double	[3]	nT	calibrated magnetic vector from FG
□	B_NEC_raw	double	[3]	nT	calibrated and aligned magnetic ve

Cadence: PT1S
Max Duration:
Resource URL: https://doi.org/10.1186/s40623-021-01373-9

Start Date: 2010-04-08T17:00:00 >= 2008-01-01T00:00:04.787680Z
Stop Date: 2010-04-08T17:59:08 <= 2017-10-31T23:59:38.945023Z

HAPI load dialogue

HAPI, the Heliophysics Data Application Programmer's Interface is a protocol for serving streamed time series data. This window is opened using the **HAPI Query** button (HAPI) in the Load Window's toolbar or the Control Window's **VO** menu. It lets you choose a HAPI service, select a dataset, and request time series data over a given time range. Queries can optionally be broken up into chunks to allow download of datasets larger than that allowed in a single download by the service.

The window is divided into the following parts.

Service Selection

This allows you to select and configure a HAPI service:

HAPI Server:
Choose from one of the known servers by name. By default the list of production servers is provided, but you can configure which are shown using the **HAPI|Server List** menu item.

HAPI URL:
This is filled in automatically when you choose an item using the selector above, but you can also type a URL by hand if you want to use an unregistered HAPI server.

Chunk Limit:
This configures whether data requests longer than the server normally allows will succeed. Most HAPI servers will reject requests larger than a certain size, but by increasing the value here larger datasets can be downloaded by breaking the request up into multiple chunks. Setting this value too high may allow abuse of a service, so use with care.

Datasets
The main part of this panel lists the datasets provided by the selected service. Clicking on one of these will populate the panels below with the details of the selected dataset.

To the right of the listing, search text may be entered into the **Filter** field to restrict the datasets listed (it is not case-sensitive). The text below gives the visible and total number of datasets available from the service.

Dataset Parameters
A table displays all the parameters available for the selected dataset. ("Parameter" is HAPI terminology for what is known as a table column elsewhere in TOPCAT). For each parameter, metadata items **Name**, **Type**, **Size**, **Units** and **Description** are shown. These are mostly self-explanatory, but in the **Size** column a value in square brackets indicates an array value, while an unbracketed number gives string length.

There is also a column of checkboxes marked **Include**. This determines which parameters will be included when the data is downloaded; if you don't need them all, you can uncheck some of the items (the initial timestamp item is always included). As a convenience, the toolbar buttons **Select All** (☑☑☑) and **Unselect All** (□□□) will check/uncheck all of the items at once.

Dataset Metadata
Shows some additional information about the selected dataset, if available:

Cadence:
Approximate interval between time series samples in ISO-8601 Duration format (e.g. "PT5M" means every 5 minutes).

Max Duration:
The longest interval for which the service is likely to permit a single query (though TOPCAT can split queries into multiple chunks).

Resource URL
A web page which may provide more information about the dataset. Clicking on it should show the page content in a desktop browser.

Interval
Selects the time interval over which data set values will be downloaded. The **Start Date** and **Stop Date** fields should be filled in to give the minimum and maximum epochs required. The format is a restricted form of ISO-8601, basically `yyyy-mm-ddThh:mm:ss` or `yyyy-dddThh:mm:ss` in which any of the trailing parts can be omitted, so for instance "`2001-01-01`" or "`2001-01-01T16`" are both allowed. The minimum and maximum values

allowed for the selected dataset are displayed to the right of the Start/Stop fields; the entered values must fall within these bounds.

As an alternative to typing in the dates, you can use the sliders at the bottom: the left hand one selects the start date, and the right hand one selects the length of interval. Or you can use the sliders to set an approximate interval and then edit the dates by hand.

The **Lock** button (🔒) prevents the Start/Stop dates fields from being updated automatically by the sliders or as a consequence of changing datasets, and can be useful when you are entering dates by hand.

OK Button

Once all the fields are filled in with values that specify a valid HAPI query, the **OK** button at the bottom will be enabled, and you can click it to download the data as a table.

HAPI Menu

The **HAPI** menu contains a few items that control the details of the client's interaction with HAPI services:

Server List

Determines how the items in the **HAPI Server** selector are populated; choosing one of these submenu items will re-populate that selector. The options here mostly correspond to different files at https://github.com/hapi-server/servers, but the **Fallback** option uses a hard-coded (perhaps out of date) list in case the online service is hard to reach.

Streaming Format

Determines the data format in which time series data will be downloaded from the chosen HAPI service. Options are:

- **Auto**: binary if supported by the server, CSV otherwise
- **csv**: force CSV format
- **binary**: force binary format

Fail on Limit

Configures what happens if too much data is requested from the service. If set false (the default), the maximum permitted amount of data will be downloaded, and the truncated table will be loaded into TOPCAT and marked "(OVERFLOW)" in the control window. If set true, the load will fail, and no table will be loaded. Note the **Chunk Limit** setting in the Service Selection panel influences how much data is too much.

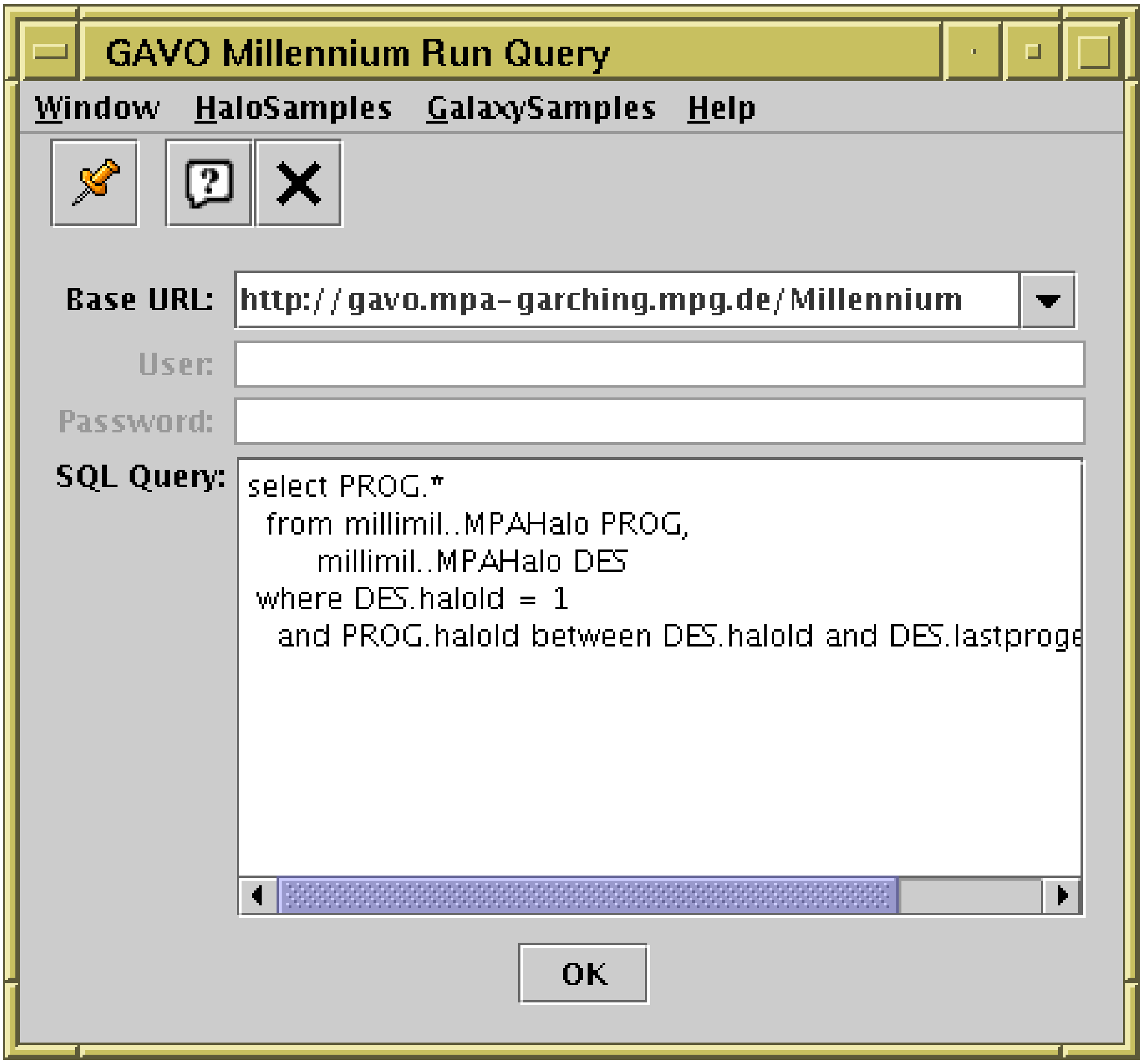
Include Header with Data

If selected, the service is requested to return the header information along with the data stream downloaded. Otherwise, the header information is acquired once with an initial metadata request and reused to interpret the data streams.

GZIP Coding

If selected, HTTP-level gzip compression is requested when downloading data from the HAPI server.

A.6.11 Virgo-Millennium Simulation Query



Base URL: http://gavo.mpa-garching.mpg.de/Millennium

User:

Password:

SQL Query:

```
select PROG.*
  from millimil..MPAHalo PROG,
       millimil..MPAHalo DES
 where DES.haloId = 1
   and PROG.haloId between DES.haloId and DES.lastproge
```

Virgo-Millennium load dialogue with an example query on the milli-Millennium database

This dialogue, selected from the Load Window's toolbar button ( ) or the main **VO** menu, permits direct SQL queries to a family of services hosting cosmological simulation databases. These services were originally developed by GAVO, the German Astrophysical Virtual Observatory, and hold results from the Millennium, Virgo and EAGLE simulations. Options are currently available to query services from MPA Garching and Durham.

To make a query, fill in the fields as required:

Base URL
This determines the database to be queried. The following options are available:

Milli-Millennium (`http://gavo.mpa-garching.mpg.de/Millennium`)
Public simulation results database, containing a fraction of the full simulation results. No username or password is required (the User and Password boxes will be disabled).

Millennium (`http://gavo.mpa-garching.mpg.de/MyMillennium`)
Full, protected simulation results database. The username and password must be filled in for these queries (register by application to GAVO).

VirgoDB (`http://virgodb.dur.ac.uk:8080/MyMillennium`)
Durham galaxy formation catalogues. Username and password required (register by application to Durham).

EAGLE (`http://virgodb.dur.ac.uk:8080/Eagle`)
Durham Eagle database. Username and password required (register by application to Durham).

Other
Other services running the similar software can be accessed from this dialogue by entering the relevant base URL here -- users of these services will know what they are. They may or may not require a username and password.

User
Password
Registration information if required (dependent on base URL as above).

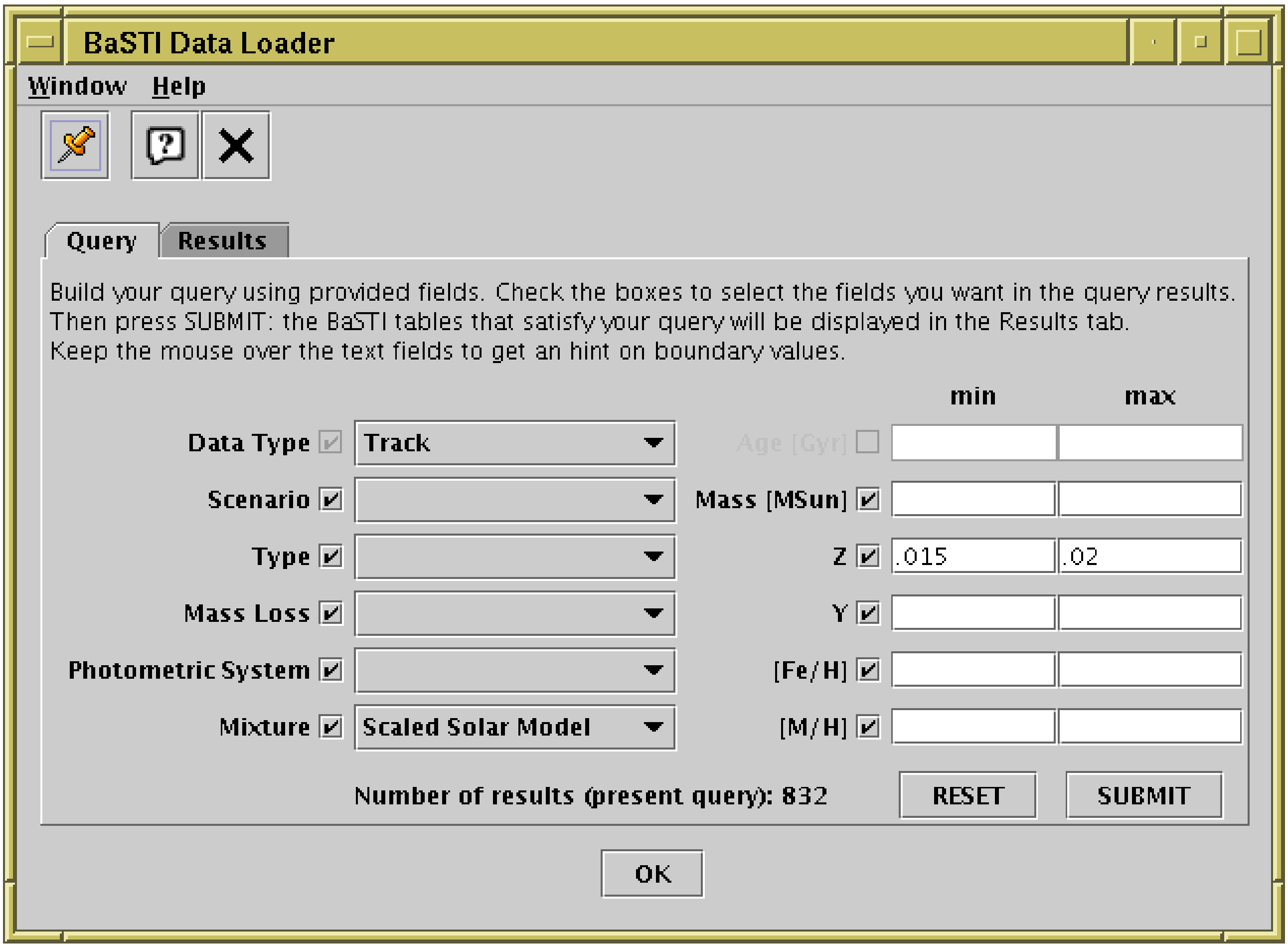
SQL Query
The text of an SQL query on the database in question.

Then click the **OK** button, and the query will execute and load the results into TOPCAT. Clicking **Cancel** while it is in progress will stop it running.

The **HaloSamples** and **GalaxySamples** menus provide some examples of queries on the Milli-Millennium database (these have been copied from the GAVO query page). If you select one of these the **SQL Query** panel will be filled in accordingly.

Much more documentation, including tutorials, descriptions of the database schemas, and information about registering for use of the authenticated services, is available online. The MPA-Garching services are documented at http://gavo.mpa-garching.mpg.de/Millennium/Help, and the Durham services at http://virgodb.dur.ac.uk/.

A.6.12 BaSTI Theory Database Query



Build your query using provided fields. Check the boxes to select the fields you want in the query results.
Then press SUBMIT: the BaSTI tables that satisfy your query will be displayed in the Results tab.
Keep the mouse over the text fields to get an hint on boundary values.

			min	max
Data Type ☑	Track	Age [Gyr] □		
Scenario ☑		Mass [MSun] ☑		
Type ☑		Z ☑	.015	.02
Mass Loss ☑		Y ☑		
Photometric System ☑		[Fe/H] ☑		
Mixture ☑	Scaled Solar Model	[M/H] ☑		

Number of results (present query): 832

BaSTI load dialog query tab

This dialogue,selected from the Load Window's toolbar button (B) or the main **VO** menu, allows direct queries to the BaSTI *(Bag of Stellar Tracks and Isochrones)* database hosted by the INAF-Teramo Astronomical Observatory. You can find more detailed documentation on the web site.

This load dialogue has two tabs: a **Query** tab to input search parameters, and a **Results** tab to display the results table with one row for each table resulting from the user's query.

The **Query** tab contains a set of parameters by which the query will be constrained, some aids to help the user while preparing the selection and two buttons. The **Reset** button simply clears query inputs and (if present) user's selections in the Results tab. The **Submit** button performs the query and switches the dialog to the Results tab. As an aid to allow a refined query without too much recursive steps, at the bottom center of the tab, a counter displays how many rows (i.e. resulting tables) the output will count. Remembering that the results will contain 30 rows at maximum, the user can than refine his/her constrains to reduce the number of results.

To do so the query helps the user in two ways mainly: automatically unselecting the unavailable parameters for a specific query (e.g. the mass range for an isochrone search) and displaying, for the ranged parameters, the value limits for that parameter, this happens just moving the mouse cursor over the input area.

Here follows a short description of the query parameters, for full details refer to the BaSTI main site.

Data Type
The type of simulation desired: isochrones, tracks or summary tables. This field is the only mandatory to submit a query.

Scenario
Stellar evolution scenario, i.e. canonical simulation or including overshooting calculations.

Type
Type of track simulation: normal or including the Asymptotic Giant Branch (AGB) simulation.

Mass Loss
Selects which value the Mass Loss parameter should have: 0.2 or 0.4 are the available choices.

Photometric System
A set of photometric systems is used to *colour* the stellar simulated tracks for comparison with observational data. This dropdown input allows for a selection within them.

Mixture
Stellar abundances mixture: scaled solar values or alpha enhanced.

Age
Age range, in values of Gyr, for the isochrones.

Mass
Mass range, in values of Solar Masses.

Z
Initial metal abundance range.

Y
Initial Helium abundance range.

[Fe/H]
Iron over Hydrogen logarithmic ratio.

[M/H]
Metals over Hydrogen logarithmic ratio.

Once the Query panel has been filled in, hit the **Submit** button. This will show the **Results** tab. This displays a table where each row refers to an available result from the BaSTI database as constrained by the user's query. On top of the table the number of results identified by the query is recalled. The user now can switch back to refine the query or selected (mouse clicking) the table/s he/she wants to load into TOPCAT. Once the selection is ready (CTRL+click or SHIFT+click for multiple selections) pressing the **OK** button will load the tables into TOPCAT.

A.6.13 Example Tables

Provided with TOPCAT are some example tables, which you can access in a number of ways. The simplest thing is to start up TOPCAT with the "`-demo`" flag on the command line, which will cause the program to start up with a few demonstration tables already loaded in.

You can also load examples in from the **Examples** menu in the Load Window however. This contains the following options:

Messier
Loads a short table containing the Messier objects

6dfgs Sample
Loads a semi-random sample of objects from the 6dfGS survey; this data is has been modified to serve as an example and is not intended for science use.

Fake Sky 1 Mrow
Loads a million-row table giving (very roughly) simulated sky objects: see Section 4.3.1 (you can easily specify similar tables with different row counts).

2d Attractor 1 Mrow
Loads a million-row table representing samples from a Clifford chaotic attractor: see Section 4.3.2 (you can easily specify similar tables with different row counts).

3d Attractor 1 Mrow
Loads a million-row table representing samples from a Rampe chaotic attractor: see Section 4.3.2 (you can easily specify similar tables with different row counts).

All Examples
Convenience item that loads all of the above tables.

Browse Demo Tree
Pops up a Hierarchy Browser (Appendix A.6.3) looking at a hierarchy of tables in different formats. This option is designed to show some of the organisational complexity which TOPCAT can handle when browsing tables.

Note these examples are a bit of a mixed bag, and are not all that exemplary in nature. They are just present to allow you to play around with some of TOPCAT's features if you don't have any real data to hand.

A.7 Save Window

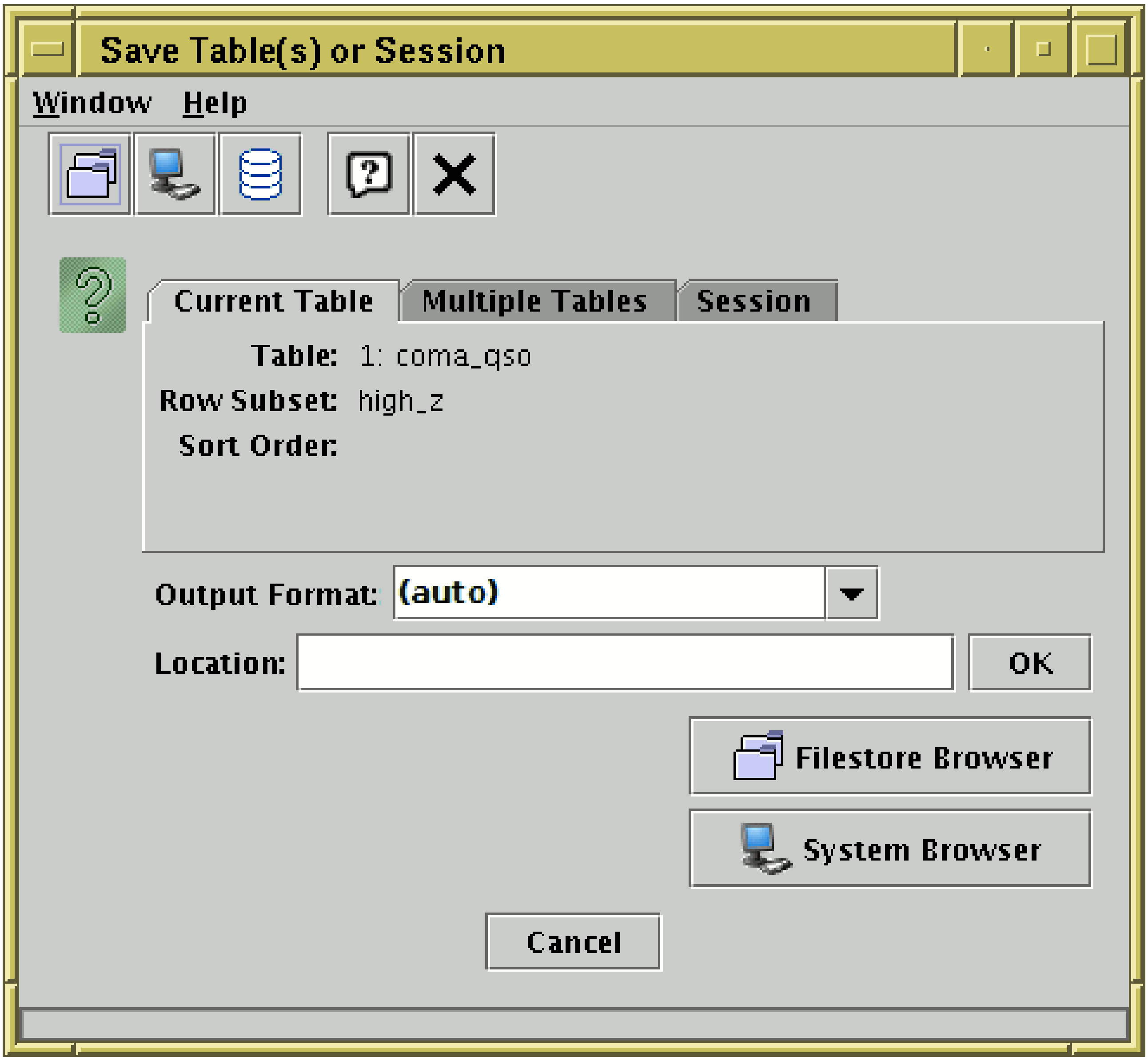


Table: 1: coma_qso
Row Subset: high_z
Sort Order:
Output Format: (auto)
Location:

Save Window

The Save Window is used to write tables out, and it is accessed using the **Save Table** button ( ) in the Control Window's toolbar or **File** menu.

The window consists of two parts. At the top is the **Content Panel**, which is used to determine exactly what table or tables are going to be saved, and below it is the **Destination Panel**, which is used to determine where they will be saved to. These panels are described separately in the subsections below.

For large tables, a progress bar will be displayed indicating how near the save is to completion. It is not advisable to edit tables which are being saved during a save operation.

In some cases, saving the table to a file with the same name as it was loaded from can cause problems (e.g. an application crash which loses the data unrecoverably). In other cases, it's perfectly OK. The main case in which it's problematic is when editing an uncompressed FITS binary table on disk. TOPCAT will attempt to warn you if it thinks you are doing something which could lead to trouble; ignore this at your own risk.

If you save a table to a format other than the one from which it was loaded, or if some new kinds of metadata have been added, it may not be possible to express all the data and metadata from the table in the new format. Some message to this effect may be output in such cases.

There is no option to compress files on output. You can of course compress them yourself once they have been written, but note that this is not usually a good idea for FITS files, since it makes them much slower to read (for TOPCAT at least).

A.7.1 Content Panel

The Content Panel is the upper part of the save window, and it is used to select what table or tables you want to save. The options are described in the following subsections.

A.7.1.1 Current Table

The **Current Table** save option saves the table currently selected in the Control Window. What is written is the Apparent Table corresponding to the current table, which takes into account any modifications you have made to its data or appearance this session. The current **Row Subset** and **Sort Order** are displayed in this window as a reminder of what you're about to save. Information about Row Subsets themselves and hidden columns will not be saved, though you can save information about subset membership by creating new boolean columns based on subsets using the **To Column** button (▥) from the Subsets Window. Alternatively you could use the Session Save option described below.

A.7.1.2 Multiple Tables

The **Multiple Tables** save option allows you to save multiple tables to the same file. If a FITS or VOTable based output format is used, this means that when the file is loaded back into TOPCAT, all the saved tables will be individually loaded in.

The list of loaded tables is shown, with a column of checkboxes on the left. By default these are selected, but you can select or unselect them by clicking. The **Select All** and **Unselect All** buttons are also provided for convenience. When the save is performed, only those tables with the checkbox selected will be saved.

As with the Current Table save, it is the Apparent Table in each case which is saved, so that only unhidden columns, and rows in the Current Subset will be included. On the right hand side of the table is information to remind you which row subset and ordering will be saved.

A.7.1.3 Session

The **Session** save option allows you to save much of the information about the loaded tables in your current TOPCAT session. Unlike the **Current Table** or **Multiple Tables** options, the whole of each loaded table, along with other information about your use of it, is saved, rather than just the Apparent Table. The saved items include:

- All rows, not just those in the current Row Subset
- All visible and hidden columns, along with their order and visibility status
- All defined Row Subsets
- The current Sort Order
- The current Row Subset
- All Activation Actions
- The label

The list of loaded tables is shown, with a column of checkboxes on the left. By default these are selected, but you can select or unselect them by clicking. The **Select All** and **Unselect All** buttons are also provided for convenience. When the save is performed, only those tables with the checkbox selected will be saved.

The tables are saved as a multi-table fits-plus (recommended) or VOTable file. This is a normal multi-table file in that any FITS or VOTable reader can read the tables it contains, but it contains some specialised metadata encoding information of special significance to TOPCAT, marking it as a saved session. The upshot is that if you save a file in this way and then load it back into TOPCAT, the tables it loads will appear in very much the same state as when you saved them, in terms of defined and current subsets, row order, visible and invisible columns, and so on. If you process it with some application other than TOPCAT, it will look quite like the table you saved, but with some additional data and metadata.

Note however that the reloaded state is not identical to that before saving. Only state belonging to tables is saved, so that for instance the state of Plot and Match windows will not be restored, and table activation actions are not saved either. It is possible that some of this may be changed in future releases.

The session save format has changed at TOPCAT version 4.5. Up to v4.4, columns and row subsets that had been defined algebraically were saved as fixed values, so the expressions were lost and it was no longer possible to edit them following a session save/load. At v4.5, the expressions themselves are saved and reloaded, which means the state is preserved better after a session save/load cycle, and may also result in smaller saved files. Post-v4.5 versions of TOPCAT should be able to load sessions saved by pre-v4.5 versions, but not vice versa. This also means that if you try to load a v4.5-format session into a non-TOPCAT reader, the algebraically-defined columns will not appear, but session-saving is not recommended for use with other table applications in any case.

A.7.2 Destination Panel

The Destination Panel is the lower part of the save window, and is used to select where and how the table or tables should be saved.

The **Output Format** selector is used to choose the format in which the table will be written from one of the supported output formats (Section 4.1.2). The available selections are somewhat different depending on what it is you are saving; for instance some formats cannot accommodate multiple tables, so these formats are not offered for the Multiple Table save. If the "(auto)" option is used, an attempt is made to guess the format based on the filename given; for instance if you specify the name "out.fits" a FITS binary table will be written. Usually just using **(auto)** or selecting one of the listed options will be appropriate, but you can type into the selector a more specific value with configuration options, for instance **votable(format=BINARY,votable=V12)**; see the output format descriptions in Section 4.1.2 for more details.

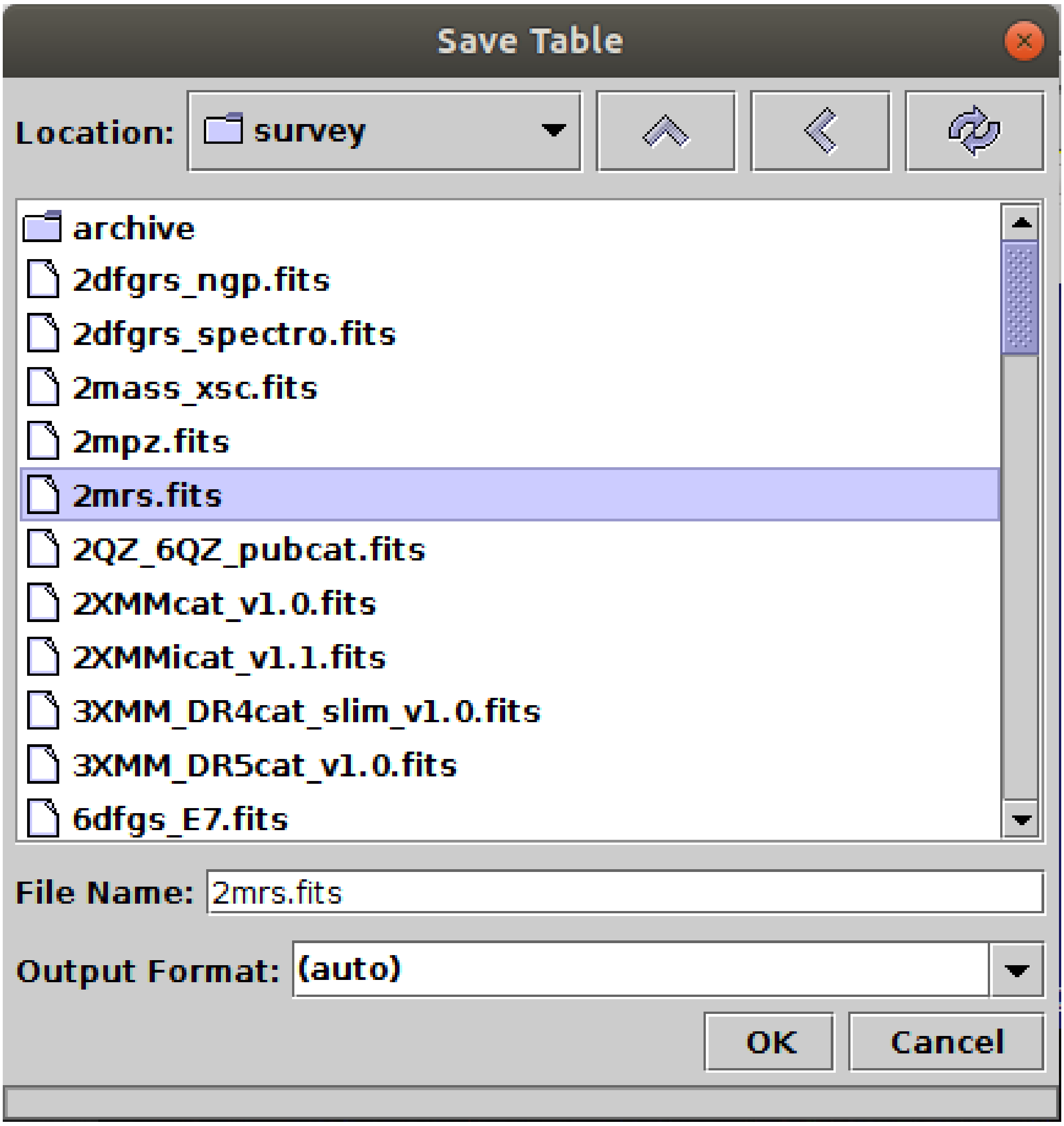
You can specify the location of the output table in these ways, which are described in the following sections:

- Type in the location directly in the Output Location field (Appendix A.7.2.1) as a filename or a URL as described in section Section 4.2
- Use the **Filestore Browser** button or toolbar button (🗂) to get a browser (Appendix A.7.2.2) that shows you local and remote files
- Use the **System Browser** button or toolbar button (🖥) to get a native browser (Appendix A.7.2.3) that uses your OS's standard file saving interface
- Use the **SQL table** toolbar button (🛢) to get the SQL Output Dialogue (Appendix A.7.2.4)

A.7.2.1 Enter Location

You can specify where to save a table by typing its location directly into the **Output Location** field of the Save Table window. This will usually be the name of a new file to write to, but could in principle be a URL or a SQL specifier.

A.7.2.2 Filestore Browser



Location: survey

File Name: 2mrs.fits

Output Format: (auto)

Filestore Browser for table saving

By clicking the **Browse Filestore** button in the **Save Table** window, you can obtain a browser which will display the files in a given directory.

The browser initially displays the current directory, but this can be changed by typing a new directory into the **File Name** field, or moving up the directory hierarchy using the selector box at the top, or navigating the file system by clicking the up-directory button or double-clicking on displayed directories. The initial default directory can be changed by setting the `user.dir` system property (Section 10.2.3).

The browser can display files in remote filestores such as on MySpace or SRB servers; see the section on the load filestore browser (Appendix A.6.1) for details.

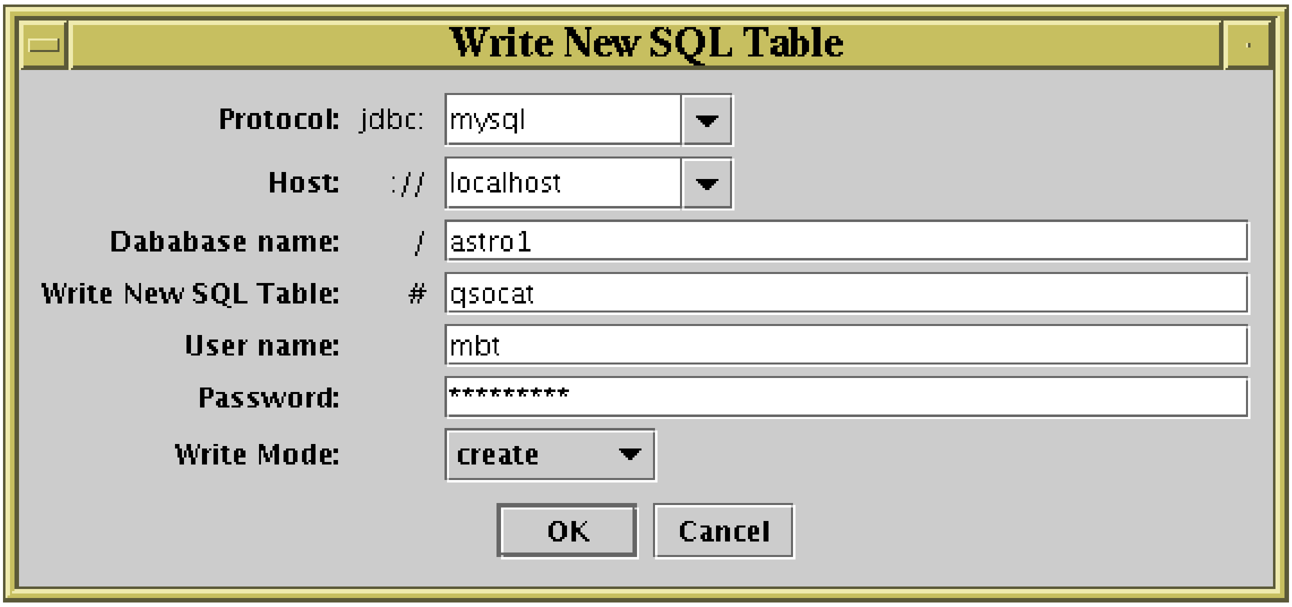
To save to an existing file, select the file name and click the **OK** button at the bottom; this will overwrite that file. To save to a new file, type it into the **File Name** field; this will save the table under that name into the directory which is displayed. You can (re)set the format in which the file will be written using the **Output Format** selector box on the right (see Appendix A.7.2 for discussion of this selector).

A.7.2.3 System Browser

By clicking the **System Browser** button or toolbar button (🖥) in the Save Window, you can use your Operating System's native file browser to decide where to save a file. What this looks like is entirely dependent on your OS; it may or may not contain system-specific features like shortcuts to commonly-used directories.

Since TOPCAT has no control over the way this dialogue looks, it cannot contain the **Output Format** selector, and certain other things such as save cancel may not work as well as for the other dialogue types. To select the output table format, you will need to use the selector in the **Save Window**.

A.7.2.4 SQL Output Dialogue



Write New SQL Table

Protocol: jdbc: mysql

Host: :// localhost

Dababase name: / astro1

Write New SQL Table: # qsocat

User name: mbt

Password: *********

Write Mode: create

SQL table writing dialogue

If you want to write a table to an SQL database, you can use a specialised dialogue to specify the table destination by clicking the **SQL Table** button in the **Save Table** window.

This provides you with a list of fields to fill in which define the new table to write, as follows:

Protocol
The name of the appropriate JDBC sub-protocol. This is defined by the JDBC driver that you are using, and is for instance "`mysql`" for MySQL's Connector/J driver or "`postgresql`" for PostgreSQL's JDBC driver.

Host
The hostname of the machine on which the database resides (may be "`localhost`" if the

database is local).

Database name
The name of the database.

New table name
The name of a new table to write into the given database. Subject to user privileges, this will overwrite any existing table in the database which has the same name, so should be used with care.

User name
The username under which you wish to access the database. This is not strictly necessary if there is no access control for the database in question.

Password
The password for the given username. Again, whether this is necessary depends on the access policy of the database.

Write Mode
Options for writing the table to the database. Choose one of:

> **create**
> A new table is created for the output. If one with the given name already exists, an error results.
>
> **dropcreate**
> A new table is created for the output. If one with the given name already exists, it is dropped first.
>
> **append**
> The results are appended to an existing table with the given name. If the table has the wrong number or types of columns, an error results.

There are a number of criteria which must be satisfied for SQL access to work within TOPCAT (installation of appropriate drivers and so on) - see the section on JDBC configuration (Section 10.3). If you don't take these steps, this dialogue may be inaccessible.

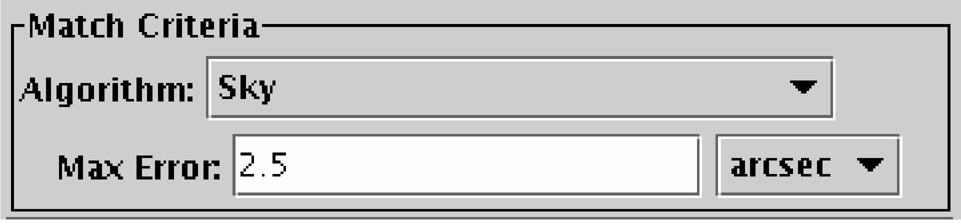
A.8 Match Windows

This section documents the windows used to crossmatch rows either between two or more local tables or within a single table. This topic is discussed in more detail in Section 5. Windows for other kinds of joins are described elsewhere: concatenation in Appendix A.11.2 and matching with remote tables via VO services in Appendix A.9 and against VizieR and SIMBAD via the CDS X-Match service in Appendix A.11.1.

The next subsection describes the features which are common to the different types of match window, and the following subsections detail the operation of each distinct match window (internal, pair and multi-table).

A.8.1 Common Features

A.8.1.1 Match Criteria



Match Criteria

Algorithm: Sky

Max Error: 2.5 arcsec

Match Criteria panel. The details will differ depending on what match type is chosen.

The match criteria box allows you to specify what counts as a match between two rows. It has two parts. First, you must select one of the options in the **Algorithm** selector. This effectively selects the coordinate space in which rows will be compared with each other. Depending on the chosen value, a number of fields will be displayed below, which you must fill in with values that determine how close two rows have to be in terms of a metric on that space to count as a match.

The following match types (algorithm values) are offered:

Sky
: Comparison of positions on the celestial sphere. In this case you will need to specify columns giving Right Ascension and Declination for each table participating in the match. The Max Error value you must fill in is the maximum separation of matched points around a great circle.

Sky with Errors
: The matching is like that for the **Sky** option above, but an error radius (positional uncertainty) is given for each row in the input tables, rather than just a single value for the whole match. Along with the Right Ascension and Declination columns, you also specify an Error column which gives the error radius corresponding to that position. Two rows are considered to match when the separation between the two RA,Dec positions is no larger than the sum of the two Error values for the corresponding rows. The Scale value should be set to a rough average of the per-row errors. It is used only to set a sensible default for the Healpix-k tuning parameter, and its value does not affect the result. If Healpix-k is set directly (see Appendix A.8.1.3), Scale is ignored. **Note:** the semantics of this matcher have changed slightly since version 3.8 of TOPCAT and earlier. In earlier versions the single parameter was named Max Error and provided an additional constraint on the maximum accepted separation between matched objects. For most uses, the old and new behaviours are expected to give the same results, but in cases of difference, the new behaviour is more likely what you want.

Sky Ellipses
: Compares elliptical regions on the sky for overlap. You will need to specify columns giving the central position, major and minor radii, and position angle of the ellipse. Two rows are considered to match if there is any overlap between the ellipses. The goodness of match is a normalised generalisation of the symmetrical case used by the **Sky with Errors** option above. The position angle is measured from north to the semi-major axis, in the direction towards the positive RA axis. The Scale value should be set to a rough average of the major radii; It is used only to set a sensible default for the Healpix-k tuning parameter, and its value does not affect the result. If Healpix-k is set directly (see Appendix A.8.1.3), Scale is ignored. The calculations are approximate since in some cases they rely on projecting the ellipses onto a Cartesian tangent plane before evaluating the match, so for larger ellipses the criterion will be less exact. For objects the size of most observed stars or galaxies, this approximation is not expected to be problematic.

Sky 3D
: Comparison of positions in the sky taking account of distance from the observer. In this case you will need to specify columns giving Right Ascension and Declination in angular units, as well as distance from the origin in arbitrary units for each table participating in the match. The Error value is a maximum separation in Cartesian space of matched points in the same units as the radial distance.

Exact Value

Requires exact matching of values. In this case you will need to specify the column containing the match key for each table participating in the match; this might typically be an object name or index number. Two rows count as matching if they have exactly the same entry in the specified field, except rows with a null value in that column, which don't match any other row. Note that the values must also be of the same type, so for instance a Long (64-bit) integer value will not match an Integer (32-bit) value.

N-dimensional Cartesian

Comparison of positions in an isotropic N-dimensional Cartesian space. In this case you will need to specify N columns giving coordinates for each table participating in the match. The Error value is the maximum spatial separation of matched points. Currently the highest dimensionality you can select is 3-d - does anyone want a higher number?

N-dimensional Cartesian, Anisotropic

Comparison of positions in an N-dimensional Cartesian space with an anisotropic metric. In this case you will need to specify N columns giving coordinates for each table participating in the match, and an error distance for each of these dimensions. Points P1 and P2 are considered to match if P2 falls within the ellipsoid with radii given by the error distances, centered on P1. This kind of match will typically be used for non-'spatial' spaces, for instance (magnitude,redshift) space, in which the metrics in different dimensions are not related to each other. Currently the highest dimensionality you can select is 4-d - does anyone want a higher number?

N-dimensional Cuboids

This matching is like that for **N-dimensional Cartesian** above, but points are considered to match if they fall within the same cuboid rather than ellipsoid. The error values are the half-axis lengths of the cuboid, like rectangular "radii". This kind of match is suitable for grouping items into regularly-spaced pixels, though it's not a very efficient way of doing that.

N-dimensional Cartesian with Errors

The matching is like that for the **N-dimensional Cartesian** option above, but an error radius (positional uncertainty) is given for each row in the input tables, rather than just a single value for the whole match. Along with the Cartesian coordinate columns, you also specify an Error column which gives the error radius corresponding to that position. Two rows are considered to match when the separation between the two positions is no larger than the sum of the two Error values for the corresponding rows. The Scale parameter should be approximately the characteristic size of the per-object error values. Its value, in conjunction with the Bin Factor tuning parameter, affects the performance of the match but not the result.

2-d Cartesian Ellipses

Compares elliptical regions in the plane for overlap. You will need to specify columns giving the central position, major and minor radii, and orientation angle of the ellipse. Two rows are considered to overlap if there is any overlap between the ellipses. The goodness of match is a generalisation of the symmetrical case used by the **2-d Cartesian with Errors** option above. The orientation angle is measured anticlockwise from the X-axis to the ellipse major axis. The Scale parameter should be set to a rough average of the major radii. Its value, in conjunction with the Bin Factor tuning parameter, affects the performance of the match but not the result.

Sky + X

Comparison of positions on the celestial sphere with an additional numeric constraint. This is a combination of the **Sky** and **1-d Cartesian** matches above, so the columns you need to supply are RA, Dec and one extra, and the errors are angular separation on the sky and the error in the extra column. A match is registered if it matches in both of the constituent tests. You could use this for instance to match objects which are both close on the sky and have similar luminosities. The "distance" measure for **Best*** matches scales the Sky distance and X distance by their maximum values and adds them in quadrature, so they have equal weight ($d=sqrt([r/r_{max}]^2 +[dX/dX_{max}]^2)$).

Note that in TOPCAT versions 4.3-5 and earlier a linear unscaled distance combination was used here, which did not give very meaningful Best match results.

Sky + X with Errors

Comparisons of positions on the celestial sphere with an additional numeric constraint that can itself vary per row. This is a combination of the **Sky** and **1-dimensional Cartesian with Errors** matches above, so the columns you will need to supply for each table are RA, Dec, X (the additional coordinate), and Error (the error on X). The values in the Match Criteria box are Max Error giving the sky position tolerance and Scale which should be approximately the characteristic size of the X error values. You could use this for instance to match tables by sky position and redshift, where the redshift uncertainty varies per source.

Sky + XY

Comparison of positions on the celestial sphere with two additional numeric constraints. This is a combination of the **Sky** and **2-d Anisotropic Cartesian** matches above, so the columns you need to supply are RA, Dec and two extra, and the errors are angular separation on the sky and the error radii corresponding to the extra columns. A match is registered if it matches in all of the constituent tests. You could use this for instance to match objects which are both close on the sky and have similar luminosities and redshifts. The "distance" measure for **Best*** matches scales the Sky, X and Y distances by their maximum values and adds them in quadrature, so they have equal weight ($d=sqrt([r/r_{max}]^2 +[dX/dX_{max}]^2 +[dY/dY_{max}]^2)$).

Note that in TOPCAT versions 4.3-5 and earlier a linear unscaled distance combination was used here, which did not give very meaningful Best match results.

HTM

Performs sky matching in just the same way as the **Sky** option above, but using a different algorithm (pixelisation of the celestial sphere is performed using the Hierarchical Triangular Mesh rather than the HEALPix scheme). The results in both cases should be identical, but HTM is much slower. Hence, this option is only useful for debugging. It may be withdrawn in future releases.

Depending on the match type, the units of the error value(s) you enter may be significant. In this case, there will be a unit selector displayed alongside the entry box. You must choose units which are correct for the number you enter.

More information is available in the GUI as a tooltip by hovering with the mouse pointer over the field in question.

See Appendix A.8.1.3 for information on optional *tuning parameters* which are sometimes displayed in this panel.

The matching framework is capable of performing even more complicated types of match, for instance Sky + 6-dimensional anisotropic Cartesian. These are not offered purely to avoid complicating the GUI. More flexible matching requirements in this in and other respects can be achieved by using the matching commands in STILTS instead.

A.8.1.2 Column Selection Boxes

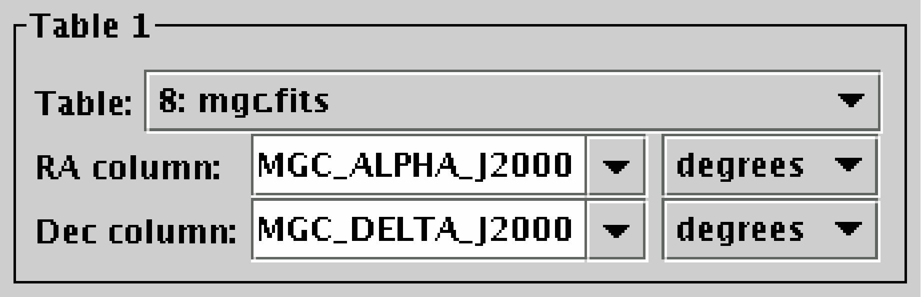


Table 1

Table: 8: mgc.fits

RA column: MGC_ALPHA_J2000 degrees

Dec column: MGC_DELTA_J2000 degrees

Column Selection Box. The details will differ depending on what match type is chosen.

The column selection boxes allow you to select which of the columns in the input tables will provide the data (the coordinates which have to match). For each table you must select the names of the required columns; the ones you need to select will depend on the match criteria you have chosen.

For some columns, such as Right Ascension and Declination in sky matches, units are important for the columns you select. In this case, there will be a selector box for the units alongside the selector box for the column itself. You must ensure that the correct units have been selected, or the results of the match will be rubbish.

In some cases these column and/or unit selectors may have a value filled in automatically (if the program thinks it can guess appropriate ones) but you should ensure that it has guessed correctly in this case. Only suitable columns are available for choosing from these column selectors; in most cases this means numeric ones.

Instead of selecting a column name from the list provided in these boxes, you may type in an expression. This can be a column name, or an algebraic expression based on column names, or even a constant. The expression language syntax is described in Section 7.

A.8.1.3 Tuning

This subsection describes the use of some toolbar buttons available in the match windows:

Tuning Parameters

Displays *tuning parameters* alongside match parameters in the Match Criteria panel.

Full Profiling

Increases the amount of timing and memory use information displayed in the Logging Panel.

Parallel Execution

Runs the match in multi-threaded mode if the tables are large and if multiple processors are available. By default this is set; unsetting it should have no effect on the result apart from (usually) slowing the matching down. A limit (currently 6) on the parallelism of the matching is imposed since adding more processors tends to lead to diminishing returns.

The parameters such as Max Error visible by default in the Match Criteria panel define what counts as a match and must be filled in for the match to proceed.

Optionally, you can see and adjust another set of parameters known as *Tuning parameters*. These will not affect the result of the match, but may affect its performance, in terms of how much CPU time and memory it takes. Most of the time, you can forget about this option, since TOPCAT attempts to pick reasonable defaults, but if your match is very slow (especially if it's unexpectedly slow given the sizes of the tables involved), or if it's failing with messages about running out of memory, then adjusting the tuning parameters may help.

To view the tuning parameters, use the **Tuning Parameters** ( ) toolbar button or menu item.

This will add display of tuning parameters to the match parameters in the Match Criteria panel. Suggested values are filled in by default, and may be affected by the match parameters that you fill in; you can play around with different values and see the effect on match performance. If you do

) toolbar button to turn on full profiling. This will cause additional information on the amount of CPU time and memory used by each step to be displayed in the logging panel at the bottom. There is a small penalty in CPU time required to gather this information, which is one reason it is not turned on by default.

What tuning parameters are available depends on the match type you have chosen. Some additional description is available as tooltips if you hover over the query field. In most cases, they are one or other of the following:

HEALPix k
Used for sky-like matches. *k* is an integer value which determines the size of pixels into which the celestial sphere is decomposed for binning rows. The legal range is between 0 (corresponding to a pixel size around 60 degrees) and 20 (around 0.2 arcsec). In HEALPix language, k is $\log_2(N_{\mathrm{side}})$.

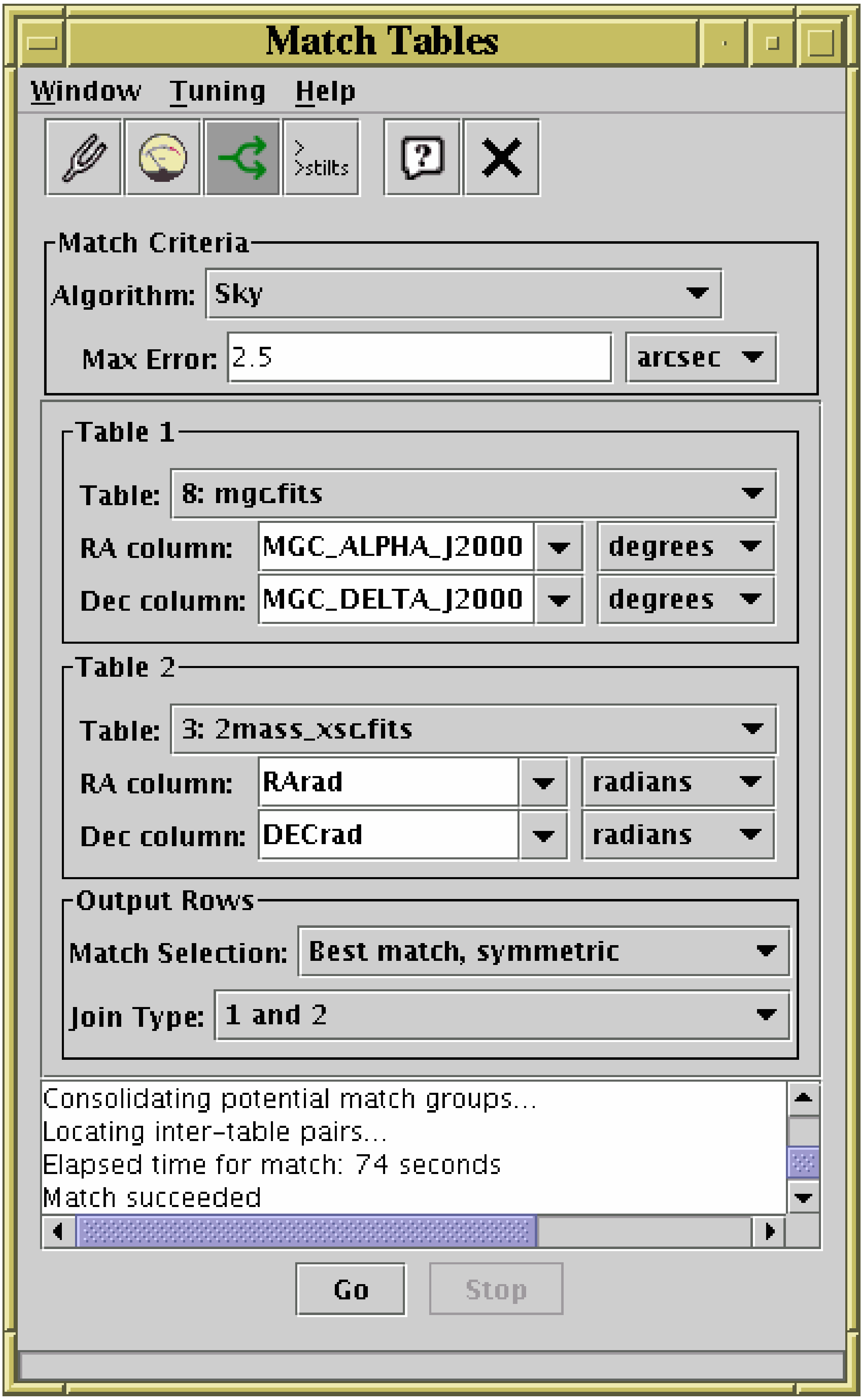
Scale Factor
Used for Cartesian-like matches. The scale factor is a floating point value which determines the size of the notional N-dimensional pixels into which the space is decomposed for binning rows, as a multiple of the match error. The smallest legal value is 1.

In either case, the number of rows per bin should be not too large, and not too small (though exactly what that means in quantitative terms is another matter). Larger bins/pixels generally mean less memory use and more CPU time, though that's not always the case.

Even if you happen to have a good understanding of how the matching algorithm works (and you probably don't), this kind of tuning is a bit of a black art, and depends considerably on the details of the particular match. In some cases however it is quite possible to improve the time taken by a match, or the size of table which can be matched in a given amount of memory, by a factor of several. If you want to try to improve performance, try the default, adjust the tuning parameters slightly, look at how the performance changes by examining the logging output, and maybe repeat to taste.

Another thing which can affect speed and memory usage is whether your Java Virtual Machine is running in 32-bit or 64-bit mode. There are pros and cons of each - sometimes one will be better, sometimes the other. If you need to improve performance, experiment!

A.8.2 Pair Match Window



Match Criteria

Algorithm: Sky

Max Error: 2.5 arcsec

Table 1

Table: 8: mgc.fits

RA column: MGC_ALPHA_J2000 degrees

Dec column: MGC_DELTA_J2000 degrees

Table 2

Table: 3: 2mass_xsc.fits

RA column: RArad radians

Dec column: DECrad radians

Output Rows

Match Selection: Best match, symmetric

Join Type: 1 and 2

```
Consolidating potential match groups...
Locating inter-table pairs...
Elapsed time for match: 74 seconds
Match succeeded
```

Pair Match Window

The Pair Match Window allows you to join two tables together side-by-side, aligning rows by matching values in some of their columns between the tables. It can be obtained using the **Pair Match** ( ) button in the Control Window toolbar or **Joins** menu.

In a typical scenario you might have two tables each representing a catalogue of astronomical objects, and you want a third table with one row for each object which has an entry in both of the original tables. An object is defined as being the same one in both tables if the co-ordinates in both rows are "similar", for instance if the difference between the positions indicated by RA and Dec columns differ by no more than a specified angle on the sky. Matching rows to produce the join requires you to specify the criteria for rows in both tables to refer to the same object and what to do when one is found - the options are discussed in more detail in Section 5.2.

The result is created from the Apparent versions of the tables being joined, so that any row subsets, hidden columns, or sorts currently in force will be reflected in the output. Progress information on the match, which may take a little while, is provided in the logging window and by a progress bar at the bottom of the window. When it is completed, you will be informed by a popup window which indicates that a new table has been created. This table will be added to the list in the Control Window and can be examined, manipulated and saved like any other. In some cases, some additional columns will be added to the output table which give you more information about how it has progressed (see Appendix A.8.2.1.

The Match Window provides a set of controls which allow you to choose how the match is done and what the results will look like. It consists of these main parts:

Match Criteria box
: Allows you to define what counts as a match between two rows.

Column Selection boxes
: Allows you to select which tables are to be joined and which columns in them supply the matching coordinates.

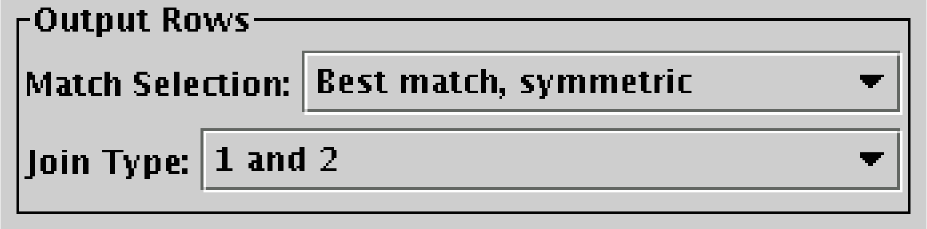
Output Rows selector
: Allows selection of which rows are to be included in the output table (for instance whether only those rows matching in both tables should be output or not).

Log window
: Reports on progress as the match is taking place. The progress bar at the bottom of the window also provides an indication of how far through each stage processing has got.

Control buttons
: The **Go** button starts the search when you are happy with the selections that you have made, and the **Stop** button interrupts it midway if you decide you no longer want the results (closing the Match Window also interrupts the calculation).

For information on the **Tuning Parameters** ( ), **Full Profiling** ( ) and **Parallel Execution** ( ) toolbar buttons, see Appendix A.8.1.3.

A.8.2.1 Output Rows Selector Box



Output Rows

Match Selection: Best match, symmetric ▾

Join Type: 1 and 2 ▾

Pair match Output Rows Selector Box

When the match is complete a new table will be created which contains rows determined by the matches which have taken place. The **Output Rows** selector box allows you to choose on what basis the rows will be included in the output table as a function of the matches that were found.

In all cases each row will refer to only one matched (or possibly unmatched) "object", so that any non-blank columns in a given row come from only rows in the input tables which match according to the specified criteria. However, you have two choices to make about which rows are produced.

The **Match Selection** selector allows you to choose what happens when a row in one table can be matched by more than one row in the other table. There are four options:

All matches
: Every match between the two tables is included in the result. Rows from both of the input tables may appear multiple times in the result.

Best match, symmetric
: The best pairs are selected in a way which treats the two tables symmetrically. Any input row which appears in one result pair is disqualified from appearing in any other result pair, so each row from both input tables will appear in at most one row in the result.

Best match for each Table 1 row
: For each row in table 1, only the best match from table 2 will appear in the result. Each row from table 1 will appear a maximum of once in the result, but rows from table 2 may appear multiple times.

Best match for each Table 2 row
: For each row in table 2, only the best match from table 1 will appear in the result. Each row from table 2 will appear a maximum of once in the result, but rows from table 1 may appear multiple times.

The "best" match in these options generally means "closest" - it is the one with the lowest match score, where the definition of this score is determined by the match criteria you have selected. The differences between the various **Best match...** options are a bit subtle. In cases where it's obvious which object in each table is the best match for which object in the other, choosing between these options will not affect the result. However, in crowded fields (where the distance between objects within one or both tables is typically similar to or smaller than the specified match radius) it will make a difference. In this case one of the asymmetric options is usually most appropriate, but you'll need to think about which of them suits your requirements. The performance (time and memory usage) of the match may also differ between these options, especially if one table is much larger than the other.

The **Join Type** selector allows you to choose what output rows result from a match in the input tables.

1 and 2
: The output table contains only rows which have an entry from both of the input tables, so that every output row represents an actual matched pair. This corresponds to an SQL *inner join*.

All from 1
: All of the matched rows are present in the output as for **1 and 2**, but additionally the unmatched rows from the first table are present with the columns from the second table blank. This corresponds to an SQL *left outer join*.

All from 2
: As for **All from 1** but the other way round. This corresponds to an SQL *right outer join*.

1 or 2
: Every row, matched and unmatched, from both of the input tables appears in the output. This is the union of rows from **All from 1** and **All from 2**. This corresponds to an SQL *full outer join*.

1 not 2

This presents all the rows in the first table which do *not* have matches in the second table. Consequently, it only contains columns from the first table, since all the entries from the second one would be blank in any case.

2 not 1

The same as **1 not 2** but the other way round.

1 xor 2

The "exclusive or" of the match - the output only contains rows from the first table which don't have matches in the second table and vice versa. It is the union of **1 not 2** and **2 not 1**.

Note that the choices of which **Match Selection** and **Join Type** to use are somewhat interlinked, and some combinations may not be very meaningful.

In most cases (all the above except for **1 not 2** and **2 not 1**), the set of columns in the output table contains all the columns from the first table followed by all the columns from the second table. If this causes a clash of column names, offending columns will be renamed with a trailing "`_1`" or "`_2`". Depending on the details of the match however, some additional useful columns may be added:

Match Score

For rows that represent a match, a numeric value representing how good the match was will usually be present. This is typically a separation in real or notional space - for instance for a **Sky** match it is the distance between the two matched celestial positions in arcseconds along a great circle. It will always be greater than or equal to zero, and a smaller value represents a better match. The name and exact meaning of this column depends on the match criteria - examine its description in the Columns Window for details.

GroupSize, GroupID

If some of the rows match more than once (which may happen for any of the Match Selection options above apart from **BEST**), two columns named **GroupID** and **GroupSize** will be added. These allow you to identify which matches are multiple. In the case of rows which represent a unique match, they are blank. But for rows which represent a set of multiple matches, the GroupSize value tells you how many rows participate in this match, and the GroupID value is an integer which is the same for all the rows which participate in the same match. So if you do a sort on the GroupID value, you'll see all the rows in the first non-unique match group together, followed by all the rows in the second non-unique group... and after them all the unique matches.

Here is an example. If your input tables are these:

```
     X          Y         Vmag
     -          -         ----
  1134.822   599.247      13.8
   659.68   1046.874      17.2
   909.613   543.293       9.3
```

and

```
    X           Y         Bmag
    -           -         ----
  909.523     543.800     10.1
  1832.114    409.567     12.3
  1135.201    600.100     14.6
   702.622   1004.972     19.0
```

then a Cartesian match of the two sets of X and Y values with an error of 1.0 using the **1 and 2** option would give you a result like this:

```
    X_1        Y_1         Vmag    X_2         Y_2          Bmag    Separation
```

```
     ---       ---         ----    ---        ---         ----    ----------
   1134.822    599.247     13.8   1135.201    600.100     14.6     0.933
    909.613    543.293      9.3    909.523    543.800     10.1     0.515
```

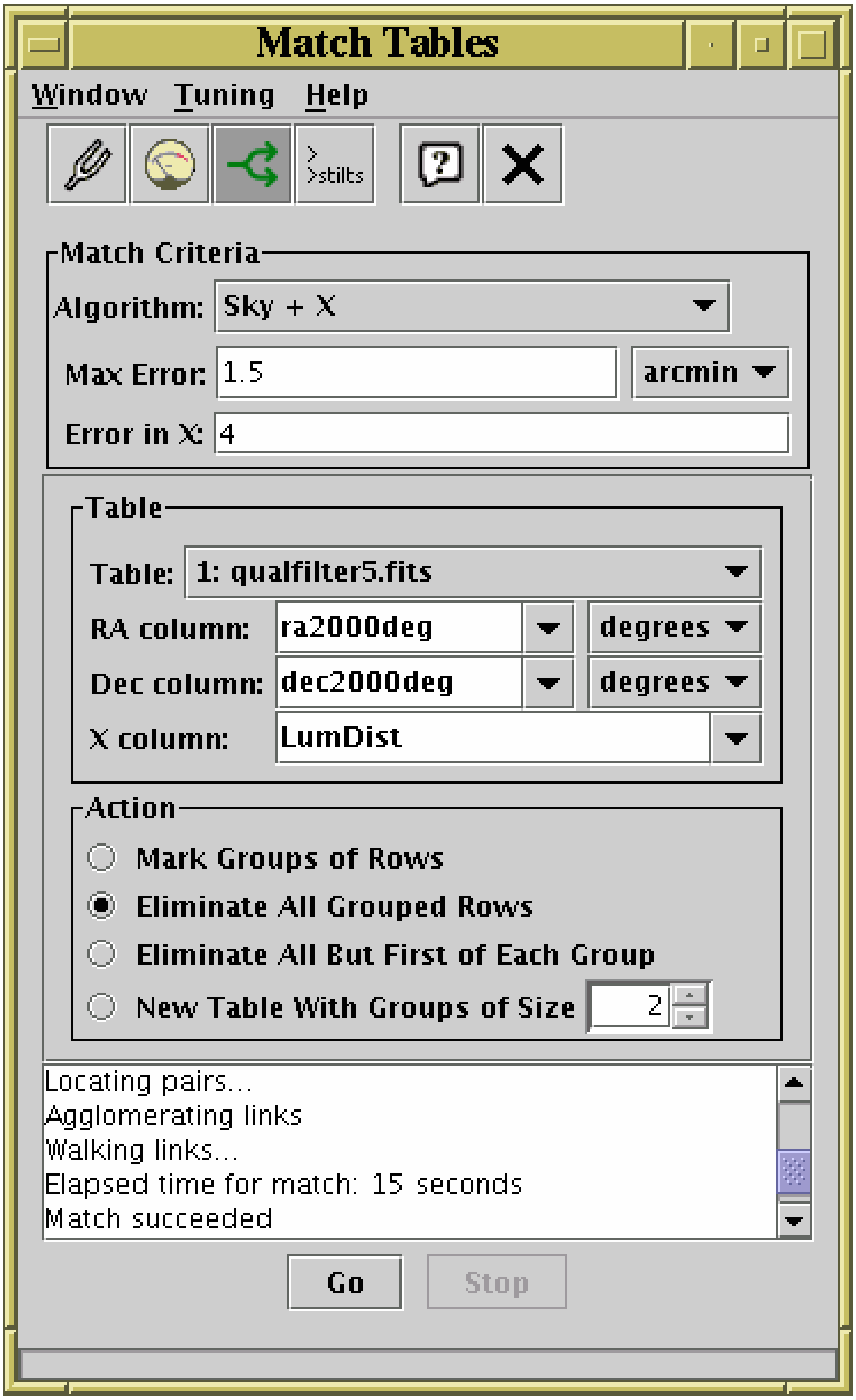
using **All from 1** would give you this:

```
     X_1       Y_1         Vmag    X_2        Y_2         Bmag    Separation
     ---       ---         ----    ---        ---         ----    ----------
   1134.822    599.247     13.8    1135.201   600.100     14.6     0.933
    659.68    1046.874     17.2
    909.613    543.293      9.3     909.523   543.800     10.1     0.515
```

and **1 not 2** would give you this:

```
     X         Y           Vmag
     -         -           ----
    659.68    1046.874     17.2
```

A.8.3 Internal Match Window



Match Criteria

Algorithm: Sky + X

Max Error: 1.5 arcmin

Error in X: 4

Table

Table: 1: qualfilter5.fits

RA column: ra2000deg degrees

Dec column: dec2000deg degrees

X column: LumDist

Action

○ Mark Groups of Rows

● Eliminate All Grouped Rows

○ Eliminate All But First of Each Group

○ New Table With Groups of Size 2

```
Locating pairs...
Agglomerating links
Walking links...
Elapsed time for match: 15 seconds
Match succeeded
```

Internal Match Window

The Internal Match Window allows you to perform matching between rows of the same table, grouping rows that have the same or similar values in specified columns and producing a new table as a result. It can be obtained by using the **Internal Match** ( ) item in the Control Window's **Joins** menu.

You might want to use this functionality to remove all rows which refer to the same object from an object catalogue, or to ensure that only one entry exists for each object, or to identify groups of several "nearby" objects in some way.

The result is created from the Apparent versions of the tables being joined, so that any row subsets, hidden columns, or sorts currently in force will be reflected in the output. Progress information on the match, which may take some time, is provided in the logging window and by a progress bar at the bottom of the window. When it is completed, you will be informed by a popup window which indicates that a new table has been created. This table will be added to the list in the Control Window and can be examined, manipulated and saved like any other.

The window has the following parts:

Match Criteria box
Allows you to define what counts as a match between two rows.

Column Selection box
Allows you to select which table to operate on and which columns supply the matching coordinates.

Match Action box
Allows you to select what will be done (what new table will be created) when the matching groups of rows have been identified.

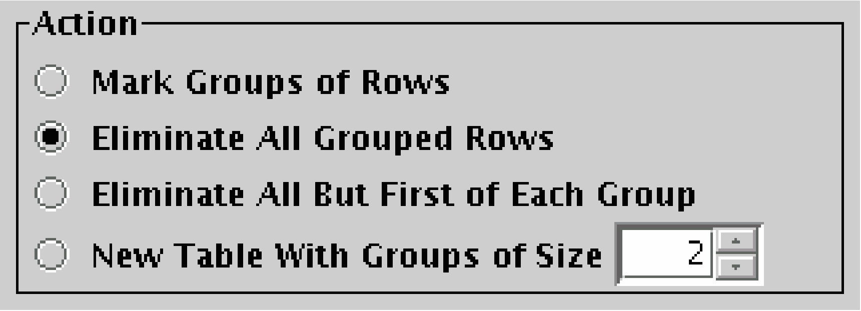
Log Panel
Displays progress as the match is taking place. The progress bar at the bottom of the window also provides an indication of how far through each stage processing has got.

Control buttons
The **Go** button starts the search when you are happy with the selections that you have made, and the **Stop** button interrupts it midway if you decide you no longer want the results (closing the Match Window also interrupts the calculation).

When considering matching beyond pairs, see the comments in Section 5.4 about the semantics of multi-object matches.

For information on the **Tuning Parameters** ( ), **Full Profiling** ( ) and **Parallel Execution** ( ) toolbar buttons, see Appendix A.8.1.3.

A.8.3.1 Internal Match Action Box



Action

○ Mark Groups of Rows
● Eliminate All Grouped Rows
○ Eliminate All But First of Each Group
○ New Table With Groups of Size 2

Internal Match Action Box

The Internal Match Action box gives a list of options for what will happen when an internal match calculation has completed. In each case a new table will be created as a result of the match. The options for what it will look like are these:

Mark Groups of Rows

The result is a table the same as the input table but with two additional columns: **GroupID** and **GroupSize**. Each group of rows which matched is assigned a unique integer, recorded in the GroupId column, and the size of each group is recorded in the GroupSize column. Rows which don't match any others (singles) have null values in both these columns. So for example by sorting the resulting table on GroupID you can group rows that match next to each other; or by sorting on GroupSize you can see all the pairs, followed by all the triples, ...

You can use this information in other ways, for instance if you create a new Row Subset using the expression "`GroupSize == 5`" you could select only those rows which form part of 5-object clusters.

Eliminate All Grouped Rows

The result is a new table containing only "single" rows, that is ones which don't match any other rows in the table according to the match criteria. Any rows which match are thrown out.

Eliminate All But First of Each Group

The result is a new table in which only one row (the first in the input table order) from each group of matching ones is retained. A subsequent internal match with the same criteria would therefore show no matches.

New Table With Groups of Size N

The result is a new "wide" table consisting of matched rows in the input table stacked next to each other. Only groups of exactly N rows in the input table are used to form the output table; each row of the output table consists of the columns of the first group member, followed by the columns of the second group member and so on. The output table therefore has N times as many columns as the input table. The column names in the new table have "_1", "_2", ... appended to them to avoid duplication.

A.8.4 Multiple Match Window

The Multiple Match Window allows you to perform matching between more than two tables. It can be obtained by using the **Triple Match**, **Quadruple Match** etc () items in the Control Window's **Joins** menu.

The result is created from the Apparent versions of the tables being joined, so that any row subsets, hidden columns, or sorts currently in force will be reflected in the output. Progress information on the match, which may take some time, is provided in the logging window and by a progress bar at the bottom of the window. When it is completed, you will be informed by a popup window which indicates that a new table has been created. This table will be added to the list in the Control Window and can be examined, manipulated and saved like any other.

The window has the following parts:

Match Criteria box

Allows you to define what counts as a match between two rows.

Column Selection box

Allows you to select which table to operate on and which columns supply the matching coordinates.

Match Action box

Allows selection of which rows are to be included in the output table (for instance whether only those matches which appear in all input tables should result in output rows).

Log window

Reports on progress as the match is taking place. The progress bar at the bottom of the window also provides an indication of how far through each stage processing has got.

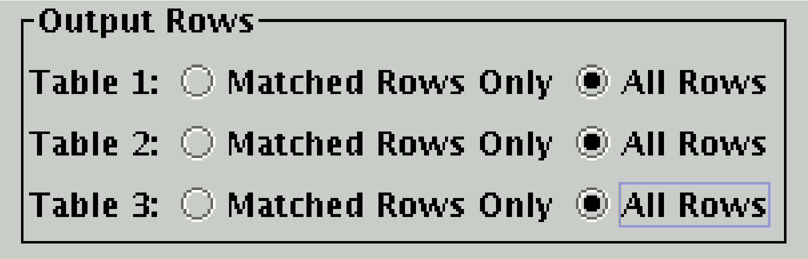
Control buttons

The **Go** button starts the search when you are happy with the selections that you have made, and the **Stop** button interrupts it midway if you decide you no longer want the results (closing the Match Window also interrupts the calculation).

See the comments in Section 5.4 about the semantics of multi-object matches.

For information on the **Tuning Parameters** ( ), **Full Profiling** ( ) and **Parallel Execution** ( ) toolbar buttons, see Appendix A.8.1.3.

A.8.4.1 Multiple Match Action Box



Output Rows

Table 1: ○ Matched Rows Only ● All Rows

Table 2: ○ Matched Rows Only ● All Rows

Table 3: ○ Matched Rows Only ● All Rows

Multiple Match Action Box

The Multiple Match Action Box allows you to choose which rows appear in the output table following a successful multi-table match. For each input table you have two options to choose from:

Matched Rows Only

Rows will appear in the output table only if they contain an entry from the table in question (but see below).

All Rows

Rows will appear in the output table if they contain an entry from the table in question, regardless of constraints on other input tables (overrides the previous option).

At present the multi-table options available within TOPCAT are not so comprehensive as those provided by the corresponding STILTS command `tmatchn`. This may be improved in future.

A.9 VO Data Access Windows

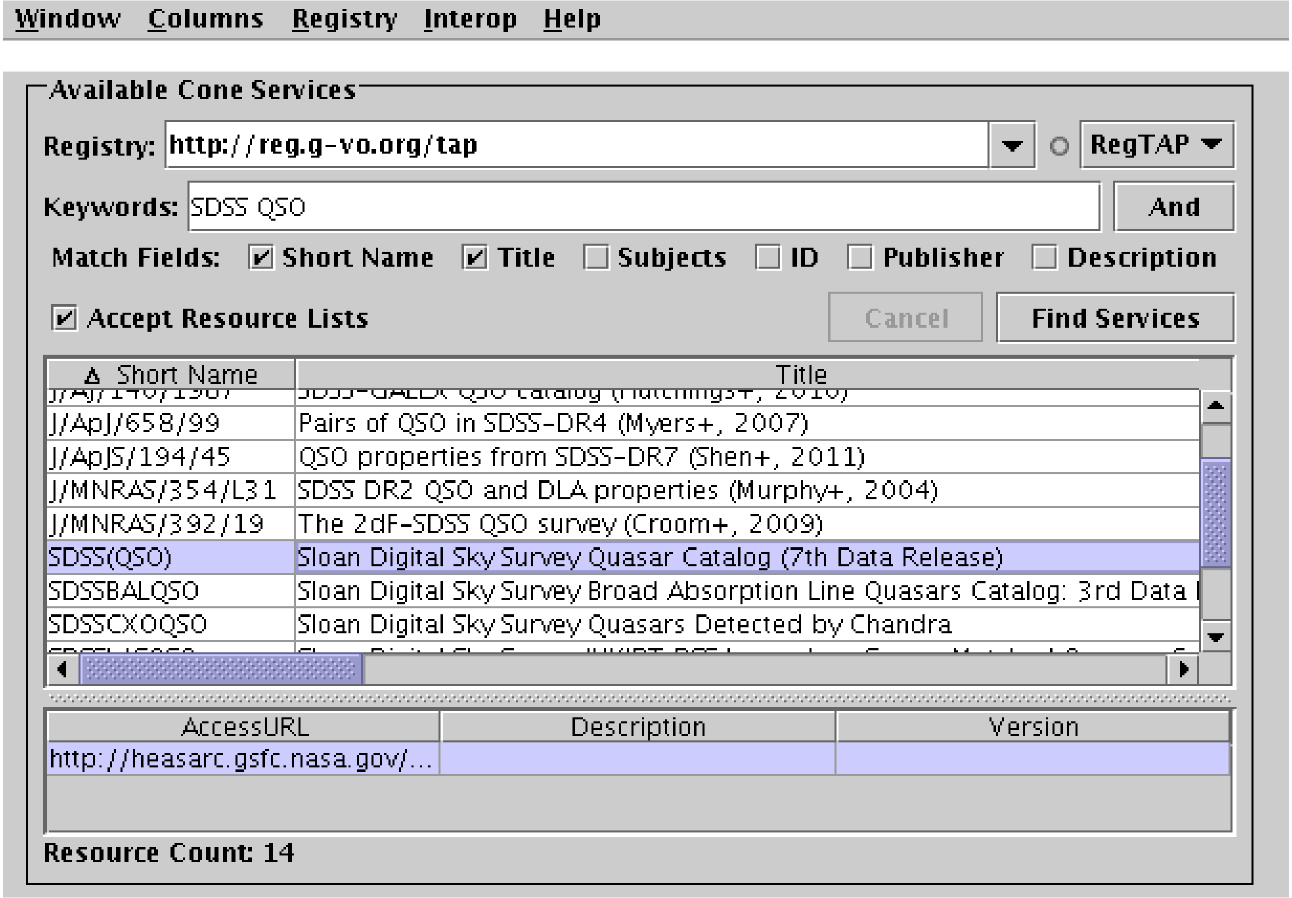
Several windows in TOPCAT allow access to standard Virtual Observatory (VO) services. These share the idea of querying the *Registry* for suitable data services, selecting one of the services thus returned, and then submitting one or more queries to that data service to generate a tabular result. These ideas are explained in more detail in Section 6, while the current section describes the windows. The next subsection describes the components from which these windows are put together. All the windows contain a registry query panel to select a suitable data access service. The positional query windows also contain either a single or a multiple search panel, and the TAP window contains its own custom panel, to define the actual query or queries to submit to the selected service. The windows themselves are described in the following subsections.

TOPCAT also provides access to two important services provided by the CDS (Centre de Données astronomiques de Strasbourg): the VizieR Catalogue Service Query and CDS X-Match Upload Window. These are not strictly speaking Virtual Observatory services, but since they provide access to the comprehensive VizieR archive of astronomical tables they can provide similar or superior functionality to their VO counterparts, so it's worth being aware of them.

A.9.1 Common Features

This section describes the graphical components which make up the VO data access windows.

A.9.1.1 Registry Query Panel



Registry: http://reg.g-vo.org/tap ○ RegTAP

Keywords: SDSS QSO

Match Fields: ☑ Short Name ☑ Title ☐ Subjects ☐ ID ☐ Publisher ☐ Description

☑ Accept Resource Lists

Short Name	Title
J/ApJ/658/99	Pairs of QSO in SDSS-DR4 (Myers+, 2007)
J/ApJS/194/45	QSO properties from SDSS-DR7 (Shen+, 2011)
J/MNRAS/354/L31	SDSS DR2 QSO and DLA properties (Murphy+, 2004)
J/MNRAS/392/19	The 2dF-SDSS QSO survey (Croom+, 2009)
SDSS(QSO)	Sloan Digital Sky Survey Quasar Catalog (7th Data Release)
SDSSBALQSO	Sloan Digital Sky Survey Broad Absorption Line Quasars Catalog: 3rd Data
SDSSCXOQSO	Sloan Digital Sky Survey Quasars Detected by Chandra

AccessURL	Description	Version
http://heasarc.gsfc.nasa.gov/...		

Resource Count: 14

Registry Query Panel

The Registry Query Panel appears in all VO query windows (though by default the TAP window uses an alternative registry search panel) and allows you to search for entries in the Registry representing data services of the type appropriate to the window. Note that if you know the service URL for the service that you want to use, you don't need to use this panel at all; you can simply fill in the URL in the lower part of the window.

The top part of this panel contains the fields you can fill in to query the registry for services matching your constraints. This consists of the following parts:

Registry

This determines the registry that will be queried for suitable data services. You can select from the available options, or enter a URL corresponding to some other registry service that you know about. The default is currently the GAVO registry, which at time of writing usually works pretty well, but other registries may have somewhat different data holdings. Some registries don't work very well, so you may have trouble if you use other ones.

To the right of this box is a selector that allows you to choose between two registry protocol options: **RegTAP** and **RI1.0**, corresponding to the IVOA Relational Registry and Registry Interface 1.0 standards respectively. At the time of writing (June 2014), RI1.0 is being phased out, and RegTAP is recommended. It is in any case considerably faster and more reliable than RI1.0. Unless you are a registry enthusiast, sticking to RegTAP is recommended.

The **Update Registry List** item in the **Registry** menu queries the currently selected registry for other registries, and adds more entries to the selector box as appropriate; the little round indicator between the URL and protocol selectors indicates progress of this query.

Keywords
Fill this in with keywords appropriate to the services you want to find. The format is space-separated words, and there is no wildcarding. So for instance if you want to find services relating to quasars in Sloan data you could enter the terms "SDSS QSO". These terms are matched against certain fields of each registry entry (as determined by the Match Fields checkboxes below) of each registry entry to see if it matches. Usually one or two short words is the appropriate level of detail. If no keywords are filled in, all the appropriate services are retrieved. Some experimentation may be required to specify a query with a reasonable number of results; if your query returns no results, try fewer or more general terms, and if it returns too many, try more or more specific ones.

And/Or button
This determines whether a registry entry is considered to match the chosen keywords if it matches all of them, or at least one of them. Clicking it toggles between "And" and "Or". The default is "And", meaning that all keywords must be matched (against any of the selected match fields) for a selection.

Match Fields
This line contains a number of checkboxes which allow you to define which fields in each registry record will be compared against the keywords you have entered when looking for matches. The contents of the **Short Name**, **Title**, **Subjects**, **ID** and **Publisher** fields are visible in the table of results. **Description** is a free-form, often long, description of the resource; it can be useful, but can often accidentally contain text which causes unwanted matches. By default the Short Name, Title, Subjects and ID fields are matched - click the checkboxes to change the selection

Find Services button
When the other fields have been filled in, hit the Find button and the registry will be searched to find data services that match the constraints you have given.

In some cases, when the window first appears, it will automatically query the registry for all known services of the appropriate type. This is currently the case for SIA and SSA services, since there is a manageable number of these. There are thousands of cone searches however, so for the cone search windows you need to fill in a query and submit it yourself. In either case, you can resubmit a query (hit the **Cancel** button first if necessary) to make another selection of registry entries.

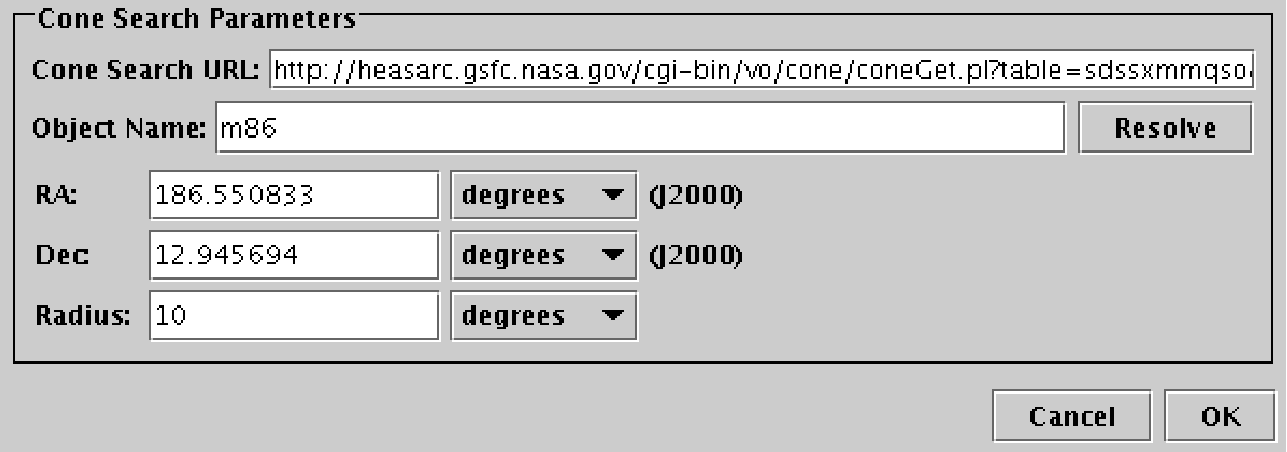
When the query completes, a table of matching entries is displayed below the Submit button. The number of resources found is displayed at the bottom, labelled **Resource Count**. The table contains one row for each matching entry, i.e. each service of the appropriate type that you can submit your data query to. The columns in the table give a short name, title, description and some other information about each service. You can sort the table rows according to the entries in different columns by clicking on the column header; click again on the same header to reverse the sort order. A little up/down arrow in a column header indicates that that column is currently determining the sort order. You can adjust the columns which are visible using the **Columns** menu, or drag the columns sideways to change their order in the usual way.

You should select the service that you want to query for data by clicking on it. When this is done, you will see one or more rows appear in a second table underneath the first one. In most cases, this contains a single row which will be automatically selected. However, some registry entries have multiple data services associated with a single record - in this case you should select the appropriate entry from the lower table as well.

Once you have selected a data service in this way, its service URL will be entered in the lower part of the window (not shown in the figure above) and you are ready to perform the actual single or multiple data query.

If a SAMP hub is running (see Section 9), it is possible to exchange lists of resources between this window and other applications. If another application sends a suitable message to TOPCAT, and the resources are of an appropriate type, and the window is visible at the time, the received list can be loaded and displayed here, replacing any which are currently displayed. You can control whether incoming lists are accepted and displayed in this way using the **Accept Resource Lists** toggle, either using the checkbox just above the resource display panel, or using the corresponding item in the **Interop** menu. If you wish to send your selection to one or more other applications which can make use of them, use the Broadcast Resource List ( ) or Send Resource List ( ) options in the **Interop** menu.

A.9.1.2 Single Positional Search Panel



Cone Search URL: http://heasarc.gsfc.nasa.gov/cgi-bin/vo/cone/coneGet.pl?table=sdssxmmqso

Object Name: m86

RA: 186.550833 degrees (J2000)

Dec 12.945694 degrees (J2000)

Radius: 10 degrees

Single Positional Search Panel

The Single Positional Search Panel appears in VO-based table load dialogues and is used to specify data queries on a single region of the sky. The purpose is to load a new table (containing entries representing catalogue entries, images or spectra). To use it you must fill in the **URL** field and the position definition, and then hit the **OK** button.

URL
This must contain the service URL for the data service that you are querying. Usually, this will be filled in by selecting one of the services obtained by the registry query. However, you can fill it in manually with the URL of a service you know about if you prefer. If you know what you're doing, it's also possible to doctor a service URL filled in by the registry selection, for instance by adding `&name=value` parameters to it.

Object Name
You can fill this in with the name of an object to be resolved by SIMBAD, and hit the **Resolve** button, which will fill in the coordinates in the RA and Dec fields below. If you enter the RA and Dec manually, you don't need to use this field.

RA and Dec
Fill in the central position in terms of J2000 Right Ascension and Declination that you are interested in here. You can either do it manually, or by using the **Object Name** field as above. Select the units as appropriate.

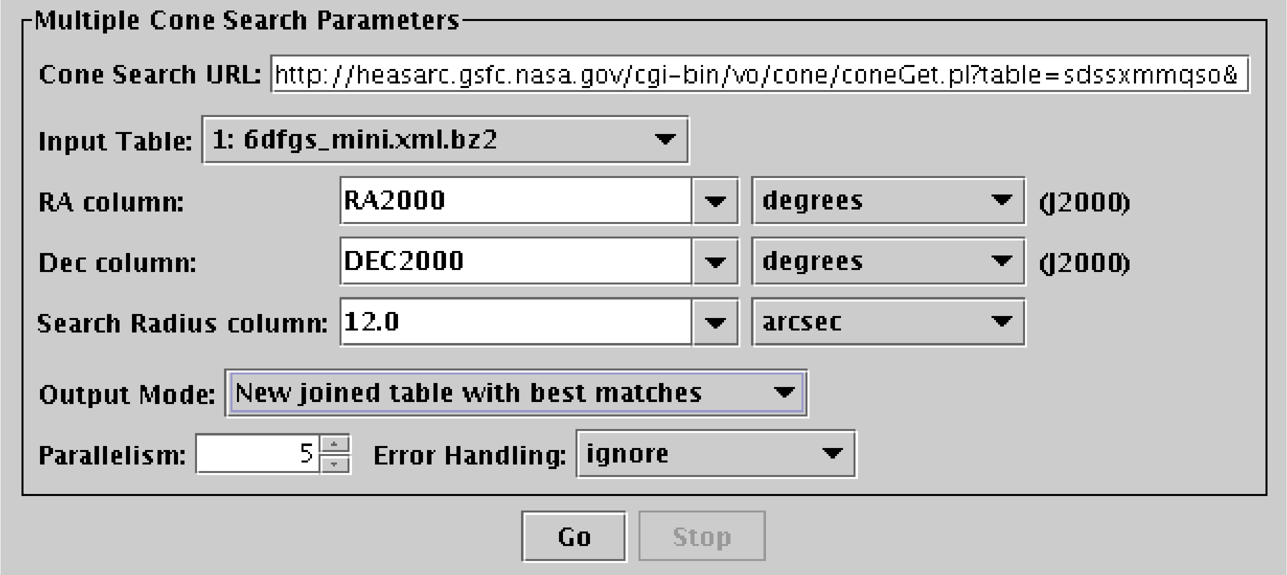
Radius
Indicates how large is the region you are interested in. This field has slightly different appearance and meaning for the different data service types; hover over it with the mouse to

see a tooltip with the details. For SIA and SSA, but not for Cone Search, it is permissible to leave it blank (though certain services don't seem to like that). Select the units as appropriate.

When the fields are filled in, hit the **OK** button and wait for the new table defined by your query parameters to load. If you get bored waiting (the service may be down, or just very slow, or you may have submitted a query with a very large return set), you can hit the **Cancel** button, and perhaps adjust the parameters and try again.

If a SAMP hub is running (see Section 9), then other running clients may be able to fill in the RA and Dec coordinates. For instance clicking on a sky position in an image viewing tool might fill in the sky position you clicked on. Usually this will happen automatically. This is often convenient, but may not be what you want. You can control whether incoming coordinate settings are accepted in this way using the **Accept Sky Positions** toggle below the Resolve button, or using the corresponding item in the **Interop** menu.

A.9.1.3 Multiple Positional Search Panel



Multiple Cone Search Parameters

Cone Search URL: http://heasarc.gsfc.nasa.gov/cgi-bin/vo/cone/coneGet.pl?table=sdssxmmqso&

Input Table: 1: 6dfgs_mini.xml.bz2

RA column: RA2000 degrees (J2000)

Dec column: DEC2000 degrees (J2000)

Search Radius column: 12.0 arcsec

Output Mode: New joined table with best matches

Parallelism: 5 Error Handling: ignore

Multiple Positional Search Panel

The Multiple Positional Search Panel appears in VO-based multiple query windows which specify a procedure in which one query is made for each row of an input table. Each of the input rows defines a (probably small) region on the sky. The purpose is to find data records (about catalogue entries, images or spectra) in a remote catalogue corresponding to each row of a local catalogue. As such, it is used to define a kind of join, of a local Apparent Table with a remote one, where the remote one is exposed via a VO data service.

To use it you must fill in the **URL** field to select the remote service, and fields defining how each row of the input table can be turned into a positional query. You also need to define what form the result will be returned in. These parts are described below.

The URL field specifies the data service which is to be queried:

URL
: This must contain the service URL for the data service that you are querying. Usually, this will be filled in by selecting one of the services obtained by the registry query. However, you can fill it in manually with the URL of a service you know about if you prefer. If you know what you're doing, it's also possible to doctor a service URL filled in by the registry selection, for instance by adding `&name=value` parameters to it.

The following fields define what queries are to be sent to the service:

Input Table
Select one of the tables loaded into TOPCAT whose row positions you want to search around.

RA and Dec columns
Fill these in with the columns in the input table which contain J2000 Right Ascension and Declination (you can perform coordinate conversions using the Sky Coordinates Window if required). Select the units as appropriate. If the input table has appropriate metadata, the correct values may be filled in by default, but it may occasionally guess wrong, so it's wise to check that they are filled in correctly.

Search Radius column
Fill this in with a value that represents the size of search that you want to make around each row's position. You can either use a constant value, or a column name, or any numeric algebraic expression based on the table (see Section 7). Select units as appropriate. This field has slightly different appearance and meaning for the different data service types; hover over it with the mouse to see a tooltip with the details. For SIA and SSA, but not for Cone Search, it is permissible to leave it blank (though certain services don't seem to like that). Select the units as appropriate.

The Output Mode selector indicates what will be done with the result.

Output Mode
When the queries have been performed, there are different ways that the results can be returned. Since the operation here is basically a crossmatch between a local table and a remote one, this effectively describes what sort of join is to be done. The options are as follows:

New joined table with best matches
A new table will be generated, containing one row for each row in the input table which returned at least one search result from the remote table. The best (closest) match will be included; any others will be discarded. The new table will have all the columns of the input table and all the columns returned by the data service.

New joined table with all matches
A new table will be generated, containing one row for each search result returned. Rows from the input table will be duplicated where more than one search result was returned corresponding to that row. The new table will have all the columns of the input table and all the columns returned by the data service.

New joined table, one row per input row
A new table will be generated containing the same number of rows as the input table. If any search results were returned for each row, the best (closest) one is included. Any others will be discarded, and cells will be empty for rows with no search results. The new table will have all the columns of the input table and all the columns returned by the data service.

Add subset for matched rows
No new table will be generated, but a new subset will be added to the table which includes only those rows which returned at least one search result.

The final controls adjust the details of how the individual queries are submitted to the remote service.

Parallelism
Controls how many cone search queries are running at once. By making several queries to the

service concurrently, the time it takes to fill in the whole table can be much quicker than making the query for the first row, waiting for the result, making the query for the second row, etc. The value here is approximately how many queries will be run at the same time. Increasing the value might make your multiple query run faster; or it might overload the server and make you unpopular with the service administrator, or result in your query taking longer or failing altogether, or both. The default value of 3 is probably reasonable, but experiment with adjusting it if you want.

Error Handling
Determines what happens if one of the queries fails with an error. The options are:

abort
terminate the operation; the whole multiple search fails

ignore
treat a failed query the same as one which returns zero rows

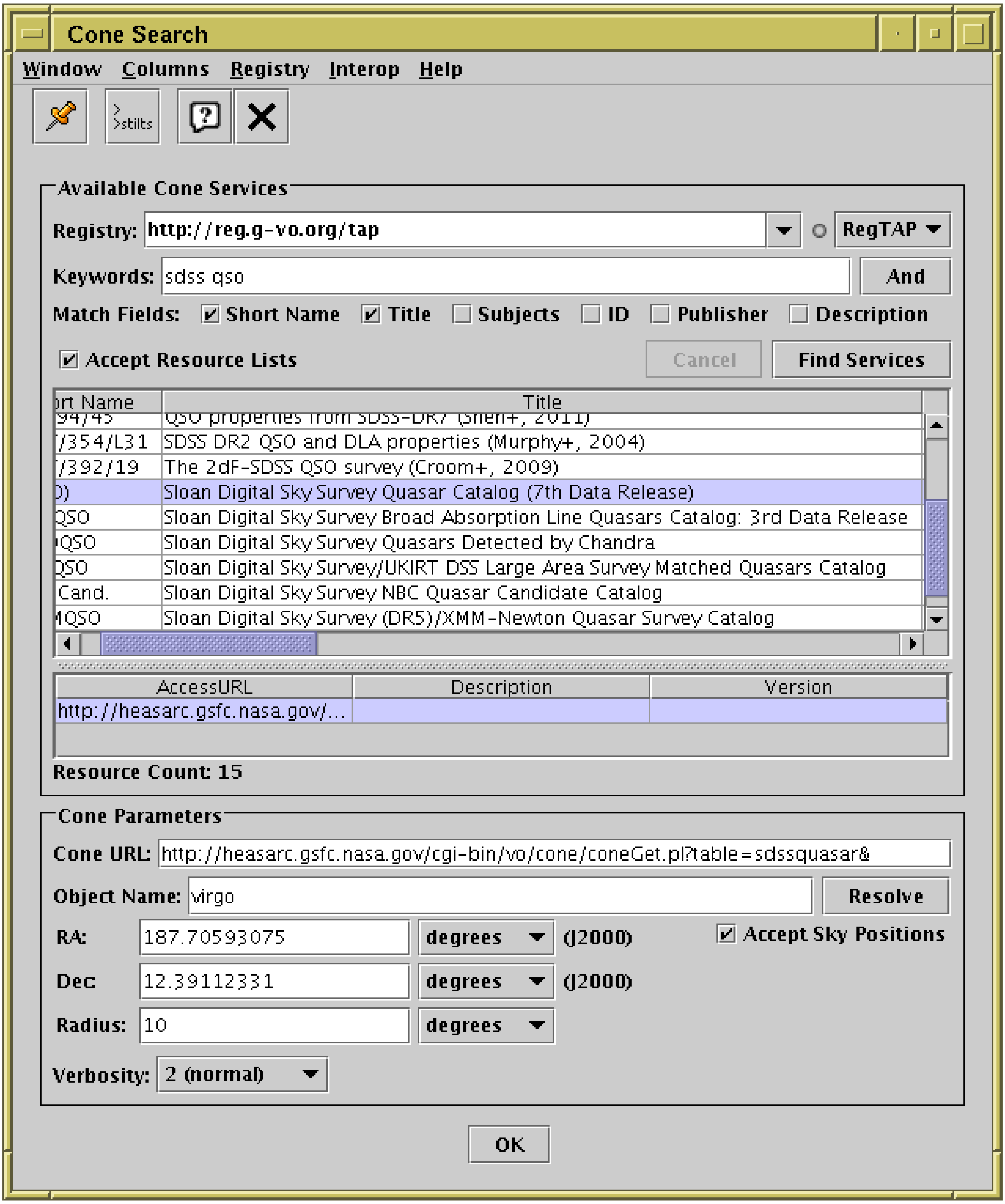
retry N times
try N times (with increasing delays) to get a non-fail result

retry indefinitely
keep trying (with increasing delays) for a non-fail result

The best setting for this depends on the way the service is set up; the default is **abort**, but for unreliable or poorly implemented services it may be better to continue operation in the face of a few failures.

When all of the fields are filled in (defaults can be used for many of them), hit the **Go** button, and the search will commence. It may be quite slow; a progress bar is shown at the bottom of the window. When it completes, a popup window summarising the result will be shown. If you get bored waiting, you can hit the **Cancel** button, and perhaps adjust the parameters and try again.

A.9.2 Cone Search



Available Cone Services

Registry: http://reg.g-vo.org/tap ○ RegTAP

Keywords: sdss qso

Match Fields: ☑ Short Name ☑ Title □ Subjects □ ID □ Publisher □ Description

☑ Accept Resource Lists

ort Name	Title
94/45	QSO properties from SDSS-DR7 (Shen+, 2011)
/354/L31	SDSS DR2 QSO and DLA properties (Murphy+, 2004)
/392/19	The 2dF-SDSS QSO survey (Croom+, 2009)
)	Sloan Digital Sky Survey Quasar Catalog (7th Data Release)
QSO	Sloan Digital Sky Survey Broad Absorption Line Quasars Catalog: 3rd Data Release
QSO	Sloan Digital Sky Survey Quasars Detected by Chandra
QSO	Sloan Digital Sky Survey/UKIRT DSS Large Area Survey Matched Quasars Catalog
Cand.	Sloan Digital Sky Survey NBC Quasar Candidate Catalog
1QSO	Sloan Digital Sky Survey (DR5)/XMM-Newton Quasar Survey Catalog

AccessURL	Description	Version
http://heasarc.gsfc.nasa.gov/...		

Resource Count: 15

Cone Parameters

Cone URL: http://heasarc.gsfc.nasa.gov/cgi-bin/vo/cone/coneGet.pl?table=sdssquasar&

Object Name: virgo

RA: 187.70593075 degrees (J2000) ☑ Accept Sky Positions

Dec: 12.39112331 degrees (J2000)

Radius: 10 degrees

Verbosity: 2 (normal)

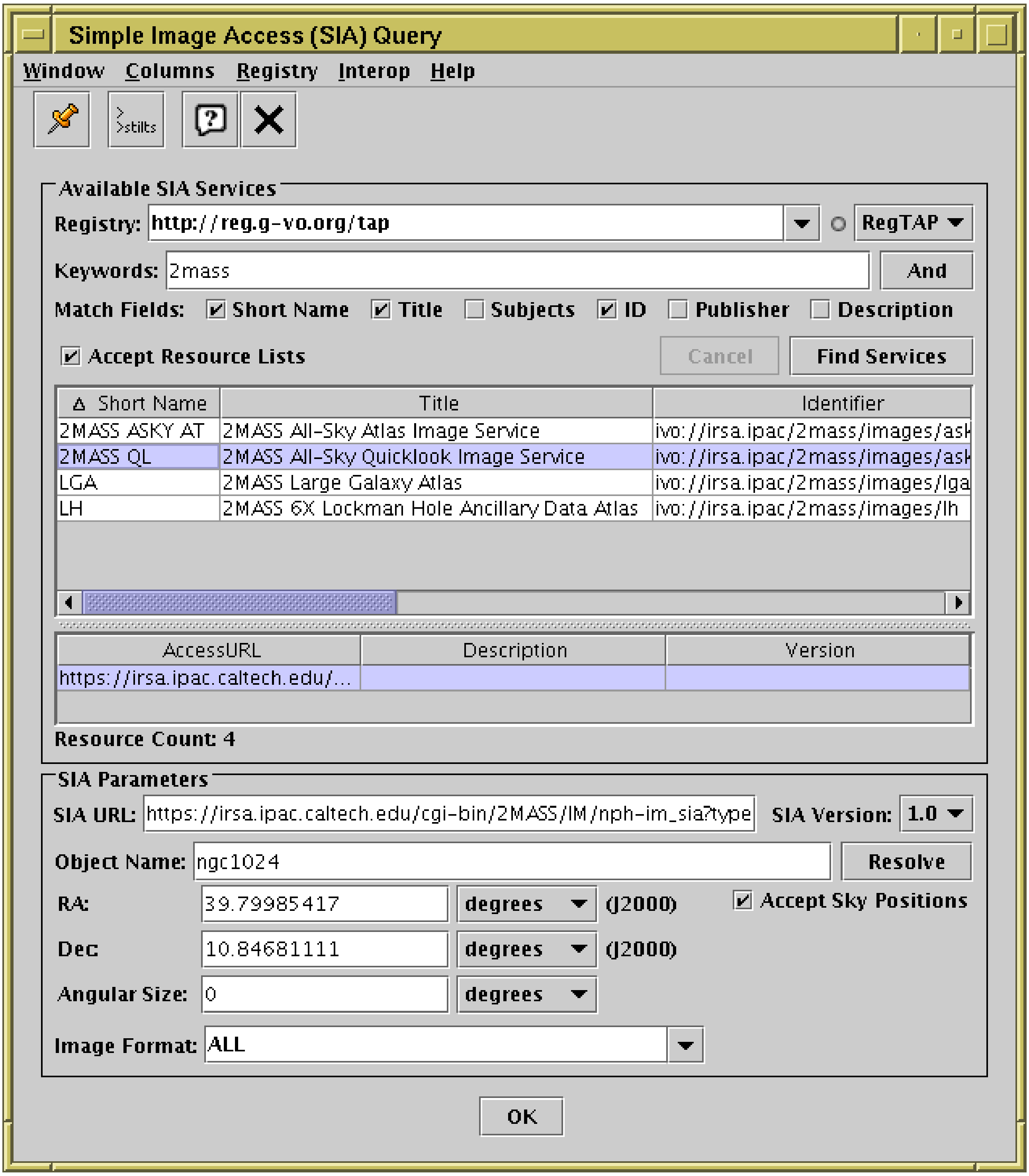
Cone table load dialogue

The cone search load dialogue can be opened using the **Cone Search** button (▼) from the Load Window's toolbar or the Control Window's **VO** menu. It allows you to query one of a number of external web services for objects from a given catalogue in a given region of the sky. Data held by cone search services are effectively source catalogues with sky position columns.

The window consists of a Registry Query Panel at the top, and a Single Positional Search Panel below.

The search panel includes a **Verbosity** selector as well as the normal features: this allows you to choose options for how many columns will appear in the output table, from 1 (minimum) to 3 (maximum). Some services will take notice of this parameter and return narrower or wider tables respectively, and others will ignore it.

A.9.3 Simple Image Access (SIA) Query



Registry: http://reg.g-vo.org/tap

Keywords: 2mass

Match Fields: ☑ Short Name ☑ Title □ Subjects ☑ ID □ Publisher □ Description

☑ Accept Resource Lists

Short Name	Title	Identifier
2MASS ASKY AT	2MASS All-Sky Atlas Image Service	ivo://irsa.ipac/2mass/images/ask
2MASS QL	2MASS All-Sky Quicklook Image Service	ivo://irsa.ipac/2mass/images/ask
LGA	2MASS Large Galaxy Atlas	ivo://irsa.ipac/2mass/images/lga
LH	2MASS 6X Lockman Hole Ancillary Data Atlas	ivo://irsa.ipac/2mass/images/lh

AccessURL	Description	Version
https://irsa.ipac.caltech.edu/...		

Resource Count: 4

SIA URL: https://irsa.ipac.caltech.edu/cgi-bin/2MASS/IM/nph-im_sia?type

SIA Version: 1.0

Object Name: ngc1024

RA: 39.79985417 degrees (J2000) ☑ Accept Sky Positions

Dec: 10.84681111 degrees (J2000)

Angular Size: 0 degrees

Image Format: ALL

SIA query load dialog

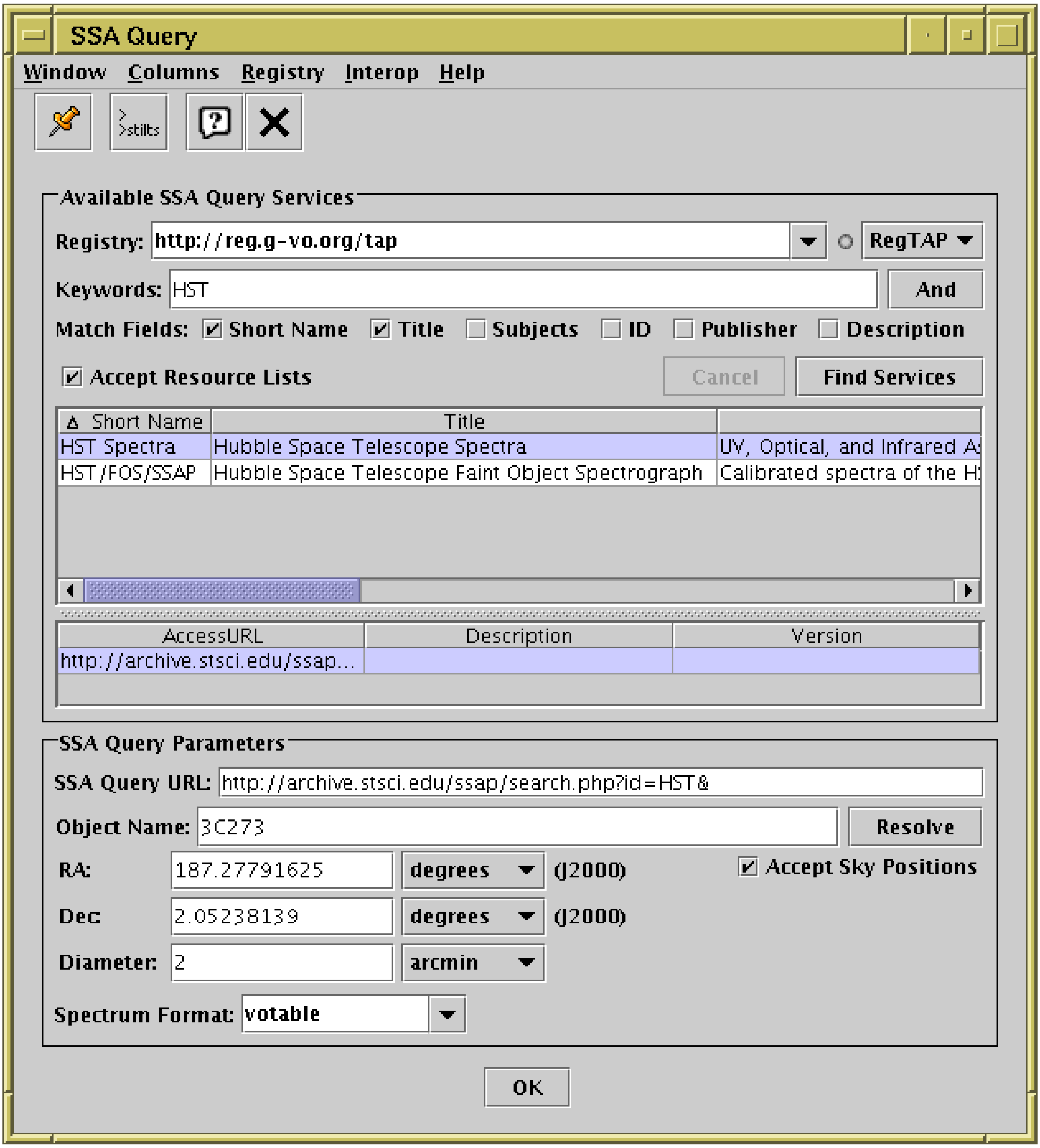
The SIA query load dialogue can be opened using the **SIA Query** button ( ) from the Load Window's toolbar or the Control Window's **VO** menu. It allows you to query one of a number of external web services for images covering a given region of the sky.

The window consists of a Registry Query Panel at the top, and a Single Positional Search Panel below. The **Angular Size** may be set to zero; this means that any image covering the chosen position will be selected. There is additionally an **Image Format** selector which allows you to restrict the result to contain only images in certain formats (the default is set to only FITS).

The result of a successful query is a table in which each row represents an image that can be downloaded from a remote site; one of the columns contains the image URL, and the other columns contain associated metadata such as image format, size, shape and so on. Different services provide different kinds of answer to these queries; some may have a static list of observed or reduced FITS files, and others may generate images on the fly. See the SIA standard, and individual registry records, for details. A useful thing to do with the resulting table might be to configure an activation action of Display Image or Send FITS Image.

TOPCAT supports two different (and incompatible) major versions of the SIA protocol, SIA version 1 and version 2. When you have selected a service to use, its protocol version number is displayed in the **SIA Version** indicator next to the **SIA URL** display. If you have entered the URL manually, you will need to make sure this is set to the appropriate value. Note that, although SIAv2 allows a number of query possibilities not offered by SIAv1, at present TOPCAT does not support this additional functionality for SIAv2 services.

A.9.4 Simple Spectral Access (SSA) Query



Available SSA Query Services

Registry: http://reg.g-vo.org/tap ○ RegTAP

Keywords: HST And

Match Fields: ☑ Short Name ☑ Title □ Subjects □ ID □ Publisher □ Description

☑ Accept Resource Lists

Short Name	Title	
HST Spectra	Hubble Space Telescope Spectra	UV, Optical, and Infrared A
HST/FOS/SSAP	Hubble Space Telescope Faint Object Spectrograph	Calibrated spectra of the H

AccessURL	Description	Version
http://archive.stsci.edu/ssap...		

SSA Query Parameters

SSA Query URL: http://archive.stsci.edu/ssap/search.php?id=HST&

Object Name: 3C273

RA: 187.27791625 degrees (J2000) ☑ Accept Sky Positions

Dec: 2.05238139 degrees (J2000)

Diameter: 2 arcmin

Spectrum Format: votable

SSA query load dialog

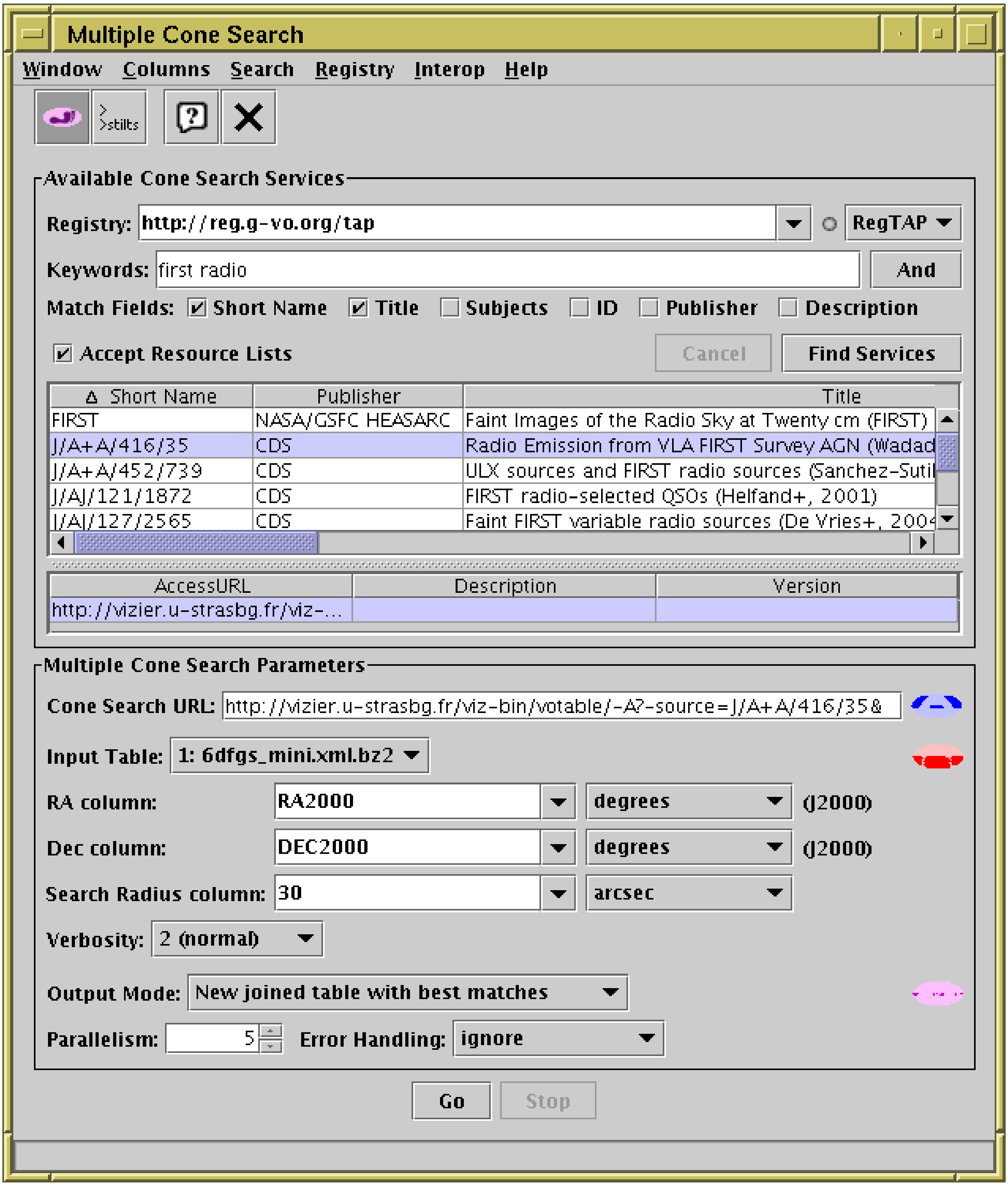
The SSA query load dialogue can be opened using the **SSA Query** button ( ) from the Load Window's toolbar or the Control Window's **VO** menu. It allows you to query one of a number of external web services for spectra in a given region of the sky.

The window consists of a Registry Query Panel at the top, and a Single Positional Search Panel below. The SSA specification says that the **Diameter** field may be left blank, in which case the service "should supply a default value intended to find nearby objects". However, experience shows that this doesn't always work, so it may be best to fill this in. It is also permitted to leave the **RA** and **Dec** fields blank, if you don't want to restrict the search by sky position. That may be a reasonable thing to do, for example, for services providing theoretical rather than observed spectra, or if you want all the data (subject to possible service restrictions). There is additionally a

Spectrum Format selector which allows you to restrict the result to contain only spectra in certain formats. By default, the service chooses which formats to return.

The result of a successful query is a table in which each row represents a spectrum that can be downloaded from a remote site; one of the columns contains the spectrum download URL, and the other columns contain associated metadata such as spectrum format, WCS, provenance information and so on. See the SSA standard for details. A useful thing to do with the resulting table might be to configure an activation action of Send Spectrum.

A.9.5 Multiple Cone Search



Available Cone Search Services

Registry: http://reg.g-vo.org/tap ○ RegTAP

Keywords: first radio And

Match Fields: ☑ Short Name ☑ Title □ Subjects □ ID □ Publisher □ Description

☑ Accept Resource Lists Cancel Find Services

Short Name	Publisher	Title
FIRST	NASA/GSFC HEASARC	Faint Images of the Radio Sky at Twenty cm (FIRST)
J/A+A/416/35	CDS	Radio Emission from VLA FIRST Survey AGN (Wadad
J/A+A/452/739	CDS	ULX sources and FIRST radio sources (Sanchez-Sutil
J/AJ/121/1872	CDS	FIRST radio-selected QSOs (Helfand+, 2001)
J/AJ/127/2565	CDS	Faint FIRST variable radio sources (De Vries+, 2004

AccessURL	Description	Version
http://vizier.u-strasbg.fr/viz-...		

Multiple Cone Search Parameters

Cone Search URL: http://vizier.u-strasbg.fr/viz-bin/votable/-A?-source=J/A+A/416/35&

Input Table: 1: 6dfgs_mini.xml.bz2

RA column: RA2000 degrees (J2000)

Dec column: DEC2000 degrees (J2000)

Search Radius column: 30 arcsec

Verbosity: 2 (normal)

Output Mode: New joined table with best matches

Parallelism: 5 Error Handling: ignore

Go Stop

Multiple cone search window

The multiple cone search window can be opened using the **Multicone** button () in the Control Window's toolbar, or its **VO** or **Joins** menus. It allows you to select one of a number of external web services which provide remote catalogue queries, and execute a query for each row of an input table. This effectively allows you to perform a positional crossmatch between a local table and a remote one.

The window consists of a Registry Query Panel at the top, and a Multiple Positional Search Panel below.

The search panel includes a **Verbosity** selector as well as the normal features: this allows you to choose options for how many columns will appear in the output table, from 1 (minimum) to 3 (maximum). Some services will take notice of this parameter and return narrower or wider tables respectively, and others will ignore it.

The following item is available in the toolbar and **Search** menu:

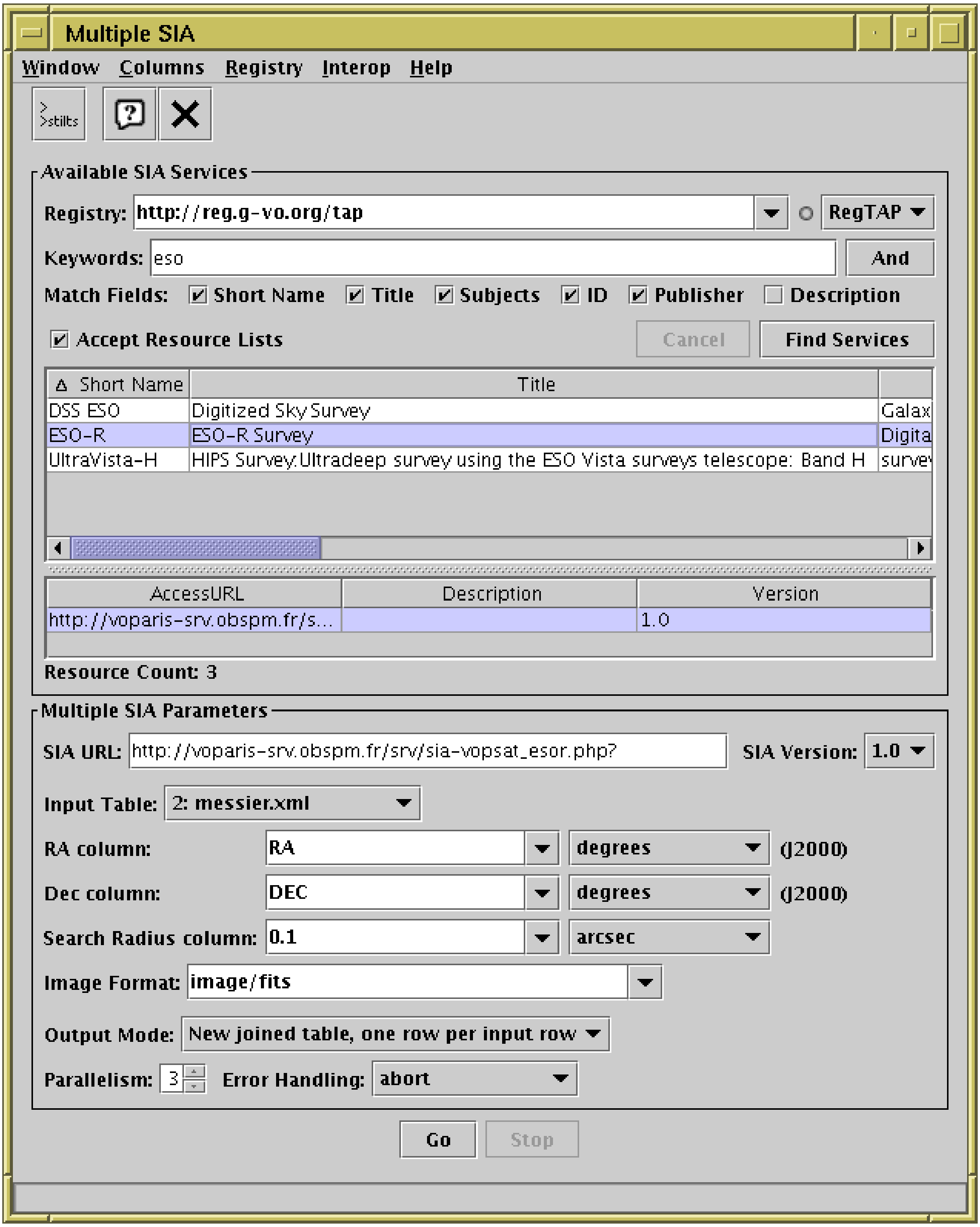
Use Service Coverage

If this option is selected, then the search attempts to query the service for the sky coverage it represents, if that information is available. Then it only dispatches queries for positions in the input table which fall within that coverage region, since any queries outside that region are bound to return an empty result. Depending on the coverage, this can substantially reduce the number of queries made, and hence the time taken. If there is no overlap between the input table and service coverage, the query is bound to return no results at all, and the time will be reduced effectively to zero.

When in operation (i.e. when this option is selected and the service provides coverage information) the coverage of the service, input table and their overlap is summarised with small (Mollweide equatorial) all-sky icons in blue, red and magenta respectively, as in the screenshot above. This makes it easy to see the sky regions in which results may be obtained, and also the amount by which this option reduces useless queries.

Currently, this option effectively works only for cone searches provided by CDS's VizieR service (publisher "CDS") which provide coverage information using Multi-Order Coverage maps (MOCs). For non-VizieR services this option has no effect.

A.9.6 Multiple SIA Query



Available SIA Services

Registry: http://reg.g-vo.org/tap ○ RegTAP

Keywords: eso And

Match Fields: ☑ Short Name ☑ Title ☑ Subjects ☑ ID ☑ Publisher □ Description

☑ Accept Resource Lists

Short Name	Title	
DSS ESO	Digitized Sky Survey	Galax
ESO-R	ESO-R Survey	Digita
UltraVista-H	HIPS Survey.Ultradeep survey using the ESO Vista surveys telescope: Band H	survey

AccessURL	Description	Version
http://voparis-srv.obspm.fr/s...		1.0

Resource Count: 3

Multiple SIA Parameters

SIA URL: http://voparis-srv.obspm.fr/srv/sia-vopsat_esor.php? SIA Version: 1.0

Input Table: 2: messier.xml

RA column: RA degrees (J2000)

Dec column: DEC degrees (J2000)

Search Radius column: 0.1 arcsec

Image Format: image/fits

Output Mode: New joined table, one row per input row

Parallelism: 3 Error Handling: abort

Multiple SIA window

The multiple Simple Image Access window can be opened using the **Multiple SIA** button () from the Control Window's **VO** or **Joins** menus. It allows you to select one of a number of external web services which provide queries on remotely accessible image holdings, and execute a query for each row of an input table.

The window consists of a Registry Query Panel at the top, and a Multiple Positional Search Panel below. The **Search Radius column** may be set to zero; this means that any image covering the chosen position will be selected. There is additionally an **Image Format** selector which allows you to restrict the result to contain only images in certain formats (the default is set to only FITS).

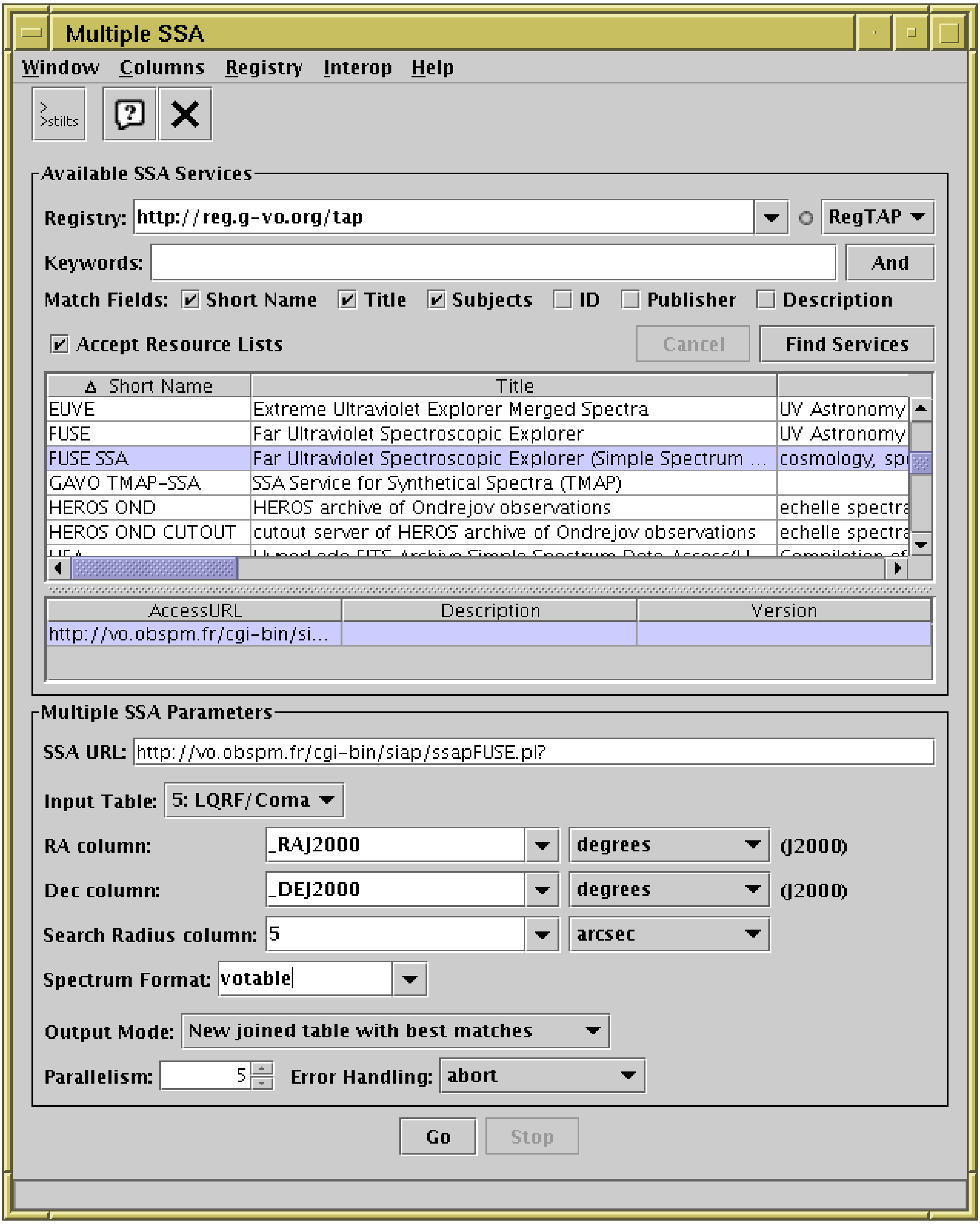
The result of each single successful query is a table in which each row represents an image that can be downloaded from a remote site; one of the columns contains the image URL, and the other columns contain associated metadata such as image format, size, shape and so on. Different services provide different kinds of answer to these queries; some may have a static list of observed or reduced FITS files, and others may generate images on the fly. See the SIA standard, and individual registry records, for details.

Note that some services return multiple images at the same positions but at different wavelengths. In this case TOPCAT's criterion for the "best" match (the one closest to the central query position) may not make much sense. For this reason, take care if using the "New joined table with best matches" or "New joined table, one row per input row" output modes.

One use of this function is to add some columns to an existing table which contain URLs of images from a given server corresponding to the sky positions of those rows. Having done this you might want to configure an activation action for the resulting table of Display Image or Send FITS Image.

TOPCAT supports two different (and incompatible) major versions of the SIA protocol, SIA version 1 and version 2. When you have selected a service to use, its protocol version number is displayed in the **SIA Version** indicator next to the **SIA URL** display. If you have entered the URL manually, you will need to make sure this is set to the appropriate value. Note that, although SIAv2 allows a number of query possibilities not offered by SIAv1, at present TOPCAT does not support this additional functionality for SIAv2 services.

A.9.7 Multiple SSA Query



Registry: http://reg.g-vo.org/tap ○ RegTAP

Keywords: And

Match Fields: ☑ Short Name ☑ Title ☑ Subjects ☐ ID ☐ Publisher ☐ Description

☑ Accept Resource Lists

Short Name	Title	
EUVE	Extreme Ultraviolet Explorer Merged Spectra	UV Astronomy
FUSE	Far Ultraviolet Spectroscopic Explorer	UV Astronomy
FUSE SSA	Far Ultraviolet Spectroscopic Explorer (Simple Spectrum ...	cosmology, spe
GAVO TMAP-SSA	SSA Service for Synthetical Spectra (TMAP)	
HEROS OND	HEROS archive of Ondrejov observations	echelle spectra
HEROS OND CUTOUT	cutout server of HEROS archive of Ondrejov observations	echelle spectra

AccessURL	Description	Version
http://vo.obspm.fr/cgi-bin/si...		

SSA URL: http://vo.obspm.fr/cgi-bin/siap/ssapFUSE.pl?

Input Table: 5: LQRF/Coma

RA column: _RAJ2000 degrees (J2000)

Dec column: _DEJ2000 degrees (J2000)

Search Radius column: 5 arcsec

Spectrum Format: votable

Output Mode: New joined table with best matches

Parallelism: 5 Error Handling: abort

Multiple SSA window

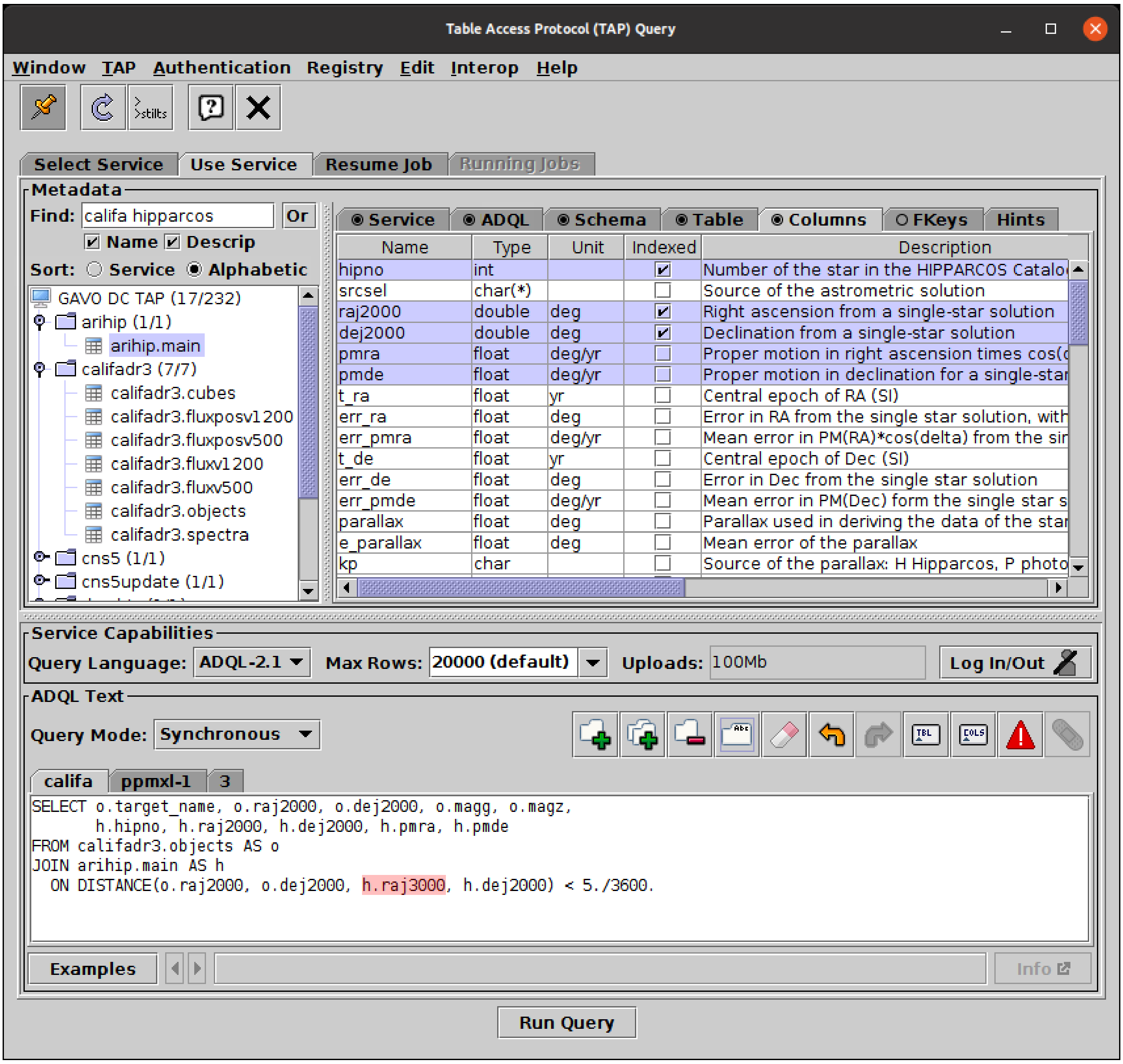
The multiple Simple Spectral Access window can be opened using the **Multiple SSA** button () from the Control Window's **VO** or **Joins** menus. It allows you to select one of a number of external web services which provide queries on remotely accessible spectral data holdings, and execute a query for each row of an input table.

The window consists of a Registry Query Panel at the top, and a Multiple Positional Search Panel below. The SSA specification says that the **Search Radius** column may be left blank, in which case the service "should supply a default value intended to find nearby objects". However, experience shows that this doesn't always work so it may be best to fill this in. There is additionally a **Spectrum Format** selector which allows you to restrict the result to contain only spectra in certain formats. By default, the service chooses which formats to return.

The result of each single successful query is a table in which each row represents a spectrum that can be downloaded from a remote site; one of the columns contains the specturm URL, and the other columns contain associated metadata such as spectrum format, WCS, provenance information and so on. See the SSA standard for details.

One use of this function is to add some columns to an existing table which contain URLs of spectra from a given server corresponding to the sky positions of those rows. Having done this you might want to configure an activation action for the resulting table of Send Spectrum.

A.9.8 Table Access Protocol (TAP) Query



Find: califa hipparcos

☑ Name ☑ Descrip

Sort: ○ Service ● Alphabetic

GAVO DC TAP (17/232)
arihip (1/1)
arihip.main
califadr3 (7/7)
califadr3.cubes
califadr3.fluxposv1200
califadr3.fluxposv500
califadr3.fluxv1200
califadr3.fluxv500
califadr3.objects
califadr3.spectra
cns5 (1/1)
cns5update (1/1)

Name	Type	Unit	Indexed	Description
hipno	int		☑	Number of the star in the HIPPARCOS Catalo
srcsel	char(*)		☐	Source of the astrometric solution
raj2000	double	deg	☑	Right ascension from a single-star solution
dej2000	double	deg	☑	Declination from a single-star solution
pmra	float	deg/yr	☐	Proper motion in right ascension times cos(
pmde	float	deg/yr	☐	Proper motion in declination for a single-star
t_ra	float	yr	☐	Central epoch of RA (SI)
err_ra	float	deg	☐	Error in RA from the single star solution, with
err_pmra	float	deg/yr	☐	Mean error in PM(RA)*cos(delta) from the sin
t_de	float	yr	☐	Central epoch of Dec (SI)
err_de	float	deg	☐	Error in Dec from the single star solution
err_pmde	float	deg/yr	☐	Mean error in PM(Dec) form the single star s
parallax	float	deg	☐	Parallax used in deriving the data of the star
e_parallax	float	deg	☐	Mean error of the parallax
kp	char		☐	Source of the parallax: H Hipparcos, P photo

Query Language: ADQL-2.1 Max Rows: 20000 (default) Uploads: 100Mb

Query Mode: Synchronous

califa | ppmxl-1 | 3

```
SELECT o.target_name, o.raj2000, o.dej2000, o.magg, o.magz,
       h.hipno, h.raj2000, h.dej2000, h.pmra, h.pmde
FROM califadr3.objects AS o
JOIN arihip.main AS h
  ON DISTANCE(o.raj2000, o.dej2000, h.raj3000, h.dej2000) < 5./3600.
```

Table Access Protocol Window

SELECT FROM) in the Control Window's toolbar or **VO** menu, or the Load Window's toolbar. It allows you to use the TAP protocol to make freeform queries of remote database services using an SQL-like language. This is a powerful capability, giving you full access to remote data holdings, and use of the IVOA-endorsed TAP standard means that you can use the same interface for many different remote datasets.

Using TAP is one of the more complicated things you can do in TOPCAT, but it's well worth getting to grips with because of the capabilities it gives you for customised access to all kinds of remote data.

In order to interrogate the remote database you will have to write queries in Astronomical Data Query Language (ADQL). ADQL is essentially a particular dialect of SQL, so if you have used SQL before, you should be able to write ADQL queries. A tutorial on writing ADQL/SQL is beyond the scope of this manual, but the Use Service tab provides a quick cheat sheet in the **Hints** tab of the metadata panel, and an **Examples** menu which will give you a good start in using it. There are introductions and tutorials elsewhere, for instance at http://docs.g-vo.org/adql (Markus Demleitner) and in the documentation of some TAP service web pages.

Once your query is written, you can submit it in either *synchronous* or *asynchronous* mode. Synchronous queries begin to execute right away and should have a relatively short runtime; asynchronous queries are submitted to a server, run for a potentially long time, and retrieve the result later.

The window is composed of four tabs, of which you can see one at a time (some of these tabs themselves contain tabbed parts). You will use two or three of them when submitting a TAP query. Each tab is described in more detail in the following subsections, but an overview of their operation is like this:

Select Service tab
Chooses the TAP service you wish to query, either using a registry search or by entering the service URL directly. Once you have chosen a service you can move to the **Use Service** tab.

Use Service tab
Displays information about the service and the tables it holds, and allows you to enter the text of the query and submit the job. If the job is submitted synchronously, loading will begin directly, but in the asynchronous case, you will move to the **Running Jobs** tab.

Running Jobs tab
Shows a list of the asynchronous query jobs you have submitted, with details on their progress.

Resume Job tab
Optionally allows you to resume an asynchronous query started in some earlier session.

When you first visit this window, the **Select Service** tab will be visible, and when an asynchronous query has been submitted the **Running Jobs** tab will be visible. If you want to submit another query to the same service, or use a different service, just select **Use Service** or **Select Service** respectively, by clicking on the appropriate tab at the top.

This window offers some menus additional to the standard VO window menus:

TAP menu
This contains some configuration options specific to Table Access Protocol service interaction. It has the following sub-items:

Reload (↻)
Reloads information displayed in the window from the server. This may be updating job

status information or reloading table or service metadata; the exact behaviour depends on which tab is currently visible.

Job Deletion submenu

Allows you to choose when asynchronously submitted jobs will by default be deleted from the server. Once they are deleted they will no longer be visible in the **Running Jobs** tab. If they are not deleted in this way, they will stay on the server until they are deleted according to the server's deletion policy. Choose one of the available options:

- **On Completion:** deleted when it has completed with either success or error status
- **On Exit:** deleted when TOPCAT shuts down
- **Never:** not deleted (except according to server policy)

You can override these settings once a job has started by using the controls at the bottom of the Running Jobs tab. The settings in this menu affect the default value of the **Delete On Exit** checkbox. If you want to resume the job in a later session, you'll have to make sure that the job is not deleted on completion or exit.

Metadata Acquisition submenu

Controls how the application loads service metadata - information about the tables and columns provided by the service. The following options are available:

- **Auto:** make a sensible decision; this generally means **TAP_SCHEMA-C** for services with very many tables, and **TableSet-VOSI1.1** otherwise.
- **TAP_SCHEMA-C:** reads metadata from TAP_SCHEMA tables; all schemas, tables and foreign keys are read at once, columns are read only when required.
- **TAP_SCHEMA-CF:** reads metadata from TAP_SCHEMA tables; all schemas and tables are read at once, columns and foreign keys are read only when required.
- **TAP_SCHEMA:** reads all metadata at once from the TAP_SCHEMA tables.
- **TableSet-VOSI1.1** Uses VOSI 1.1-style `detail`-sensitive requests to the `/tables` endpoint. No `detail` specification is made initially, so the service decides whether up-front loading is done. Still compatible with VOSI 1.0 services.
- **TableSet-VOSI1.1-1step** Uses VOSI 1.1-style `detail`-sensitive requests to the `/tables` endpoint. `detail=max` is specified initially to indicate that full metadata up-front metadata load is preferred. Still compatible with VOSI 1.0 services.
- **TableSet-VOSI1.1-2step** Uses VOSI 1.1-style `detail`-sensitive requests to the `/tables` endpoint. `detail=min` is specified initially to indicate that on-demand column load is preferred. Still compatible with VOSI 1.0 services.
- **TableSet-VOSI1.0:** reads the document at the `/tables` endpoint of the TAP service as defined by VOSI 1.0; all table metadata is loaded up front.
- **VizieR:** uses the 2-stage TableSet from `/tables` as implemented (at time of writing) by the TAPVizieR service; columns are read only when required (experimental, may be withdrawn).

You will usually want to stick with the default value of **Auto**. The main purpose of this menu is for TAP service implementers or testers who want to experiment with different options. However, if you're on a slow link, and it's taking a long time to display any metadata for medium-sized services, you might want to switch to TAP_SCHEMA-C/CF, or VizieR for talking to TAPVizier.

Response Format submenu

Configures how the application requests that VOTable results are to be returned. The options (**TABLEDATA**, **BINARY**, **BINARY2**) correspond to the different defined VOTable serialization formats. The selected format is only actually requested if the service has indicated support for it, otherwise the service default is used.

Upload Format submenu

Configures how tables are transmitted to the service for upload if an upload query (one involving a `TAP_UPLOAD.*` table) is performed. The options (**TABLEDATA**, **BINARY**,

BINARY2) correspond to the different defined VOTable serialization formats. The choice shouldn't affect any results, but it may affect required bandwidth and some services may (though shouldn't) have non-standard requirements for serialization format.

Service Discovery submenu

Configures how TOPCAT searches for available TAP services in the **By Table Properties** sub-tab of the Select Service tab. There are currently three options, **GloTS**, which uses the Global TAP Schema service list maintained by GAVO, and two variants which use the IVOA Registry: **RegTAP 1.2** which uses some features of the RegTAP v1.2 standard, and **Reg Prototype** which uses a proposed convention related to auxiliary resources. Unless you are a registry nerd you will probably want to stick with the default option (currently GloTS).

HTTP gzip

If checked, this option sends `Accept-Encoding: gzip` headers with most of the HTTP requests when communicating with the TAP service. That ought to reduce required bandwidth considerably, but only if the service chooses to honour this request. The setting should have no effect on the actual results, but may affect performance.

Authentication menu

Provides options concerned with logging in and out of TAP services.

Log In/Out (👤)

Logs you in to the currently selected TAP service, or logs you out if you are already logged in. Only applies to services that have optional or mandatory authentication. This action is the same as the corresponding button in the Use Service tab, and it is documented in more detail there.

Reset Authentication (🔒)

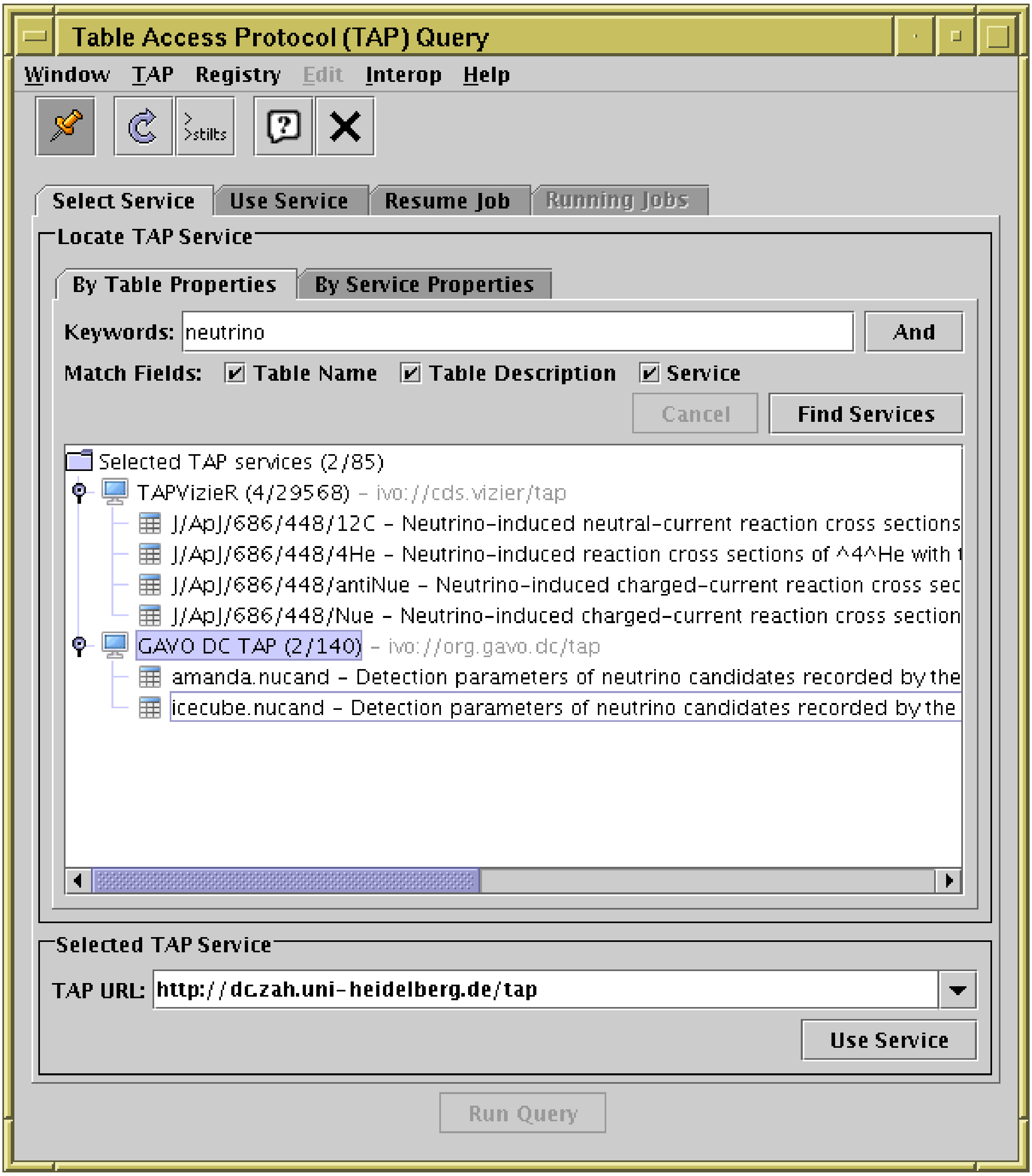
Logs out of any external services that you have logged in to. If you try to access these services again, you will either access them anonymously or (in cases where authentication is mandatory) be prompted to log in again.

Edit menu

Provides options concerned with editing ADQL text. This is only active when the **Use Service** tab is visible, and it is described in that section.

Note: TAP is a complex protocol, and at time of writing, many TAP services still do not provide full and error-free implementations. Unexpected behaviour may be a result of service implementation deficiencies. Hopefully that will improve over time.

A.9.8.1 Select Service tab



Keywords: neutrino

Match Fields: ☑ Table Name ☑ Table Description ☑ Service

Selected TAP services (2/85)
- TAPVizieR (4/29568) – ivo://cds.vizier/tap
  - J/ApJ/686/448/12C – Neutrino-induced neutral-current reaction cross sections
  - J/ApJ/686/448/4He – Neutrino-induced reaction cross sections of ^4^He with t
  - J/ApJ/686/448/antiNue – Neutrino-induced charged-current reaction cross sec
  - J/ApJ/686/448/Nue – Neutrino-induced charged-current reaction cross section
- GAVO DC TAP (2/140) – ivo://org.gavo.dc/tap
  - amanda.nucand – Detection parameters of neutrino candidates recorded by the
  - icecube.nucand – Detection parameters of neutrino candidates recorded by the

TAP URL: http://dc.zah.uni-heidelberg.de/tap

TAP window with Select Service tab visible

The **Select Service** tab of the TAP load dialogue allows you to choose which TAP service you wish to query. You need to do this before submitting a new TAP query.

There are two separate tabs within the **Locate TAP Service** sub-panel, corresponding to two different ways to select the service you want to use.

By Table Properties
Use this tab to find TAP services that contain particular tables, or tables relating to particular topics. When first opened (or if no search terms are entered), it shows a list by name of all the known TAP services, ordered by the number of tables they contain. If you enter one or more search terms in the **Keywords** field and hit **Find Services** (or return), then the list will be redisplayed as a tree in which the children of each service are the tables whose name and/or

description matches the search terms you entered (click on the node handles to display/hide the children). You can choose whether to match the **Table Name** or **Table Description** using the **Match Fields** checkboxes; basic **Service** metadata can also be matched (just name, title, ID; for more complex searches by service metadata use the By Service Properties tab, described below). Clicking the **And**/**Or** button to toggle its value determines whether the text from the displayed tables/service text are required to match all of your search terms, or just one of them. When you've decided which service you want to use, click it, and its access URL will be filled in the **TAP URL** selector at the bottom. If you double-click it, it will take you directly to the Use Service tab. If you select or double-click on a table from the tree, then when you look at the service, that table's metadata will be displayed first.

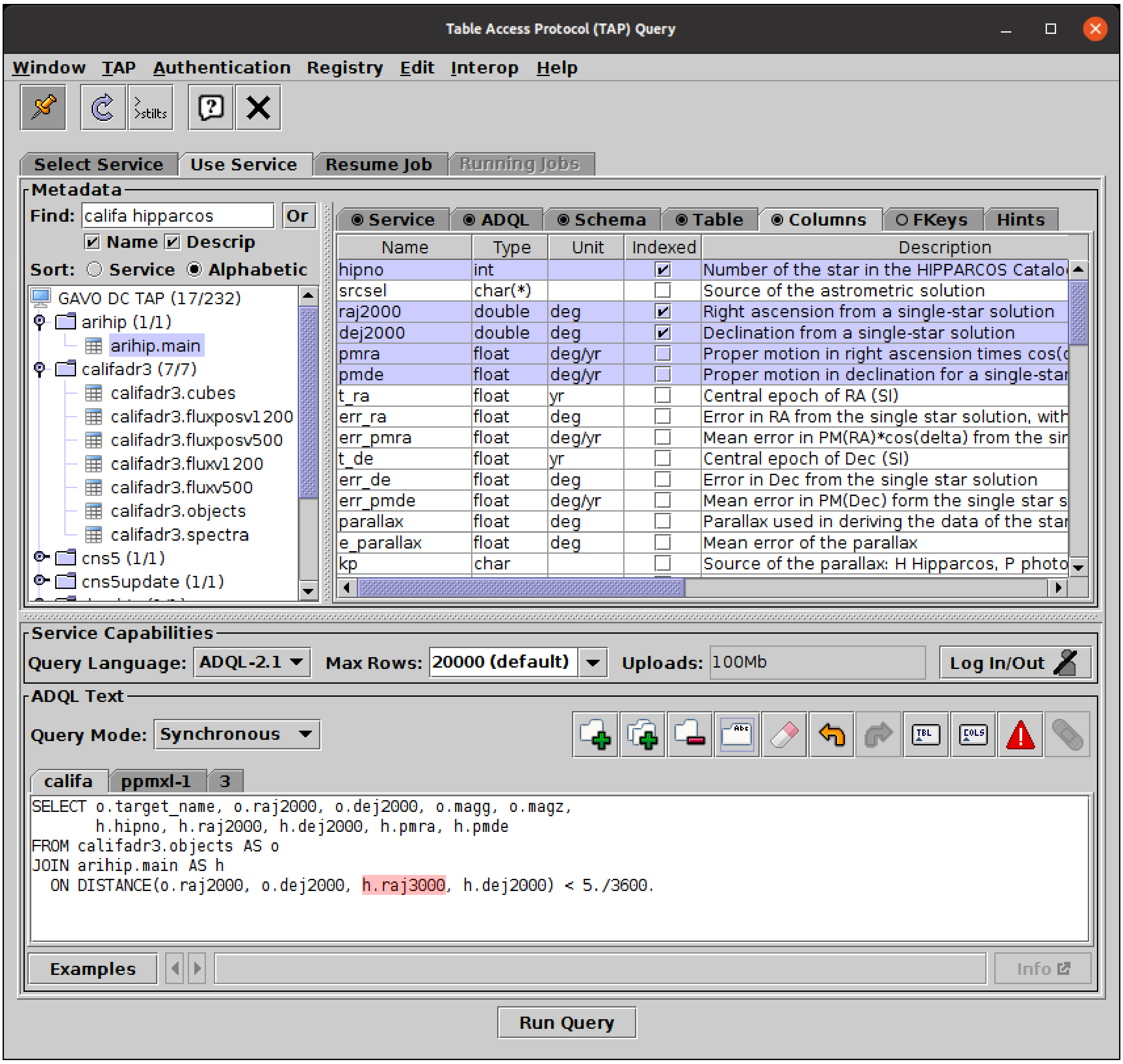
By Service Properties

Use this tab to find TAP services based on properties of the service itself, such as the service name or publisher. It is the same as the Registry Query Panel of the other VO query windows. Having queried the registry, click on one of the rows to enter its service URL in the **TAP URL** field at the bottom. At time of writing there are only about a hundred TAP services registered, so it's feasible to query the registry for them all (hit **Find Services** without entering any keywords), and choose the one you want by eye, perhaps sorting the displayed services by one of the metadata items (click on the column header to sort).

If you know the URL of the TAP service you wish to query, you can enter it directly into the **TAP URL** field at the bottom of the window without a registry search. This field remembers URLs previously entered into it, so if you click the little down-arrow on the right, you can easily return to services you visited earlier in the same session. The state (entered ADQL etc) of those sessions is also retained.

Once a service URL has been chosen in one of these ways, you can click the **Use Service** button at the bottom (or equivalently the tab header at the top), and you will be moved to the Use Service tab.

A.9.8.2 Use Service tab



Find: califa hipparcos

☑ Name ☑ Descrip

Sort: ○ Service ● Alphabetic

- GAVO DC TAP (17/232)
  - arihip (1/1)
    - arihip.main
  - califadr3 (7/7)
    - califadr3.cubes
    - califadr3.fluxposv1200
    - califadr3.fluxposv500
    - califadr3.fluxv1200
    - califadr3.fluxv500
    - califadr3.objects
    - califadr3.spectra
  - cns5 (1/1)
  - cns5update (1/1)

Name	Type	Unit	Indexed	Description
hipno	int		☑	Number of the star in the HIPPARCOS Catalo
srcsel	char(*)		□	Source of the astrometric solution
raj2000	double	deg	☑	Right ascension from a single-star solution
dej2000	double	deg	☑	Declination from a single-star solution
pmra	float	deg/yr	□	Proper motion in right ascension times cos(
pmde	float	deg/yr	□	Proper motion in declination for a single-sta
t_ra	float	yr	□	Central epoch of RA (SI)
err_ra	float	deg	□	Error in RA from the single star solution, with
err_pmra	float	deg/yr	□	Mean error in PM(RA)*cos(delta) from the sir
t_de	float	yr	□	Central epoch of Dec (SI)
err_de	float	deg	□	Error in Dec from the single star solution
err_pmde	float	deg/yr	□	Mean error in PM(Dec) form the single star s
parallax	float	deg	□	Parallax used in deriving the data of the star
e_parallax	float	deg	□	Mean error of the parallax
kp	char		□	Source of the parallax: H Hipparcos, P photo

Query Language: ADQL-2.1

Max Rows: 20000 (default)

Uploads: 100Mb

Query Mode: Synchronous

califa | ppmxl-1 | 3

```
SELECT o.target_name, o.raj2000, o.dej2000, o.magg, o.magz,
       h.hipno, h.raj2000, h.dej2000, h.pmra, h.pmde
FROM califadr3.objects AS o
JOIN arihip.main AS h
  ON DISTANCE(o.raj2000, o.dej2000, h.raj3000, h.dej2000) < 5./3600.
```

TAP window with Use Service tab visible

The **Use Service** tab of the TAP load dialogue displays information about the service you have selected to query, including what tables are available and what columns they contain, and allows you to enter the query text and some additional options about how the query is to be executed. Once you have entered suitable query text, click the **Run Query** button at the bottom of the window to submit the job.

The various parts of the window are described in detail below.

Metadata Panel

The **Metadata** panel displays information about the tables held by the selected service. On the left of the panel is a searchable **tree** summarising the schemas and tables in the service, and on the right are some **tabs** containing more detail. Note all this information has to be loaded from the server, so in some cases you may have to wait before it is visible.

The tree on the left contains a listing of the service's *Schemas*, and under them the *Tables* they contain. A Schema in this context is just a subject grouping that contains one or more tables. Each schema notes in brackets the number of tables it contains; you can reveal or hide the tables by clicking on the schema node's handle. If the service contains only a small number of tables it may be convenient just to scroll up and down to see them all. But for services with hundreds or thousands of tables, you might need some help to find the tables that you're interested in by using the controls above the tree:

Find

You can restrict the displayed tables to those that you're interested in. To do this, type one or more search terms into the **Find** field. The nodes displayed in the tree will be filtered to include only those that match the search terms as you type them. The **Name** and **Descrip** checkboxes below the field indicate whether you want your search terms to match tables/schemas by name and/or by description. The **And**/**Or** toggle button determines whether each displayed table/schema has to match all of the search terms or just one. So for instance if you want to list just two tables, you can type in both table full names, uncheck the Descrip control, and set the toggle button to "Or".

Sort

You can choose between two options to determine the order in which entries are displayed in the tree. **Alphabetic** means that the schemas are listed in alphabetical order, and within each schema the tables are listed in alphabetical order. **Service** attempts to use information provided by the data service about how to order the schemas and tables. Depending on how the data provider has organised its metadata this may list more important schemas and tables first, or it may show them in a more or less random order, or it may be alphabetical as with the other option.

To the right of the tree is a tabbed panel giving detail relating to the currently selected tree node. Select a tree node by clicking on it. Each tab has a little circle next to the title which may be empty or filled, according to whether it contains information. If it's empty it's either because that information doesn't make sense for the selected node in the tree (e.g. if a schema is selected, there is no column information) or because the information has not been retrieved from the service; either it's on its way and will be filled in later, or there's an error. The tabs are as follows:

Service tab

Provides various items of information about the TAP service itself, gathered from the service and its registry record if available. This includes the service's name, Service URL, Reference URL, curation and contact information, detailed description, declared data models, and a summary of optional and custom features of the ADQL implementation (these are listed with more detail in the next tab). It also displays your currently authenticated identity for service interaction, if any. If a Reference URL or Examples URL are provided, clicking on the URLs or the little link icon (↗) in the panel will try to open the relevant pages in an external web browser (or you can copy and paste them yourself). The content of this tab is not sensitive to which node of the tree is currently selected.

ADQL tab

Lists language features associated with the service. This includes User-Defined Functions (UDFs) alongside all other standard mandatory and optional functions the service provides, as well as supported optional and custom language features such as Common Table Expressions. These functions and features are displayed in a tree which can be explored by clicking on node handles (⊸). The content of this tab is not sensitive to which node of the tree is currently selected, but it may change according to which ADQL version is selected.

Schema tab

Provides a textual description of the currently selected schema, or the schema of the currently selected table. A schema is just a higher-level grouping of tables within the TAP service. Depending on how the service has arranged its tables this may or may not provide useful information that's not available at the table level.

Table tab

Provides a textual description of the currently selected table (if any). Some additional information like the number of columns, rows and foreign keys may also be displayed if available.

Columns tab
Tabulates a list of all the columns in the currently selected table; column names, data types, units, descriptions, UCDs, and possibly other metadata are shown. This is generally the information you will need to know about a table before you can write queries. You can sort the entries by clicking on the column header (clicking cycles between *ascending*, *descending*, and *unsorted*), so for instance to list the columns in alphabetical order click on the **Name** column header, or to group together all the columns with units of `mag` click on the **Unit** column header. If no table is selected in the tree, this tab will be empty.

FKeys tab
If the service provides information about foreign keys (links between fields in different tables in the database) they will be tabulated here for the currently selected table. If any are present, they may help you to formulate efficient queries. If no table is selected in the tree, this tab will be empty.

Hints tab
This panel gives you a very basic "Cheat Sheet" for writing ADQL queries. There are just a few hints to jog your memory about the required syntax for some common operations. But the very best advice it gives you is to use the **Examples** menu, and the service-provided examples (from the Examples menu or via a link) if available. The content of this panel is somewhat sensitive to the capabilities of the service, for instance the ADQL version selected and the availability of service-provided examples.

Service Capabilities Panel

The **Service Capabilities** panel shows capability metadata that the service provides about itself. This has to be loaded from the server and may not appear immediately. It contains the following information:

Query Language
Shows what query languages, and what versions, are supported by the service; the default selection is the most recent version of ADQL on offer. If there is more than one option listed, you can choose which language you want to submit your query in, but if you choose something which is not a variant of ADQL it may not work. The ADQL version selected here will affect the syntax highlighting in the ADQL Text panel below, as well as the content of the ADQL and Hints tabs, and the exact form of the example queries in some of the Examples menu entries. It's usually best to stick with the default language selection, but experimenting with earlier ADQL versions if available can help to understand differences between the ADQL versions.

Max Rows
Selector which shows the maximum number of output rows that the service is willing to deliver as the result of a query under normal circumstances. Entries in this list value may be marked "(default)" or "(max)". You can change this value by typing in a different number, as long as it does not exceed the server's maximum. This value is here to protect the user, as well as the service, from inadvertently requesting an unduly large output table. Note that if the construction "`TOP nnn`" is used in the ADQL, the `nnn` limit may override the value supplied here.

Uploads
Indicates whether table uploads are permitted, and perhaps what is the largest upload size (in terms of rows and/or bytes) which will be accepted. See below for more about table uploads.

Log In/Out
This button is used to log in or out of the TAP service in cases where this is permitted or required. If the service has no authentication, this button will be disabled. See the Authentication item below for more detail.

ADQL Text Panel

The **ADQL Text** panel is where you can enter the actual query text for the request. It has the following parts:

Query Mode selector
Provides the following options to control what happens when you hit the **Run Query** button:

- **Synchronous** means that TOPCAT sends the query to the server and waits for the result as the response of the same HTTP request. It's simpler and faster than async, but if the job takes a long time, the request may time out and the results be lost. When the query is submitted, the Load Window will appear with a load progress indicator, and once it's complete the table will be loaded into TOPCAT in the usual way.
- **Asynchronous** means that TOPCAT sends the query to the server and gets a message back telling it where to look for information about the job's progress; the application then keeps polling the server to find out when it has completed. When the query is submitted, you will be taken to the Running Jobs tab where you can see the progress of your submitted job; when it's ready it will be loaded into TOPCAT.
- **Quick Look** runs the query as for Synchronous mode above, but when the result comes back the data is just displayed in a new popup window rather than loading it into TOPCAT for further analysis.

Synchronous mode is suitable for short jobs (it may typically execute with a few seconds less delay), but for longer jobs aysnchronous is more reliable. Some TAP services permit longer queries to be executed in asynchronous than in synchronous mode. It is possible to submit an asynchronous job on one day from one machine, and pick up the results on a different day from another machine, using another invocation of TOPCAT or of a different application altogether. Quick look is good if you just want to see the result but don't want to do any detailed analysis of it. One example is if you just want to know the number of rows in a table (`SELECT COUNT(*) FROM table`); the result of the query is the row count in a one-column, one-row table, and there's not much point loading that table into TOPCAT.

Edit toolbar
This contains a number of actions that affect the **Text Entry panel**:

Add Tab
Adds a new blank editing tab, so you can start editing a new query without affecting any existing ones.

Copy Tab
Adds a new editing tab with text initially copied from that in the currently visible one.

Remove Tab
Deletes the currently visible tab.

Title Tab
Sets the title of the currently visible tab. By default the titles are consecutive integers (the first is "1", the second "2" etc), but you can give them more meaningful names if you like.

Clear

Erases all the text in the currently visible tab. The text can be retrieved using the Undo action.

Undo

Undoes the most recent editing action in the currently visible tab. You can do the same thing from the keyboard using **Ctrl-Z**.

Redo

Reverses the most recent Undo action in the currently visible tab. You can do the same thing from the keyboard using **Ctrl-Shift-Z** or **Ctrl-Y**.

Insert Table

Inserts into the text at the current cursor position the name of the table (if any) that is currently selected in the metadata tree above. This is a convenience that may save some typing, especially for long or complicated table names.

Insert Columns

Inserts into the text at the current cursor position a comma-separated list of the names of any columns that are currently selected in the **Columns** tab of the Metadata panel above. In order to select columns, click on the rows in the Columns panel; you may need to hold down the Ctrl key while clicking or something similar (depends on your OS) to select more than one. Columns are inserted in the text in the order they were selected. This is a convenience that may save some typing, but note that you may still need to edit the inserted text, for instance to add table name/alias prefixes.

Parse Errors

If this button is enabled, it means that the parser has detected an error in the currently visible ADQL text. The error will usually be highlighted in pink in the editing panel. Clicking this button pops up the actual parser error message, which may give you more idea what's wrong.

Fix Errors

If this button is enabled, it can fix some common errors in the currently visible ADQL. This includes things like putting quotes around table or column names that non-alphanumeric or that clash with ADQL reserved words. There are plenty of problems it can't fix!

These actions are also available from the **Edit** menu.

Text Entry area

This panel is where you type ADQL text that forms the query to send to the TAP service for execution. The **Edit** actions described above, along with Undo/Redo actions from the keyboard, can also help. You can have multiple queries on the go at once in different tabs; add/copy/remove/retitle tabs with the toolbar actions, and move between them by clicking on the tab headers at the top. The tab you can see is the one that will be executed if you hit the **Run Query** button.

As you enter the query text, it will be checked for syntax errors. If an error is found, the place in the text which appears to have caused it will be highlighted in pink (as in the figure above). To see more detail on the error, click the **Parse Errors (** ⚠ **)** button. In some cases, the parser can guess how to fix such errors; if the **Fix Errors (🩹)** button is enabled, you can try clicking that.

The checking itself is reasonably reliable, but the position of the highlighted error, and the text of the error message, can sometimes be a bit misleading, so don't take the details too seriously. The error highlighting is just used as a warning, you are not prevented from submitting a query marked as erroneous, since in some cases TAP services may accept ADQL which is not strictly correct, and in some cases the error checking may not be perfect. Note also that for various reasons the service may not be able to accept all queries that count as syntactically valid ADQL, so even if TOPCAT doesn't report any errors, the service may still respond with an error message.

The details of the syntax checking depend on the table metadata, ADQL version and optional language features declared by the service. If multiple ADQL versions are available in the Query Language selector, it is also be possible to change the language version checked by making a different selection.

Examples line

Clicking on the **Examples** button at the bottom left of the panel will pop up a menu that allows you to select from a number of example ADQL queries. If you select one, the relevant ADQL text will be pasted into the editing panel; you should be able to hit the **Run Query** button straight away and execute it to get a result. In some cases the example text is influenced by the table you're currently looking at in the Metadata panel, and by the ADQL version of the Query Language currently selected in the Service Capabilities panel. Until you are experienced with ADQL, starting from one of these examples and editing it to taste is often a good way to write a query of your own.

The Examples menu is divided into a number of sub-menus, though not all will be enabled for all services. The sub-menus are:

- **Basic:** Basic examples that should be applicable to most services.
- **Upload:** Examples that involve uploading a tables from TOPCAT to the TAP server and executing queries using it. Only enabled if the service supports table uploads and at least one table is currently loaded into TOPCAT.
- **Service-Provided:** Examples provided by the TAP service. These are likely to provide some insight into the kinds of query that make sense and make good use of the particular tables available from the service, so it is often instructive to look at these. Only enabled if the service actually provides some examples (at time of writing, not many do, but hopefully this will improve). *Note to TAP service implementers: at time of writing, there are two variants of the standard way for services to provide examples (DALI sec 2.3 and TAPNotes sec 4.2.1), and it's not clear which is preferred. For now, TOPCAT will recognise either.*
- **TAP_SCHEMA:** Examples using the tables in the TAP_SCHEMA schema, which describes metadata about the tables in the service. You can use these to do things like count tables or columns in the current service, or locate tables with columns having particular physical meaning.
- **ObsTAP:** Examples using the IVOA ObsCore data model (`ivoa.obscore` table), which describes astronomical observations in a standardised way. You can use these to do things like search for all observations at a particular sky position, or observation time, or waveband etc. Only available if the service declares support for the ObsCore data model.
- **RegTAP:** Examples using the IVOA RegTAP data model (tables in the `rr` schema). You can use these to do things like search for all TAP services containing tables having columns with particular physical meanings, or all cone search services relating to particular wavebands, etc. You can use queries like this to perform more sophisticated searches for TAP services than you can do using the Select Service tab. Only available if the TAP service declares support for the RegTAP data model (i.e. if it hosts an IVOA

searchable registry).

When you select an example from one of the sub-menus, its name and title will be displayed to the right of the Examples button. Little left and right arrow buttons allow you to cycle through the examples in the current submenu so you can browse what's available.

Some examples come with additional explanation on the web. If the example you're currently looking at has such additional documentation then the **Info** button to the right will be enabled, and clicking on that should open the relevant page in a browser on your desktop.

Some TAP services permit **Table Uploads**. What this means is that you can upload tables from TOPCAT into the remote database and perform queries on your table directly. In this way you can perform joins between your own tables (anything loaded into TOPCAT) and any of the tables in the remote database. This is an extremely powerful feature. To take advantage of it, you just need to use a special form for the table name to reference a TOPCAT-based table: you write "`TAP_UPLOAD.t<n>`", where `<n>` is the ID number of the table in TOPCAT, that is the number before the colon in the Control Window Table List. If the table's label has a suitable form (roughly, if it's alphanumeric and unique) you can use that instead of "`t<n>`". So, if a table is identified as "`1: messier`" in the table list, you can reference it in ADQL as "`TAP_UPLOAD.t1`" or as "`TAP_UPLOAD.messier`" - see the **Upload** sub-menu of the **Examples** menu described above for some examples. Note the table uploaded in this way is the Apparent Table corresponding to the given ID number, i.e. current subset and column selections apply. It's a good idea to ensure that any table you are uploading has columns with sensible names, otherwise the service may rename the columns or otherwise have trouble handling it.

Authentication

Some TAP services (at time of writing, a minority) permit or require you to log in before using the service, as discussed in Section 6.3. The behaviour of this window depends on whether the service provides mandatory, optional or no authentication, and can be controlled using the **Log In/Out** button on the right of the **Service Capabilities** panel in the middle of the window.

Mandatory Authentication
For services that require authentication, when you open the **Use Service** panel an authentication dialogue asking you for your username and password will appear before you can do anything else.

Optional Authentication
For services that allow optional authentication, the panel will be displayed initially in anonymous mode, but you can use the **Log In/Out** button ( ) to supply your username and password. The advantage that authenticating confers on your usage depends on the service; in some cases you will have access to more of the hosted data, in other cases it may entitle you to higher resource limits or enable protection of your asynchronous query results. TOPCAT can't tell you what the difference will be if you authenticate, but following successful login it will reload and display the service metadata in case it has changed for the new authenticated context.

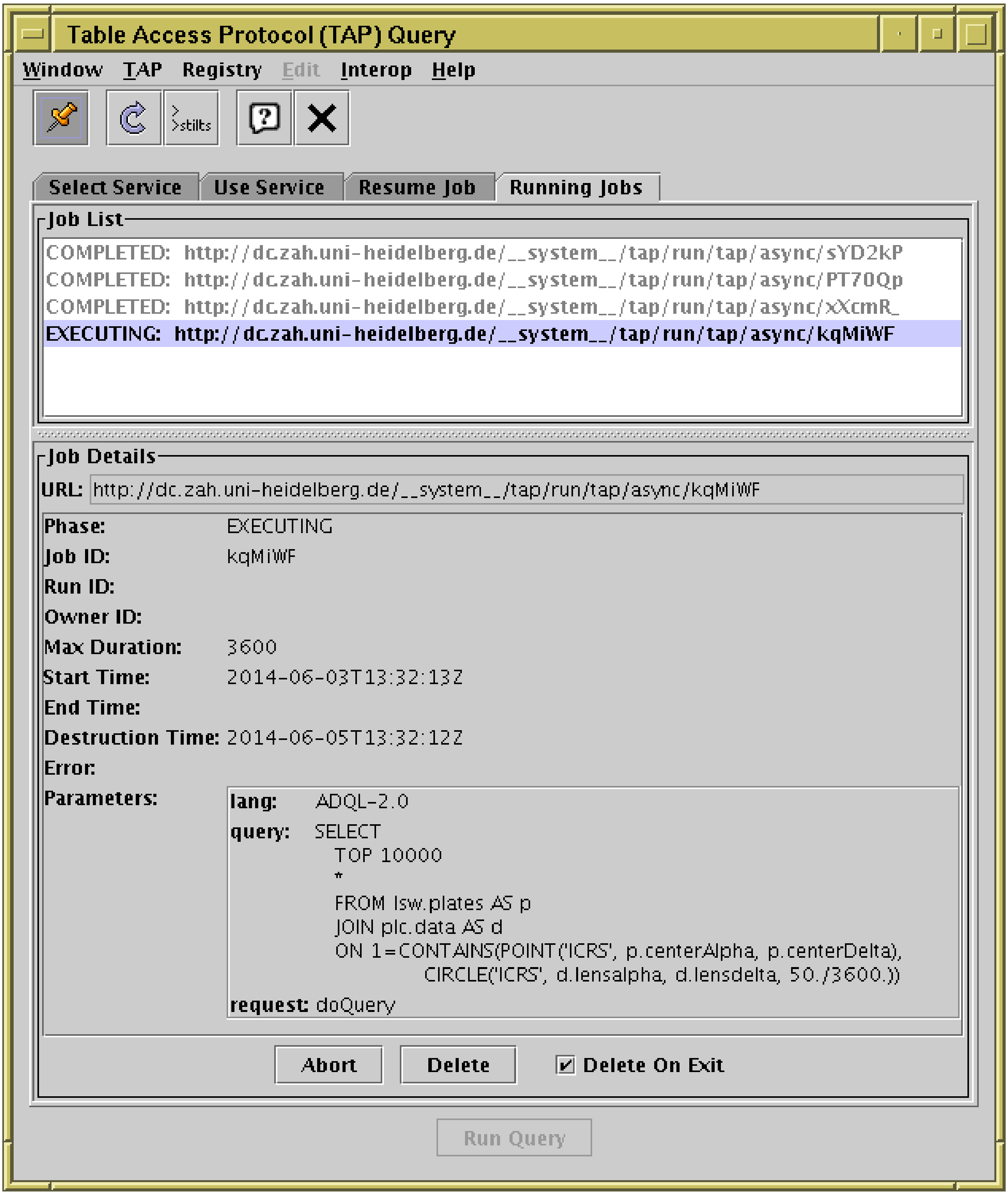
No Authentication
For services with no possibility of authentication, the **Log In/Out** button will be disabled.

The **Log In/Out** button's appearance changes depending on whether you are logged in ( ) or not ( ). To check your authenticated identity, you can hover over the button or look under the **Authentication** item in the **Service** tab of the **Metadata Panel** above.

If you have already logged in, clicking this button will pop up the authentication dialogue again and you can enter a different username and password, or select Anonymous to log out. It is also possible to log out of all services you are currently authenticated with by using the **Reset Authentication** option from the **Authentication** menu.

Note: These authentication arrangements in TOPCAT are new at version 4.9, and rely on VO standards that are still under discussion. The behaviour and user interface may change in future releases, and at time of writing not all TAP services that provide authentication advertise it in a way that TOPCAT can work with. It is hoped that authentication interoperability will improve in future versions of TOPCAT and of server-side software.

A.9.8.3 Running Jobs tab



```
COMPLETED:  http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async/sYD2kP
COMPLETED:  http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async/PT70Qp
COMPLETED:  http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async/xXcmR_
EXECUTING:  http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async/kqMiWF
```

URL: http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async/kqMiWF

Phase:	EXECUTING
Job ID:	kqMiWF
Run ID:	
Owner ID:	
Max Duration:	3600
Start Time:	2014-06-03T13:32:13Z
End Time:	
Destruction Time:	2014-06-05T13:32:12Z
Error:	
Parameters:	lang: ADQL-2.0

```
lang:    ADQL-2.0
query:   SELECT
           TOP 10000
           *
           FROM lsw.plates AS p
           JOIN plc.data AS d
           ON 1=CONTAINS(POINT('ICRS', p.centerAlpha, p.centerDelta),
                         CIRCLE('ICRS', d.lensalpha, d.lensdelta, 50./3600.))
request: doQuery
```

TAP window with Running Jobs tab visible

The **Running Jobs** tab of the TAP load dialogue shows a list of the asynchronous jobs you have submitted, with information about their current progress.

The upper part of the screen shows a list of the jobs submitted, along with a one-word summary of their status (their *Phase*). If one of these jobs is selected, its details will be shown in the lower part. By default the most recently submitted job will be shown, but you can select a job by clicking on it.

The information displayed is retrieved from the server, and can only be seen as long as the job has not been deleted, either by you the user, or as a result of server policy (for instance a maximum job lifetime). The following items, if provided by the service, are shown:

URL
The URL at which the job information is held on the server. This value can be used later in the Resume Job tab, or entered directly into a web browser to see job information.

Phase
The current stage of job processing. This usually progresses in the sequence QUEUED, EXECUTING, COMPLETED (or ERROR).

Job ID
Job identifier, unique to the server. Forms part of the URL.

Run ID
Run identifier, often blank.

Owner Id
Owner identifier, blank unless user authentication is in use.

Max Duration
The maximum amount of time in seconds that the server will allow the job to run for. This is "wall-clock" time rather than CPU time.

Start Time
Time at which actual processing of the job (as opposed to queuing) started.

End Time
Time at which processing of the job finished.

Destruction Time
Time in the future at which the server will delete the job, if it has not been manually removed by then.

Error
Message and possibly other information about error status if an error occurred during processing. This will normally be blank unless **Phase** has the value "ERROR".

Parameters
Displays parameters which were supplied to define the job to the server. This will normally include the text of the ADQL being executed.

You can cut and paste from these items, so for instance if you want to take the URL and paste it into a web browser you can do.

There are three buttons at the bottom of the screen which affect the status of the currently displayed job:

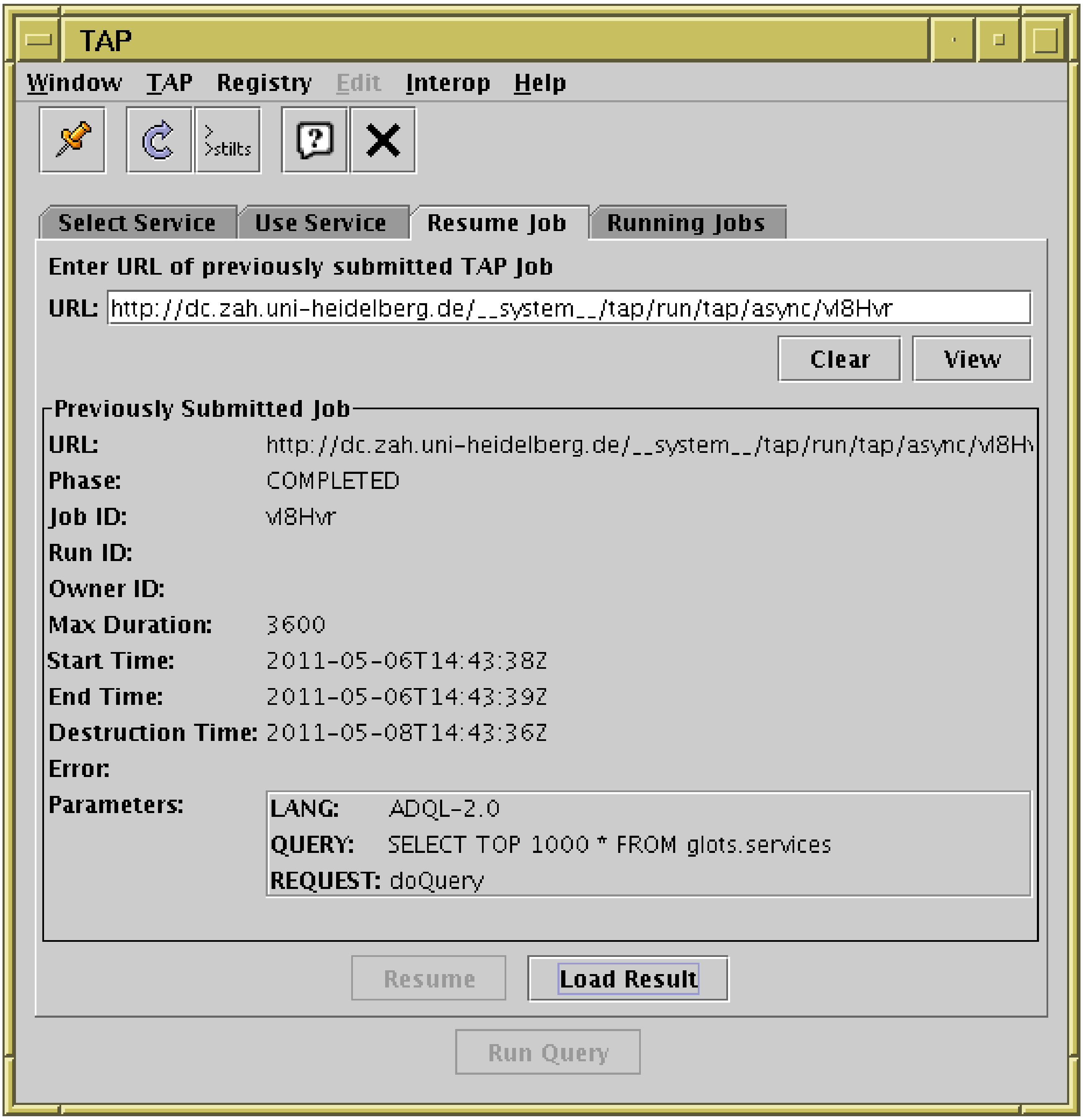
Abort
This button stops the job from executing. It cannot be restarted. The job still exists on the server however, so it can be examined in this window.

Delete
This button stops the job as for Abort, but also removes all trace of it from the server. No details can be seen any more.

Delete On Exit
This checkbox determines whether the job will be deleted from the server when TOPCAT exits. If you want to come back to a long-running job in a later session, you should uncheck this so that the job remains on the server. The default action is normally that jobs are deleted on exit (i.e. this box starts off checked), but you can change the default by using the **Deletion** menu.

If no jobs are currently visible (none have been submitted or they have all been deleted), this tab is inaccessible.

A.9.8.4 Resume Job tab



Enter URL of previously submitted TAP Job

URL: http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async/vl8Hvr

Previously Submitted Job

URL: http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async/vl8Hv

Phase: COMPLETED

Job ID: vl8Hvr

Run ID:

Owner ID:

Max Duration: 3600

Start Time: 2011-05-06T14:43:38Z

End Time: 2011-05-06T14:43:39Z

Destruction Time: 2011-05-08T14:43:36Z

Error:

Parameters:

LANG:	ADQL-2.0
QUERY:	SELECT TOP 1000 * FROM glots.services
REQUEST:	doQuery

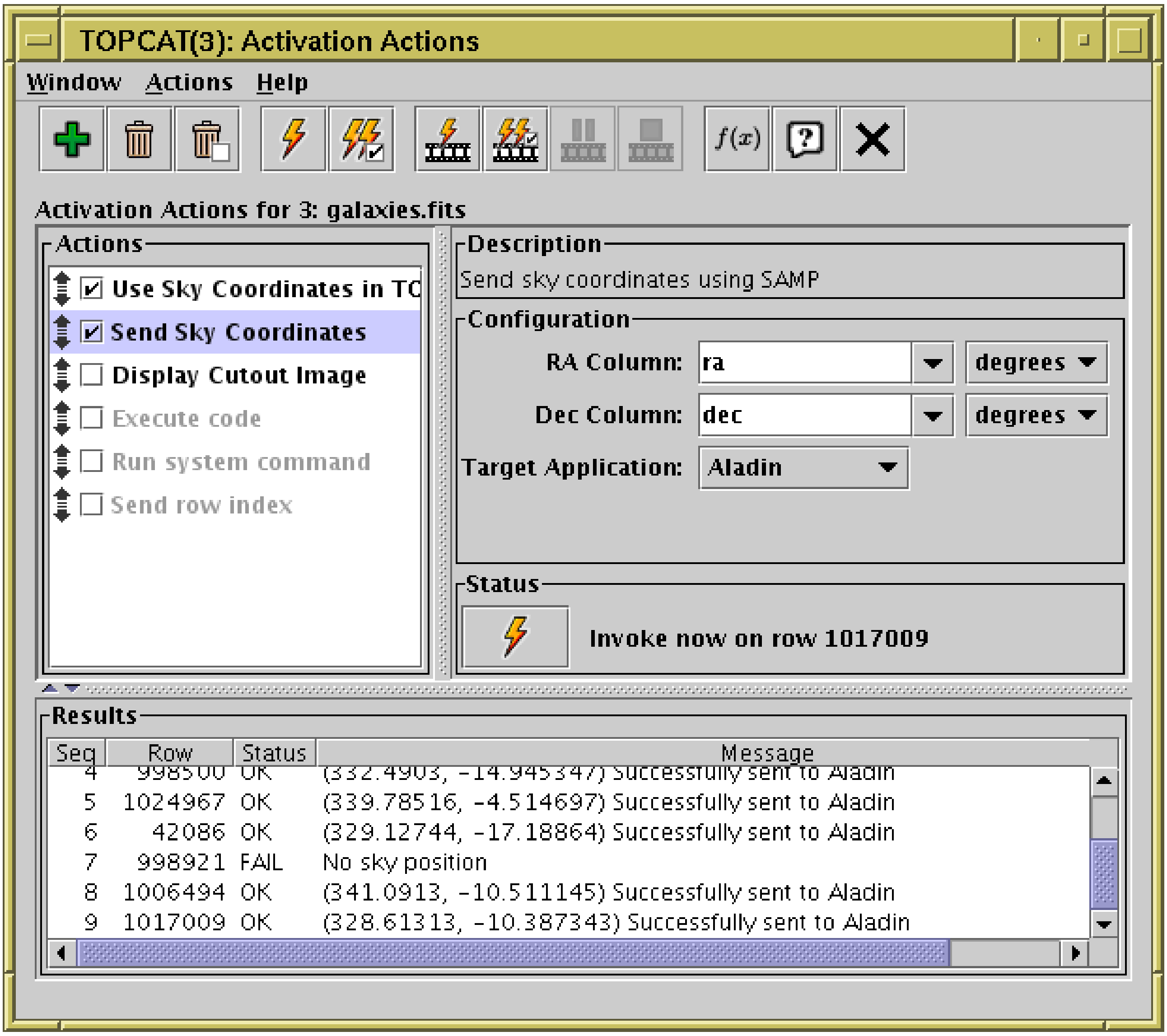
TAP window with Resume Job tab visible

The **Resume Job** tab of the TAP load dialogue allows you to continue execution of an asynchronous job which was started outside of this run of TOPCAT, either during an earlier TOPCAT run, or using some other mechanism. This may be useful if you have submitted a long-running job on an earlier day or on a different machine and want to monitor its progress or pick up the results.

To use it, enter the Job URL into the **URL** field at the top and click the **View** button. If a job with that URL exists, its details will be displayed in the panel below, in the same format as for the Running Jobs tab.

Depending on whether the job has completed or not, either the **Resume** or **Load Result** button at the bottom of the window will be enabled. You can click the appropriate one either to monitor its progress further or to pick up the completed result.

A.10 Activation Window



Activation Actions for 3: galaxies.fits

- ☑ Use Sky Coordinates in TC
- ☑ Send Sky Coordinates
- ☐ Display Cutout Image
- ☐ Execute code
- ☐ Run system command
- ☐ Send row index

Description: Send sky coordinates using SAMP

RA Column: ra degrees

Dec Column: dec degrees

Target Application: Aladin

Invoke now on row 1017009

Seq	Row	Status	Message
4	998500	OK	(332.4903, -14.945347) Successfully sent to Aladin
5	1024967	OK	(339.78516, -4.514697) Successfully sent to Aladin
6	42086	OK	(329.12744, -17.18864) Successfully sent to Aladin
7	998921	FAIL	No sky position
8	1006494	OK	(341.0913, -10.511145) Successfully sent to Aladin
9	1017009	OK	(328.61313, -10.387343) Successfully sent to Aladin

Activation Window

The Activation Window lets you configure what happens when a row is activated (Section 8). You can display it using the **Activation Window** ( ) button when the chosen table is selected in the Control Window's Table List.

A list of suggested Activation Actions is displayed in the top left panel, and if you select one of these its details, including configuration and invocation options and the results of invoking these actions to date, will be shown in the rest of the window. Those actions in the list which are enabled (not greyed-out) and have their checkbox checked will be invoked every time a table row is activated. Unchecked ones can be invoked manually. More description of each of the panels is given below.

The **Actions list** lists a number of activation actions that may be appropriate for the table this window applies to. Others, including duplicates of ones already listed, can be added from the **Add**

Action (✚) button or menu item. The only reason the full list is not visible by default is to avoid too much clutter in the window. Each action has a checkbox; if checked, an attempt will be made to invoke that action whenever a row is activated (e.g. clicked on). If the checkbox is not checked, the action can still be invoked manually for the current row using the **Invoke Selected Action on Current Row** (⚡) or **Invoke Selected Action on All Rows** (⚡) buttons. If the action name is greyed out, it means that it can't currently be invoked for some reason, perhaps because it needs more configuration first. To find out the reason, look in the **Status** panel, which should have a helpful message indicating why it's disabled. The grab handles (↕) can be used to re-order items in the list, though this order doesn't have much effect except visually. Actions can be removed from the list by selecting them and using the **Remove Current Action** (🗑) button.

The **Description panel** gives a short description of the behaviour of the currently selected action.

The **Configuration panel** displays a user interface to configure the details of what the selected action will do. The format depends on the action; see Appendix A.10.1 for details.

The **Status panel** contains a button (identical to that in the toolbar) that will invoke the currently selected action on the most recently activated row (or the first row, if no row has been activated so far in this table). It may be in one of two states. If the action is sufficiently configured to be used immediately, the button will be enabled, and clicking on it will cause the action to happen immediately, with the result being displayed in the Results panel below. However, if the action cannot be performed for some reason, the button will be greyed out, and some text will explain why it's disabled. Sometimes that means that the configuration in the panel above needs to be adjusted (e.g. some values filled in).

The **Results panel** contains a line added every time the currently selected action is invoked, either by the **Invoke** buttons (⚡, ⚡, ⚡, ⚡) or because a row has been activated elsewhere in the application. Each line contains four entries:

- **Seq:** A count that starts at 1 and increments for each invocation
- **Row:** The index of the row on which the action has been invoked
- **Status:** one of:
    - OK: action success
    - FAIL: action failure
    - CANCELLED: if the action was interrupted because the next invocation started before it had finished
    - (blank): if the result is not yet available
- **Message:** Some indication of what happened (OK) or what went wrong (FAIL) when the action was performed

A **Security panel** may occasionally also be visible. If you see this, consult Appendix A.10.3 for an explanation.

The following items are available in the toolbar or menus:

✚ **Add Action**

Add a new action to the Actions List from the full menu of available actions, as enumerated in Appendix A.10.1. This can be used to add an action that's not visible by default, because TOPCAT didn't think you would want to use it for this table, or to add a duplicate of an existing visible action, for instance to send a similar SAMP message to two different external tools.

Remove Selected Action

Removes the currently selected action from the Actions list.

Remove Inactive Actions

Removes all the actions from the Actions list, except the currently active ones (ones with their checkbox checked). This can be useful to clear out the list and remove clutter. Any actions you want to restore can be added back later using the **Add Action** button.

Approve All Actions

Removes security blockers from all actions that currently have them. This has security implications, and should only be used if you are confident that the activation actions and corresponding data have not been configured maliciously. See Appendix A.10.3 for explanation.

Invoke Selected Action on Current Row

This causes the currently selected action to be activated on the most recently activated row (or the first row if none have been activated yet). You can use it to invoke the action without having to actually click on a row in the view window or a plot. It will invoke the action (if it can) whether or not it is currently active, i.e. even when its checkbox is not checked.

Activate Current Row

Activates all the currently enabled actions on the most recently activated row (or the first row if none have been activated yet). This does the same as clicking on a row in the Data Window or a plot.

Invoke Selected Action on All Rows

This invokes the currently selected action to be activated for each row in turn in the apparent table (i.e. each row in the current row subset). As each row is activated, the result will be displayed in the Results panel, and a progress bar will indicate how far through the table it's got.

Activate All Rows

Activates all the currently enabled actions for each row in turn in the apparent table (i.e. each row in the current row subset). As each row is activated, the result will be displayed in the Results panel of each active action, and a progress bar will indicate how far through the table it's got. The effect is the same as clicking on every row of the table displayed in the Data Window one at a time. If this is being used to generate a "slideshow", the Delay action may be useful.

Pause Sequence

Pauses/restarts a sequence of activations that was started by either of the **... All Rows** actions above. Once paused, the sequence can be restarted from the same position by hitting this button again.

Cancel Sequence

Cancels a sequence of activations that was started by either of the **... All Rows** actions above. This may be done either while the sequence is running, or while it is paused. Once cancelled, re-running it will start again from the beginning.

Available Functions

Displays a window containing all the functions which can be used for writing algebraic expressions (see Section 7).

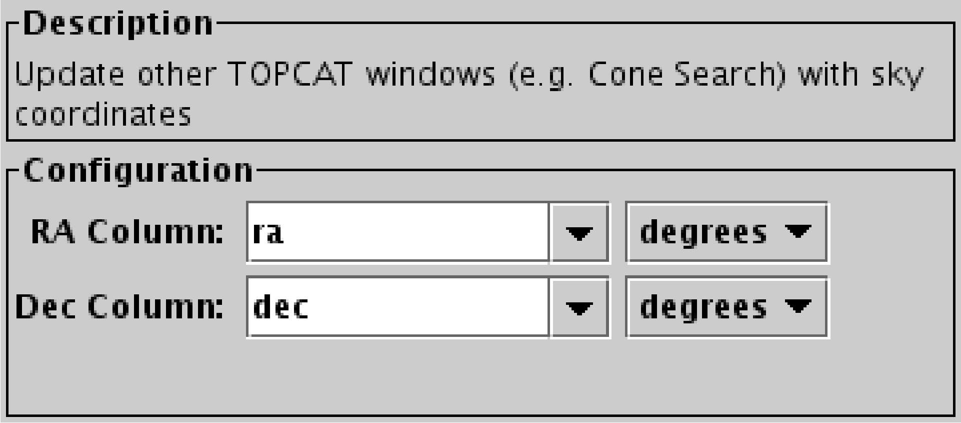
All the state of this window is saved and restored if you save its corresponding table using the Save Session option.

A.10.1 Activation Actions

A range of activation actions is available. Which ones are displayed for selection by default depends on the characteristics of the table, but if you want to use one that's not displayed, you can use the **Actions|Add Action** (✚) menu item to insert it into the list.

The available actions are listed in the following sections.

A.10.1.1 Use Sky Coordinates in TOPCAT



Description

Update other TOPCAT windows (e.g. Cone Search) with sky coordinates

Configuration

RA Column: ra degrees

Dec Column: dec degrees

Configuration for Use Sky Coordinates in TOPCAT action

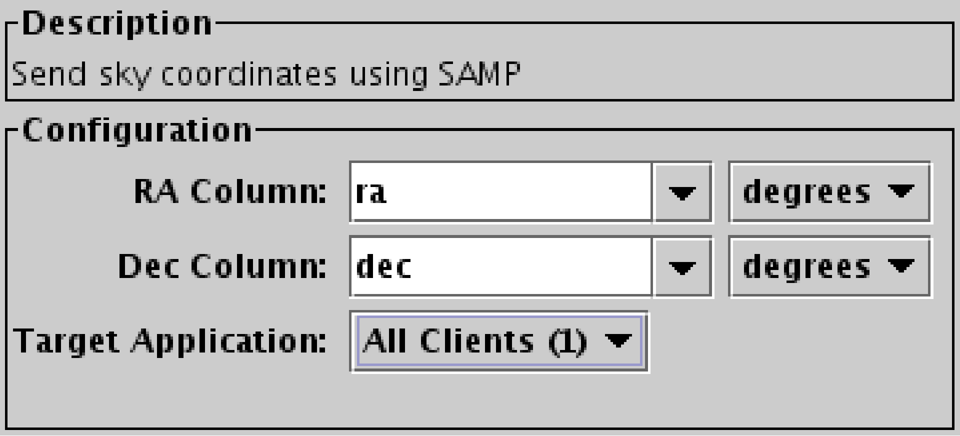
The **Use Sky Coordinates in TOPCAT** action causes the sky coordinates of an activated row to be communicated to certain other TOPCAT windows that have sky position entry widgets, for instance to update the query position in the Cone Search and Simple Image Access windows, and to be used as the default position in certain queries from the **Examples** menu in the TAP window.

This is in most cases useful behaviour so if sky coordinates can be identified this action will be enabled by default. If you find it unhelpful however (e.g. you have typed a position into the Cone Search window position selector and you don't want it to be overwritten) it can be disabled.

Configuration:

RA Column
Dec Column
Identifies the table columns representing sky coordinates.

A.10.1.2 Send Sky Coordinates



Description

Send sky coordinates using SAMP

Configuration

RA Column: ra degrees

Dec Column: dec degrees

Target Application: All Clients (1)

Configuration for Send Sky Coordinates action

The **Send Sky Coordinates** action causes the sky coordinates associated with the activated row to be sent using SAMP to one or all suitable applications. It sends a message with the MType `coord.pointAt.sky`, so at least one external SAMP client subscribed to such messages must be registered.

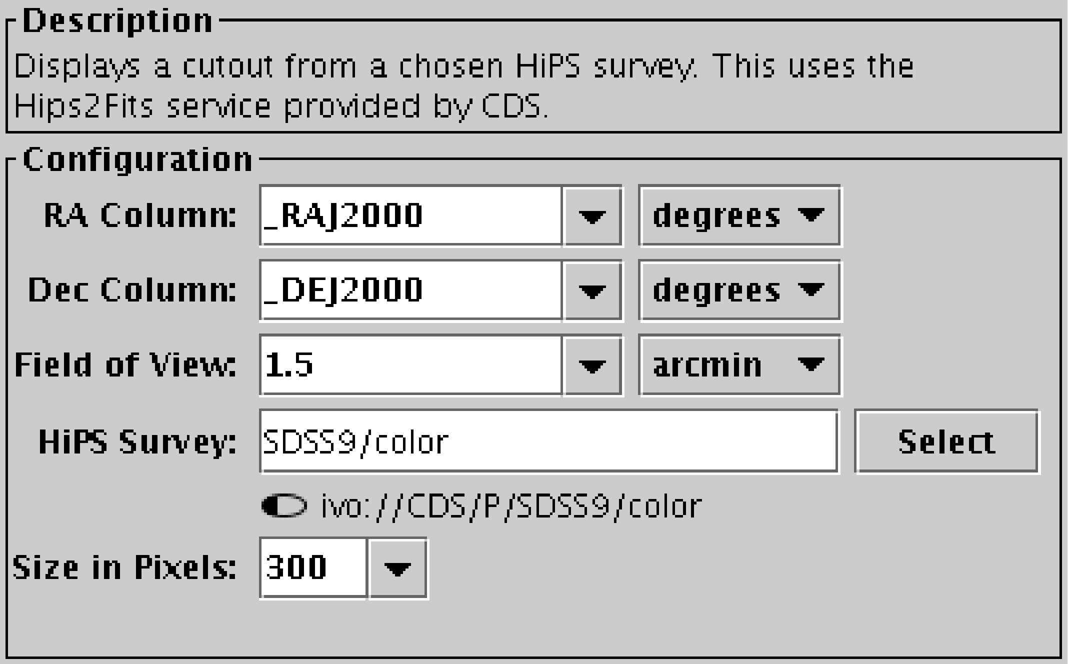
One way to use this action is to configure it to transmit to a viewer of all-sky imagery such as Aladin. Then whenever a row is activated, Aladin will change its view to show imagery at the sky coordinates corresponding to that row.

Configuration:

RA Column
Dec Column
Identifies the table columns representing sky coordinates.

Target Application
Determines which external client(s) to send the messages to. The default is "All Clients" which will send it to all eligible applications, but you can select just one. Clicking the selector also serves to indicate which clients capable of acting on this message type are currently registered. If there are none, the action will be disabled.

A.10.1.3 Display HiPS Cutout



Description

Displays a cutout from a chosen HiPS survey. This uses the Hips2Fits service provided by CDS.

Configuration

RA Column: _RAJ2000 degrees

Dec Column: _DEJ2000 degrees

Field of View: 1.5 arcmin

HiPS Survey: SDSS9/color

ivo://CDS/P/SDSS9/color

Size in Pixels: 300

Configuration for Display HiPS Cutout action

The **Display HiPS Cutout** action uses the hips2fits service at CDS to acquire a resampled image at the activated row's sky (or planetary) position and displays it in a no-frills image viewer window. The data is taken from a Hierarchical Progressive Survey (HiPS), an all-sky image data set. There are several hundred HiPS available providing imagery for many important sky and planetary survey datasets, so this easy to use activation action is often a good option for viewing imagery associated with activated positions.

Configuration:

RA Column
Dec Column
Identifies the table columns representing coordinates. For sky data, these are in ICRS, but in the case of planetary data they just represent longitude and latitude.

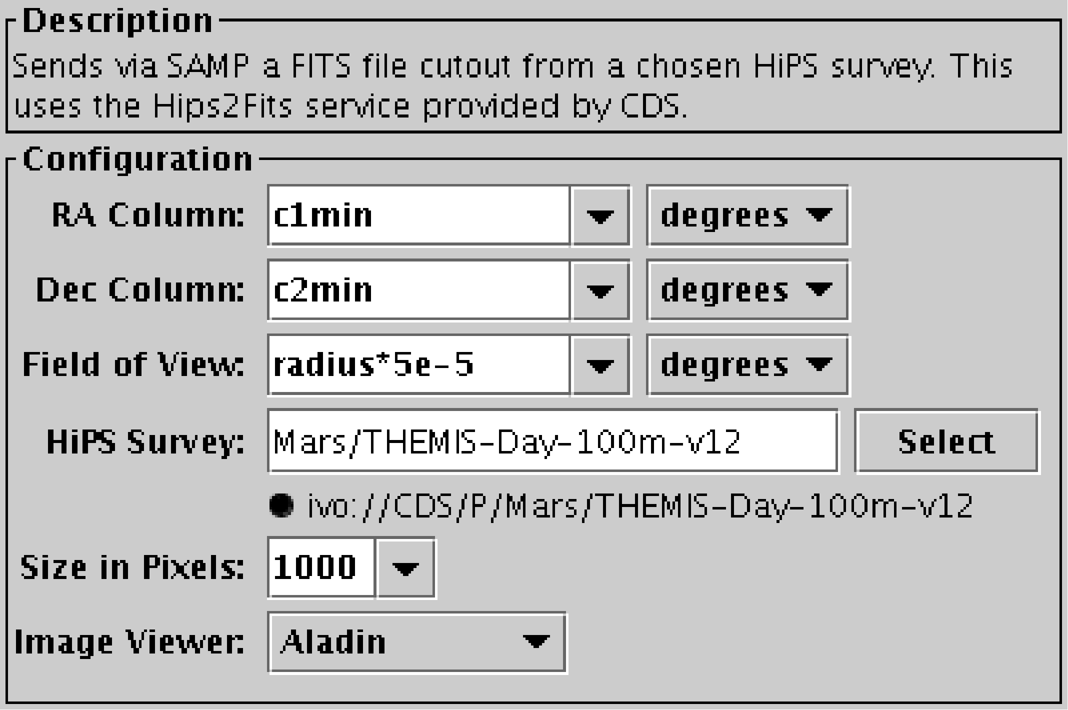
Field of View
Gives the angular size of the cutout on the sky. In most cases this will be a fixed value such as 1 degree or 45 arcsec, but you can choose a table expression instead, for instance the name of a column giving the source radius.

HiPS Survey
Selects the survey from which the cutout is taken. The **Select** button on the right opens a menu with a hierarchical list of available surveys, including numerous sky and planetary/solar system object datasets. A little icon is displayed next to each survey in the menu indicating fractional sky coverage - a filled in ellipse indicates that the whole sky (or planet) is covered. If a HiPS has been selected from the menu, its full identifier and coverage icon is displayed below the entry field; hovering over these with the mouse gives a bit more information. It is also possible to type the full or partial survey name directly into the entry field.

Size in Pixels
The width of the acquired image in pixels. The image is square, so this is also its height.

Alternatives: The hips2fits service is very flexible. If this action doesn't invoke it in exactly the way you want it to, you can customise the behaviour by using the Display Image action instead constructing your own custom hips2fits URL, perhaps by using the `hips2fitsUrl` function from the URLs class. You can also use the Send HiPS Cutout action if you want to view the image in an external viewer such as Aladin or ds9. If you want to be able to zoom around the HiPS image however, the best thing might be to open the HiPS in Aladin and use the Send Sky Coordinates action to communicate with it.

A.10.1.4 Send HiPS Cutout



Description

Sends via SAMP a FITS file cutout from a chosen HiPS survey. This uses the Hips2Fits service provided by CDS.

Configuration

RA Column: c1min degrees

Dec Column: c2min degrees

Field of View: radius*5e-5 degrees

HiPS Survey: Mars/THEMIS-Day-100m-v12

● ivo://CDS/P/Mars/THEMIS-Day-100m-v12

Size in Pixels: 1000

Image Viewer: Aladin

Configuration for Send HiPS Cutout action

The **Send HiPS Cutout** action uses the hips2fits service at CDS to get the URL of a resampled image at the activated row's sky position and asks an external image viewer application to load it via SAMP. The data is taken from a Hierarchical Progressive Survey (HiPS), an all-sky image data set.

Since the Aladin viewer already has all-sky viewing of HiPS datasets built in, you may find it better to load the HiPS directly into Aladin and use the Send Sky Coordinates action to communicate with it.

Configuration:

RA Column
Dec Column
Identifies the table columns representing coordinates. For sky data, these are in ICRS, but in the case of planetary data they just represent longitude and latitude.

Field of View
Gives the angular size of the cutout on the sky. In most cases this will be a fixed value such as 1 degree or 45 arcsec, but you can choose a table expression instead, for instance the name of a column giving the source radius.

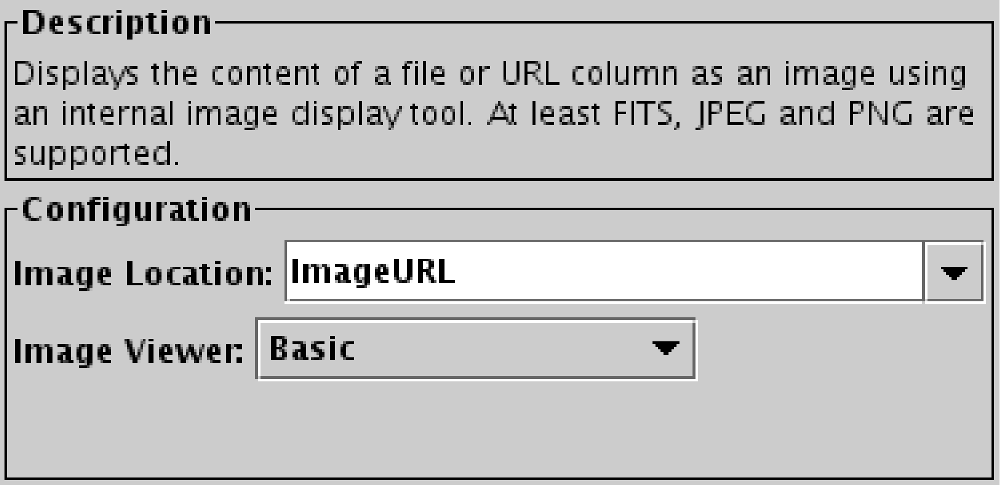
HiPS Survey
Selects the survey from which the cutout is taken. The **Select** button on the right opens a menu with a hierarchical list of available surveys, including numerous sky and planetary/solar system object datasets; note this only includes surveys with FITS (rather than just image) content. A little icon is displayed next to each survey in the menu indicating fractional sky coverage - a filled in ellipse indicates that the whole sky (or planet) is covered. If a HiPS has been selected from the menu, its full identifier and coverage icon is displayed below the entry

field; hovering over these with the mouse gives a bit more information. It is also possible to type the full or partial survey name directly into the entry field.

Size in Pixels
The width of the acquired image in pixels. The image is square, so this is also its height.

Image Viewer
Determines which external client(s) to send the image URLs to. The default is "All Clients" which will send it to all eligible applications, but you can select just one. Clicking the selector also serves to indicate which clients capable of acting on this message type are currently registered. If there are none, the action will be disabled.

A.10.1.5 Display Image



Description

Displays the content of a file or URL column as an image using an internal image display tool. At least FITS, JPEG and PNG are supported.

Configuration

Image Location: ImageURL

Image Viewer: Basic

Configuration for Display Image action

The **Display Image** action uses the contents of a column to provide a URL or filename giving the location of an image to display in an internal viewer. The image can be in a graphics format such as JPEG, PNG, or GIF, or it can be a FITS image. The standard image support is implemented by Java's ImageIO library, so the details of installation determine what formats are supported. Java version 9 has more extensive image format support than earlier Java versions.

Note that for more sophisticated image handling, especially for FITS images, the Send FITS Image option in conjunction with an external image viewer application may be more appropriate.

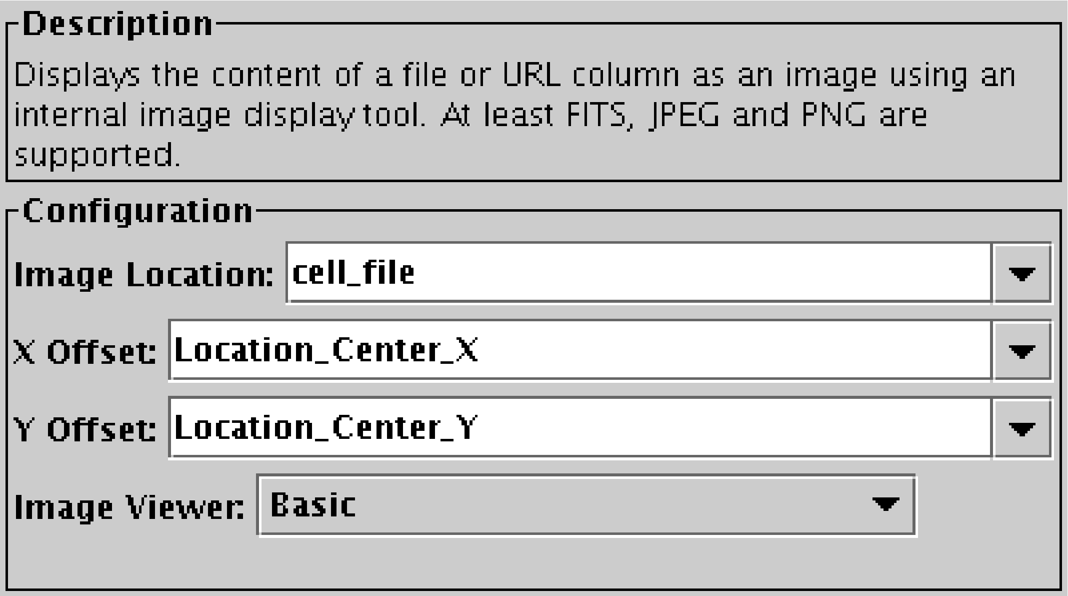
Configuration:

Image Location
Gives the URL or filename of the image to display for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a location string from other columns.

Image Viewer
Selects which of the internal image viewers will be used for display. At least some of the following options will be available; see Appendix A.10.2 for more detail:

- **Basic:** A minimal image viewer.
- **Basic (allow preprocessing):** The same image viewer as Basic, but the image location is permitted to contain a preprocessing system command, for instance "`< convert " + filename + " -normalize png:-`", using the ImageMagick `convert` command. This rather advanced usage may only work on Un*x-like OSs. This option is usually not required, and has security implications; if it is used in an action loaded from a session file, the security panel will be displayed.

A.10.1.6 Display Image Region



Description

Displays the content of a file or URL column as an image using an internal image display tool. At least FITS, JPEG and PNG are supported.

Configuration

Image Location: cell_file

X Offset: Location_Center_X

Y Offset: Location_Center_Y

Image Viewer: Basic

Configuration for Display Image Region action

The **Display Image Region** action behaves like Display Image, but additionally identifies an X/Y position on the displayed image, in pixel coordinates. This is currently done by scrolling the image if necessary to ensure the identified position is in view, and highlighting it with green crosshairs.

Configuration:

Image Location
: Gives the URL or filename of the image to display for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a location string from other columns.

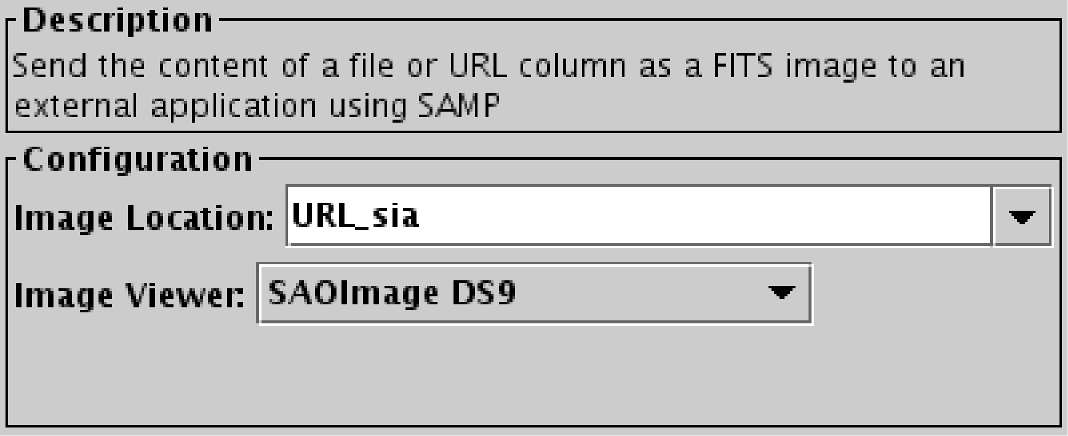
X Offset
: Horizontal offset in pixels from the left of the image for the position of interest. In most cases you will select a column from the drop-down list, but this could also be a constant or algebraic expression.

Y Offset
: Vertical offset in pixels from the top of the image for the position of interest. In most cases you will select a column from the drop-down list, but this could also be a constant or algebraic expression. For most image formats, Y values increasing downwards is the natural direction; for FITS however Y is normally considered to increase upwards. This viewer displays FITS images upside down so that the offsets probably do what you want, but the images may be inverted. This somewhat confusing behaviour may (or may not) be changed in the future.

Image Viewer
: Selects which of the internal image viewers will be used for display. The following options are available;

- **Basic:** A minimal image viewer.
- **Basic (allow preprocessing):** The same image viewer as Basic, but the image location is permitted to contain a preprocessing system command, for instance `"< convert " + filename + " -normalize png:-"`, using the ImageMagick `convert` command. This rather advanced usage may only work on Un*x-like OSs. This option is usually not required, and has security implications; if it is used in an action loaded from a session file, the security panel will be displayed.

A.10.1.7 Send FITS Image



Description

Send the content of a file or URL column as a FITS image to an external application using SAMP

Configuration

Image Location: URL_sia

Image Viewer: SAOImage DS9

Configuration for Send FITS Image action

The **Send FITS Image** action takes a URL or filename in the table that points to a FITS image, and sends it using SAMP to an external image viewer for display. It sends a message with the MType `image.load.fits`, so at least one external SAMP client subscribed to such messages must be registered.

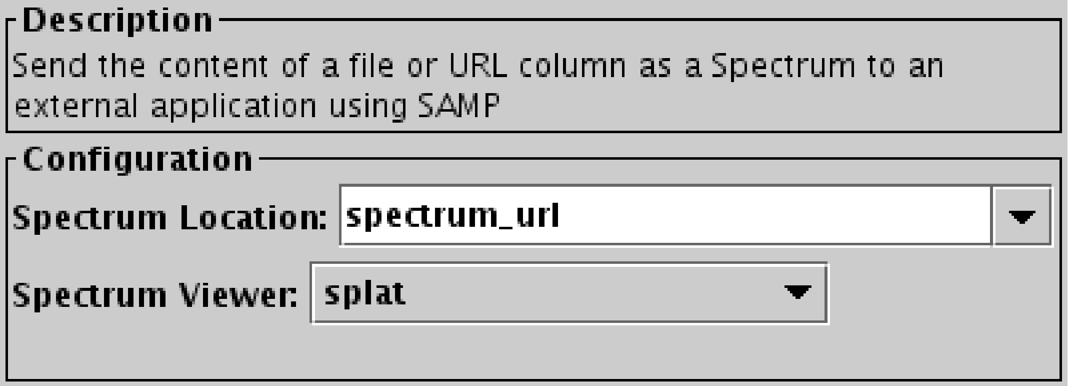
Suitable image viewers for use with this action include Aladin, GAIA and SAOImage DS9.

Configuration:

Image Location
Gives the URL or filename of the image to display for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a location string from other columns.

Image Viewer
Determines which external client(s) to send the image URLs to. The default is "All Clients" which will send it to all eligible applications, but you can select just one. Clicking the selector also serves to indicate which clients capable of acting on this message type are currently registered. If there are none, the action will be disabled.

A.10.1.8 Send Spectrum



Description

Send the content of a file or URL column as a Spectrum to an external application using SAMP

Configuration

Spectrum Location: spectrum_url

Spectrum Viewer: splat

Configuration for Send Spectrum action

The **Send Spectrum** action takes a URL or filename in the table that points to a spectrum resource, and sends it using SAMP to an external viewer for display. The supported spectrum data formats are dependent on the external application in question. It sends a message with the MType `spectrum.load.ssa-generic`, so at least one external SAMP client subscribed to such messages must be registered.

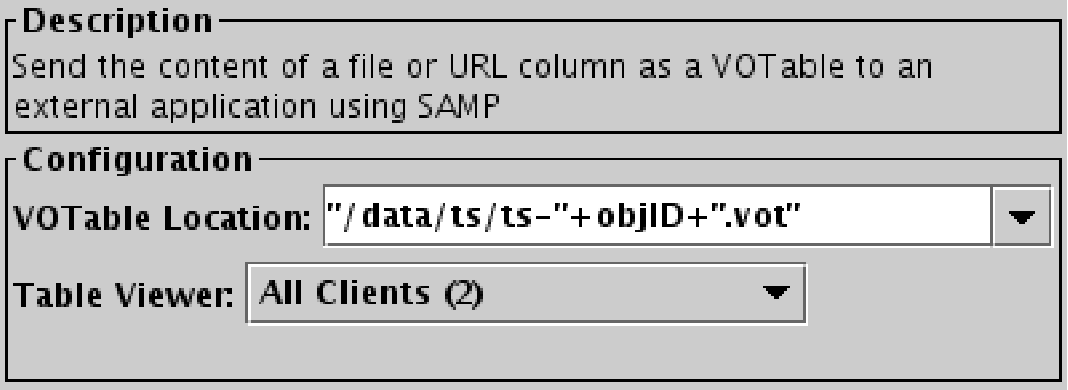
Suitable spectrum viewers for use with this action include SPLAT and VOSpec.

Configuration:

Spectrum Location
Gives the URL or filename of the spectrum to display for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a location string from other columns.

Spectrum Viewer
Determines which external client(s) to send the spectrum URLs to. The default is "All Clients" which will send it to all eligible applications, but you can select just one. Clicking the selector also serves to indicate which clients capable of acting on this message type are currently registered. If there are none, the action will be disabled.

A.10.1.9 Send VOTable



Description

Send the content of a file or URL column as a VOTable to an external application using SAMP

Configuration

VOTable Location: "/data/ts/ts-"+objID+".vot"

Table Viewer: All Clients (2)

Configuration for Send VOTable action

The **Send VOTable** action takes a URL or filename in the table that points to a VOTable document, and sends it as a table using SAMP to an external program to load. It sends a message with the MType `table.load.votable`, so at least one external SAMP client subscribed to such messages must be registered.

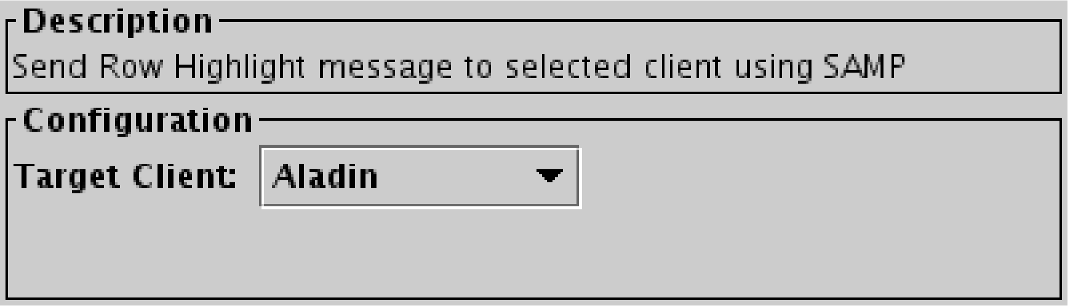
This action can be used to send tables to other running instances of TOPCAT, or to different applications such as GAIA, Aladin or SAMP-enabled Python scripts.

Configuration:

VOTable Location
Gives the URL or filename of the VOTable to send for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a location string from other columns.

Table Viewer
Determines which external client(s) to send the VOTable URLs to. The default is "All Clients" which will send it to all eligible applications, but you can select just one. Clicking the selector also serves to indicate which clients capable of acting on this message type are currently registered. If there are none, the action will be disabled.

A.10.1.10 Send Row Index



Description

Send Row Highlight message to selected client using SAMP

Configuration

Target Client: Aladin ▾

Configuration for Send Row Index action

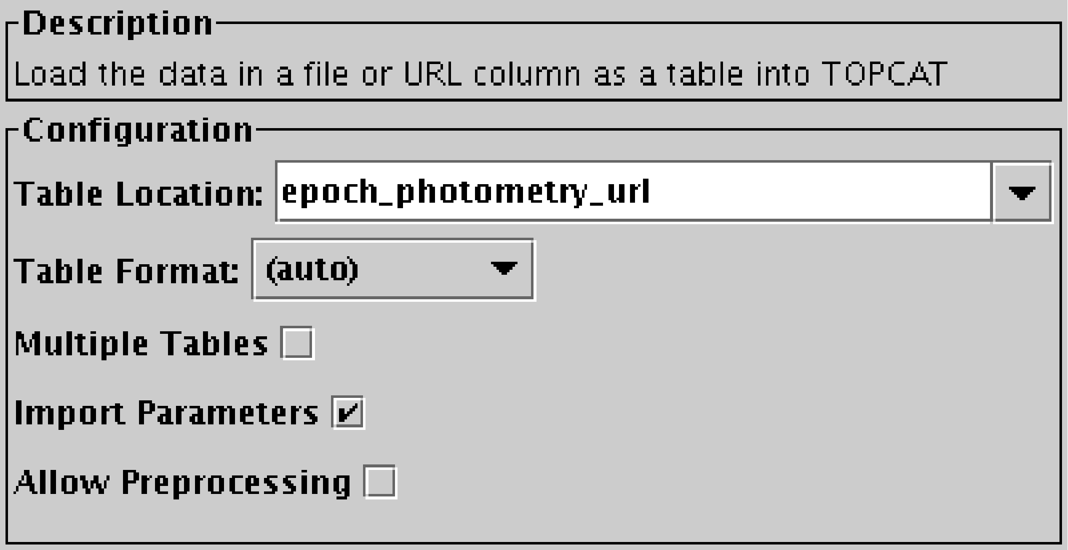
The **Send Row Index** action sends a message to other SAMP applications giving the row index of the activated row. This is in general only useful to other applications if they are already aware of the table whose row indices are being communicated like this, so a usual pattern of usage would be first to send the table to one or several external applications, and then to set up this action to indicate which rows are being selected. If it is known that the table in question has never been communicated to other applications, the action will be disabled. The message sent has the MType `table.highlight.row`, so at least one external SAMP client subscribed to such messages must be registered.

Suitable clients for use with this action include image viewers such as Aladin, GAIA and SAOImage DS9. If you send a table to one of (e.g. using the **Broadcast Table** (📡) button in the Control Window toolbar) and then set up this action to send row indices, then the image viewer will plot the catalogue positions over its displayed imagery, and whenever a row is activated within TOPCAT, will highlight the corresponding plotted point in some way.

Configuration:

Target Client
Determines which external client(s) to send the row indices to. The default is "All Clients" which will send it to all eligible applications, but you can select just one. Clicking the selector also serves to indicate which clients capable of acting on this message type are currently registered. If there are none, the action will be disabled.

A.10.1.11 Load Table



Description

Load the data in a file or URL column as a table into TOPCAT

Configuration

Table Location: epoch_photometry_url ▾

Table Format: (auto) ▾

Multiple Tables □

Import Parameters ☑

Allow Preprocessing □

Configuration for Load Table action

The **Load Table** action takes a URL or filename in the table that points to a table in one of TOPCAT's supported formats, and loads that into TOPCAT as a new table.

Configuration:

Table Location:
Gives the URL or filename of the table to load for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a location string from other columns.

Table Format:
Selects the file format of the table to load. If the table is in one of the formats that TOPCAT is able to auto-detect (VOTable, FITS, ECSV, CDF, PDS4, Feather) then the default value `(auto)` may be used, otherwise the correct format must be chosen from the list.

Multiple Tables:
Determines what happens if the loaded file contains multiple tables; some formats such as FITS and VOTable permit this. If this checkbox is selected, then all tables in the file will be loaded, otherwise only the first one is loaded and the others are ignored.

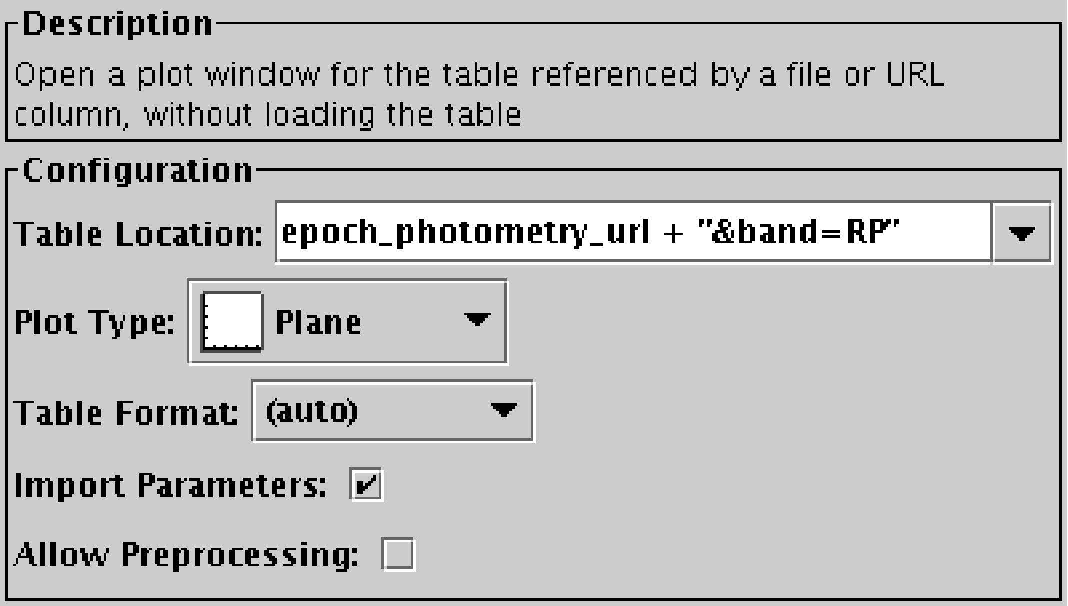
Import Parameters:
If this option is set, then all the values in the activated row of the activated table will appear in the newly loaded table as *table parameters*. That means they can be viewed in the Parameter Window and accessed from the expression language using the param$*name* syntax.

Allow Preprocessing:
If set, the "`<syscmd`" or "`|syscmd`" syntax described in Section 4.2 may be used to preprocess the input stream before the table is loaded. This option is usually not required, and has security implications; if it is switched on in an action loaded from a session file, the security panel will be displayed.

A.10.1.12 Plot Table

The **Plot Table** action takes a URL or filename in the table that points to a table in one of TOCAT's supported formats, and displays a plot window that can plot the data from that table. This is like loading the table (e.g. with the Load Table action) and then opening a plot window in the usual way, but doing it this way does not actually load the table into TOPCAT's list of loaded tables, and every time the action is activated, the data from the last invocation is just replaced with that from the current one (as far as possible - the tables need to have a similar structure for each row). In the plot window, the referenced table is listed first in the Table selector (or selectors, if multiple layers are configured) with the special name "**0: Activated**". All the currently loaded tables are available in the window as well as the activated one, so you can plot those for comparison in the same window if you want.

The configuration of this action lets you select the type of plot window to use, but other than that does not let you specify in detail how you want the plot to look. However, after you have activated the action once, you can interactively configure the plot as required using all the usual plot options, and subsequent activations of the same action will just update the plot with the data from the newly activated row, without affecting the rest of the configuration. This makes it easy to visually compare column data from tables linked in different rows of the original table. Note however that the details of the plot window configuration are not included when a session is saved.



Description

Open a plot window for the table referenced by a file or URL column, without loading the table

Configuration

Table Location: epoch_photometry_url + "&band=RP"

Plot Type: Plane

Table Format: (auto)

Import Parameters: ☑

Allow Preprocessing: □

Configuration for Plot Table action

Configuration:

Table Location:
Gives the URL or filename of the table to plot for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a location string from other columns.

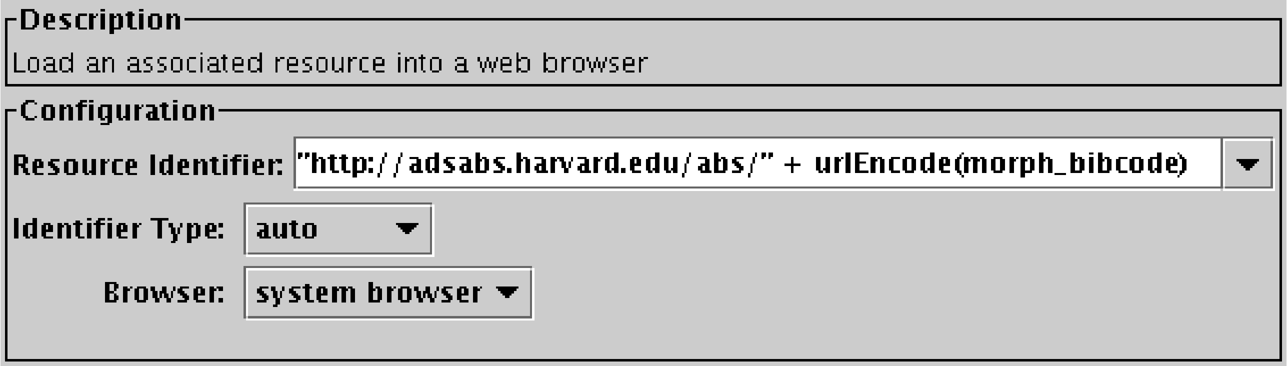
Plot Type:
Selects which type of plot window will be displayed. The options are as listed in Appendix A.4.

Table Format:
Selects the file format of the table to load. If the table is in one of the formats that TOPCAT is able to auto-detect (VOTable, FITS, ECSV, CDF, PDS4, Feather) then the default value `(auto)` may be used, otherwise the correct format must be chosen from the list.

Import Parameters:
If this option is set, then all the values in the activated row of the activated table will be available in the plotted table as *table parameters*. That means they can accessed from the expression language (e.g. as coordinate expressions) using the param$*name* syntax.

Allow Preprocessing:
If set, the "`<syscmd`" or "`|syscmd`" syntax described in Section 4.2 may be used to preprocess the input stream before the table is loaded. This option is usually not required, and has security implications; if it is switched on in an action loaded from a session file, the security panel will be displayed.

A.10.1.13 View in Web Browser



Description

Load an associated resource into a web browser

Configuration

Resource Identifier: "http://adsabs.harvard.edu/abs/" + urlEncode(morph_bibcode)

Identifier Type: auto

Browser: system browser

Configuration for View in Web Browser action

The **View in Web Browser** action uses some kind of location indicator from the activated row and causes it to be displayed in a web browser. If the web page is HTML it should get displayed directly, other content types may get sent to other desktop applications depending on the capabilities and configuration of the browser in use.

In the most straightforward case, the location is a URL supplied by one of the table columns. However TOPCAT knows how to turn various other types of value such as DOIs and filenames into web page addresses, depending on the **Identifier Type** that's chosen. If none of the identifier types listed below is suitable for you (for instance if you want to use NED with some non-standard options), you can construct your own URL based on column values using the expression language.

Configuration:

Resource Identifier
: Gives the URL, filename or other identifier for the resource to display for each row. A column name may be selected from the drop-down list, or the expression language may be used to assemble a string from other columns. How this string is interpreted depends on the **Identifier Type** setting.

Identifier Type
: Determines how the value from the **Resource Identifier** is interpreted. In most cases you can leave it to the default **auto** setting, but for some specific identifier types (ones that cannot be recognised just by looking at them) you may need to make a more specific choice. The options are:

  auto:
  : Looks at the string, and tries to interpret it as one of the other types listed here, depending on its form.

  URL:
  : The string is interpreted directly as a URL.

  Filename:
  : The string is interpreted as a pathname of a file on the local filesystem. If the pathname is not absolute (e.g. it doesn't start with a "/" on Un*x) it is taken to be relative to the directory in which the application was started.

  Bibcode:
  : The string is interpreted as a Bibcode, and turned into an ADS URL, e.g. "`2018A&A...616A...2L`" turns into https://ui.adsabs.harvard.edu/abs/2018A%26A...616A...2L. See the `bibcodeUrl` function.

  DOI:
  : The string is interpreted as a DOI (Digital Object Identifier), and turned into a DOI display URL, e.g. "`10.3390/informatics4030018`" turns into https://doi.org/10.3390/informatics4030018. See the `doiUrl` function.

arXiv:
The string is interpreted as an arXiv identifier and turned into a URL for the relevant paper, e.g. "`arXiv:1804.09379`" turns into https://arxiv.org/abs/1804.09381. See the `arxivUrl` function.

SIMBAD:
The string is assumed to be a source identfier, and the corresponding SIMBAD URL is used, e.g. "`Beta Pictoris`" turns into http://simbad.u-strasbg.fr/simbad/sim-id?Ident=Beta%20Pictoris. See the `simbadUrl` function.

NED:
The string is assumed to be a source identifier, and the corresponding NED URL is used, e.g. "`NGC 3952`" turns into http://ned.ipac.caltech.edu/byname?objname=NGC%203952. See the `nedUrl` function.

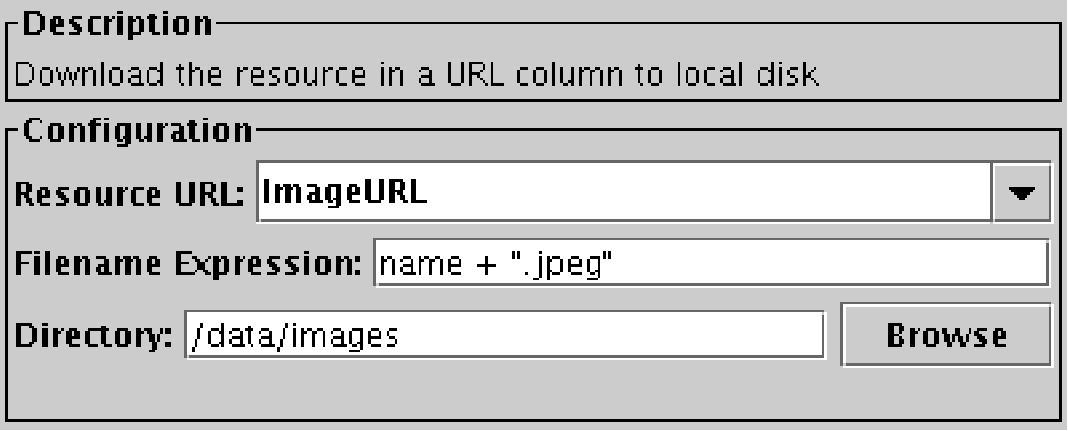
Browser
Determines where the web resource will be displayed; see also Appendix A.10.2.3.

JavaFX Browser:
Displays the URL content in a basic HTML viewer implemented within TOPCAT using *JavaFX* classes. The same window is re-used every time the action is invoked. This generally does a better job than the **Basic Browser**, including being able to render JavaScript as well as static HTML. However, this option may or may not be available, depending on your java installation.

Basic Browser:
Displays the URL content in a basic HTML viewer implemented within TOPCAT using *Swing* classes. The same window is re-used every time the action is invoked. This is OK for basic HTML and images, but more complex or non-web-page content may not work, and rendering may not be very beautiful. This option is always present.

System Browser:
Attempts to open the URL in the default desktop web browser. There is, unfortunately, no control over whether the new web page will appear in the same tab/window or if the browser will open a new one each time, it's down to the browser itself. This option may or may not be available, depending on your java installation.

A.10.1.14 Download URL



Description

Download the resource in a URL column to local disk

Configuration

Resource URL: ImageURL

Filename Expression: name + ".jpeg"

Directory: /data/images

Configuration for Download URL action

The **Download URL** action uses a URL column or expression defined in the table and downloads its contents to a local file.

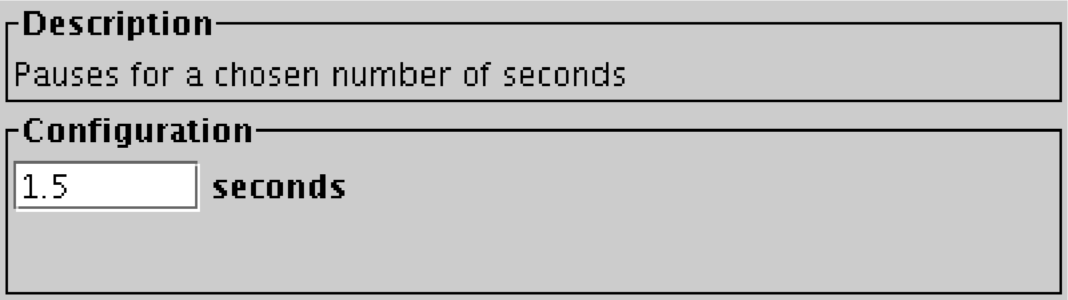
Configuration:

Resource URL
Gives the URL of the remote resource to download. A column name may be selected from the drop-down list, or the expression language may be used to assemble a URL string from other columns.

Filename Expression
An expression for the name of the local file to which the resource should be saved. This is an expression based on table columns in the usual way; it might just be a column name, but is more likely to be a string expression based on columns in the table. For instance in the above example, the expression `name + ".jpeg"` (or equivalently `concat(name, ".jpeg")`) takes the content of the `name` column in the table, and appends the extension ".jpeg" to make a filename appropriate for the given row.

Directory
This selector is a convenience to let you choose the local directory into which files will be downloaded. You can type in a pathname, or use the **Browse** button to select a directory interactively. If this field is left blank, then the Filename expression supplied above will be used on its own to give an absolute or relative (to the current application directory) path.

A.10.1.15 Delay



Description

Pauses for a chosen number of seconds

Configuration

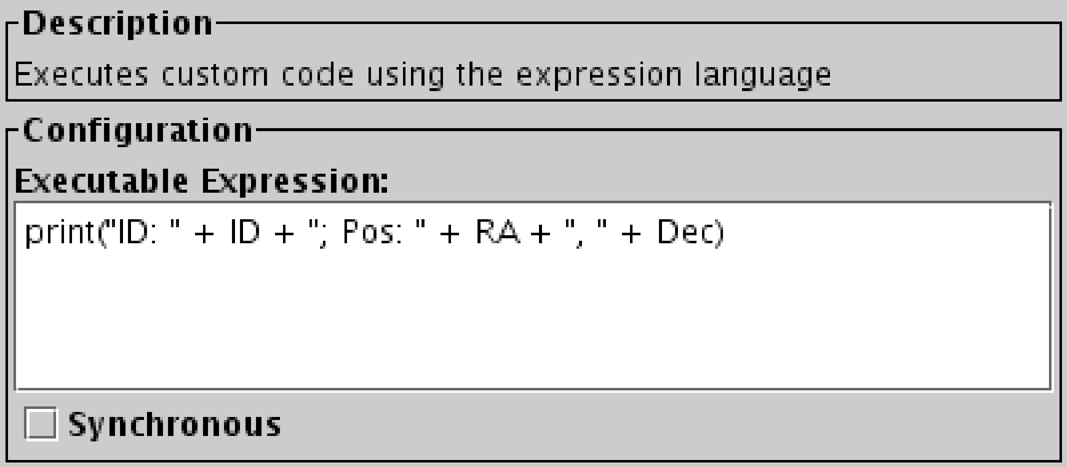
1.5 seconds

Configuration for Delay action

The **Delay** action just waits for the specified number of seconds before proceeding. Any whole or fractional number of seconds can be specified. It's not much use on its own, but it can be useful in combination with other actions when preparing a "slide show" with the **Activate All Rows** () action.

This action is a convenience; a similar effect can be achieved using, for instance, the `sleep` function in the Execute Code action.

A.10.1.16 Execute Code



Description

Executes custom code using the expression language

Configuration

Executable Expression:

```
print("ID: " + ID + "; Pos: " + RA + ", " + Dec)
```

□ Synchronous

Configuration for Execute Code action

The **Execute Code** action can execute arbitrary code using the expression language described in Section 7. Most of the expressions and functions just evaluate an expression, and in that case the only effect is that the result of the expression is displayed in the **Results** panel at the bottom of the activation window every time a row is activated. However, some functions are defined especially for use in this window, and have some additional effect such as popping up a viewer window, writing output to the console, or downloading a file. These are listed below.

In principle, this window can provide very flexible options for defining your own activation actions. In practice, the available activation functions don't give you much more sophisticated options than what is available in the other activation types. However, some more flexible options may be added in the future, and it is also possible to implement your own as explained in Section 7.10. Note that another option for configuring your own custom actions is to use the Run System Command to invoke a shell script or similar with supplied arguments.

Configuration:

Executable Expression
Type in the expression to be evaluated on row activation. Column names act as variables which evaluate to the column value at the activation row, as explained in Section 7. The result of the expression is displayed in the Activation Window **Results** panel, but there may be other effects as well depending on the functions invoked. The expression is parsed as it is typed in, and if the current text does not represent a legal expression, the action is disabled, and an error message is displayed in the **Status** panel that should help to understand what's wrong.

Synchronous
This checkbox determines whether the expression is evaluated on the Java Event Dispatch Thread or not. You can usually leave it alone, but if the expression is known to be fast to evaluate, multiple evaluations may behave a bit more predictably with it checked. For slow expressions, it should be unchecked (its default state).

The available *Activation Functions* (ones that have some effect other than just returning a value) are summarised below, and listed in detail in Appendix B.2. You can also browse them interactively in the **Activation Functions** branch of the Available Functions Window. Note some of these offer rather out of date functionality, and may be withdrawn (replaced by better alternatives) in future releases.

BasicImageDisplay
Functions for display of graphics-format images in a no-frills viewing window (an `ImageWindow`).

Browsers
Displays URLs in web browsers.

Image
Functions for display of images in a window.

Mgc
Specialist functions for use with data from the the Millennium Galaxy Survey.

Output
Functions for writing text to standard output.

Sdss
Specialist display functions for use with the Sloan Digital Sky Server.

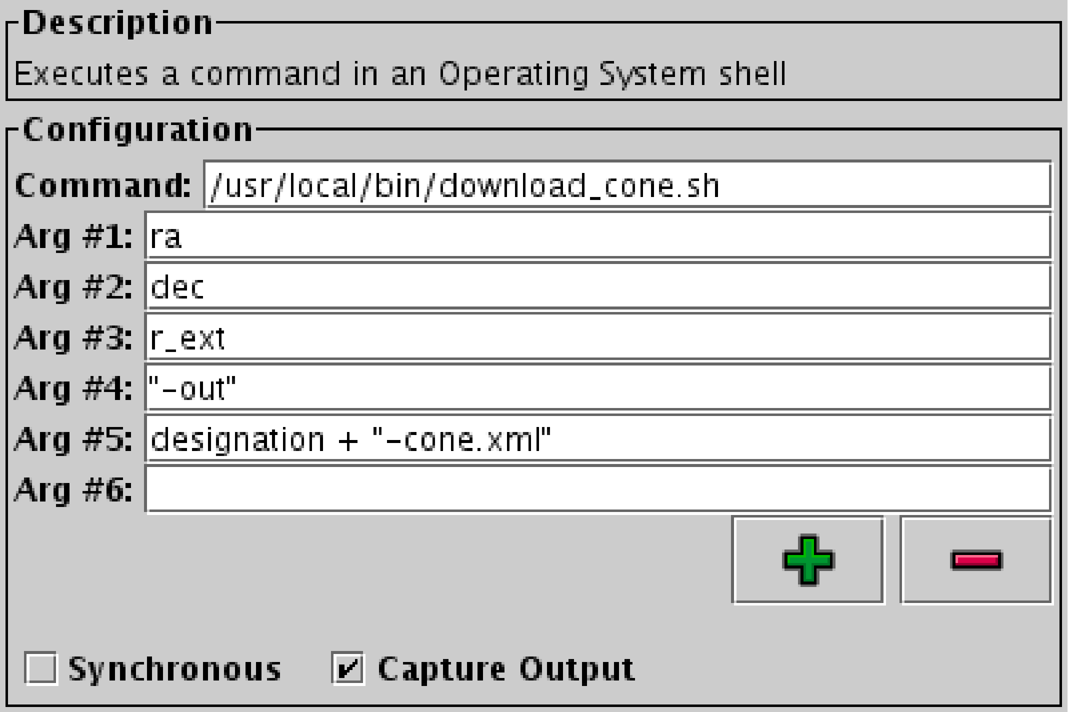
SuperCosmos
Specialist display functions for use with the SuperCOSMOS survey.

System
Functions for executing shell commands on the local operating system and other system-level operations.

TwoQZ
Specialist functions for use with data from the the 2QZ survey.

Invoking certain activation functions configured by an unknown party has security implications. If this action is loaded from a session file, the security panel will be shown.

A.10.1.17 Run System Command


Description

Executes a command in an Operating System shell

Configuration

Command: /usr/local/bin/download_cone.sh

Arg #1: ra

Arg #2: dec

Arg #3: r_ext

Arg #4: "-out"

Arg #5: designation + "-cone.xml"

Arg #6:

□ Synchronous ☑ Capture Output

Configuration for Run System Command action

The **Run System Command** action allows you to invoke an operating system command, for instance running a shell script, when a row is activated, with command parameters taken from table cell values as required.

This functionality has been tested on Unix-like operating systems; it may or may not work well on other OSs (user reports welcome!).

Configuration:

Command
The first word of the system command; this is generally the name of the command to execute. This argument is implicitly quoted.

Arg #N
A list of zero or more command-line arguments to the specified command. Each item in this list is presented to the OS as a separate word, so for instance embedded spaces within one of these lines do not indicate multiple arguments. **Note** that each of these arguments is an expression in the expression language described in Section 7, not a literal value. So if you want (e.g.) the *value* of a given column to appear as an argument, you should enter the column's name. If you want to supply a fixed string, rather than the result of evaluating an expression, then surround the string in double quotes ("). The expressions are parsed as they are typed in, and if the current text of any of the entries does not represent a legal expression, the action is disabled, and an error message is displayed in the Status panel that should help to understand what's wrong.

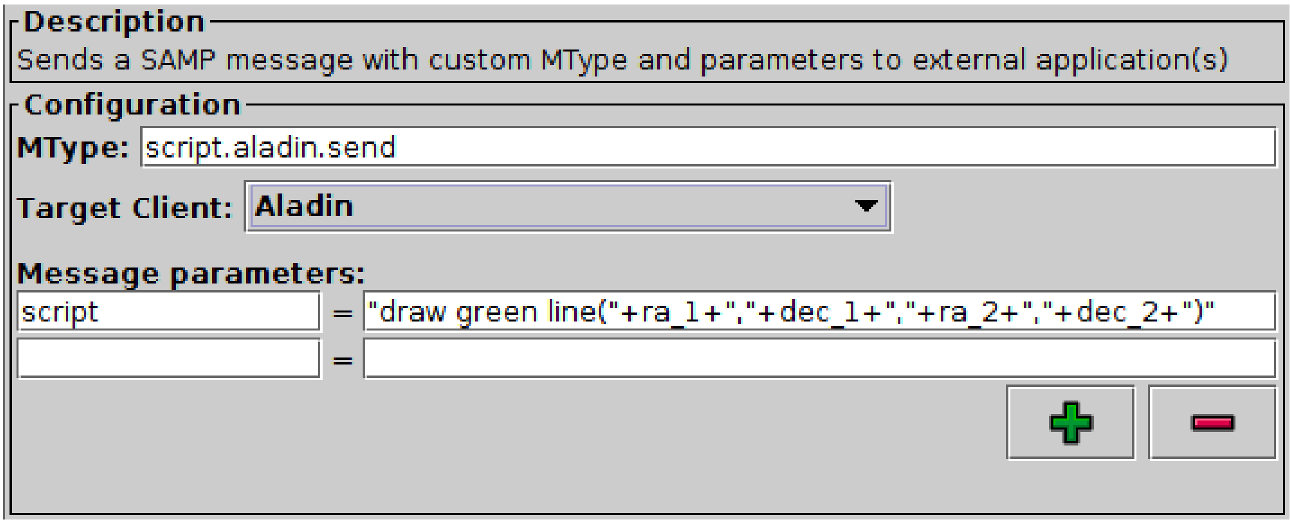
Add/Remove Argument Field (➕/➖)
Use these buttons to add or remove entries to the Arg #N argument list. Empty entries at the end of the list are ignored, so you don't have to remove trailing ones.

Synchronous
This checkbox determines whether the expression is evaluated on the Java Event Dispatch Thread or not. You can usually leave it alone, but if the command is known to be fast to execute, multiple invocations may behave a bit more predictably with it checked. For slow commands, it should be unchecked (its default state).

Capture Output
Determines what happens to the output (standard output and standard error streams) of the executed command. If this checkbox is checked then the output is displayed in the Results panel of the Activation Window, otherwise it goes to the standard output/error stream of the TOPCAT application (and may appear on the console).

Executing system commands configured by an unknown party has obvious security implications. If this action is loaded from a session file, the security panel will be shown.

A.10.1.18 SAMP Message



Description
Sends a SAMP message with custom MType and parameters to external application(s)

Configuration
MType: script.aladin.send
Target Client: Aladin
Message parameters:
script = "draw green line("+ra_1+","+dec_1+","+ra_2+","+dec_2+")"

Configuration for SAMP Message action

The **SAMP Message** action allows you to send a custom SAMP message to other appliacations on the desktop. In most cases it is more straightforward to use one of the standard **Send...** actions like Send Sky Coordinates, Send FITS Image or Send VOTable, but if you want to send a SAMP message that is not covered by those options, you can use this action instead to control exactly what goes in the message.

To use this action, you need to supply the **MType** string and (if applicable) name-value pairs for the message parameters. You will of course need to know the MType-specific parameter syntax for the message that you want to send.

Configuration:

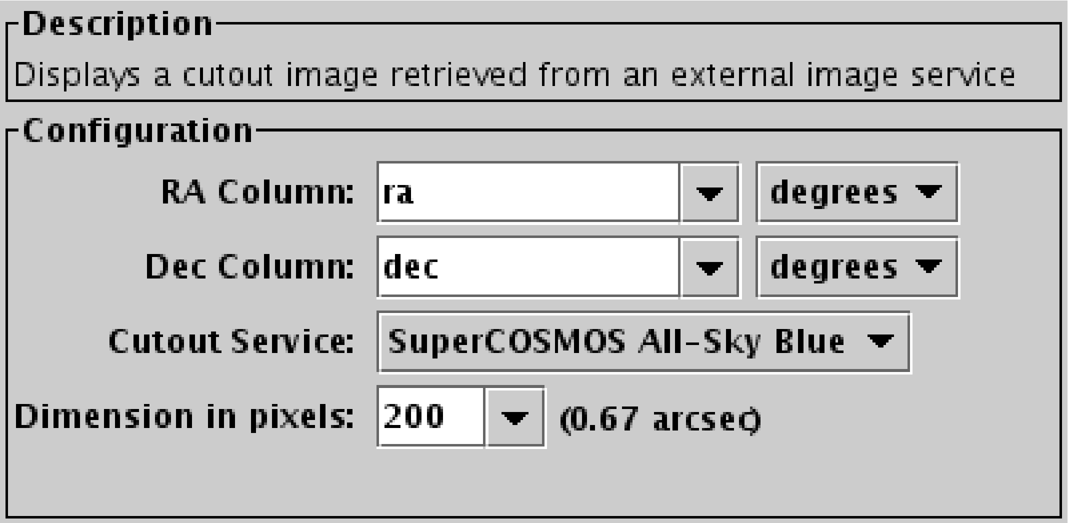
MType
The MType string identifying the message type to send.

Target Client
Determines which external client(s) to send the messages to. Only those which declare support for the chosen MType are listed. The default is "All Clients" which will send it to all eligible applications, but you can select just one. Clicking the selector also serves to indicate which clients capable of acting on the supplied MType are currently registered. If there are none, the action will be disabled.

Message Parameters
Each line represents a name=value pair to be sent as a parameter assignment in the message that is sent. The first field gives the name of the parameter, and the second gives its value. Note that the name part is interpreted as a literal string, but the value part is an expression in TOPCAT's expression language, evaluated for each table row as it is activated.

Add/Remove Parameter Field (➕/➖)
Use these buttons to add or remove *name = expression* fields in the parameter list. Empty entries at the end of the list are ignored, so you don't have to remove trailing ones.

A.10.1.19 Display Cutout Image



Description
Displays a cutout image retrieved from an external image service

Configuration
RA Column: ra ▾ degrees ▾
Dec Column: dec ▾ degrees ▾
Cutout Service: SuperCOSMOS All-Sky Blue ▾
Dimension in pixels: 200 ▾ (0.67 arcsec)

Configuration for Display Cutout Image action

This option presents an easy-to-use way of popping up a cutout image from an image server displaying a region of sky around an activated row. The size of cutout can be selected, and a small selection of image services is provided. When you activate the row, the program will attempt to contact the web server which provides these images, retrieve the image, and display it in one of the internal image viewers.

Note: the list of image services provided by this action is very limited. In most cases, you will be better off using the Display HiPS cutout action.

Configuration:

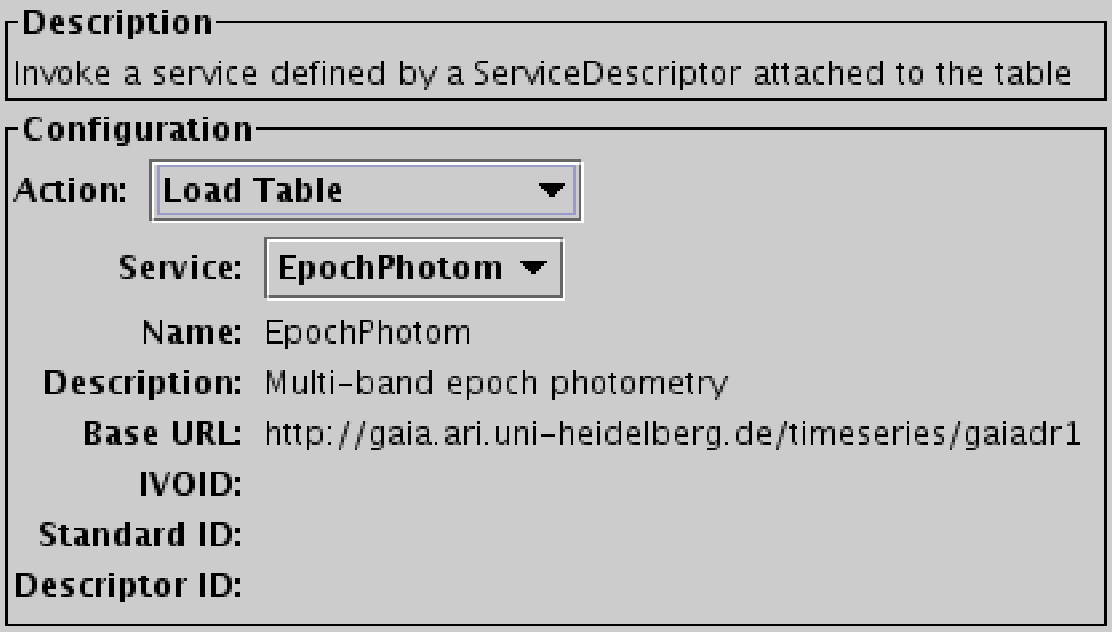
RA Column
Dec Column
Identifies the table columns representing sky coordinates for the central cutout position.

Cutout Service
Select which of the provided cutout services will provide the displayed image. The list is currently:

- SuperCOSMOS All-Sky Blue
- SuperCOSMOS All-Sky Red
- 2MASS Quick-Look J-band
- 2MASS Quick-Look H-band
- 2MASS Quick-Look K-band
- SDSS Colour Images (note does not provide all-sky coverage)

Dimension in pixels
Gives the number of pixels (width and height) of cutout images. The text to the right of the selector indicates the size of a pixel for currently selected service.

A.10.1.20 Invoke Service



Description
Invoke a service defined by a ServiceDescriptor attached to the table

Configuration
Action: Load Table
Service: EpochPhotom
Name: EpochPhotom
Description: Multi-band epoch photometry
Base URL: http://gaia.ari.uni-heidelberg.de/timeseries/gaiadr1
IVOID:
Standard ID:
Descriptor ID:

Configuration for Invoke Service action

The **Invoke Service** action is only appropriate (and only visible by default) for tables in the VOTable (or FITS-plus) format which include **Service Desriptor** elements as defined in the DataLink standard. These service descriptors can provide information on how to invoke an external data service of some kind, which is usually related to the data in the table, and usually can be invoked differently for each row in the table (for instance using one of the table columns as a parameter of the invocation URL). For instance a Service Descriptor may be attached to a table to provide a rule for finding an additional data product (like a time series or preview image) for each row in a table.

This action invokes one of these service descriptors for the activated row.

Configuration:

Action
Determines what to do with the URL that invoking the chosen service on an activated row defines. An attempt is made to come up with a sensible default based on the available information, but you may need to select one manually from the list. Most of the action options have behaviour that is similar to a corresponding activation action, as noted below. The options are:

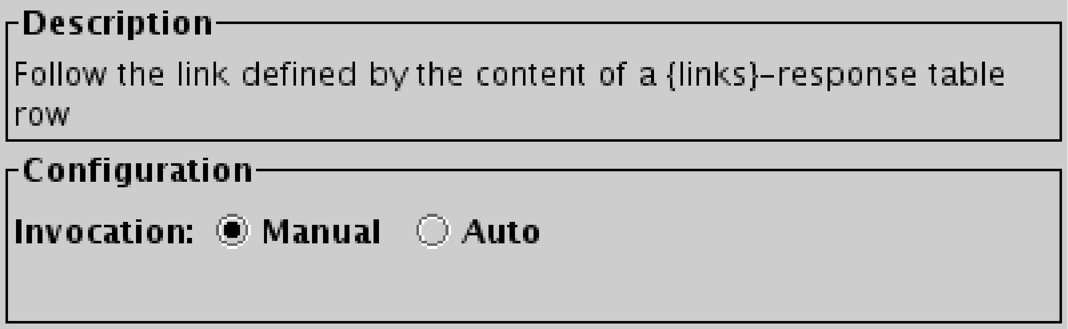
- **Report URL:** displays the URL in the Result field
- **View image internally:** displays the linked resource (an image in a format like PNG, GIF, JPEG, FITS) in an internal image viewer (like Display Image action)
- **Load Table:** loads the first table from the linked resource (a FITS or VOTable table) into this TOPCAT application (like Load Table action)
- **Load Tables:** loads all tables from the linked resource (a FITS or VOTable table) into this TOPCAT application (like Load Table action with Multiple Tables option selected)
- **View DataLink Table:** displays the linked resource (a DataLink table) in a new window like this one (like View Datalink Table action)
- **Send FITS Image:** sends the linked resource (a FITS image) using SAMP to external image viewers (like Send FITS Image action)
- **Send Spectrum:** sends the linked resource (a spectrum) using SAMP to external spectrum viewers (like Send Spectrum action)
- **Send Table:** sends the linked resource (a VOTable) using SAMP to external table viewers (like Send VOTable action)
- **Download URL:** downloads the linked resource to a local file, chosen using a popup filesystem browser
- **Show Web Page:** opens the linked resource in the system web browser (like View in Web Browser action)

Note that there currently is not much opportunity to customise the behaviour of the various Action options supplied by this window; these actions are less configurable than their corresponding Activation Actions.

Service
If the current table has multiple service descriptors, this allows you to select which one to use. In the common case that there is only one service descriptor for a table, this option does not appear. If there are no service descriptors, this action is disabled.

Service Parameters
Some service descriptors have additional user-supplied parameters. In this case, a set of fields will be displayed at the bottom of the configuration panel (defaults may or may not be present) allowing you to supply values. Note that the user interface for supplying these parameters is currently very basic, and may be improved in future releases.

A.10.1.21 Invoke Datalink Row



Description

Follow the link defined by the content of a {links}-response table row

Configuration

Invocation: ● Manual ○ Auto

Configuration for Invoke Datalink Row action

The **Invoke Datalink Row** action is only useful (and only visible by default) for a particular type of table, namely the *{links}-response* table format defined by the DataLink standard. This format is sometimes loosely known as the DataLink format, and is used to represent links of various kinds to external data resources.

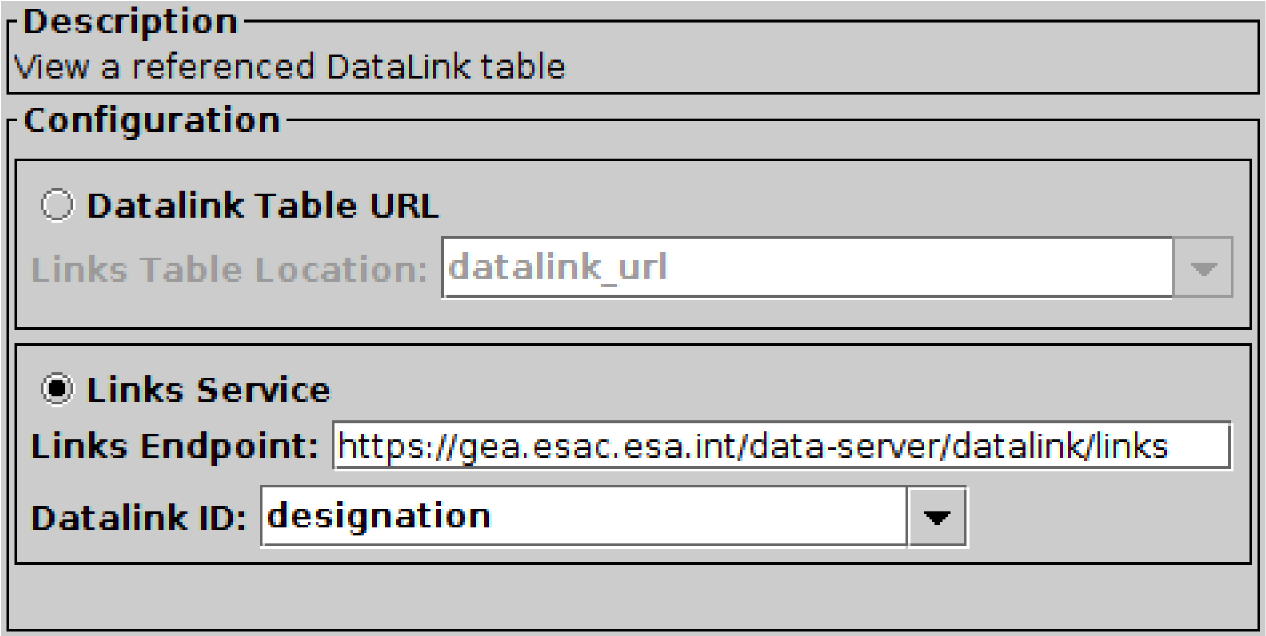
Each row of a DataLink table represents a labelled link to some external resource, and this activation action follows that link in some way when the row is activated, providing options for how it is followed. **Note** however, that if you have a DataLink table loaded and want to invoke its links, the DataLink Window may provide a more straightforward way to do it than this activation action (the functionality is quite similar to what's here, but fewer popup windows are involved).

Because different rows typically have different kinds of links and so can be invoked in different ways, most of the configuration of link-following behaviour is not done in the Configuration panel of the Activation Window, but in a popup window specific to the row when activation takes place. The per-row popup window, if displayed, has the same content as the lower panel of the DataLink Window (Appendix A.3.6). Refer to the documentation of that window for more details.

Configuration:

Auto/Manual
In Manual mode, activating the row pops up the per-row configuration popup window to give you a chance to adjust the options before actually invoking the link. In Auto mode, the link is invoked immediately on row activation, using whatever are the current default options.

A.10.1.22 View Datalink Table



Description

View a referenced DataLink table

Configuration

○ Datalink Table URL

Links Table Location: datalink_url

● Links Service

Links Endpoint: https://gea.esac.esa.int/data-server/datalink/links

Datalink ID: designation

Configuration for View Datalink Table action

The **View Datalink Table** action is for tables whose rows that can reference files in the *{links}-response* table format defined by the DataLink standard. Activating a row with this action defined will retrieve the DataLink file from the location indicated and display it in a new popup window. The display is the same as that of the DataLink Window (Appendix A.3.6); see the documentation of that window for details. There are a couple of differences however:

- In this case, the DataLink table is not loaded into TOPCAT as a new table.
- There is an extra checkbox, **Auto-Invoke**, next to the **Invoke** button. If this is selected, then as soon as the row is activated, the invoke action will be performed automatically on that row's DataLink table without needing any more user intervention. This can work well, and reduces the number of clicks required (it's not necessary to hit Invoke every time), but it may be worth trying it manually on a few rows to make sure the right action is taking place. TOPCAT tries to use the corresponding row of the DataLink table each time a new DataLink table is loaded, based on column values; it usually guesses correctly, but may get confused if the structure of the tables is much different for different rows.

There are two alternative ways to configure this action; you can either specify the **Datalink Table URL** or the **Links Service** parameters. Select the radio button for one or the other and fill in the details as appropriate.

Datalink Table URL:

This option is suitable if you have the complete URL of a DataLink table supplied by one of the columns in the activated table.

Links Table Location

Gives the URL or filename of the DataLink table for each row. This is normally one of the table columns and can be selected from the drop-down list, but alternatively the expression language may be used to assemble a location string from other columns.

Links Service:

This option is suitable if you know where to find a DataLink "{links}" service, as defined by the DataLink standard, and have suitable ID values supplied in the activated table.

Links Endpoint

The "{links} endpoint" as defined by e.g. Section 2 of the DataLink 1.0 standard. This is a base URL for the service, and is entered as a fixed string without quotes.

Datalink ID

The (textual or numeric) value for each activated row of the **ID** parameter to the DataLink service defined by the above endpoint. This identifies the item whose links are to be retrieved, and is normally one of the table columns that can be selected from the drop-down list; alternatively the expression language may be used to assemble an ID value from other columns. TOPCAT will try to choose a default value for this field based on table metadata (it looks for columns with UCD "`meta.id`" or variants).

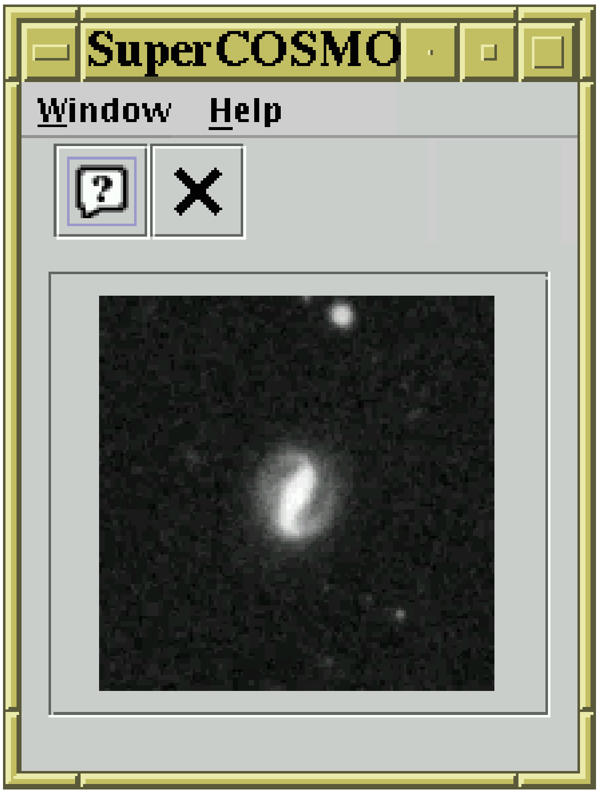
A.10.2 Viewer Windows

Some of the activation actions listed in the previous sections result in popping up windows to view related resources in some way. In many cases, the best thing is to send data to external applications that are optimised for working with different (non-table) data types, but for simple cases TOPCAT provides internal viewers. They are listed in the following subsections.

A.10.2.1 Image Viewer Applications

If you choose the Display Image or Display Cutout Image activation actions, or the **View image internally** option in one of the DataLink-related actions, TOPCAT will try to display an image in an internal image viewer.

This image viewer is very basic, and looks like this:



Basic Image Viewer

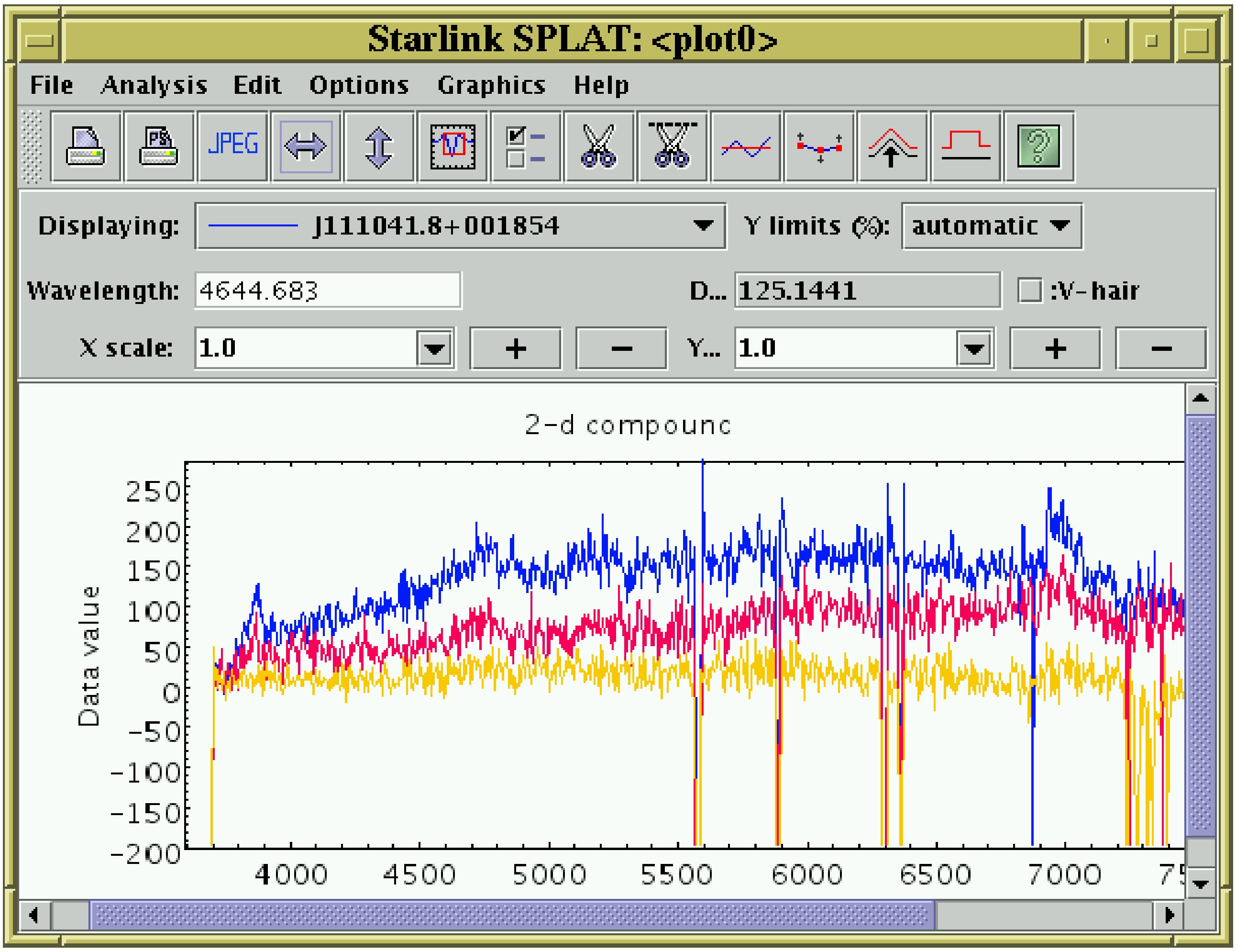
Alternative options are available to communicate with external image viewers.

Note: earlier versions of TOPCAT offered a more capable image viewer called SoG under some circumstances. However, this only worked when the Java Advanced Imaging classes were present at runtime, and these are hard to obtain now, so this option has been withdrawn.

A.10.2.2 Spectrum Viewers

If you try to display a spectrum internally, TOPCAT may be able to pop up a SPLAT sub-window.

SPLAT (http://www.starlink.ac.uk/splat/) is a sophisticated multi-spectrum analysis program. This requires the presence of a component named JNIAST, which may or may not have been installed with TOPCAT (it depends on some non-Java, i.e. platform-specific code). There is currently no fallback spectrum viewer, so if JNIAST is not present, then spectra cannot be displayed internally. An example of SPLAT display of multiple spectra is shown below.



Displaying: J111041.8+001854
Y limits (%): automatic
Wavelength: 4644.683
D... 125.1441
□ :V-hair
X scale: 1.0
Y... 1.0

SPLAT Spectrum Viewer

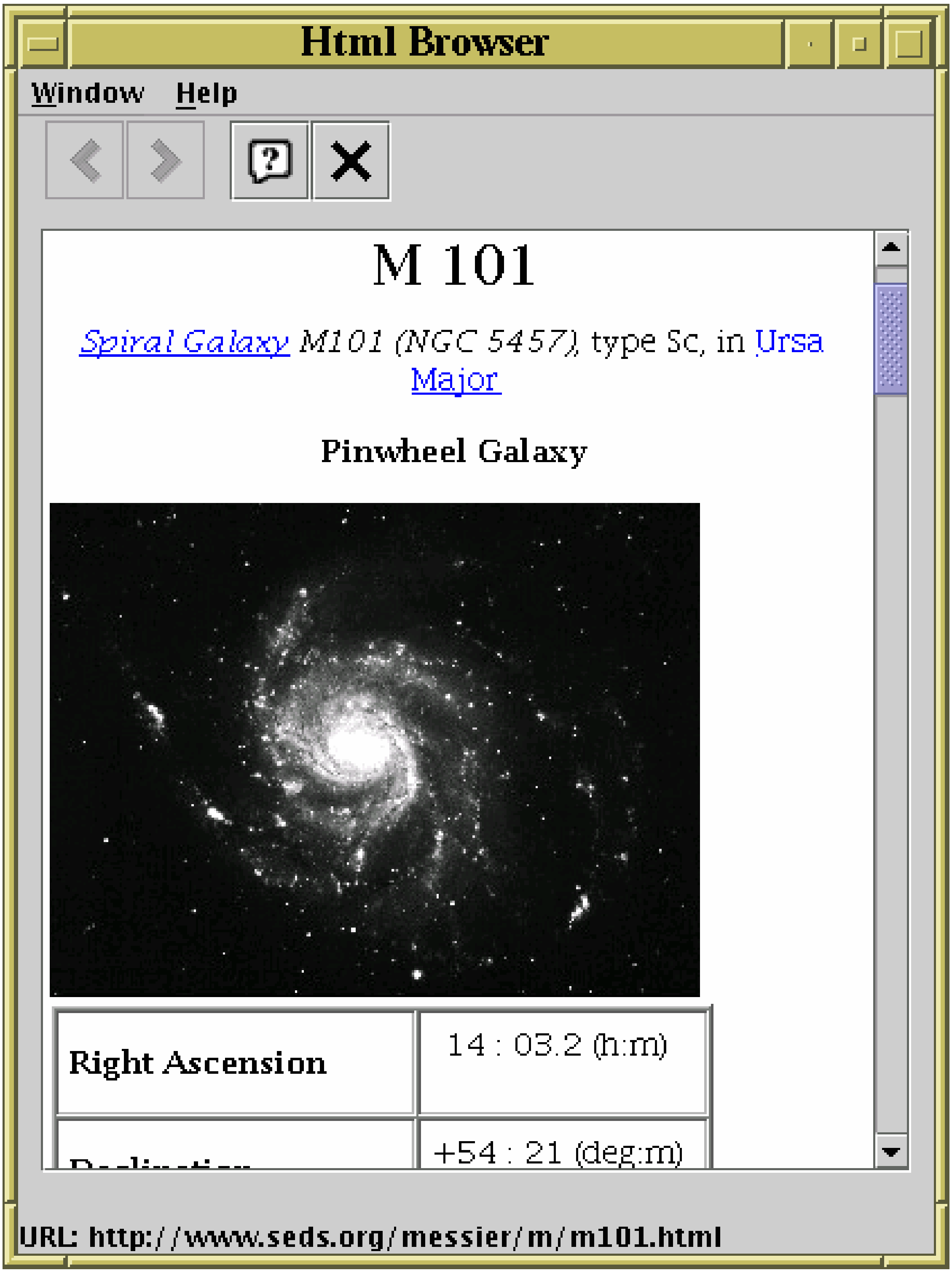
Full documentation for SPLAT is available on-line within the program, or in SUN/243.

In general it is more reliable and configurable to use SAMP to send spectra to an external running instance of SPLAT, via the Send Spectrum action.

A.10.2.3 Web Browsers

If you choose the View in Web Browser action then activating a row will display the web page whose URL is in one of the columns in a web browser. You are given the option of what browser you would like to use in this case; the options available depend on your java installation.

The **Basic Browser** and **JavaFX Browser** options use an internal browser implementation that runs inside the TOPCAT application itself. This can view HTML or plain text pages and has forward and back buttons which work as you'd expect. The JavaFX variant has more capabilities and generally works better, for instance it can render JavaScript as well as static HTML, so it's generally better to select this if it's available, but it's not present in all Java Runtime Environments. The Basic variant is always available. These browsers look approximately like this:



M 101

Spiral Galaxy M101 (NGC 5457), type Sc, in Ursa Major

Pinwheel Galaxy

Right Ascension	14 : 03.2 (h:m)
Declination	+54 : 21 (deg:m)

URL: http://www.seds.org/messier/m/m101.html

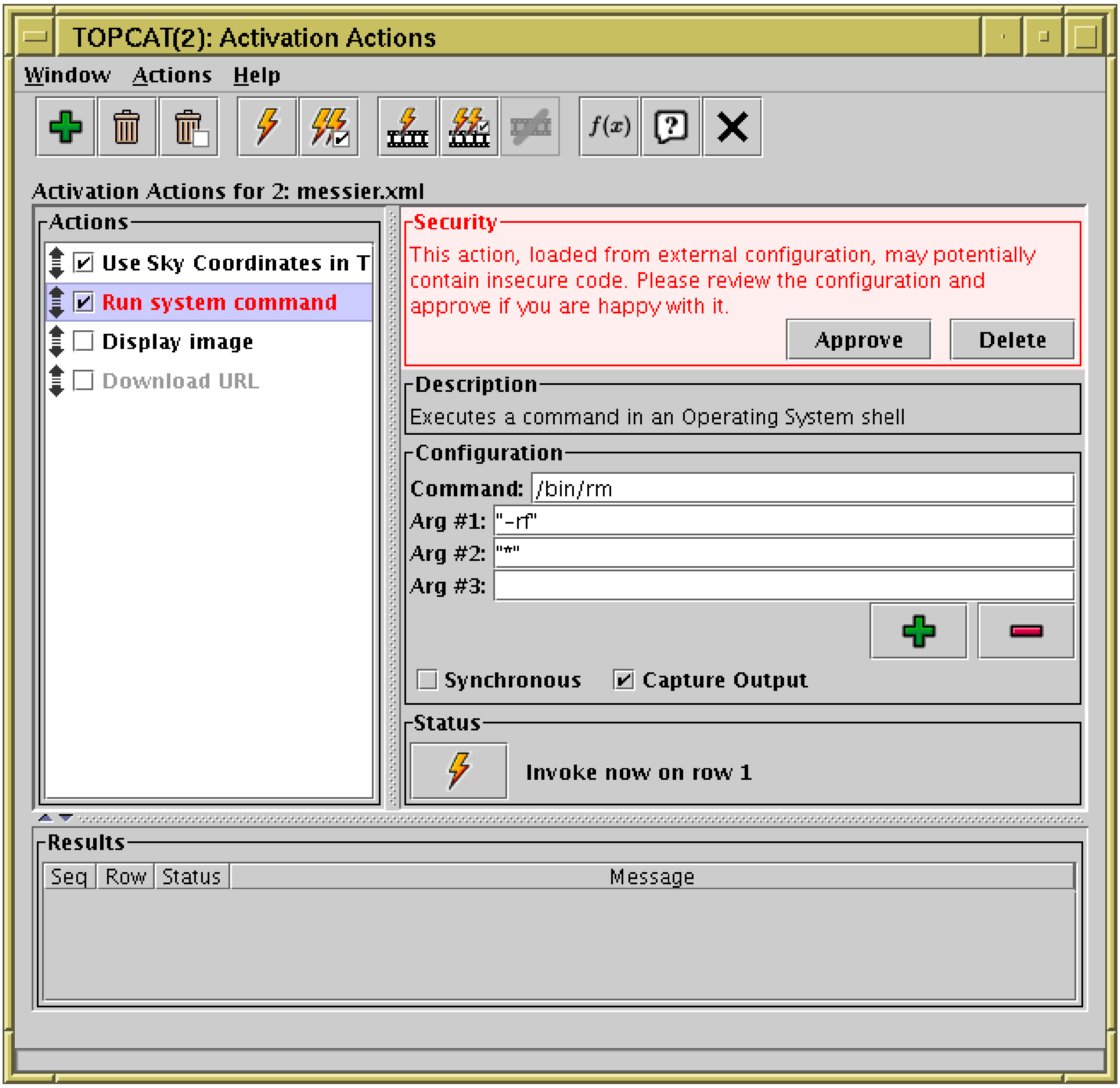
Basic HTML browser

Alternatively, the **System Browser** option may be available. This will send the URL to the default browser installed on your desktop. This will certainly be a more capable browser implementation than the internal options, but there may be annoyances; for instance in some cases it opens a new tab every time a new URL is opened.

A.10.3 Activation Security

Activation actions can be configured to perform a wide variety of user-defined actions, in some cases including actions that could have damaging consequences. For instance the Run System Command action can execute arbitrary system (shell) commands with the permissions of the user running the TOPCAT application. As long as the TOPCAT user is the person who has configured these commands, this is no more dangerous than the same user sitting down at the keyboard to type shell commands.

In the case of Activation Actions that have been loaded from a Session file however, it is possible that the person who saved the session might have configured it to perform malicious actions affecting whoever re-loads it. TOPCAT can't determine whether an action is actually malicious, but it can identify actions which could be configured to do malicious things. In the case that it detects an Activation Action that has been loaded from a Session file that *might* be doing something dangerous (invoking arbitrary system commands, or in general anything that has effects outside of the TOPCAT application itself, excepting some known-harmless SAMP messages) it will ask the user for explicit approval before invoking it.



Security: This action, loaded from external configuration, may potentially contain insecure code. Please review the configuration and approve if you are happy with it.

Description: Executes a command in an Operating System shell

Command: /bin/rm
Arg #1: "-rf"
Arg #2: "*"
Arg #3:

□ Synchronous ☑ Capture Output

Invoke now on row 1

Activation Window configured with security implications

This screenshot shows what happens. When called upon to invoke a potentially dangerous pre-configured action, it displays the Security panel in red, and the user has to click the **Approve** button before the action will be invoked. If you see this, you should, as instructed, check that the action configuration is not doing anything it shouldn't. Once you have approved it, the security panel will disappear and later invocations will take place with no further user interaction.

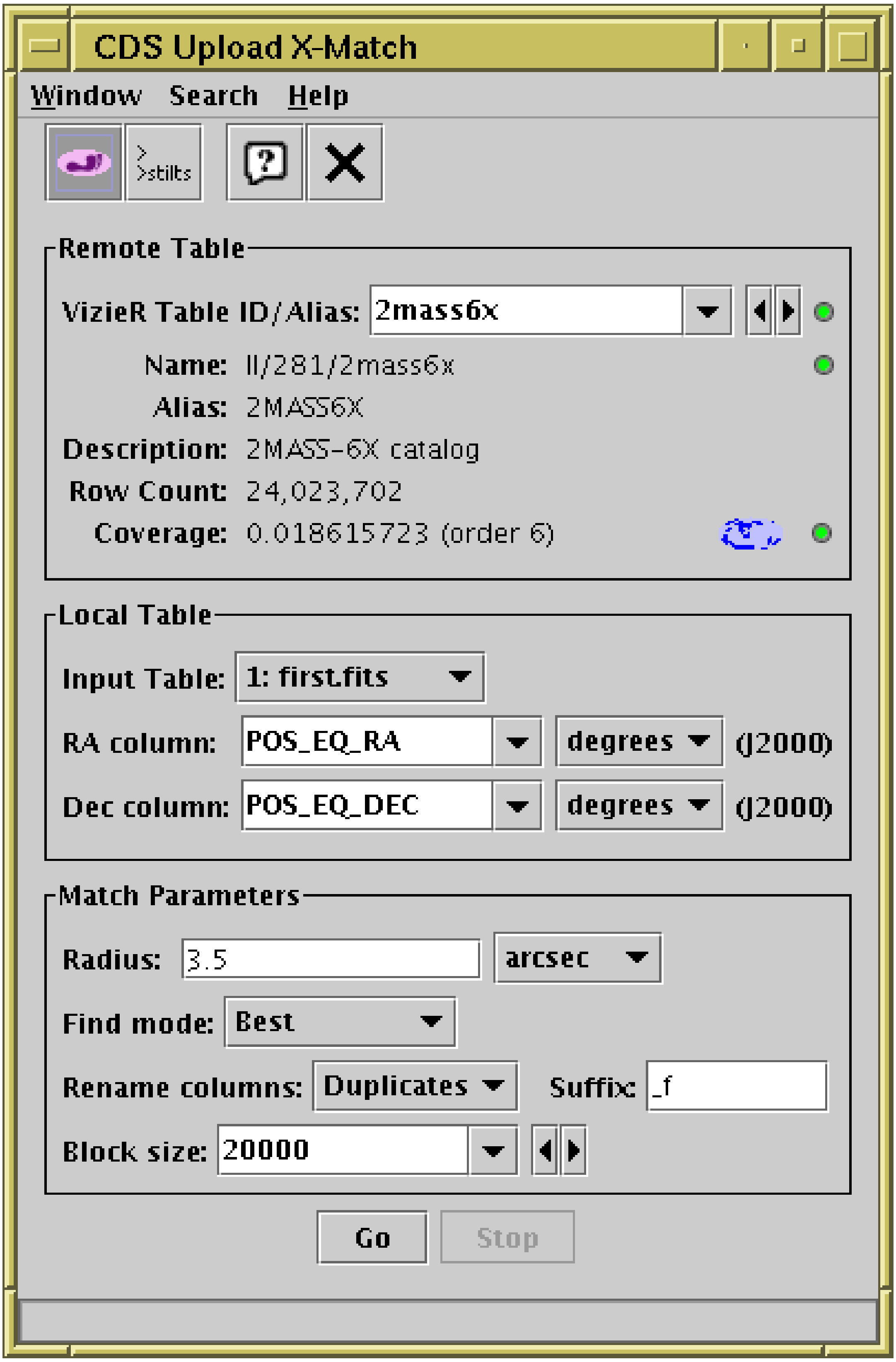
If you know that the session file is harmless, and you don't want to be bothered by these approval messages, you can use the **Approve All Actions** (☑) menu item in the **Actions** menu.

The Run System Command and Execute Code actions can trigger these warnings for fairly obvious reasons. They can however also be triggered under some circumstances by Display Image, Load Table and Plot Table. That's because in some configurations these actions can allow file/URL location values that perform preprocessing of the input streams using the "`<syscmd`" or "`syscmd|`" syntax described in Section 4.2, and that means that referencing image/table locations either entered directly, or acquired from table data, could result in execution of arbitrary system commands. This preprocessing is always switched off by default, and you shouldn't switch it on unless you need it, so it will not apply unless you have turned it on explicitly or loaded it from a session file.

In nearly all cases, you don't need to think about these issues, but if you are loading a session file that may have been prepared by somebody else, TOPCAT will warn you in case anything malicious *might* be capable of affecting you.

A.11 Other Windows

A.11.1 CDS Upload X-Match Window



Remote Table

VizieR Table ID/Alias: 2mass6x

Name: II/281/2mass6x

Alias: 2MASS6X

Description: 2MASS-6X catalog

Row Count: 24,023,702

Coverage: 0.018615723 (order 6)

Local Table

Input Table: 1: first.fits

RA column: POS_EQ_RA degrees (J2000)

Dec column: POS_EQ_DEC degrees (J2000)

Match Parameters

Radius: 3.5 arcsec

Find mode: Best

Rename columns: Duplicates Suffix: _f

Block size: 20000

CDS Upload X-Match Window

The CDS Upload X-Match Window allows you to join a local table with any table provided by the VizieR database of astronomical tables or with SIMBAD. You can access the window from the main Control Window using the **CDS Upload Match** button ( ) in the toolbar, or the **Joins** or **VO** menus. This window is an interface to the excellent CDS X-Match service provided by the Centre de Données astronomiques de Strasbourg (CDS). The service is very fast, and in most cases it is the best way to match a local table against a large external table hosted by a service. In particular, it is almost certainly much better than using the Multi-Cone window, though it's less flexible than TAP.

The local table is uploaded to the X-Match service in blocks, and the matches for each block are retrieved in turn and eventually stitched together to form the final result. The tool only uploads sky position and an identifier for each row of the input table, but all columns of the input table are reinstated in the result for reference. These details are mostly transparent when using the service, though it may help to understand how the progress bar moves.

For a better understanding of the details of how this service operates, including exactly what coordinates are matched against the uploaded positions (roughly: integrated to J2000 using proper motions if available) and what columns are included in the output (roughly: a subset of the most commonly used columns), please consult the service documentation. Note that because of the way the service is implemented not all the columns from the VizieR table may be present in the result; if the columns you need are not present, you're out of luck, there's no way to request additional ones.

The window has three panels, described below.

The **Remote Table** panel allows you to select the CDS table against which to match. The table must be entered by name in the **VizieR Table ID/Alias** field, and may be in one of the following forms:

- "simbad", to indicate the SIMBAD database of known astronomical objects.
- A VizieR table identifier, for instance "II/246/out". *Note* these must name the table not just the catalogue (which may contain several tables), so for instance "II/246" will not work. No help is currently provided in this window to search for these IDs, you have to use one of the services or web pages provide by CDS to locate them. A good tip is the table searching facility at the TAPVizieR service, http://tapvizier.u-strasbg.fr/adql/.
- A VizieR table alias, for instance "2mass". A few tens of these are defined for the largest and most-used tables in VizieR, and these can be selected by clicking on the selection box.

When a table name has been entered, additional information (Name, Alias, Description and Row Count) will be downloaded from CDS and displayed in the lower part of the Remote Table panel. If nothing shows up here, then the table name is not legal for the X-Match service, and the **Go** button will be disabled.

Sky coverage information is also displayed, including the proportion of the sky covered by the table and a graphical indication of the covered regions on the sky.

Service interaction is monitored by the little "lights" to the right of this panel. If things are operating correctly they should be, or quickly turn, green. If they are amber or red, the service may be slow or not responding. Hovering over them with the mouse will give more information (apologies to red-green colourblind users).

The **Local Table** panel allows you to indicate the local table (the one loaded into TOPCAT) that you want to match. You must select the required table and indicate its Right Ascension and Declination columns.

The **Match Parameters** panel supplies the other information about how the match will operate:

Radius
The maximum distance between a local table and remote table position to count as a match. There is a maximum enforced on this value by the CDS X-Match service, currently 2 arcminutes.

Find mode

Determines in what form the result is generated and used. The following options are available:

Best
Load a new table with one row for each local row that matches a remote row, giving the closest match. Unmatched local rows are not included.

All
Load a new table with one row for each match between a local row and a remote row. The match is symmetric between local and remote tables.

Each
Load a new table with the same number of rows as the local table, in the same order. The best remote match, if any, appears alongside the local row, but if there is no match the remote columns are blank.

Best Remote
Load a new table with one row for each remote table row that matches the local table. *Note* there is currently a bug in this mode; if the match is done in N blocks then a remote row may appear up to N times rather than just one. This may be fixed in a future release, but to avoid it you can make sure that the **Block size** is greater than the row count.

Add Subset
No new table is loaded, but a new Row Subset is added to the local table indicating which rows had at least one match to the remote table.

Rename columns
If columns from two input tables are combined, it's possible that some may share the same name. Multiple columns in a table with the same name can cause confusion, so this selector allows you to determine whether and how such name clashes are dealt with. Columns from the remote table can be renamed by appending a specified suffix, either always or only in case of duplicate names.

Block size
The number of rows uploaded to the X-Match service at a time. This should not affect the result, though it may affect performance.

Large blocksizes tend to be good (up to a point) for reducing the total amount of time a large xmatch operation takes, but they can make it harder to see the job progressing. There is also the danger (for ALL-type find modes) of exceeding the return size limit, which will result in truncation of the returned result. At time of writing, the upload limit is 100Mbyte (about 3Mrow), and the maximum return size is 2Mrow.

When all the fields have been filled in, hit the **Go** button and the match will start; progress will be logged in the progress bar at the bottom of the window. To stop the match before it has completed, hit the **Stop** button. Note that the **Go** button is only enabled when a legal table name has been entered at the top.

The remote table in most cases contains only a subset of the the columns in the relevant VizieR table, including the most useful ones. The service currently provides no straightforward way to acquire columns which are not returned by default.

The following item is available in the toolbar and **Search** menu:

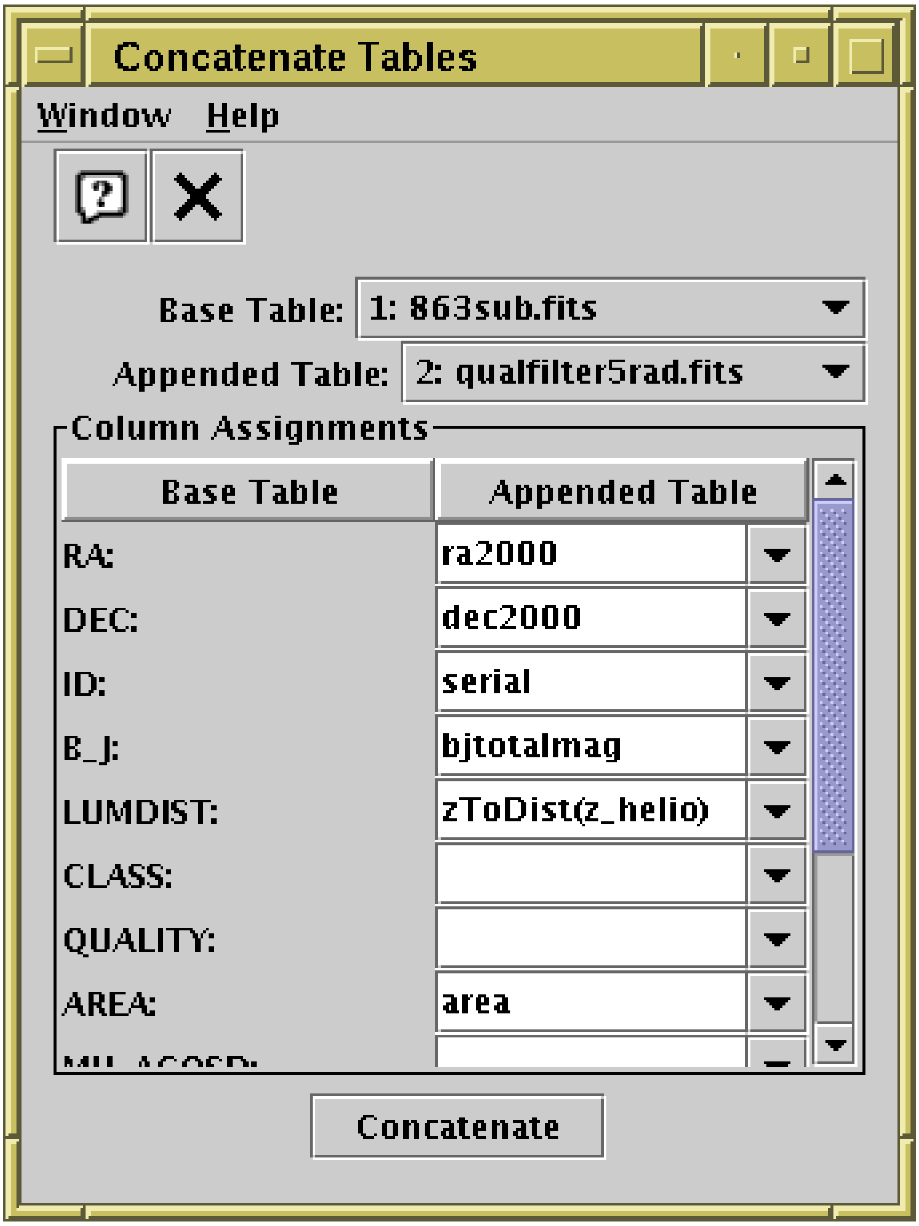
Use Service Coverage

If this option is selected, then the sky coverage of the service is used to pre-filter local table rows. In this case those rows that fall outside the coverage area of the remote table are not uploaded for matching, since it is known that they will not match. This option should therefore not affect the result, but improve performance, and as such should generally be turned on (the default).

Acknowledgement: CDS note that if the use of the X-Match service is useful to your research, they would appreciate the following acknowledgement:

> *"This research made use of the cross-match service provided by CDS, Strasbourg."*

A.11.2 Concatenation Window



Base Table: 1: 863sub.fits

Appended Table: 2: qualfilter5rad.fits

Column Assignments

Base Table	Appended Table
RA:	ra2000
DEC:	dec2000
ID:	serial
B_J:	bjtotalmag
LUMDIST:	zToDist(z_helio)
CLASS:	
QUALITY:	
AREA:	area

Concatenation Window

The Concatenation Window allows you to join two tables together top-to-bottom. It can be obtained using the **Concatenate Tables** item ( ) in the Control Window **Joins** menu.

When two windows are concatenated all the rows of the first ("base") table are followed by all the rows of the second ("appended") table. The result is a new table which has a number of rows equal to the sum of the two it has been made from. The columns in the resulting table are the same as those of the base table. To perform the concatenation, you have to specify which columns from the appended table correspond to which ones in the base table. Of course, this sort of operation only makes sense if at least some of the columns in both tables have the same meaning. This process is discussed in more detail in Section 5.1.

The concatenation window allows you to select the base and appended tables, and for each column in the base table to specify what quantity in the appended table corresponds to it. You can either select a column from the appended table from the selection box, or type in an expression in the expression language referring to columns from the appended table. If you leave the field blank, the column in question will have all null entries in the resulting table. Only suitable columns are available for choosing from these column selectors, that is ones matching the data type of the base column.

In some cases these column selectors may have a value filled in automatically if the program thinks it can guess appropriate ones, but you should ensure that it has guessed correctly in this case. If the two tables have essentially the same structure, the corresponding columns should all get filled in automatically.

When you have filled in the fields to your satisfaction, hit the **Concatenate** button at the bottom of the window, and a new table will be created and added to the table list in the Control Window (a popup window will inform you this has happened).

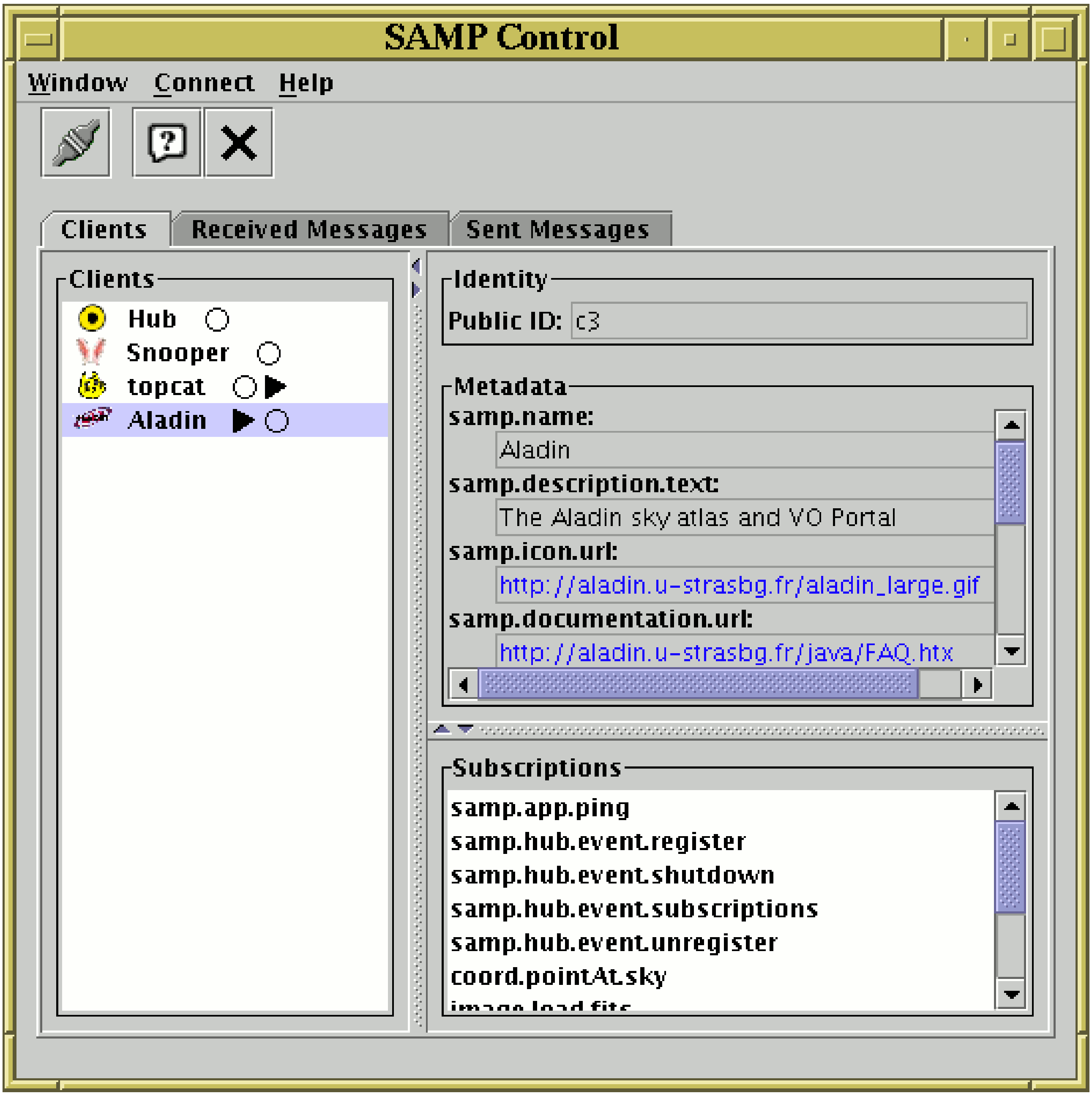
The result is created from the Apparent (Section 3) versions of the base and appended tables, so that any row subsets, hidden columns, or sorts currently in force will be reflected in the output.

A.11.3 SAMP Window

The SAMP Window displays the status of SAMP messaging and allows some control over its configuration. You can display it using the **SAMP Status** item () in the Control Window **Interop** menu or toolbar. SAMP is a messaging protocol which allows TOPCAT to exchange information with other desktop applications - see Section 9 for details.

The main part of the window is a tabbed panel which displays the status of the SAMP connection in detail.

The first tab is labelled **Clients**:



Clients

- Hub ○
- Snooper ○
- topcat ○ ▶
- Aladin ▶ ○

Identity

Public ID: c3

Metadata

samp.name:
Aladin

samp.description.text:
The Aladin sky atlas and VO Portal

samp.icon.url:
http://aladin.u-strasbg.fr/aladin_large.gif

samp.documentation.url:
http://aladin.u-strasbg.fr/java/FAQ.htx

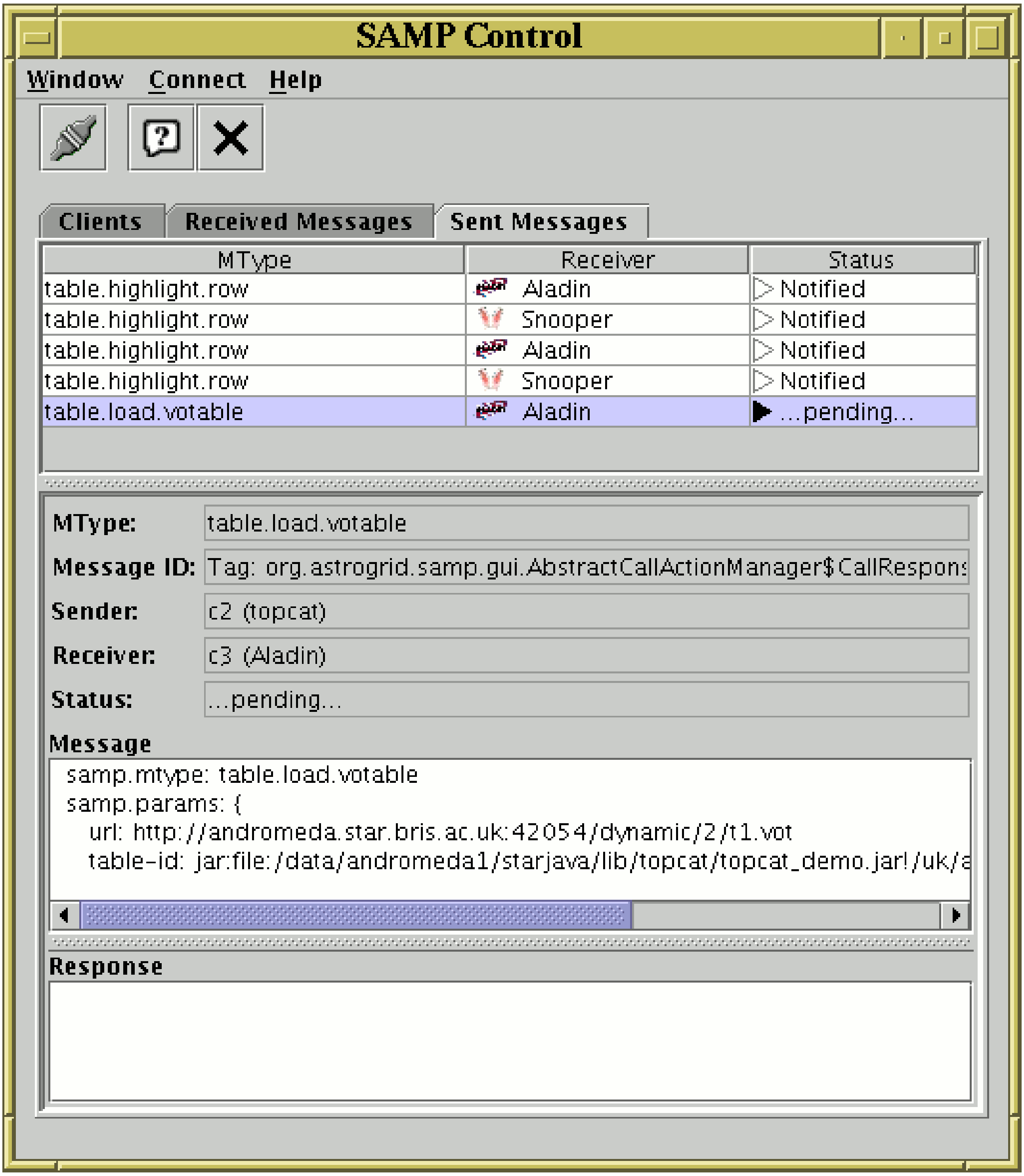
Subscriptions

samp.app.ping
samp.hub.event.register
samp.hub.event.shutdown
samp.hub.event.subscriptions
samp.hub.event.unregister
coord.pointAt.sky
image.load.fits

SAMP Window Clients tab

This displays details of all applications (including TOPCAT itself) which are currently registered with the hub. If you select a client in the list on the left hand side of the panel, you will see its details on the right hand side. This includes its **Metadata** (which may include things like its name, author, documentation URL etc) and its **Subscriptions** which describes what types of messages (MTypes) it will respond to. Also in the list on the left is for each application a graphical indication of messages recently received by/sent from it, in which TOPCAT is the other partner. These are represented by little arrows before/after the little circle following the application name. In the figure, a message is currently in progress from TOPCAT to Aladin. More detail on this is shown in the other tabs.

The other two tabs are labelled **Received Messages** and **Sent Messages**. They look very similar to each other:



MType	Receiver	Status
table.highlight.row	Aladin	▷ Notified
table.highlight.row	Snooper	▷ Notified
table.highlight.row	Aladin	▷ Notified
table.highlight.row	Snooper	▷ Notified
table.load.votable	Aladin	▶ ...pending...

MType: table.load.votable

Message ID: Tag: org.astrogrid.samp.gui.AbstractCallActionManager$CallRespons

Sender: c2 (topcat)

Receiver: c3 (Aladin)

Status: ...pending...

Message

```
samp.mtype: table.load.votable
samp.params: {
  url: http://andromeda.star.bris.ac.uk:42054/dynamic/2/t1.vot
  table-id: jar:file:/data/andromeda1/starjava/lib/topcat/topcat_demo.jar!/uk/a
```

Response

SAMP Window Sent Messages tab

They display details of SAMP messages recently sent from or received by TOPCAT. A table near the top contains one row for each message. Shown in different columns are the message's MType (which describes the message semantics), the application receiving/sending it from/to TOPCAT, and a summary of its status. If you click on the row, all the details of the message, including its content and that of the response if one has been received, are visible in the lower panel. Messages remain in this table for a few seconds after they have completed, but are eventually removed to avoid clutter.

The following toolbar button is available:

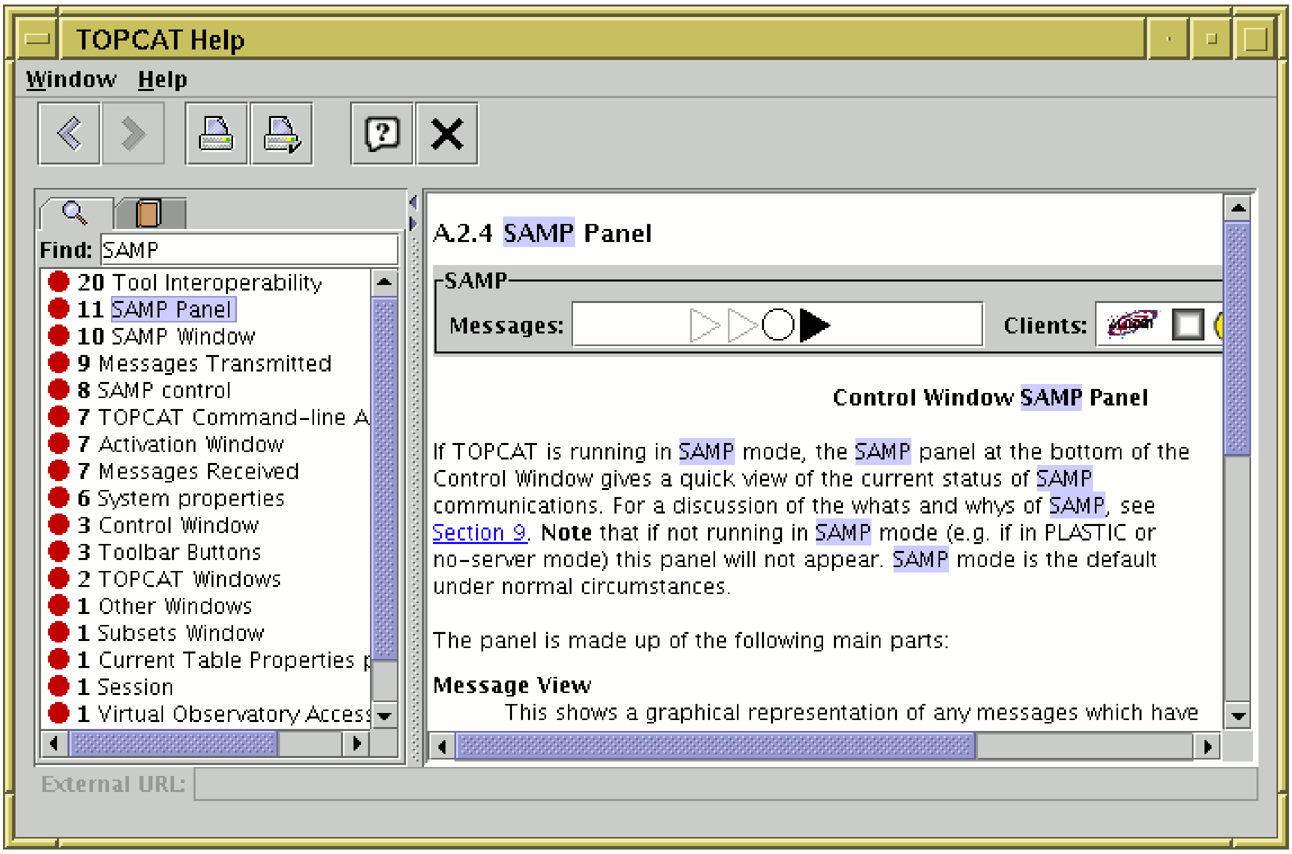
Connect/Disconnect

This button controls connection to the SAMP hub; without a connection no SAMP commmunication can take place. If a hub is running, clicking this button toggles whether TOPCAT is connected or not.

If no hub is running, clicking this button will pop up a dialogue box which allows you to start a hub. You are given the option to start either an **internal** or an **external** hub. An internal one runs in the same Java Virtual Machine as TOPCAT itself, and an external one runs in a separate process. The main important difference is that an internal hub will shut down when TOPCAT does, while an external hub will keep running until it is shut down explicitly. If an external hub is started, a window which shows its status is displayed on the screen; closing this window (which does not belong to TOPCAT) will shut down the hub.

More facilities may be introduced into this window in future releases.

A.11.4 Help Window



Find: SAMP

- 20 Tool Interoperability
- 11 SAMP Panel
- 10 SAMP Window
- 9 Messages Transmitted
- 8 SAMP control
- 7 TOPCAT Command-line A
- 7 Activation Window
- 7 Messages Received
- 6 System properties
- 3 Control Window
- 3 Toolbar Buttons
- 2 TOPCAT Windows
- 1 Other Windows
- 1 Subsets Window
- 1 Current Table Properties p
- 1 Session
- 1 Virtual Observatory Access

A.2.4 SAMP Panel

Control Window SAMP Panel

If TOPCAT is running in SAMP mode, the SAMP panel at the bottom of the Control Window gives a quick view of the current status of SAMP communications. For a discussion of the whats and whys of SAMP, see Section 9. **Note** that if not running in SAMP mode (e.g. if in PLASTIC or no-server mode) this panel will not appear. SAMP mode is the default under normal circumstances.

The panel is made up of the following main parts:

Message View
This shows a graphical representation of any messages which have

Help Window

The help window is a browser for displaying help information on TOPCAT; it can be obtained by clicking the **Help** (?) button that appears in the toolbar of most windows. It views the text contained in this document, so it may be what you are looking at now.

The panel on the right hand side displays the currently selected section of help text itself. The panel on the left gives a couple of ways of selecting this view:

Text Search
Allows you to search for words in the manual. Enter a word or words as search terms, and a list of sections which fully or partially match your search terms will be displayed. You can click on one of these to take you to the section that has been found.

Table of Contents

A hierarchical view of the available help topics. This is the table of contents of the manual; clicking on an entry will take you to the relevant section.

The bar in between the two panels can be dragged with the mouse to affect the relative sizes of these windows.

The toolbar contains these extra buttons:

Back

Moves backward through the list of topics in the order you have looked at them.

Forward

Moves forward through the list of topics in the order you have looked at them.

Print

Pops up a dialogue to permit printing of the current page to a file or printer (but see below).

Page Setup

Pops up a dialogue to do printer setup.

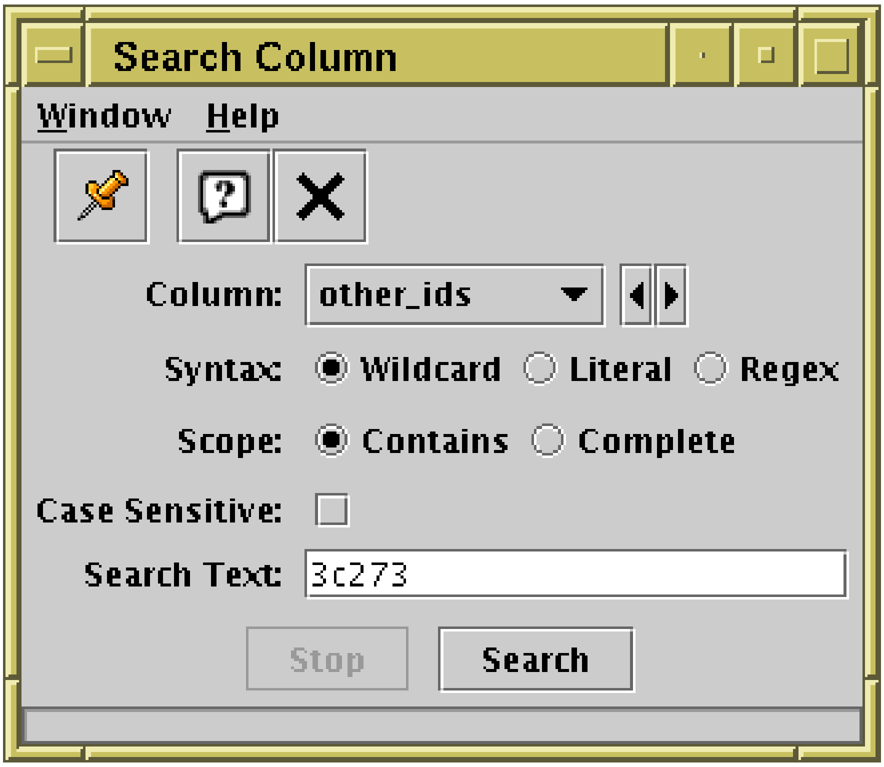
To Browser

Displays the currently displayed help page in your normal WWW browser.

Although the printing buttons work, if you want to print out the whole of this document rather than just a few sections you may be better off printing the PDF version, or printing the single-page HTML version through a web browser. The most recent version of these should be available on the web at http://www.starlink.ac.uk/topcat/sun253/sun253.html and http://www.starlink.ac.uk/topcat/sun253.pdf; you can also find the HTML version in the topcat jar file at `uk/ac/starlink/topcat/help/sun253.html` or, if you have a full TOPCAT installation, in `docs/topcat/sun253/sun253.html` and `docs/topcat/sun253.pdf` (the single-page HTML version is available here in the HTML version).

The help browser is an HTML browser and some of the hyperlinks in the help document point to locations outside of the help document itself. Selecting these links will go to the external documents. When the viewer is displaying an external document, its URL will be displayed in a line at the bottom of the window. You can cut and paste from this using your platform's usual mechanisms for this.

A.11.5 Column Search Window



Column: other_ids

Syntax: ● Wildcard ○ Literal ○ Regex

Scope: ● Contains ○ Complete

Case Sensitive: □

Search Text: 3c273

Column Search window

The Column Search window lets you search through the values in a chosen column of the displayed JTable to find occurrences of some particular text. It is currently available from the Data Window and Columns Window. In each case you can get it by using the **Search Column** (🔍) action in one of two ways: either click the toolbar button or menu item, in which case you have to choose the column of interest using the selector, or select it from the popup menu that appears when you right-click on a column, in which case that column is selected automatically.

To perform a search, ensure that the column selector is filled in with the column you want to search, fill in the matching options, enter the text to search for, and hit the **Search** button. Any rows matching the query are then highlighted in the corresponding JTable, and the display is scrolled so that the first match, if any, is visible. If at least one match is found, the search window usually closes (though this can be prevented with the **Stay Open** (📌) option).

When used with the Data Window only, if exactly one matching row was found, it is also activated. You can see how many matches there were from the **Selected** value indicated at the bottom of the Data Window.

The fields to fill in are as follows:

Column:
The column whose cell values will be matched against. This may be filled in automatically (if you open the window using the column popup menu), or you can select a column by hand.

Syntax:
Controls the way that the search text is interpreted:

- **Wildcard:** basic string matching with two special characters; "*" matches any sequence of characters, and "?" matches any single character
- **Literal:** no special characters; the strings must match exactly
- **Regex:** uses the powerful *regular expression* matching syntax

Scope:
Determines which part of the cell text has to match the search text:

- **Contains:** the search text has to appear somewhere in the cell content
- **Complete:** the search text has to match the whole of the cell content

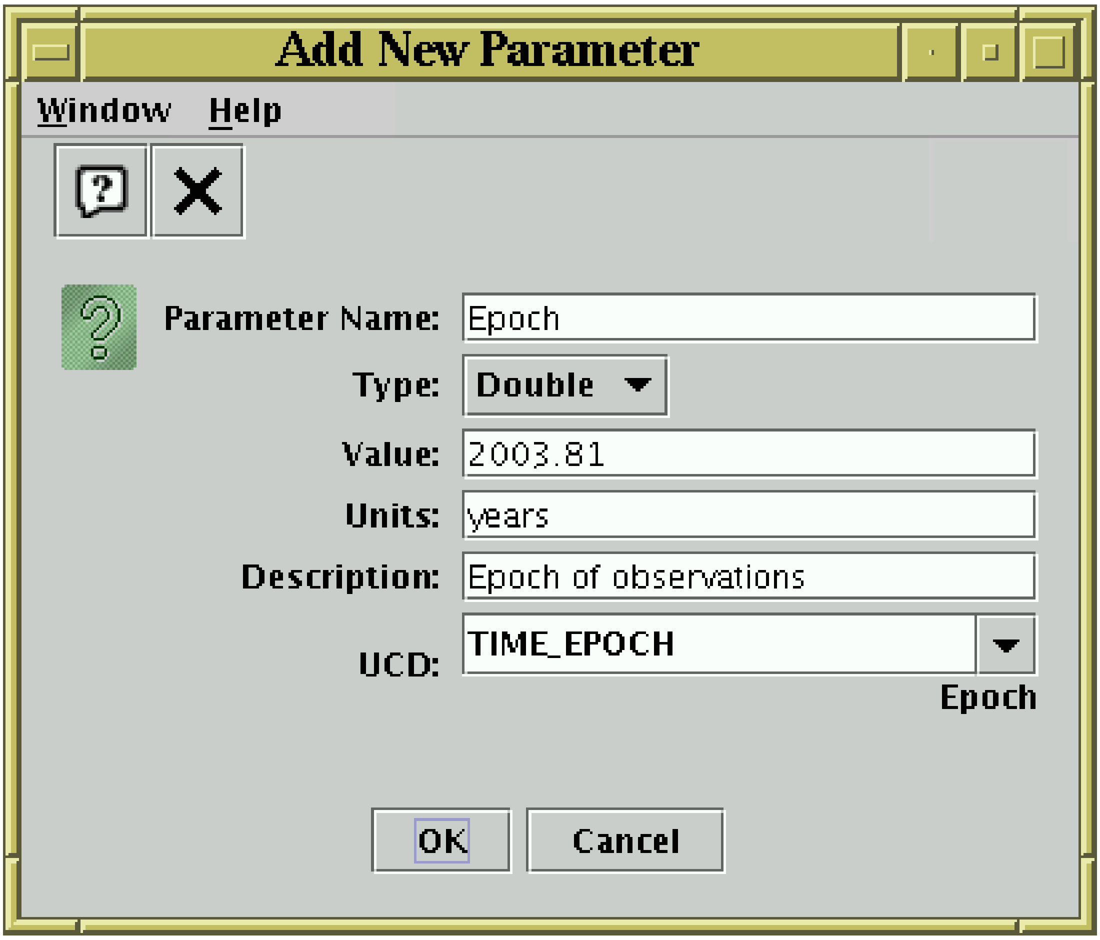
Case Sensitive:
If checked, matching is case-sensitive, otherwise, upper/lower case discrepancies are ignored.

Search Text:
The text to match cell values in the given column according to the constraints defined by the other controls.

The **Search** button will only be enabled if the controls specify a legal search condition. If for instance the search text is empty, or *Regex* syntax is chosen and the search text is not a legal regular expression, it will be disabled.

The search will usually complete straight away, but if the table is very long it may take a significant amount of time. In this case progress is displayed with the progress bar at the bottom of the window, and you can interrupt the running search using the **Stop** button.

A.11.6 New Parameter Window



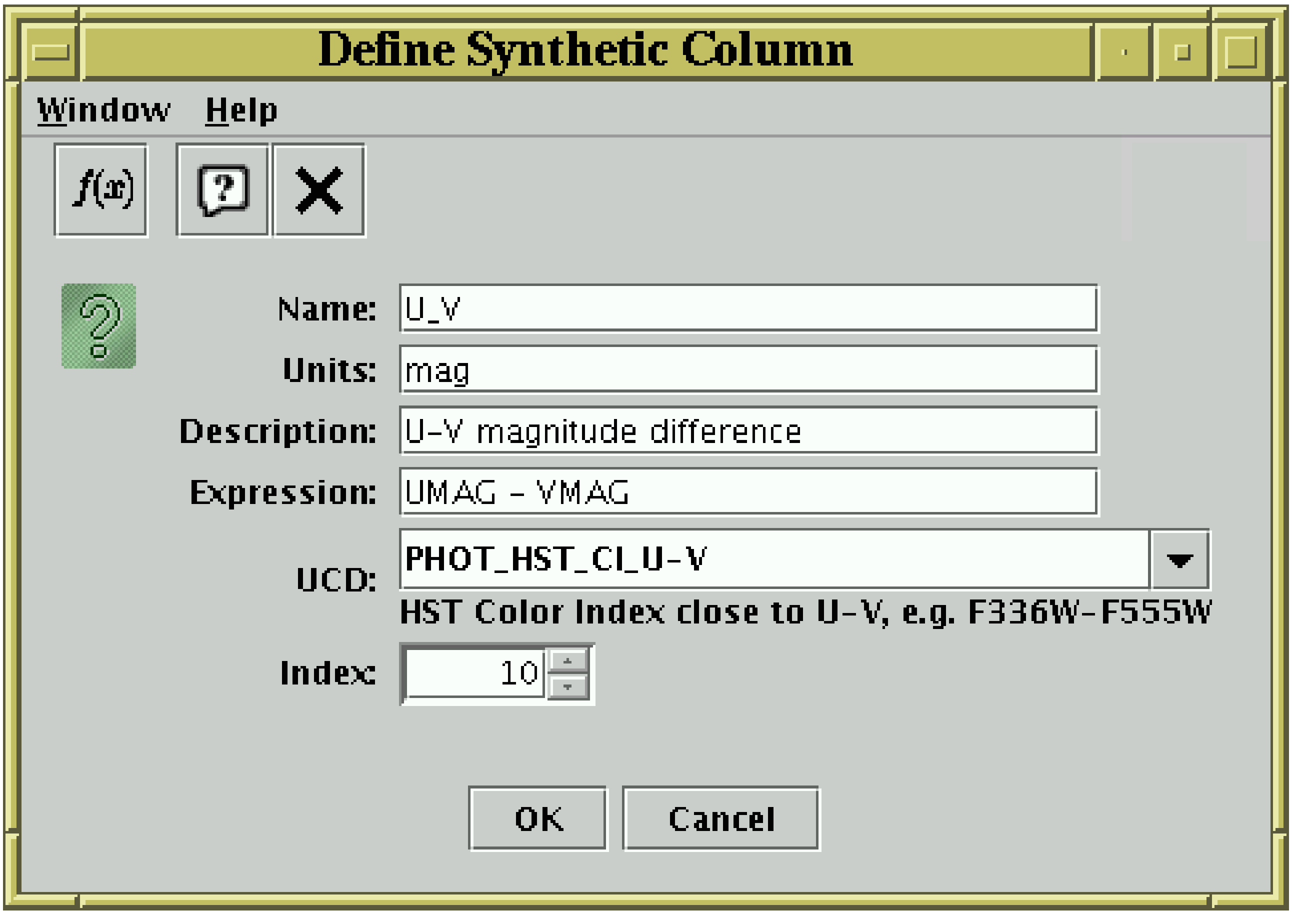
Parameter Name: Epoch
Type: Double
Value: 2003.81
Units: years
Description: Epoch of observations
UCD: TIME_EPOCH
Epoch

New Parameter dialogue window

The New Parameter window allows you to enter a new table parameter to be added to a table. It can be obtained by clicking the **New Parameter** (✚) button in the Appendix A.3.2. A parameter is simply a fixed value attached to a table and can contain information which is a string, a scalar, an array... in fact exactly the same sorts of values which can appear in table cells.

The window is pretty straightforward to use: fill in the fields and click **OK** to complete the addition. The **Type** selector allows you to select what kind of value you have input. The only compulsory field is **Parameter Name**; any of the others may be left blank, though you will usually want to fill in at least the **Value** field as well. Often, the parameter will have a string value, in which case the **Units** field is not very relevant.

A.11.7 Synthetic Column Window



Name: U_V

Units: mag

Description: U-V magnitude difference

Expression: UMAG - VMAG

UCD: PHOT_HST_CI_U-V

HST Color Index close to U-V, e.g. F336W-F555W

Index: 10

Synthetic Column dialogue window

The Synthetic Column Window allows you to define a new "Synthetic" column, that is one whose values are defined using an algebraic expression based on the values of other columns in the same row. The idea is that the value of the cells in a given row in this column will be calculated on demand as a function of the values of cells of other columns in that row. You can think of this as providing functionality like that of a column-oriented spreadsheet. You can activate the dialogue using the **Add Column** (✚) or **Replace Column** (↪) buttons in the Columns Window or from the (right-click) popup menu in the Data Window.

The window consists of a number of fields you must fill in to define the new column:

Name
The name of the new column. This should preferably be unique (different from all the other column names). It will be easier to use it in algebraic expressions if it is also:

- Different from other columns even apart from upper/lower case distinctions
- In the form of a java identifier (starts with a letter, continues with alphanumerics

including underscore)
- Not too long

Expression

This is the algebraic expression which defines the values that the cells in the new column of the table will have. The rules for writing algebraic expressions are described in Section 7, and detailed documentation of the functions you can use can be seen in the Available Functions Window, which you can see by clicking the **Show Functions** (■) button on the toolbar.

Units

The units of the column. If the quantity it represents is dimensionless or you don't know the units, this can be left blank. It would be a good idea to use a similar format for the units to that used for the existing columns in the table.

Description

A short textual description of what the values contained by this column are. May be left blank.

UCD

A Unified Content Descriptor for the column; a UCD is a semantic label attached to the column indicating what kind of quantity it contains by picking one option from a list defined by the CDS. The list of known UCDs is available via a selection box, or you can type a UCD in by hand. You may leave this blank if the you do not wish to assign a UCD to the column. A brief description of the UCD selected is visible below selection box itself.

Index

Determines the position in the displayed table at which the new column will initially appear.

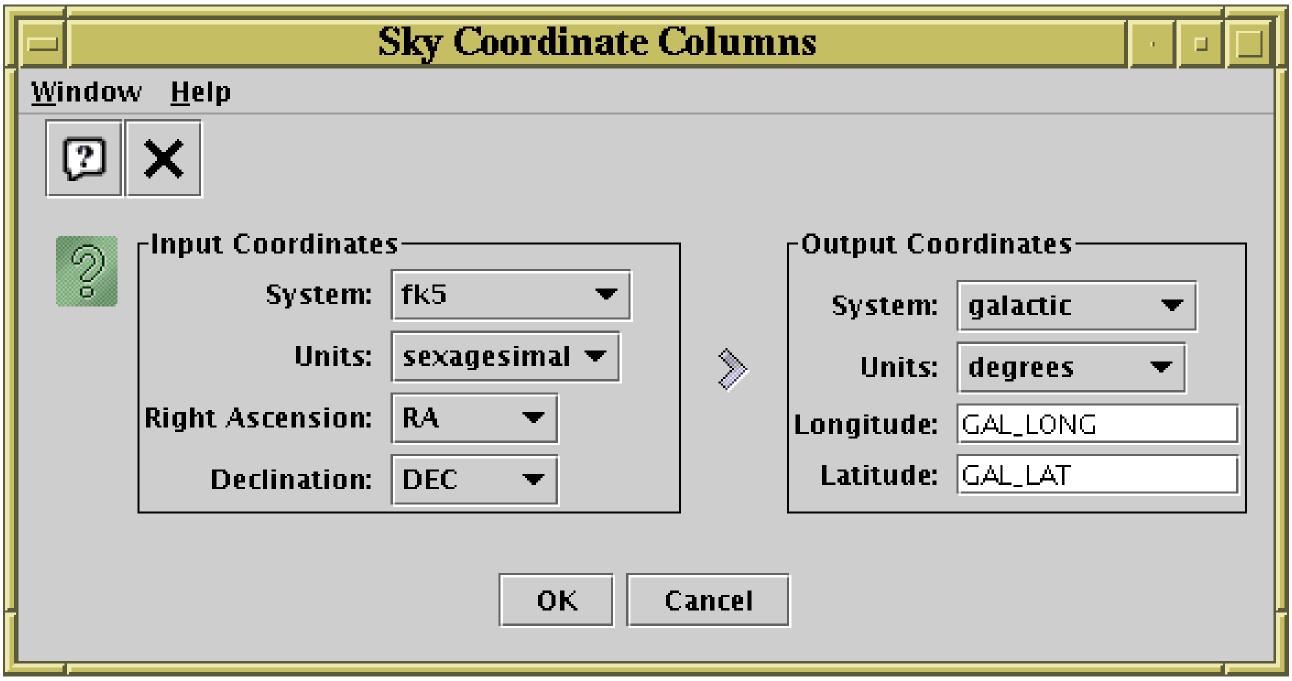
Of these, the **Expression** is the only one which must be filled in.

Having filled in the form to your satisfaction, hit the **OK** button at the bottom and the new column will be added to the table. If you have made some mistake in filling in the fields, a popup window will give you a message describing the problem. This message may be a bit arcane - try not to panic and see if you can rephrase the expression in a way that the parser might be happier with. If you can't work out the problem, it's time to consult your friendly local Java programmer (failing that, your friendly local C programmer may be able to help) or, by all means, contact the author.

If you wish to add more metadata items you can edit the appropriate cells in the Columns Window. You can edit the expression of an existing synthetic column in the same way.

Once created, a synthetic column is added to the Apparent Table and behaves just like any other; it can be moved, hidden/revealed, used in expressions for other synthetic columns and so on. If the table is saved the new column and its contents will be written to the new output table.

A.11.8 Sky Coordinates Window



Input Coordinates

System: fk5

Units: sexagesimal

Right Ascension: RA

Declination: DEC

Output Coordinates

System: galactic

Units: degrees

Longitude: GAL_LONG

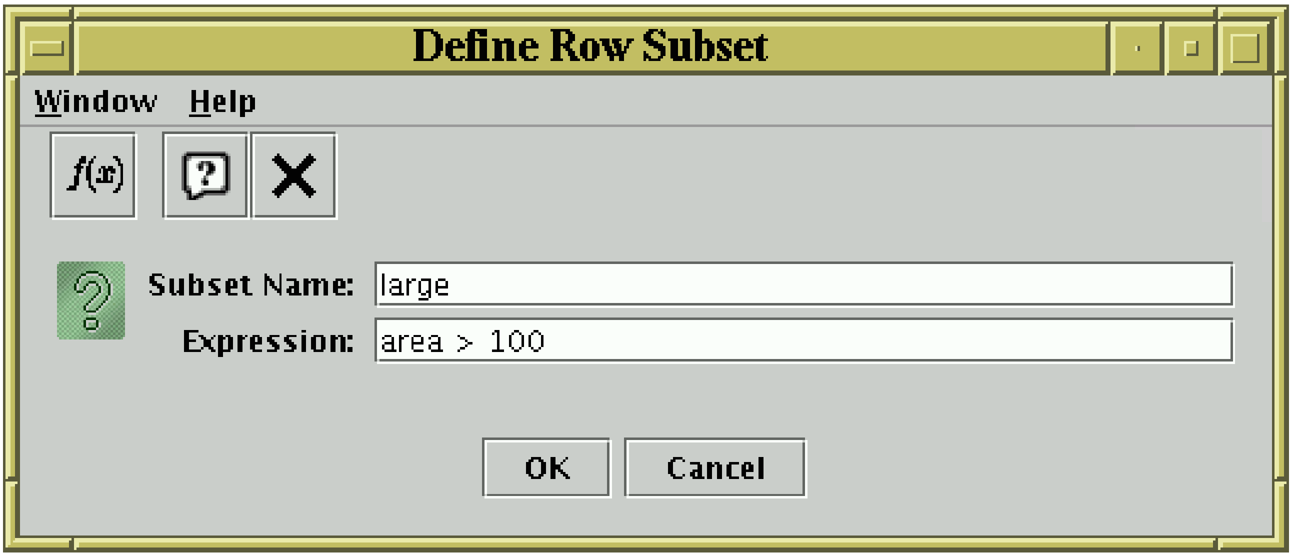
Latitude: GAL_LAT

Sky Coordinates Window

The Sky Coordinates Window allows you to add new columns to a table, representing coordinates in a chosen sky coordinate system. The table must already contain columns which represent sky coordinates; by describing the systems of the existing and of the new coordinates, you provide enough information to calculate the values in the new columns. You can activate this dialogue using the **New Sky Coordinate Columns** ( ) button in the Columns Window.

The dialogue window has two halves; on the left you give the existing columns which represent sky coordinates, their coordinate system (ICRS, fk5, fk4, galactic, supergalactic or ecliptic) and the units (degrees, radians or sexagesimal) that they are in. Note that the columns available for selection will depend on the units you have selected; for degrees or radians only numeric columns will be selectable, while for sexagesimal (dms/hms) units only string columns will be selectable. On the right you make the coordinate system and units selections as before, but enter the names of the new columns in the text fields. Then just hit the **OK** button, and the columns resulting from the coordinate conversion will be appended at the right of the table.

A.11.9 Algebraic Subset Window



Subset Name: large

Expression: area > 100

Algebraic Subset dialogue window

The Algebraic Subset Window allows you to define a new Row Subset which uses an algebraic expression to define which rows are included. The expression must be a boolean one, i.e. its value is either true or false for each row of the table. You can activate this dialogue using the **Add Subset** ( ) button in the Subsets Window.

The window consists of two fields which must be filled in to define the new subset:

Subset Name

The name of the new subset. This should preferably be unique (different from existing subset names). It will be easier to use it in other expressions if it is also:

- Different from other columns even apart from upper/lower case distinctions
- In the form of a java identifier (starts with a letter, continues with alphanumerics including underscore)
- Not too long

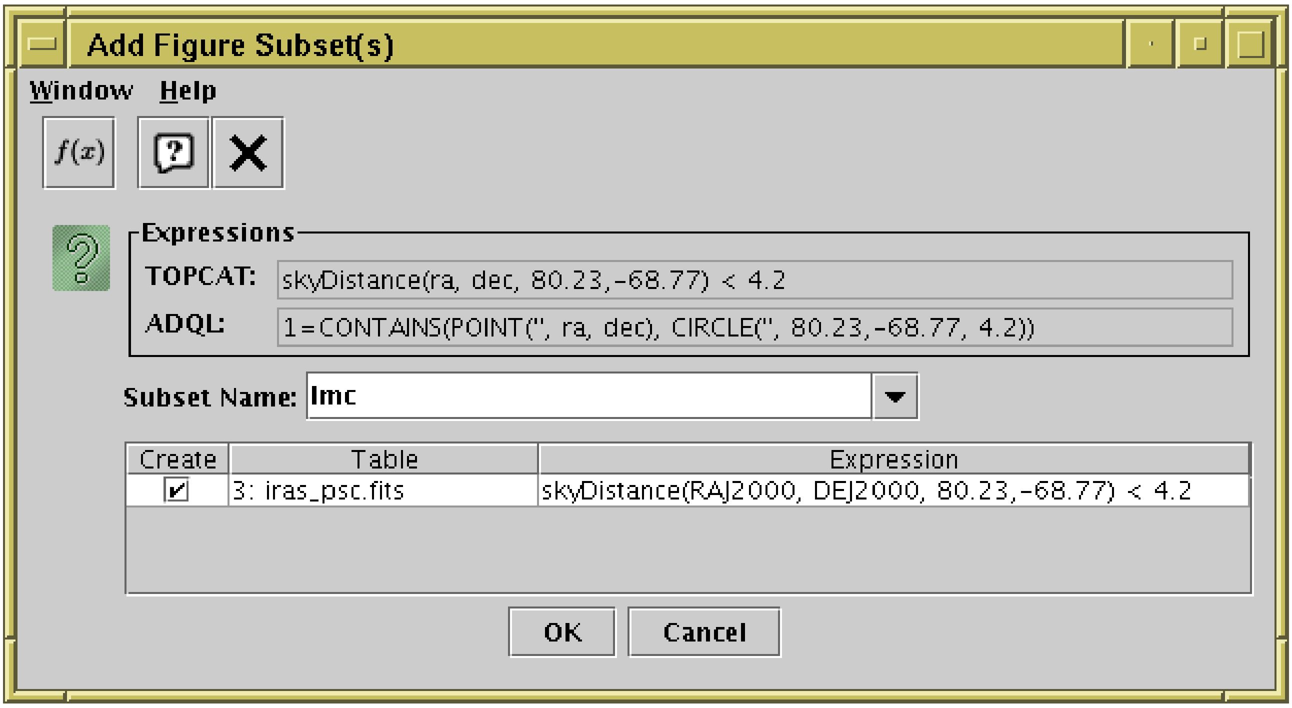
Expression

This is a boolean expression which defines the subset; it is a function of the values of any combination of the columns; only rows for which it evaluates to true will be included in the subset. The values of the other columns in the same row are referenced using their names or $ID identifiers, and other subsets may be referenced using their names or _ID identifiers. The rules for expression syntax are described in Section 7, and detailed documentation of the functions you can use can be seen in the Available Functions Window, which you can see by clicking the **Show Functions** ( ) button on the toolbar.

Having filled in the form to your satisfaction, hit the **OK** button at the bottom and the new subset will be added to the list that can be seen in the Subsets Window where it behaves like any other. If you have made some mistake in filling in the fields, a popup window will give you a message describing the problem.

A.11.10 Multi Algebraic Subset Window

The Multi Algebraic Subset Window is displayed when an action can create more than one subset with algebraic definitions at once. This happens on completion of the **Draw Algebraic Subset** ( ) or **Algebraic Subset From Visible** ( ) actions; more than one subset may result if more than one data set is plotted. If there is more than one subset to be added, they will be shown here in separate rows. Often however, only one subset will be presented.



Expressions

TOPCAT: skyDistance(ra, dec, 80.23,-68.77) < 4.2

ADQL: 1=CONTAINS(POINT('', ra, dec), CIRCLE('', 80.23,-68.77, 4.2))

Subset Name: lmc

Create	Table	Expression
☑	3: iras_psc.fits	skyDistance(RAJ2000, DEJ2000, 80.23,-68.77) < 4.2

Multi Algebraic Subset Window showing a subset expression to add

The **Subset Name** selector gives the name of the subset that will be added. You can type in a new name, or select one from the list of existing subsets if you wish to redefine one (re-use an existing subset name). Note: if you re-use an existing name, the expression must not reference that name directly or indirectly. If it does, an error will be displayed, and you will have to choose again.

Each row has the following entries:

Create
Indicates whether the subset should be added; you can uncheck it if you don't want to add that subset.

Table
Indicates which table the subset will be added to.

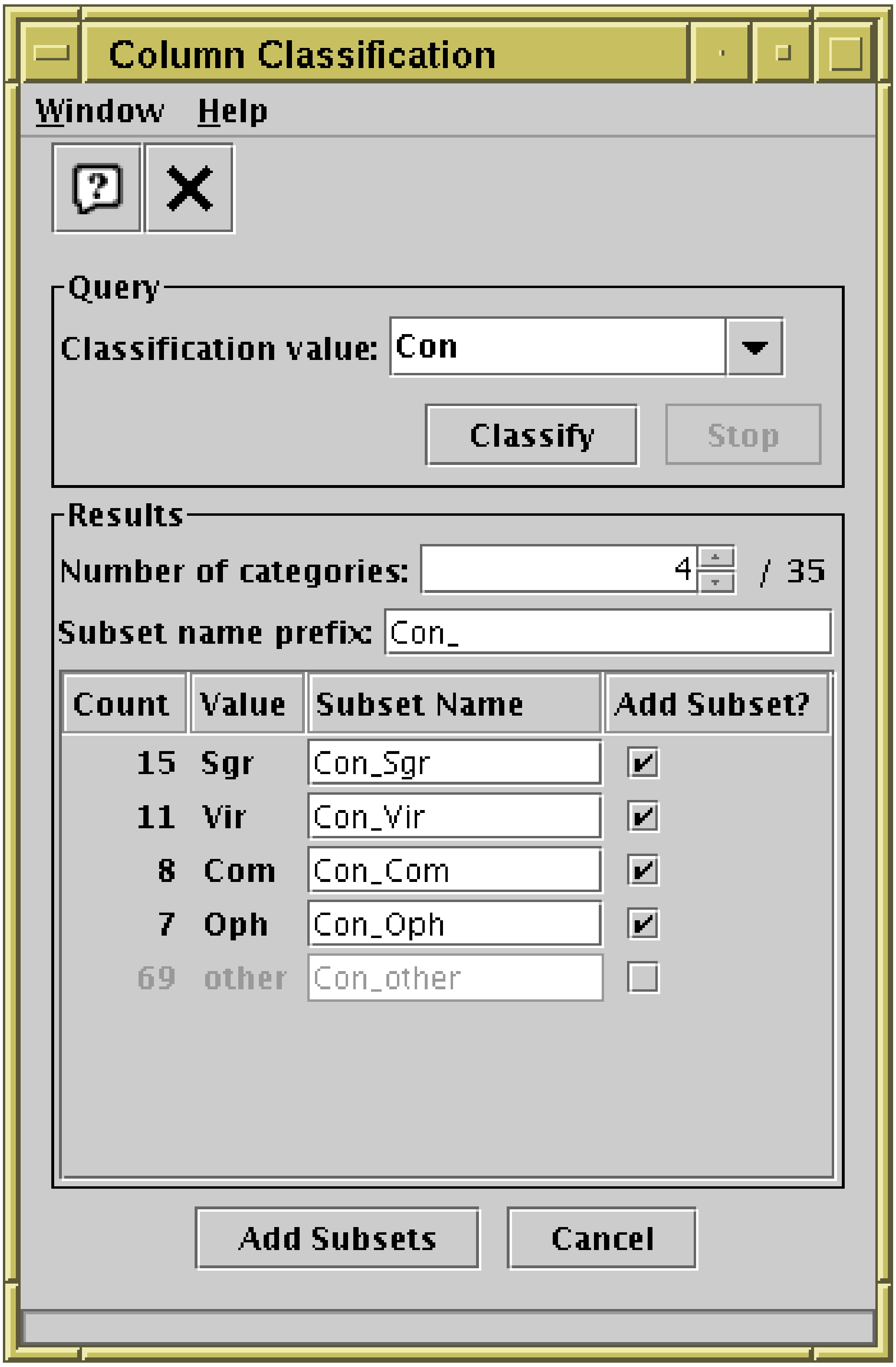
Expression
Gives the algebraic expression corresponding to the region you have defined. You can edit this before accepting it if you like by double-clicking on the field in the usual way. If the expression contains a syntax error, it will be greyed out, and attempting to add it will cause an error; you can uncheck the Create checkbox if you want to exclude it.

Once you are happy with the configuration, hit the **OK** button and the subset(s) will be created.

An **Expressions** panel also appears at the top of the window. This will, if possible, show representations of the selected region: **TOPCAT** uses the expression language used elsewhere in the application, and **ADQL** uses ADQL syntax. The ADQL form can be cut and pasted for use in the WHERE clause of a query in the TAP window. The variable names in these expressions are generic (e.g. "x" and "y" for Cartesian axes), instead of the actual quantities plotted, which are used in the subset expressions below.

Note: *the display of expression information in this window is somewhat experimental; better support for region descriptions may be provided elsewhere in the application in some future release.*

A.11.11 Column Classification Window



Classification value: Con

Number of categories: 4 / 35

Subset name prefix: Con_

Count	Value	Subset Name	Add Subset?
15	Sgr	Con_Sgr	☑
11	Vir	Con_Vir	☑
8	Com	Con_Com	☑
7	Oph	Con_Oph	☑
69	other	Con_other	☐

Column Classification Window

The Column Classification Window takes a column or other algebraic expression, and generates a number of mutually exclusive Row Subsets based on its contents. For instance if you have a column with different integer values representing different object types, this window will let you add a new subset to the table for each of the distinct values (object types) appearing in the table. Each subset contains rows with a single value of the classification expression you supply. In the above example, a new subset will be created for each of the four most commonly-occurring constellations in the table.

You can activate this dialogue using the **Classify By Column** () button in the Subsets

Window.

In the **Query** panel, you specify the expression you want to classify on. You can select a simple column name from the drop-down list, or type in an algebraic expression as described in Section 7. If you want to classify on ranges of values rather than exact equivalence you can use an expression that rounds to an integer, for instance "`toInteger(RMAG/2.0)*2`" would give you subsets corresponding to bins of width 2 in magnitude. When you have entered the expression, you may need to click the **Classify** button to start the classification (or it may happen automatically). The classification will often complete straight away, but for large tables it could take a noticeable amount of time, in which case a progress bar is shown at the bottom of the window. You can stop a long-running classification in progress with the **Stop** button. If you have a very large table with many distinct categories in the given column, the process can take a lot of memory - if the application runs out of memory, a warning will pop up and the classification will not complete.

When the classification has run, the results are displayed in the **Results** panel. Two fields control the way these results are displayed:

Number of Categories
Since in general there may be a large number of different values in the column of interest (as many as there are rows), only the few most popular ones are shown. This field controls the maximum number shown, you can adjust it as required. The fixed value after the entry field shows the total number of distinct values discovered in the data; increasing the field's value beyond this value will have no further effect.

Subset name prefix
Defines the prefix added to the column values to give the default name for each subset that will be added to the table. The application tries to come up with something sensible based on the classification expression, but doesn't always succeed. If you type in a new prefix here, all the subsets listed below are given new default names accordingly.

The results are displayed in a table, each row corresponding to a Row Subset that can be added to the table based on the classification. The subsets are shown in decreasing order of popularity (the subset containing the most rows is listed first). The final row, labelled "**other**", groups all the rows which are not in any of the other currently selected and displayed subsets. The following columns are shown:

Count
Number of rows in the subset.

Value
The classification value shared by all rows in the subset.

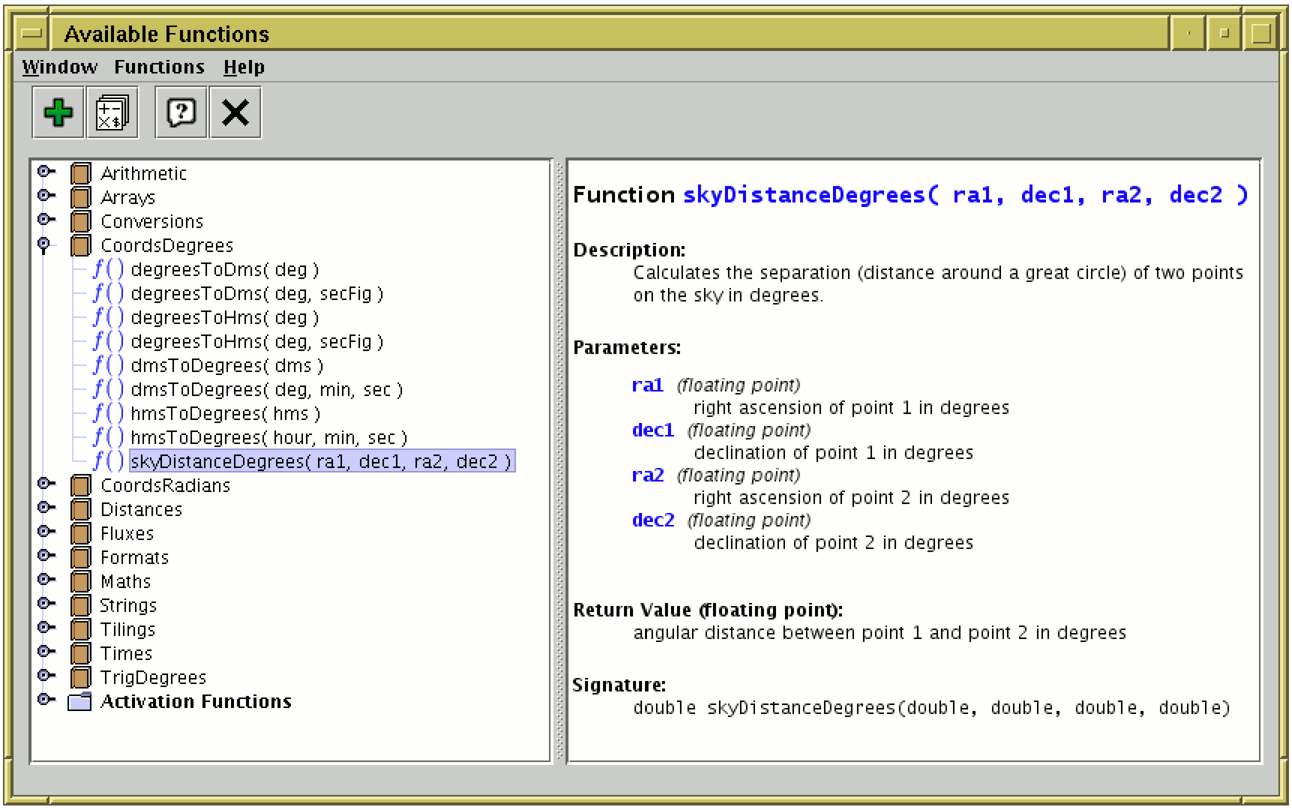
Subset Name
The name of the subset that will be added. The applicaiton tries to come up with sensible default names, but doesn't always succeed - you can edit this field as required. If the name matches the name of any subset already present in the table, when the new subset is added it will replace the old one.

Add Subset?
If this box is checked, the **Add Subsets** button below will add a subset corresponding to this row; if not, it will be ignored. If unchecked, the rows from this subset are considered part of the **other** subset listed at the bottom (you can see the **Count** field in the **other** row adjusting when you include or exclude subsets by checking/uncheking this box).

When you have adjusted subset names and selected the ones you want to add, click the **Add Subsets** button at the bottom, and one subset will be added to the table for each of the items with the **Add Subset?** checkbox ticked. If you don't want to add any of these subsets after all, just hit

Cancel or close the window.

A.11.12 Available Functions Window



- Arithmetic
- Arrays
- Conversions
- CoordsDegrees
  - degreesToDms(deg)
  - degreesToDms(deg, secFig)
  - degreesToHms(deg)
  - degreesToHms(deg, secFig)
  - dmsToDegrees(dms)
  - dmsToDegrees(deg, min, sec)
  - hmsToDegrees(hms)
  - hmsToDegrees(hour, min, sec)
  - skyDistanceDegrees(ra1, dec1, ra2, dec2)
- CoordsRadians
- Distances
- Fluxes
- Formats
- Maths
- Strings
- Tilings
- Times
- TrigDegrees
- **Activation Functions**

Function skyDistanceDegrees(ra1, dec1, ra2, dec2)

Description:
Calculates the separation (distance around a great circle) of two points on the sky in degrees.

Parameters:

ra1 *(floating point)*
right ascension of point 1 in degrees
dec1 *(floating point)*
declination of point 1 in degrees
ra2 *(floating point)*
right ascension of point 2 in degrees
dec2 *(floating point)*
declination of point 2 in degrees

Return Value (floating point):
angular distance between point 1 and point 2 in degrees

Signature:

```
double skyDistanceDegrees(double, double, double, double)
```

Available Functions Window

This window displays all the functions (Java methods) which are available for use when writing algebraic expressions (Section 7). This includes both the built-in expressions and any extended ones you might have added. You can find this window by using the **Show Functions** ( ) button in the Synthetic Column or Algebraic Subset window toolbars.

On the left hand side of the window is a tree-like representation of the functions you can use. Each item in this tree is one of the following:

Folder
A group of classes. There's only one of these, marked "Activation Functions", and it contains functions which are only available for use in Activation Actions (Section 8). When defining a new synthetic columns or algebraic subsets they are not used.

Class
A set of functions and/or constants; it doesn't matter what class a function is in when you use it, but since the functions in a class are usually related this makes it easier to find the one you're looking for in this window.

f() **Function**
A function that you can use in an expression.

C **Constant**
A constant value which you can refer to by name in an expression (as long as it doesn't clash with a column name).

Of these, the Folder and Class items have a 'handle' (⊸), which means that they contain other items (classes and functions/constants respectively). By clicking on the handle (or equivalently double-clicking on the name) you can toggle whether the item is open (so you can see its contents) or closed (so you can't). So to see the functions in a class, click on its handle and they will be revealed.

You can click on any of these items and information about it will appear in the right hand panel. In the case of functions this describes the function, its arguments, what it does, and how to use it. The explanations should be fairly self-explanatory; for instance the description in the figure above indicates that you could use the invocation `skyDistanceDegrees(RA1,DEC1,RA2,DEC2)` as the expression for a new table column which gives the angular distance between two sky positions represented by columns with the names RA1, DEC1 and RA2, DEC2. Examples of a number of these functions are given in Section 7.9.

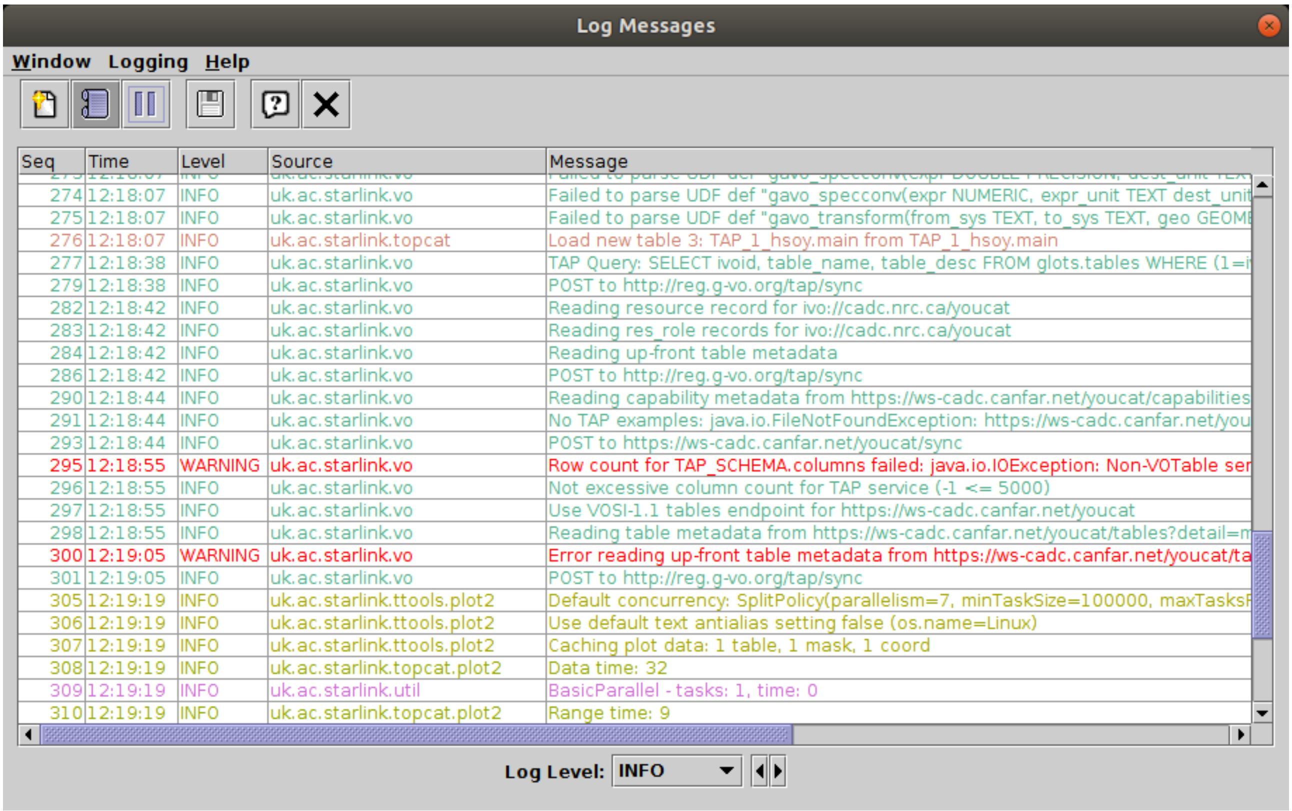
There are two menu/toolbar actions:

Syntax Help
: Displays the section of the manual which discusses the general syntax used for algebraic expressions, including what operators are available, what the rules are for referring to columns, etc. The help is displayed in the Help Window (and can from there be passed to a browser if that's more convenient). The displayed material is simply a shortcut to Section 7 in this document.

Add Class
: Allows you to add the name of a class to those available. You should enter the fully-qualified class name (i.e. including the dot-separated package path). The class that you specify must be on the class path which was current when TOPCAT was started, as explained in Section 10.2.1. Note however it would be more usual to specify these using the system property `jel.classes` or `jel.classes.activation` at startup, as described in Section 7.10.

Classes added either using the **Add Class** action above or the `jel.classes` or `jel.classes.activation` system properties will be visible in the tree in this window, but they may not have proper documentation (clicking on them may not reveal a description in the right hand panel).

A.11.13 Log Window



Seq	Time	Level	Source	Message
274	12:18:07	INFO	uk.ac.starlink.vo	Failed to parse UDF def "gavo_specconv(expr NUMERIC, expr_unit TEXT dest_unit
275	12:18:07	INFO	uk.ac.starlink.vo	Failed to parse UDF def "gavo_transform(from_sys TEXT, to_sys TEXT, geo GEOME
276	12:18:07	INFO	uk.ac.starlink.topcat	Load new table 3: TAP_1_hsoy.main from TAP_1_hsoy.main
277	12:18:38	INFO	uk.ac.starlink.vo	TAP Query: SELECT ivoid, table_name, table_desc FROM glots.tables WHERE (1=i
279	12:18:38	INFO	uk.ac.starlink.vo	POST to http://reg.g-vo.org/tap/sync
282	12:18:42	INFO	uk.ac.starlink.vo	Reading resource record for ivo://cadc.nrc.ca/youcat
283	12:18:42	INFO	uk.ac.starlink.vo	Reading res_role records for ivo://cadc.nrc.ca/youcat
284	12:18:42	INFO	uk.ac.starlink.vo	Reading up-front table metadata
286	12:18:42	INFO	uk.ac.starlink.vo	POST to http://reg.g-vo.org/tap/sync
290	12:18:44	INFO	uk.ac.starlink.vo	Reading capability metadata from https://ws-cadc.canfar.net/youcat/capabilities
291	12:18:44	INFO	uk.ac.starlink.vo	No TAP examples: java.io.FileNotFoundException: https://ws-cadc.canfar.net/you
293	12:18:44	INFO	uk.ac.starlink.vo	POST to https://ws-cadc.canfar.net/youcat/sync
295	12:18:55	WARNING	uk.ac.starlink.vo	Row count for TAP_SCHEMA.columns failed: java.io.IOException: Non-VOTable ser
296	12:18:55	INFO	uk.ac.starlink.vo	Not excessive column count for TAP service (-1 <= 5000)
297	12:18:55	INFO	uk.ac.starlink.vo	Use VOSI-1.1 tables endpoint for https://ws-cadc.canfar.net/youcat
298	12:18:55	INFO	uk.ac.starlink.vo	Reading table metadata from https://ws-cadc.canfar.net/youcat/tables?detail=m
300	12:19:05	WARNING	uk.ac.starlink.vo	Error reading up-front table metadata from https://ws-cadc.canfar.net/youcat/ta
301	12:19:05	INFO	uk.ac.starlink.vo	POST to http://reg.g-vo.org/tap/sync
305	12:19:19	INFO	uk.ac.starlink.ttools.plot2	Default concurrency: SplitPolicy(parallelism=7, minTaskSize=100000, maxTasksF
306	12:19:19	INFO	uk.ac.starlink.ttools.plot2	Use default text antialias setting false (os.name=Linux)
307	12:19:19	INFO	uk.ac.starlink.ttools.plot2	Caching plot data: 1 table, 1 mask, 1 coord
308	12:19:19	INFO	uk.ac.starlink.topcat.plot2	Data time: 32
309	12:19:19	INFO	uk.ac.starlink.util	BasicParallel - tasks: 1, time: 0
310	12:19:19	INFO	uk.ac.starlink.topcat.plot2	Range time: 9

Log Window

The log window can be obtained using the **View Log** ( ) option on the **File** menu of the Control Window.

This window displays any log messages which the application has generated. These messages are *not* required reading for users, but they can be useful for debugging in case of problems, and they can also tell you some useful information about what the application is doing, for instance external URLs that are being accessed or ADQL query text. These messages can be useful if you want to reproduce some parts of TOPCAT's behaviour outside the application, e.g. using `curl` or a web browser.

The messages are colour-coded to some extent. WARNING- and SEVERE-level messages, which generally indicate some kind of problem that may cause trouble, are displayed in red and black respectively. INFO- and CONFIG-level messages, which generally just provide information about what's going on under the hood, are assigned an arbitrary colour based on the code subsystem (the *Source* value) that has generated them. These colours are not really significant, they are just intended as a visual aid.

The **Level** selector at the bottom can be used as a filter so that only messages beyond a certain severity are displayed.

Depending on whether the `-verbose` flag has been specified, some or all of the displayed messages may have been written to the console as well (if there is a console - this depends on how you have invoked TOPCAT).

There is a limit (currently 1000) to the number of logging messages retained, so for long-running instances the older ones may be discarded.

The following toolbar/menu items are available:

Clear

Clears all the existing messages from the display.

Auto-scroll

When this toggle is on (the default), the display will scroll to the bottom any time a new log message is added, so you can always see the latest ones. That can be annoying if you're trying to examine messages earlier in the list however, so by turning it off you can prevent this behaviour.

Pause logging

While this toggle is on, no more messages will be added to the display.

Export to file

Writes all the current content of the logging buffer to a text file. Even those messages currently filtered out by level from the display are included. The output is in a somewhat different format than what's displayed. This output may be a useful accompaniment to (certain kinds of) bug reports.

Note: the messages displayed here are those written through Java's logging system. Although it tries not to disturb things too much, TOPCAT's manipulation of the logging infrastructure affects how it is set up, so if you have customised your logging setup using, e.g., the `java.util.logging.config.*` system properties, you may find that it's not behaving exactly as you expected. Sorry.

B Algebraic Functions

This appendix lists the functions which can be used in algebraic expressions (see Section 7). They are listed in two sections: the first gives the functions available for use anywhere an expression can be used, and the second gives those only for use in defining custom Activation Actions (Section 8).

This information is also available within TOPCAT from the Available Functions Window (Appendix A.11.12), obtained using the ■ toolbar button.

B.1 General Functions

The following functions can be used anywhere that you can write an algebraic expression in TOPCAT. They will typically be used for defining new synthetic columns or algebraically-defined row subsets. The documentation in the following subsections is also available from within TOPCAT in the Available Functions Window.

B.1.1 Arithmetic

Standard arithmetic functions including things like rounding, sign manipulation, and maximum/minimum functions. Phase folding operations, and a convenient form of the modulus operation on which they are based, are also provided.

`roundUp( x )`

Rounds a value up to an integer value. Formally, returns the smallest (closest to negative infinity) integer value that is not less than the argument.

- Parameters:
  - `x` *(floating point)*: a value.
- Return value
  - *(integer)*: `x` rounded up

`roundDown( x )`

Rounds a value down to an integer value. Formally, returns the largest (closest to positive infinity) integer value that is not greater than the argument.

- Parameters:
  - `x` *(floating point)*: a value
- Return value
  - *(integer)*: `x` rounded down

`round( x )`

Rounds a value to the nearest integer. Formally, returns the integer that is closest in value to the argument. If two integers are equally close, the result is the even one.

- Parameters:
  - `x` *(floating point)*: a floating point value.
- Return value
  - *(integer)*: `x` rounded to the nearest integer

`roundDecimal( x, dp )`

Rounds a value to a given number of decimal places. The result is a `float` (32-bit floating point value), so this is only suitable for relatively low-precision values. It's intended for truncating the number of apparent significant figures represented by a value which you know has been obtained by combining other values of limited precision. For more control, see the functions in the `Formats` class.

- Parameters:
  - `x` *(floating point)*: a floating point value
  - `dp` *(integer)*: number of decimal places (digits after the decimal point) to retain
- Return value
  - *(floating point)*: floating point value close to `x` but with a limited apparent precision
- Example:
  - `roundDecimal(PI,2) = 3.14f`

`abs( x )`

Returns the absolute value of an integer value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

- Parameters:
  - `x` *(integer)*: the argument whose absolute value is to be determined
- Return value
  - *(integer)*: the absolute value of the argument.

`abs( x )`

Returns the absolute value of a floating point value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

- Parameters:
  - `x` *(floating point)*: the argument whose absolute value is to be determined
- Return value
  - *(floating point)*: the absolute value of the argument.

`max( a, b )`

Returns the greater of two integer values. If the arguments have the same value, the result is that same value.

Multiple-argument maximum functions are also provided in the `Arrays` and `Lists` packages.

- Parameters:
  - `a` *(integer)*: an argument.
  - `b` *(integer)*: another argument.
- Return value
  - *(integer)*: the larger of `a` and `b`.

`maxNaN( a, b )`

Returns the greater of two floating point values. If the arguments have the same value, the result is that same value. If either value is blank, then the result is blank.

- Parameters:
    - `a` *(floating point)*: an argument.
    - `b` *(floating point)*: another argument.
- Return value
    - *(floating point)*: the larger of `a` and `b`.

`maxReal( a, b )`

Returns the greater of two floating point values, ignoring blanks. If the arguments have the same value, the result is that same value. If one argument is blank, the result is the other one. If both arguments are blank, the result is blank.

Multiple-argument maximum functions are also provided in the `Arrays` and `Lists` packages.

- Parameters:
    - `a` *(floating point)*: an argument
    - `b` *(floating point)*: another argument
- Return value
    - *(floating point)*: the larger non-blank value of `a` and `b`

`min( a, b )`

Returns the smaller of two integer values. If the arguments have the same value, the result is that same value.

Multiple-argument minimum functions are also provided in the `Arrays` and `Lists` packages.

- Parameters:
    - `a` *(integer)*: an argument.
    - `b` *(integer)*: another argument.
- Return value
    - *(integer)*: the smaller of `a` and `b`.

`minNaN( a, b )`

Returns the smaller of two floating point values. If the arguments have the same value, the result is that same value. If either value is blank, then the result is blank.

- Parameters:
    - `a` *(floating point)*: an argument.
    - `b` *(floating point)*: another argument.
- Return value
    - *(floating point)*: the smaller of `a` and `b`.

`minReal( a, b )`

Returns the smaller of two floating point values, ignoring blanks. If the arguments have the same value, the result is that same value. If one argument is blank, the result is the other one. If both arguments are blank, the result is blank.

Multiple-argument minimum functions are also provided in the `Arrays` and `Lists` packages.

- Parameters:
  - `a` *(floating point)*: an argument
  - `b` *(floating point)*: another argument
- Return value
  - *(floating point)*: the larger non-blank value of `a` and `b`

`mod( a, b )`

Returns the non-negative remainder of `a/b`. This is a modulo operation, but differs from the expression `a%b` in that the answer is always >=0 (as long as `b` is not zero).

- Parameters:
  - `a` *(floating point)*: dividend
  - `b` *(floating point)*: divisor
- Return value
  - *(floating point)*: non-negative remainder when dividing `a` by `b`
- Examples:
  - `modulo(14, 5) = 4`
  - `modulo(-14, 5) = 1`
  - `modulo(2.75, 0.5) = 0.25`

`phase( t, period )`

Returns the phase of a value within a period.

For positive period, the returned value is in the range [0,1).

- Parameters:
  - `t` *(floating point)*: value
  - `period` *(floating point)*: folding period
- Return value
  - *(floating point)*: mod(t,period)/period
- Examples:
  - `phase(7, 4) = 0.75`
  - `phase(-1000.5, 2.5) = 0.8`
  - `phase(-3300, 33) = 0`

`phase( t, period, t0 )`

Returns the phase of an offset value within a period. The reference value `t0` corresponds to phase zero.

For positive period, the returned value is in the range [0,1).

- Parameters:
  - `t` *(floating point)*: value
  - `period` *(floating point)*: folding period
  - `t0` *(floating point)*: reference value, corresponding to phase zero
- Return value
  - *(floating point)*: phase(t-t0, period)
- Examples:

  - `phase(5003,100,0) = 0.03`
  - `phase(5003,100,2) = 0.01`
  - `phase(5003,100,4) = 0.99`

`phase( t, period, t0, phase0 )`

Returns the offset phase of an offset value within a period. The reference value `t0` corresponds to integer phase value, and the phase offset `phase0` determines the starting value for the phase range.

For positive period, the returned value is in the range [`phase0`,`phase0+1`).

- Parameters:
  - `t` *(floating point)*: value
  - `period` *(floating point)*: folding period
  - `t0` *(floating point)*: reference value, corresponding to phase zero
  - `phase0` *(floating point)*: offset for phase
- Return value
  - *(floating point)*: offset phase
- Examples:
  - `phase(23,10,1,99) = 99.2`
  - `phase(8.6125,0.2,0.0125,-0.3) = 0`
  - `phase(8.6125,0.2,0.1125,-0.7) = -0.5`

B.1.2 Arrays

Functions which operate on array-valued cells. The array parameters of these functions can only be used on values which are already arrays (usually, numeric arrays). In most cases that means on values in table columns which are declared as array-valued. FITS and VOTable tables can have columns which contain array values, but other formats such as CSV cannot.

If you want to calculate aggregating functions like sum, min, max etc on multiple values which are not part of an array, it's easier to use the functions from the `Lists` class.

Note that none of these functions will calculate statistical functions over a whole column of a table.

The functions fall into a number of categories:

- Aggregating operations, which map an array value to a scalar, including `size`, `count`, `countTrue`, `maximum`, `minimum`, `sum`, `mean`, `median`, `quantile`, `stdev`, `variance`, `join`.
- Operations on one or more arrays which produce array results, including `add`, `subtract`, `multiply`, `divide`, `reciprocal`, `condition`, `slice`, `pick`. Mostly these work on any numeric array type and return floating point (double precision) values, but some of them (`slice`, `pick`) have variants for different array types.
- The function `array`, which lets you assemble a floating point array value from a list of scalar numbers. There are variants (`intArray`, `stringArray`) for some different array types.

`sum( array )`

Returns the sum of all the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

- Parameters:

    - `array` *(Object)*: array of numbers
- Return value
    - *(floating point)*: sum of all the numeric values in `array`

`mean( array )`

Returns the mean of all the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

- Parameters:
    - `array` *(Object)*: array of numbers
- Return value
    - *(floating point)*: mean of all the numeric values in `array`

`variance( array )`

Returns the population variance of all the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

- Parameters:
    - `array` *(Object)*: array of numbers
- Return value
    - *(floating point)*: variance of the numeric values in `array`

`stdev( array )`

Returns the population standard deviation of all the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

- Parameters:
    - `array` *(Object)*: array of numbers
- Return value
    - *(floating point)*: standard deviation of the numeric values in `array`

`minimum( array )`

Returns the smallest of the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

- Parameters:
    - `array` *(Object)*: array of numbers
- Return value
    - *(floating point)*: minimum of the numeric values in `array`

`maximum( array )`

Returns the largest of the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

- Parameters:
    - `array` *(Object)*: array of numbers

- Return value
    - *(floating point)*: maximum of the numeric values in `array`

`median( array )`

Returns the median of the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

- Parameters:
    - `array` *(Object)*: array of numbers
- Return value
    - *(floating point)*: median of the numeric values in `array`

`quantile( array, quant )`

Returns a quantile value of the non-blank elements in the array. Which quantile is determined by the `quant` value; values of 0, 0.5 and 1 give the minimum, median and maximum respectively. A value of 0.99 would give the 99th percentile.

- Parameters:
    - `array` *(Object)*: array of numbers
    - `quant` *(floating point)*: number in the range 0-1 deterining which quantile to calculate
- Return value
    - *(floating point)*: quantile corresponding to `quant`

`size( array )`

Returns the number of elements in the array. If `array` is not an array, zero is returned.

- Parameters:
    - `array` *(Object)*: array
- Return value
    - *(integer)*: size of `array`

`count( array )`

Returns the number of non-blank elements in the array. If `array` is not an array, zero is returned.

- Parameters:
    - `array` *(Object)*: array (may or may not be numeric)
- Return value
    - *(integer)*: number of non-blank elements in `array`

`countTrue( array )`

Returns the number of true elements in an array of boolean values.

- Parameters:
    - `array` *(array of boolean)*: array of true/false values
- Return value

  - *(integer)*: number of true values in `array`

`join( array, joiner )`

Returns a string composed of concatenating all the elements of an array, separated by a joiner string. If `array` is not an array, null is returned.

- Parameters:
  - `array` *(Object)*: array of numbers or strings
  - `joiner` *(String)*: text string to interpose between adjacent elements
- Return value
  - *(String)*: string composed of `array` elements separated by `joiner` strings
- Example:
  - `join(array(1.5,2.1,-3.9), "; ") = "1.5; 2.1; -3.9"`

`dotProduct( array1, array2 )`

Returns the dot (scalar) product of two numeric arrays. If either argument is not an array, or if the arrays are not of the same length, a blank value is returned.

- Parameters:
  - `array1` *(Object)*: first array
  - `array2` *(Object)*: second array
- Return value
  - *(floating point)*: sum of element-wise products of input arrays
- Example:
  - `dotProduct(array(3,4,5), array(1,2,3)) = 26`

`add( arrayOrScalar1, arrayOrScalar2 )`

Returns the element-by-element result of adding either two numeric arrays of the same length, or an array and a scalar considered as if an array of the right length.

If the arguments are not as expected (e.g. arrays of different lengths, both scalars, not numeric) then null is returned.

- Parameters:
  - `arrayOrScalar1` *(Object)*: first numeric array/scalar
  - `arrayOrScalar2` *(Object)*: second numeric array/scalar
- Return value
  - *(array of floating point)*: element-by-element result of `arrayOrScalar1 + arrayOrScalar2`, the same length as the input array(s)
- Examples:
  - `add(array(1,2,3), array(0.1,0.2,0.3)) = [1.1, 2.2, 3.3]`
  - `add(array(1,2,3), 10) = [11,12,13]`

`subtract( arrayOrScalar1, arrayOrScalar2 )`

Returns the element-by-element result of subtracting either two numeric arrays of the same length, or an array and a scalar considered as if an array of the right length.

If the arguments are not as expected (e.g. arrays of different lengths, both scalars, not numeric) then null is returned.

- Parameters:
  - `arrayOrScalar1` *(Object)*: first numeric array/scalar
  - `arrayOrScalar2` *(Object)*: second numeric array/scalar
- Return value
  - *(array of floating point)*: element-by-element result of `arrayOrScalar1 - arrayOrScalar2`, the same length as the input array(s)
- Examples:
  - `subtract(array(1,2,3), array(0.1,0.2,0.3)) = [0.9, 1.8, 2.7]`
  - `subtract(array(1,2,3), 1.0) = [0, 1, 2]`

`multiply( arrayOrScalar1, arrayOrScalar2 )`

Returns the element-by-element result of multiplying either two numeric arrays of the same length, or an array and a scalar considered as if an array of the right length.

If the arguments are not as expected (e.g. arrays of different lengths, both scalars, not numeric) then null is returned.

- Parameters:
  - `arrayOrScalar1` *(Object)*: first numeric array/scalar
  - `arrayOrScalar2` *(Object)*: second numeric array/scalar
- Return value
  - *(array of floating point)*: element-by-element result of `arrayOrScalar1 * arrayOrScalar2`, the same length as the input array(s)
- Examples:
  - `multiply(array(1,2,3), array(2,4,6)) = [2, 8, 18]`
  - `multiply(2, array(1,2,3)) = [2, 4, 6]`

`divide( arrayOrScalar1, arrayOrScalar2 )`

Returns the element-by-element result of dividing either two numeric arrays of the same length, or an array and a scalar considered as if an array of the right length.

If the arguments are not as expected (e.g. arrays of different lengths, both scalars, not numeric) then null is returned.

- Parameters:
  - `arrayOrScalar1` *(Object)*: first numeric array/scalar
  - `arrayOrScalar2` *(Object)*: second numeric array/scalar
- Return value
  - *(array of floating point)*: element-by-element result of `arrayOrScalar1 / arrayOrScalar2`, the same length as the input array(s)
- Examples:
  - `divide(array(0,9,4), array(1,3,8)) = [0, 3, 0.5]`
  - `divide(array(50,60,70), 10) = [5, 6, 7]`

`reciprocal( array )`

Returns the result of taking the reciprocal of every element of a numeric array. If the supplied `array` argument is not a numeric array, `null` is returned.

- Parameters:
  - `array` *(Object)*: array input
- Return value
  - *(array of floating point)*: array output, the same length as the `array` parameter
- Example:
  - `reciprocal(array(1,2,0.25) = [1, 0.5, 4]`

`condition( flagArray, trueValue, falseValue )`

Maps a boolean array to a numeric array by using supplied numeric values to represent true and false values from the input array.

This has the same effect as applying the expression `outArray[i] = flagArray[i] ? trueValue : falseValue`.

- Parameters:
  - `flagArray` *(array of boolean)*: array of boolean values
  - `trueValue` *(floating point)*: output value corresponding to an input true value
  - `falseValue` *(floating point)*: output value corresponding to an input false value
- Return value
  - *(array of floating point)*: output numeric array, same length as `flagArray`
- Example:
  - `condition([true, false, true], 1, 0) = [1, 0, 1]`

`constant( n, value )`

Returns a fixed-size array filled with a given constant value.

Note: This documents the double-precision version of the routine. Corresponding routines exist for other data types (`float`, `long`, `int`, `short`, `byte`).

- Parameters:
  - `n` *(integer)*: size of output array
  - `value` *(floating point)*: value of every element in the output array
- Return value
  - *(array of floating point)*: `n`-element array with every element set to `value`
- Example:
  - `constant(5, 23.5) = [23.5, 23.5, 23.5, 23.5, 23.5]`

`slice( array, i0, i1 )`

Returns a sub-sequence of values from a given array.

The semantics are like python array slicing, though both limits have to be specified: the output array contains the sequence of elements in the input array from `i0` (inclusive) to `i1` (exclusive). If a negative value is given in either case, it is added to the length of the input array, so that -1 indicates the last element of the input array. The indices are capped at 0 and the input array length respectively, so a large positive value may be used to indicate the end of the array. If the end index is less than or equal to the start index, a zero-length array is returned.

Note: This documents the double-precision version of the routine. Corresponding routines exist for other data types (`float`, `long`, `int`, `short`, `byte`, `String`, `Object`).

- Parameters:
  - `array` *(array of floating point)*: input array
  - `i0` *(integer)*: index of first element, inclusive (may be negative to count back from the end)
  - `i1` *(integer)*: index of the last element, exclusive (may be negative to count back from the end)
- Return value
  - *(array of floating point)*: array giving the sequence of elements specified by `i0` and `i1`
- Examples:
  - `slice(array(10,11,12,13), 0, 3) = [10, 11, 12]`
  - `slice(array(10,11,12,13), -2, 999) = [12, 13]`

`pick( array, indices, ... )`

Returns a selection of elements from a given array.

The output array consists of one element selected from the input array for each of the supplied index values. If a negative value is supplied for an index value, it is added to the input array length, so that -1 indicates the last element of the input array. If the input array is null, null is returned. If any of the index values is out of the range of the extent of the input array, an error results.

Note: This documents the double-precision version of the routine. Corresponding routines exist for other data types (`float`, `long`, `int`, `short`, `byte`, `String`, `Object`).

- Parameters:
  - `array` *(array of floating point)*: input array
  - `indices` *(integer, one or more)*: one or more index into the input array (may be negative to count back from the end)
- Return value
  - *(array of floating point)*: array giving the elements specified by `indices`
- Examples:
  - `pick(array(10,11,12,13), 0, 3) = [10, 13]`
  - `pick(array(10,11,12,13), -1, -2, -3) = [13, 12, 11]`

`arrayFunc( expr, inArray )`

Returns a floating-point array resulting from applying a given function expression element-by-element to an input array. The output array is the same length as the input array.

The supplied expression can use the variable "`x`" to refer to the corresponding element of the input array, and "`i`" to refer to its (zero-based) index. The various functions and operators from the expression language can all be used, but it is currently **not** possible to reference other table column values.

If there is an error in the expression, a blank value (not an array) will be returned.

- Parameters:
  - `expr` *(String)*: expression mapping input to output array values
  - `inArray` *(Object)*: input array
- Return value

  - *(array of floating point)*: floating point array with the same number of elements as `inArray`, or null for a bad `expr`
- Examples:
  - `arrayFunc("3*x",array(0,1,2,3,NaN)) = [0, 3, 6, 9, NaN]`
  - `arrayFunc("pow(2,i)+x", array(0.5,0.5,0.5,0.5)) = [1.5, 2.5, 4.5, 8.5]`

`intArrayFunc( expr, inArray )`

Returns an integer array resulting from applying a given function expression element-by-element to an input array. The output array is the same length as the input array.

The supplied expression can use the variable "`x`" to refer to the corresponding element of the input array, and "`i`" to refer to its (zero-based) index. The various functions and operators from the expression language can all be used, but it is currently **not** possible to reference other table column values.

If there is an error in the expression, a blank value (not an array) will be returned.

- Parameters:
  - `expr` *(String)*: expression mapping input to output array values
  - `inArray` *(Object)*: input array
- Return value
  - *(array of integer)*: floating point array with the same number of elements as `inArray`, or null for a bad `expr`
- Example:
  - `intArrayFunc("-x",sequence(5)) = [0, -1, -2, -3, -4]`

`indexOf( array, item )`

Returns the position in a supplied array at which a given item appears. The result is zero-based, so if the supplied `item` is the first entry in the `array`, the return value will be zero.

If the item does not appear in the array, -1 is returned. If it appears multiple times, the index of its first appearance is returned.

If `indexOf(array, item)==n`, then `array[n]` is equal to `item`.

Note: This documents the `Object` version of the routine. Corresponding routines exist for other data types (`double`, `float`, `long`, `int`, `short`).

- Parameters:
  - `array` *(array of Object)*: array which may contain the supplied item
  - `item` *(Object)*: entry to look for in the array
- Return value
  - *(integer)*: the index of `item` in `array`, or -1
- Examples:
  - `indexOf(stringArray("QSO", "BCG", "SNR"), "BCG") = 1`
  - `indexOf(stringArray("QSO", "BCG", "SNR"), "TLA") = -1`

`sequence( n )`

Returns an integer array of a given length with the values 0, 1, 2, ...

See also the `loop` functions, which provide similar functionality.

- Parameters:
  - `n` *(integer)*: length of array
- Return value
  - *(array of integer)*: `n`-element array, (0, 1, 2, ... `n`-1)
- Example:
  - `sequence(4) = (0, 1, 2, 3)`

`sequence( n, start, step )`

Returns a floating point array of a given length with values starting at a given value and increasing with a given increment.

See also the `loop` functions, which provide similar functionality.

- Parameters:
  - `n` *(integer)*: length of array
  - `start` *(floating point)*: value of first element
  - `step` *(floating point)*: increment to apply to each element
- Return value
  - *(array of floating point)*: `n`-element array, (start, start+step, start+2*step, ... start+(n-1)*step)
- Example:
  - `sequence(4, 100, 0.1) = (100.0, 100.1, 100.2, 100.3)`

`loop( start, end )`

Returns an integer array like the values taken in a for-loop with given start and end elements and a step of 1. The notional loop corresponds to:

```
for (int x = start; x < end; x++)
```

If you want a floating point array, or one with a non-unit step, you can use the three-parameter version of the `loop` function. See also the `sequence` functions.

- Parameters:
  - `start` *(integer)*: value for first element of output array
  - `end` *(integer)*: value one greater than last element of output array
- Return value
  - *(array of integer)*: array with `end-start` (or 0) elements `(start, start+1, start+2, ..., end-1)`
- Examples:
  - `loop(0, 5) = (0, 1, 2, 3, 4)`
  - `loop(5, 0) = ()`

`loop( start, end, step )`

Returns a floating point array like the values taken in a for-loop with given start, end, and step arguments. For a positive step, the notional loop corresponds to:

```
for (double x = start; x < end; x += step)
```

Note that numerical precision issues may result in the output array differing from its expected length by 1 (it is generally risky to rely on exact comparison of floating point values). If you want to be sure of the number of elements you can use the `sequence` function instead.

- Parameters:
  - `start` *(floating point)*: value for first element of output array
  - `end` *(floating point)*: first value beyond last element of output array
  - `step` *(floating point)*: increment between output array values; may not be zero
- Return value
  - *(array of floating point)*: array with approximately `(end-start)/step` (or 0) elements
- Examples:
  - `loop(10, 12, 0.5) = (10.0, 10.5, 11.0, 11.5)`
  - `loop(0, 10, 3) = (0., 3., 6., 9.)`
  - `loop(5, 0, -1) = (5., 4., 3., 2., 1.)`

`array( values, ... )`
Returns a floating point numeric array built from the given arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more array elements
- Return value
  - *(array of floating point)*: array

`intArray( values, ... )`
Returns an integer numeric array built from the given arguments.

- Parameters:
  - `values` *(integer, one or more)*: one or more array elements
- Return value
  - *(array of integer)*: array

`stringArray( values, ... )`
Returns a String array built from the given arguments.

- Parameters:
  - `values` *(String, one or more)*: one or more array elements
- Return value
  - *(array of String)*: array

B.1.3 Bits

Bit manipulation functions.

Note that for bitwise AND, OR, XOR of integer values etc you can use the java bitwise operators "`&`", "`|`", "`^`".

`hasBit( value, bitIndex )`

Determines whether a given integer has a certain bit set to 1.

- Parameters:
  - `value` *(long integer)*: integer whose bits are to be tested
  - `bitIndex` *(integer)*: index of bit to be tested in range 0..63, where 0 is the least significant bit
- Return value
  - *(boolean)*: true if bit is set; more or less equivalent to `(value & 1L<<bitIndex) != 0`
- Examples:
  - `hasBit(64, 6) = true`
  - `hasBit(63, 6) = false`

`bitCount( i )`

Returns the number of set bits in the 64-bit two's complement representation of the integer argument.

- Parameters:
  - `i` *(long integer)*: integer value
- Return value
  - *(integer)*: number of "1" bits in the binary representation of `i`
- Examples:
  - `bitCount(64) = 1`
  - `bitCount(3) = 2`

`toBinary( value )`

Converts the integer argument to a binary string consisting only of 1s and 0s.

- Parameters:
  - `value` *(long integer)*: integer value
- Return value
  - *(String)*: binary representation of `value`
- Examples:
  - `toBinary(42) = "101010"`
  - `toBinary(255^7) = "11111000"`

`fromBinary( binVal )`

Converts a string representing a binary number to its integer value.

- Parameters:
  - `binVal` *(String)*: binary representation of value
- Return value
  - *(integer)*: integer value represented by binary string `binVal`
- Example:

- `fromBinary("101010") = 42`

B.1.4 Conversions

Functions for converting between strings and numeric values.

`toString( fpVal )`
Turns a numeric value into a string.

- Parameters:
  - `fpVal` *(floating point)*: floating point numeric value
- Return value
  - *(String)*: a string representation of `fpVal`

`toString( intVal )`
Turns an integer numeric value into a string.

- Parameters:
  - `intVal` *(long integer)*: integer numeric value
- Return value
  - *(String)*: a string representation of `intVal`

`toString( charVal )`
Turns a single character value into a string.

- Parameters:
  - `charVal` *(char)*: character numeric value
- Return value
  - *(String)*: a string representation of `charVal`

`toString( byteVal )`
Turns a byte value into a string.

- Parameters:
  - `byteVal` *(byte)*: byte numeric value
- Return value
  - *(String)*: a string representation of `byteVal`

`toString( booleanVal )`
Turns a boolean value into a string.

- Parameters:
  - `booleanVal` *(boolean)*: boolean value (true or false)
- Return value
  - *(String)*: a string representation of `booleanVal` ("`true`" or "`false`")

`toString( objVal )`

Turns any object value into a string. As applied to existing string values this isn't really useful, but it means that you can apply `toString` to any object value without knowing its type and get a useful return from it.

- Parameters:
  - `objVal` *(Object)*: non-primitive value
- Return value
  - *(String)*: a string representation of `objVal`

`parseByte( str )`

Attempts to interpret a string as a byte (8-bit signed integer) value. If the input string can't be interpreted in this way, a blank value will result.

- Parameters:
  - `str` *(String)*: string containing numeric representation
- Return value
  - *(byte)*: byte value of `str`

`parseShort( str )`

Attempts to interpret a string as a short (16-bit signed integer) value. If the input string can't be interpreted in this way, a blank value will result.

- Parameters:
  - `str` *(String)*: string containing numeric representation
- Return value
  - *(short integer)*: byte value of `str`

`parseInt( str )`

Attempts to interpret a string as an int (32-bit signed integer) value. If the input string can't be interpreted in this way, a blank value will result.

- Parameters:
  - `str` *(String)*: string containing numeric representation
- Return value
  - *(integer)*: byte value of `str`

`parseLong( str )`

Attempts to interpret a string as a long (64-bit signed integer) value. If the input string can't be interpreted in this way, a blank value will result.

- Parameters:
  - `str` *(String)*: string containing numeric representation
- Return value
  - *(long integer)*: byte value of `str`

`parseFloat( str )`

Attempts to interpret a string as a float (32-bit floating point) value. If the input string can't be interpreted in this way, a blank value will result.

- Parameters:
  - `str` *(String)*: string containing numeric representation
- Return value
  - *(floating point)*: byte value of `str`

`parseDouble( str )`

Attempts to interpret a string as a double (64-bit signed integer) value. If the input string can't be interpreted in this way, a blank value will result.

- Parameters:
  - `str` *(String)*: string containing numeric representation
- Return value
  - *(floating point)*: byte value of `str`

`parseBigInteger( str )`

Attempts to interpret a string as a "BigInteger" value. This can be used for working with string representations of integers that can't be stored as an unsigned 64-bit value.

The result is a `BigInteger` object, which can't be used in normal numeric expressions, but has a number of methods defined on it for comparison, arithmetic, bit manipulation etc. See the java.math.BigInteger (https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html) javadocs for details.

- Parameters:
  - `str` *(String)*: string containing numeric representation
- Return value
  - *(BigInteger)*: BigInteger value of `str`
- Examples:
  - `parseBigInteger("-20000000000000000023").doubleValue() = -2e19`
  - `parseBigInteger("18446744073709551616").testBit(64) = true`

`parseBigDecimal( str )`

Attempts to interpret a string as a "BigDecimal" value. This can be used for working with string representations of non-integer values that require more precision or range than is possible in a 64-bit IEEE-754 double precision variable.

The result is a `BigDecimal` object, which can't be used in normal numeric expressions, but has a number of methods defined on it for comparison, arithmetic, bit manipulation etc. See the java.math.BigDecimal (https://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html) javadocs for details.

- Parameters:
  - `str` *(String)*: string contining numeric representation
- Return value
  - *(BigDecimal)*: BigDecimal value of `str`

- Example:
  - `parseBigDecimal("101").compareTo(parseBigDecimal("102")) = -1`

`parseInts( str )`

Attempts to interpret a string as an array of integer values. An ad-hoc algorithm is used that tries to extract a list of integers from a string; a comma- or space-separated list of integer values will work, and other formats may or may not.

The details of this function's behaviour may change in future releases.

- Parameters:
  - `str` *(String)*: string containing a list of integer values
- Return value
  - *(array of integer)*: array of integer values
- Examples:
  - `parseInts("9 8 -23") = [9, 8, -23]`
  - `parseInts("tiddly-pom") = []`

`parseDoubles( str )`

Attempts to interpret a string as an array of floating point values. An ad-hoc algorithm is used that tries to extract a list of numeric values from a string; a comma- or space-separated list of floating point values will work, and other formats may or may not.

This function can be used as a hacky way to extract the numeric values from an STC-S (for instance ObsCore/EPNcore `s_region`) string.

The details of this function's behaviour may change in future releases.

- Parameters:
  - `str` *(String)*: string containing a list of floating point values
- Return value
  - *(array of floating point)*: array of floating point values
- Examples:
  - `parseDoubles("1.3, 99e1, NaN, -23") = [1.3, 990.0, NaN, -23.0]`
  - `parseDoubles("Polygon ICRS 0.8 2.1 9.0 2.1 6.2 8.6") = [0.8, 2.1, 9.0, 2.1, 6.2, 8.6]`
  - `parseDoubles("La la la") = []`

`toByte( value )`

Attempts to convert the numeric argument to a byte (8-bit signed integer) result. If it is out of range, a blank value will result.

- Parameters:
  - `value` *(floating point)*: numeric value for conversion
- Return value
  - *(byte)*: `value` converted to type byte

`toShort( value )`

Attempts to convert the numeric argument to a short (16-bit signed integer) result. If it is out of range, a blank value will result.

- Parameters:
  - `value` *(floating point)*: numeric value for conversion
- Return value
  - *(short integer)*: `value` converted to type short

`toInteger( value )`
Attempts to convert the numeric argument to an int (32-bit signed integer) result. If it is out of range, a blank value will result.

- Parameters:
  - `value` *(floating point)*: numeric value for conversion
- Return value
  - *(integer)*: `value` converted to type int

`toLong( value )`
Attempts to convert the numeric argument to a long (64-bit signed integer) result. If it is out of range, a blank value will result.

- Parameters:
  - `value` *(floating point)*: numeric value for conversion
- Return value
  - *(long integer)*: `value` converted to type long

`toFloat( value )`
Attempts to convert the numeric argument to a float (32-bit floating point) result. If it is out of range, a blank value will result.

- Parameters:
  - `value` *(floating point)*: numeric value for conversion
- Return value
  - *(floating point)*: `value` converted to type float

`toDouble( value )`
Converts the numeric argument to a double (64-bit signed integer) result.

- Parameters:
  - `value` *(floating point)*: numeric value for conversion
- Return value
  - *(floating point)*: `value` converted to type double

`toHex( value )`
Converts the integer argument to hexadecimal form.

- Parameters:
  - `value` *(long integer)*: integer value

- Return value
  - *(String)*: hexadecimal representation of `value`
- Example:
  - `toHex(42) = "2a"`

`fromHex( hexVal )`

Converts a string representing a hexadecimal number to its integer value.

- Parameters:
  - `hexVal` *(String)*: hexadecimal representation of value
- Return value
  - *(integer)*: integer value represented by `hexVal`
- Example:
  - `fromHex("2a") = 42`

B.1.5 CoordsDegrees

Functions for angle transformations and manipulations, with angles generally in degrees. In particular, methods for translating between degrees and HH:MM:SS.S or DDD:MM:SS.S type sexagesimal representations are provided.

`degreesToDms( deg )`

Converts an angle in degrees to a formatted degrees:minutes:seconds string. No fractional part of the seconds field is given.

- Parameters:
  - `deg` *(floating point)*: angle in degrees
- Return value
  - *(String)*: DMS-format string representing `deg`

`degreesToDms( deg, secFig )`

Converts an angle in degrees to a formatted degrees:minutes:seconds string with a given number of decimal places in the seconds field.

- Parameters:
  - `deg` *(floating point)*: angle in degrees
  - `secFig` *(integer)*: number of decimal places in the seconds field
- Return value
  - *(String)*: DMS-format string representing `deg`

`degreesToHms( deg )`

Converts an angle in degrees to a formatted hours:minutes:seconds string. No fractional part of the seconds field is given.

- Parameters:

    - `deg` *(floating point)*: angle in degrees
- Return value
    - *(String)*: HMS-format string representing `deg`

`degreesToHms( deg, secFig )`

Converts an angle in degrees to a formatted hours:minutes:seconds string with a given number of decimal places in the seconds field.

- Parameters:
    - `deg` *(floating point)*: angle in degrees
    - `secFig` *(integer)*: number of decimal places in the seconds field
- Return value
    - *(String)*: HMS-format string representing `deg`

`dmsToDegrees( dms )`

Converts a formatted degrees:minutes:seconds string to an angle in degrees. Delimiters may be colon, space, characters `dm[s]`, or some others. Additional spaces and leading +/- are permitted. The :seconds part is optional.

- Parameters:
    - `dms` *(String)*: formatted DMS string
- Return value
    - *(floating point)*: angle in degrees specified by `dms`

`hmsToDegrees( hms )`

Converts a formatted hours:minutes:seconds string to an angle in degrees. Delimiters may be colon, space, characters `hm[s]`, or some others. Additional spaces and leading +/- are permitted. The :seconds part is optional.

- Parameters:
    - `hms` *(String)*: formatted HMS string
- Return value
    - *(floating point)*: angle in degrees specified by `hms`

`dmsToDegrees( deg, min, sec )`

Converts degrees, minutes, seconds to an angle in degrees.

In conversions of this type, one has to be careful to get the sign right in converting angles which are between 0 and -1 degrees. This routine uses the sign bit of the `deg` argument, taking care to distinguish between +0 and -0 (their internal representations are different for floating point values). It is illegal for the `min` or `sec` arguments to be negative.

- Parameters:
    - `deg` *(floating point)*: degrees part of angle
    - `min` *(floating point)*: minutes part of angle
    - `sec` *(floating point)*: seconds part of angle
- Return value
    - *(floating point)*: specified angle in degrees

`hmsToDegrees( hour, min, sec )`

Converts hours, minutes, seconds to an angle in degrees.

In conversions of this type, one has to be careful to get the sign right in converting angles which are between 0 and -1 hours. This routine uses the sign bit of the `hour` argument, taking care to distinguish between +0 and -0 (their internal representations are different for floating point values).

- Parameters:
  - `hour` *(floating point)*: degrees part of angle
  - `min` *(floating point)*: minutes part of angle
  - `sec` *(floating point)*: seconds part of angle
- Return value
  - *(floating point)*: specified angle in degrees

`skyDistanceDegrees( ra1, dec1, ra2, dec2 )`

Calculates the separation (distance around a great circle) of two points on the sky in degrees.

- Parameters:
  - `ra1` *(floating point)*: right ascension of point 1 in degrees
  - `dec1` *(floating point)*: declination of point 1 in degrees
  - `ra2` *(floating point)*: right ascension of point 2 in degrees
  - `dec2` *(floating point)*: declination of point 2 in degrees
- Return value
  - *(floating point)*: angular distance between point 1 and point 2 in degrees

`posAngDegrees( ra1, dec1, ra2, dec2 )`

Calculates the position angle between two points on the sky in degrees. The result is in the range +/-180. If point 2 is due east of point 1, the result is +90. Zero is returned if the points are coincident.

- Parameters:
  - `ra1` *(floating point)*: right ascension of point 1 in degrees
  - `dec1` *(floating point)*: declination of point 1 in degrees
  - `ra2` *(floating point)*: right ascension of point 2 in degrees
  - `dec2` *(floating point)*: declination of point 2 in degrees
- Return value
  - *(floating point)*: bearing in degrees of point 2 from point 1.

`polarDistanceDegrees( ra1, dec1, radius1, ra2, dec2, radius2 )`

Calculates the distance in three dimensional space between two points specified in spherical polar coordinates.

- Parameters:
  - `ra1` *(floating point)*: right ascension of point 1 in degrees
  - `dec1` *(floating point)*: declination of point1 in degrees
  - `radius1` *(floating point)*: distance from origin of point1
  - `ra2` *(floating point)*: right ascension of point 2 in degrees
  - `dec2` *(floating point)*: declination of point2 in degrees

    - `radius2` *(floating point)*: distance from origin of point2

- Return value

    - *(floating point)*: the linear distance between point1 and point2; units are as for `radius1` and `radius2`

B.1.6 CoordsRadians

Functions for angle transformations and manipulations, based on radians rather than degrees. In particular, methods for translating between radians and HH:MM:SS.S or DDD:MM:SS.S type sexagesimal representations are provided.

`radiansToDms( rad )`

Converts an angle in radians to a formatted degrees:minutes:seconds string. No fractional part of the seconds field is given.

- Parameters:

    - `rad` *(floating point)*: angle in radians

- Return value

    - *(String)*: DMS-format string representing `rad`

`radiansToDms( rad, secFig )`

Converts an angle in radians to a formatted degrees:minutes:seconds string with a given number of decimal places in the seconds field.

- Parameters:

    - `rad` *(floating point)*: angle in radians
    - `secFig` *(integer)*: number of decimal places in the seconds field

- Return value

    - *(String)*: DMS-format string representing `rad`

`radiansToHms( rad )`

Converts an angle in radians to a formatted hours:minutes:seconds string. No fractional part of the seconds field is given.

- Parameters:

    - `rad` *(floating point)*: angle in radians

- Return value

    - *(String)*: HMS-format string representing `rad`

`radiansToHms( rad, secFig )`

Converts an angle in radians to a formatted hours:minutes:seconds string with a given number of decimal places in the seconds field.

- Parameters:

    - `rad` *(floating point)*: angle in radians
    - `secFig` *(integer)*: number of decimal places in the seconds field

- Return value
  - *(String)*: HMS-format string representing `rad`

`dmsToRadians( dms )`

Converts a formatted degrees:minutes:seconds string to an angle in radians. Delimiters may be colon, space, characters `dm[s]`, or some others. Additional spaces and leading +/- are permitted. The :seconds part is optional.

- Parameters:
  - `dms` *(String)*: formatted DMS string
- Return value
  - *(floating point)*: angle in radians specified by `dms`

`hmsToRadians( hms )`

Converts a formatted hours:minutes:seconds string to an angle in radians. Delimiters may be colon, space, characters `hm[s]`, or some others. Additional spaces and leading +/- are permitted. The :seconds part is optional.

- Parameters:
  - `hms` *(String)*: formatted HMS string
- Return value
  - *(floating point)*: angle in radians specified by `hms`

`dmsToRadians( deg, min, sec )`

Converts degrees, minutes, seconds to an angle in radians.

In conversions of this type, one has to be careful to get the sign right in converting angles which are between 0 and -1 degrees. This routine uses the sign bit of the `deg` argument, taking care to distinguish between +0 and -0 (their internal representations are different for floating point values). It is illegal for the `min` or `sec` arguments to be negative.

- Parameters:
  - `deg` *(floating point)*: degrees part of angle
  - `min` *(floating point)*: minutes part of angle
  - `sec` *(floating point)*: seconds part of angle
- Return value
  - *(floating point)*: specified angle in radians

`hmsToRadians( hour, min, sec )`

Converts hours, minutes, seconds to an angle in radians.

In conversions of this type, one has to be careful to get the sign right in converting angles which are between 0 and -1 hours. This routine uses the sign bit of the `hour` argument, taking care to distinguish between +0 and -0 (their internal representations are different for floating point values).

- Parameters:
  - `hour` *(floating point)*: degrees part of angle
  - `min` *(floating point)*: minutes part of angle
  - `sec` *(floating point)*: seconds part of angle

- Return value
  - *(floating point)*: specified angle in radians

`skyDistanceRadians( ra1, dec1, ra2, dec2 )`

Calculates the separation (distance around a great circle) of two points on the sky in radians.

- Parameters:
  - `ra1` *(floating point)*: right ascension of point 1 in radians
  - `dec1` *(floating point)*: declination of point 1 in radians
  - `ra2` *(floating point)*: right ascension of point 2 in radians
  - `dec2` *(floating point)*: declination of point 2 in radians
- Return value
  - *(floating point)*: angular distance between point 1 and point 2 in radians

`posAngRadians( ra1, dec1, ra2, dec2 )`

Calculates the position angle between two points on the sky in radians. The result is in the range +/-pi. If point 2 is due east of point 1, the result is +pi/2. Zero is returned if the points are coincident.

- Parameters:
  - `ra1` *(floating point)*: right ascension of point 1 in radians
  - `dec1` *(floating point)*: declination of point 1 in radians
  - `ra2` *(floating point)*: right ascension of point 2 in radians
  - `dec2` *(floating point)*: declination of point 2 in radians
- Return value
  - *(floating point)*: bearing in radians of point 2 from point 1

`polarDistanceRadians( ra1, dec1, radius1, ra2, dec2, radius2 )`

Calculates the distance in three dimensional space between two points specified in spherical polar coordinates.

- Parameters:
  - `ra1` *(floating point)*: right ascension of point 1 in radians
  - `dec1` *(floating point)*: declination of point1 in radians
  - `radius1` *(floating point)*: distance from origin of point1
  - `ra2` *(floating point)*: right ascension of point 2 in radians
  - `dec2` *(floating point)*: declination of point2 in radians
  - `radius2` *(floating point)*: distance from origin of point2
- Return value
  - *(floating point)*: the linear distance between point1 and point2; units are as for `radius1` and `radius2`

`hoursToRadians( hours )`

Converts hours to radians.

- Parameters:
  - `hours` *(floating point)*: angle in hours
- Return value

  - *(floating point)*: angle in radians

`degreesToRadians( deg )`

Converts degrees to radians.

- Parameters:
  - `deg` *(floating point)*: angle in degrees
- Return value
  - *(floating point)*: angle in radians

`radiansToDegrees( rad )`

Converts radians to degrees.

- Parameters:
  - `rad` *(floating point)*: angle in radians
- Return value
  - *(floating point)*: angle in degrees

`raFK4toFK5radians( raFK4, decFK4 )`

Converts a B1950.0 FK4 position to J2000.0 FK5 at an epoch of B1950.0 yielding Right Ascension. This assumes zero proper motion in the FK5 frame.

- Parameters:
  - `raFK4` *(floating point)*: right ascension in B1950.0 FK4 system (radians)
  - `decFK4` *(floating point)*: declination in B1950.0 FK4 system (radians)
- Return value
  - *(floating point)*: right ascension in J2000.0 FK5 system (radians)

`decFK4toFK5radians( raFK4, decFK4 )`

Converts a B1950.0 FK4 position to J2000.0 FK5 at an epoch of B1950.0 yielding Declination This assumes zero proper motion in the FK5 frame.

- Parameters:
  - `raFK4` *(floating point)*: right ascension in B1950.0 FK4 system (radians)
  - `decFK4` *(floating point)*: declination in B1950.0 FK4 system (radians)
- Return value
  - *(floating point)*: declination in J2000.0 FK5 system (radians)

`raFK5toFK4radians( raFK5, decFK5 )`

Converts a J2000.0 FK5 position to B1950.0 FK4 at an epoch of B1950.0 yielding Declination. This assumes zero proper motion, parallax and radial velocity in the FK5 frame.

- Parameters:
  - `raFK5` *(floating point)*: right ascension in J2000.0 FK5 system (radians)
  - `decFK5` *(floating point)*: declination in J2000.0 FK5 system (radians)
- Return value
  - *(floating point)*: right ascension in the FK4 system (radians)

decFK5toFK4radians(raFK5, decFK5)

Converts a J2000.0 FK5 position to B1950.0 FK4 at an epoch of B1950.0 yielding Declination. This assumes zero proper motion, parallax and radial velocity in the FK5 frame.

- Parameters:
  - `raFK5` *(floating point)*: right ascension in J2000.0 FK5 system (radians)
  - `decFK5` *(floating point)*: declination in J2000.0 FK5 system (radians)
- Return value
  - *(floating point)*: right ascension in the FK4 system (radians)

raFK4toFK5Radians(raFK4, decFK4, bepoch)

Converts a B1950.0 FK4 position to J2000.0 FK5 yielding Right Ascension. This assumes zero proper motion in the FK5 frame. The `bepoch` parameter is the epoch at which the position in the FK4 frame was determined.

- Parameters:
  - `raFK4` *(floating point)*: right ascension in B1950.0 FK4 system (radians)
  - `decFK4` *(floating point)*: declination in B1950.0 FK4 system (radians)
  - `bepoch` *(floating point)*: Besselian epoch
- Return value
  - *(floating point)*: right ascension in J2000.0 FK5 system (radians)

decFK4toFK5Radians(raFK4, decFK4, bepoch)

Converts a B1950.0 FK4 position to J2000.0 FK5 yielding Declination. This assumes zero proper motion in the FK5 frame. The `bepoch` parameter is the epoch at which the position in the FK4 frame was determined.

- Parameters:
  - `raFK4` *(floating point)*: right ascension in B1950.0 FK4 system (radians)
  - `decFK4` *(floating point)*: declination in B1950.0 FK4 system (radians)
  - `bepoch` *(floating point)*: Besselian epoch
- Return value
  - *(floating point)*: declination in J2000.0 FK5 system (radians)

raFK5toFK4Radians(raFK5, decFK5, bepoch)

Converts a J2000.0 FK5 position to B1950.0 FK4 yielding Declination. This assumes zero proper motion, parallax and radial velocity in the FK5 frame.

- Parameters:
  - `raFK5` *(floating point)*: right ascension in J2000.0 FK5 system (radians)
  - `decFK5` *(floating point)*: declination in J2000.0 FK5 system (radians)
  - `bepoch` *(floating point)*: Besselian epoch
- Return value
  - *(floating point)*: right ascension in the FK4 system (radians)

decFK5toFK4Radians(raFK5, decFK5, bepoch)

Converts a J2000.0 FK5 position to B1950.0 FK4 yielding Declination. This assumes zero proper motion, parallax and radial velocity in the FK5 frame.

- Parameters:
  - `raFK5` *(floating point)*: right ascension in J2000.0 FK5 system (radians)
  - `decFK5` *(floating point)*: declination in J2000.0 FK5 system (radians)
  - `bepoch` *(floating point)*: Besselian epoch
- Return value
  - *(floating point)*: right ascension in the FK4 system (radians)

`DEGREE_RADIANS`
The size of one degree in radians.

`HOUR_RADIANS`
The size of one hour of right ascension in radians.

`ARC_MINUTE_RADIANS`
The size of one arcminute in radians.

`ARC_SECOND_RADIANS`
The size of one arcsecond in radians.

B.1.7 Coverage

Functions related to coverage and footprints.

One coverage standard is **Multi-Order Coverage maps**, described at http://www.ivoa.net/Documents/MOC/ (http://www.ivoa.net/Documents/MOC/). MOC positions are always defined in ICRS equatorial coordinates.

MOCs may be specified using a string argument of the functions in one of the following ways:

- The filename of a MOC FITS file
- The URL of a MOC FITS file
- The identifier of a VizieR table, for instance "`V/139/sdss9`" (SDSS DR9)
- An ASCII MOC string, for instance "`1/1 2 4 2/12-14 21 23 25 8/`"

A list of all the MOCs available from VizieR can currently be found at http://alasky.u-strasbg.fr/footprints/tables/vizier/ (http://alasky.u-strasbg.fr/footprints/tables/vizier/). You can search for VizieR table identifiers from the VizieR web page (http://vizier.u-strasbg.fr/ (http://vizier.u-strasbg.fr/)); note you must use the *table* identifier (like "`V/139/sdss9`") and not the *catalogue* identifier (like "`V/139`").

`inMoc( moc, ra, dec )`
Indicates whether a given sky position falls strictly within a given MOC (Multi-Order Coverage map). If the given `moc` value does not represent a MOC (for instance no file exists or the file/string is not in MOC format) a warning will be issued the first time it's referenced, and the result will be false.

- Parameters:
  - `moc` *(String)*: a MOC identifier; a filename, a URL, a VizieR table name, or an ASCII

MOC string
- `ra` *(floating point)*: ICRS right ascension in degrees
- `dec` *(floating point)*: ICRS declination in degrees

- Return value
  - *(boolean)*: true iff the given position falls within the given MOC

`nearMoc( moc, ra, dec, distanceDeg )`

Indicates whether a given sky position either falls within, or is within a certain distance of the edge of, a given MOC (Multi-Order Coverage map). If the given `moc` value does not represent a MOC (for instance no file exists or the file/string is not in MOC format) a warning will be issued the first time it's referenced, and the result will be false.

- Parameters:
  - `moc` *(String)*: a MOC identifier; a filename, a URL, a VizieR table name, or an ASCII MOC string
  - `ra` *(floating point)*: ICRS right ascension in degrees
  - `dec` *(floating point)*: ICRS declination in degrees
  - `distanceDeg` *(floating point)*: permitted distance from MOC boundary in degrees
- Return value
  - *(boolean)*: true iff the given position is within `distance` degrees of the given MOC

`mocSkyProportion( moc )`

Returns the proportion of the sky covered by a given MOC.

If the given `moc` value does not represent a MOC (for instance no file exists or the file/string is not in MOC format) a warning will be issued the first time it's referenced, and the result will be NaN.

- Parameters:
  - `moc` *(String)*: a MOC identifier; a filename, a URL, a VizieR table name, or an ASCII MOC string
- Return value
  - *(floating point)*: a fractional value in the range 0..1

`mocTileCount( moc )`

Returns the number of unique tiles within a given MOC.

If the given `moc` value does not represent a MOC (for instance no file exists or the file/string is not in MOC format) a warning will be issued the first time it's referenced, and the result will be 0.

- Parameters:
  - `moc` *(String)*: a MOC identifier; a filename, a URL, a VizieR table name, or an ASCII MOC string
- Return value
  - *(long integer)*: number of tiles in the MOC

`mocUniq( order, index )`

Converts a HEALPix order and and tile index into a UNIQ-encoded integer as used in MOC encoding. The result is `index + 4**(1+order)`.

If the order or index are out of bounds, behaviour is undefined.

- Parameters:
  - `order` *(integer)*: HEALPix order, in range 0..29
  - `index` *(long integer)*: tile index within the given level
- Return value
  - *(long integer)*: uniq-encoded value

`uniqToOrder( uniq )`

Extracts the HEALPix order from a UNIQ-encoded integer as used in MOC encoding.

If the supplied value is not a legal UNIQ integer, behaviour is undefined.

- Parameters:
  - `uniq` *(long integer)*: uniq-encoded value
- Return value
  - *(integer)*: HEALPix order

`uniqToIndex( uniq )`

Extracts the HEALPix pixel index from a UNIQ-encoded integer as used in MOC encoding.

If the supplied value is not a legal UNIQ integer, behaviour is undefined.

- Parameters:
  - `uniq` *(long integer)*: uniq-encoded value
- Return value
  - *(long integer)*: pixel index

`SPHERE_STERADIAN`

The number of steradians on the sphere, 4 PI.

`SPHERE_SQDEG`

The number of square degrees on the sphere, approx 41253.

B.1.8 Distances

Functions for converting between different measures of cosmological distance.

The following parameters are used:

- **z**: redshift
- **H0**: Hubble constant in km/sec/Mpc (example value ~70)
- **omegaM**: Density ratio of the universe (example value 0.3)
- **omegaLambda**: Normalised cosmological constant (example value 0.7)

For a flat universe, `omegaM+omegaLambda=1`

The terms and formulae used here are taken from the paper by D.W.Hogg, *Distance measures in* *cosmology*, astro-ph/9905116 (http://arxiv.org/abs/astro-ph/9905116) v4 (2000).

`MpcToM( distMpc )`

Converts from MegaParsecs to metres.

- Parameters:
  - `distMpc` *(floating point)*: distance in Mpc
- Return value
  - *(floating point)*: distance in m

`mToMpc( distM )`

Converts from metres to MegaParsecs.

- Parameters:
  - `distM` *(floating point)*: distance in m
- Return value
  - *(floating point)*: distance in Mpc

`zToDist( z )`

Quick and dirty function for converting from redshift to distance.

Warning: this makes some reasonable assumptions about the cosmology and returns the luminosity distance. It is only intended for approximate use. If you care about the details, use one of the more specific functions here.

- Parameters:
  - `z` *(floating point)*: redshift
- Return value
  - *(floating point)*: some distance measure in Mpc

`zToAge( z )`

Quick and dirty function for converting from redshift to time.

Warning: this makes some reasonable assumptions about the cosmology. It is only intended for approximate use. If you care about the details use one of the more specific functions here.

- Parameters:
  - `z` *(floating point)*: redshift
- Return value
  - *(floating point)*: 'age' of photons from redshift `z` in Gyr

`comovingDistanceL( z, H0, omegaM, omegaLambda )`

Line-of-sight comoving distance.

- Parameters:
  - `z` *(floating point)*: redshift
  - `H0` *(floating point)*: Hubble constant in km/sec/Mpc
  - `omegaM` *(floating point)*: density ratio of the universe
  - `omegaLambda` *(floating point)*: normalised cosmological constant

- Return value
  - *(floating point)*: line-of-sight comoving distance in Mpc

`comovingDistanceT( z, H0, omegaM, omegaLambda )`

Transverse comoving distance.

- Parameters:
  - `z` *(floating point)*: redshift
  - `H0` *(floating point)*: Hubble constant in km/sec/Mpc
  - `omegaM` *(floating point)*: density ratio of the universe
  - `omegaLambda` *(floating point)*: normalised cosmological constant
- Return value
  - *(floating point)*: transverse comoving distance in Mpc

`angularDiameterDistance( z, H0, omegaM, omegaLambda )`

Angular diameter distance.

- Parameters:
  - `z` *(floating point)*: redshift
  - `H0` *(floating point)*: Hubble constant in km/sec/Mpc
  - `omegaM` *(floating point)*: density ratio of the universe
  - `omegaLambda` *(floating point)*: normalised cosmological constant
- Return value
  - *(floating point)*: angular diameter distance in Mpc

`luminosityDistance( z, H0, omegaM, omegaLambda )`

Luminosity distance.

- Parameters:
  - `z` *(floating point)*: redshift
  - `H0` *(floating point)*: Hubble constant in km/sec/Mpc
  - `omegaM` *(floating point)*: density ratio of the universe
  - `omegaLambda` *(floating point)*: normalised cosmological constant
- Return value
  - *(floating point)*: luminosity distance in Mpc

`lookbackTime( z, H0, omegaM, omegaLambda )`

Lookback time. This returns the difference between the age of the universe at time of observation (now) and the age of the universe at the time when photons of redshift `z` were emitted.

- Parameters:
  - `z` *(floating point)*: redshift
  - `H0` *(floating point)*: Hubble constant in km/sec/Mpc
  - `omegaM` *(floating point)*: density ratio of the universe
  - `omegaLambda` *(floating point)*: normalised cosmological constant
- Return value
  - *(floating point)*: lookback time in Gyr

comovingVolume(z, H0, omegaM, omegaLambda)

Comoving volume. This returns the all-sky total comoving volume out to a given redshift `z`.

- Parameters:
  - `z` *(floating point)*: redshift
  - `H0` *(floating point)*: Hubble constant in km/sec/Mpc
  - `omegaM` *(floating point)*: density ratio of the universe
  - `omegaLambda` *(floating point)*: normalised cosmological constant
- Return value
  - *(floating point)*: comoving volume in Gpc^3

SPEED_OF_LIGHT

Speed of light in m/s.

METRE_PER_PARSEC

Number of metres in a parsec.

SEC_PER_YEAR

Number of seconds in a year.

B.1.9 Fluxes

Functions for conversion between flux and magnitude values. Functions are provided for conversion between flux in Janskys and AB magnitudes.

Some constants for approximate conversions between different magnitude scales are also provided:

- Constants `JOHNSON_AB_*`, for Johnson <-> AB magnitude conversions, from Frei and Gunn, Astronomical Journal *108*, 1476 (1994), Table 2 (1994AJ....108.1476F (http://adsabs.harvard.edu/abs/1994AJ....108.1476F)).
- Constants `VEGA_AB_*`, for Vega <-> AB magnitude conversions, from Blanton et al., Astronomical Journal *129*, 2562 (2005), Eqs. (5) (2005AJ....129.2562B (http://adsabs.harvard.edu/abs/2005AJ....129.2562B)).

abToJansky(magAB)

Converts AB magnitude to flux in Jansky.

$F/Jy=10^{(23-(AB+48.6)/2.5)}$

- Parameters:
  - `magAB` *(floating point)*: AB magnitude value
- Return value
  - *(floating point)*: equivalent flux in Jansky

janskyToAb(fluxJansky)

Converts flux in Jansky to AB magnitude.

$AB=2.5*(23-\log_{10}(F/Jy))-48.6$

- Parameters:
  - `fluxJansky` *(floating point)*: flux in Jansky
- Return value
  - *(floating point)*: equivalent AB magnitude

`luminosityToFlux( lumin, dist )`

Converts luminosity to flux given a luminosity distance.

F=lumin/(4 x Pi x dist^2)

- Parameters:
  - `lumin` *(floating point)*: luminosity
  - `dist` *(floating point)*: luminosity distance
- Return value
  - *(floating point)*: equivalent flux

`fluxToLuminosity( flux, dist )`

Converts flux to luminosity given a luminosity distance.

lumin=(4 x Pi x dist^2) F

- Parameters:
  - `flux` *(floating point)*: flux
  - `dist` *(floating point)*: luminosity distance
- Return value
  - *(floating point)*: equivalent luminosity

`JOHNSON_AB_V`

Approximate offset between Johnson and AB magnitudes in V band. V_J~=V_{AB}+`JOHNSON_AB_V`.

`JOHNSON_AB_B`

Approximate offset between Johnson and AB magnitudes in B band. B_J~=B_{AB}+`JOHNSON_AB_B`.

`JOHNSON_AB_Bj`

Approximate offset between Johnson and AB magnitudes in Bj band. Bj_J~=Bj_{AB}+`JOHNSON_AB_Bj`.

`JOHNSON_AB_R`

Approximate offset between Johnson and AB magnitudes in R band. R_J~=R_{AB}+`JOHNSON_AB_R`.

`JOHNSON_AB_I`

Approximate offset between Johnson and AB magnitudes in I band. I_J~=I_{AB}+`JOHNSON_AB_I`.

`JOHNSON_AB_g`

Approximate offset between Johnson and AB magnitudes in g band. g_J~=g_{AB}+`JOHNSON_AB_g`.

`JOHNSON_AB_r`

Approximate offset between Johnson and AB magnitudes in r band. r_J~=r_{AB}+`JOHNSON_AB_r`.

`JOHNSON_AB_i`

Approximate offset between Johnson and AB magnitudes in i band. i_J~=i_{AB}+`JOHNSON_AB_i`.

`JOHNSON_AB_Rc`

Approximate offset between Johnson and AB magnitudes in Rc band. Rc_J~=Rc_{AB}+`JOHNSON_AB_Rc`.

`JOHNSON_AB_Ic`

Approximate offset between Johnson and AB magnitudes in Ic band. Ic_J~=Ic_{AB}+`JOHNSON_AB_Ic`.

`JOHNSON_AB_uPrime`

Offset between Johnson and AB magnitudes in u' band (zero). u'_J=u'_{AB}+`JOHNSON_AB_uPrime`=u'_{AB}.

`JOHNSON_AB_gPrime`

Offset between Johnson and AB magnitudes in g' band (zero). g'_J=g'_{AB}+`JOHNSON_AB_gPrime`=g'_{AB}.

`JOHNSON_AB_rPrime`

Offset between Johnson and AB magnitudes in r' band (zero). r'_J=r'_{AB}+`JOHNSON_AB_rPrime`=r'AB.

`JOHNSON_AB_iPrime`

Offset between Johnson and AB magnitudes in i' band (zero). i'_J=i'_{AB}+`JOHNSON_AB_iPrime`=i'AB.

`JOHNSON_AB_zPrime`

Offset between Johnson and AB magnitudes in z' band (zero). z'_J=z'_{AB}+`JOHNSON_AB_zPrime`=z'_{AB}.

`VEGA_AB_J`

Approximate offset between Vega (as in 2MASS) and AB magnitudes in J band. J_{Vega}~=J_{AB}+`VEGA_AB_J`.

`VEGA_AB_H`

Approximate offset between Vega (as in 2MASS) and AB magnitudes in H band. H_{Vega}~=HAB+`VEGA_AB_H`.

`VEGA_AB_K`

Approximate offset between Vega (as in 2MASS) and AB magnitudes in K band. K_{Vega}~=KAB+VEGA_AB_K.

B.1.10 Formats

Functions for formatting numeric values.

`formatDecimal( value, dp )`

Turns a floating point value into a string with a given number of decimal places using standard settings.

- Parameters:
  - `value` *(floating point)*: value to format
  - `dp` *(integer)*: number of decimal places (digits after the decmal point)
- Return value
  - *(String)*: formatted string
- Examples:
  - `formatDecimal(PI,0) = "3."`
  - `formatDecimal(0,10) = ".0000000000"`
  - `formatDecimal(E*10,3) = "27.183"`

`formatDecimalLocal( value, dp )`

Turns a floating point value into a string using current locale settings. For instance if language is set to French, decimal points will be represented as a comma "," instead of a full stop ".". Otherwise behaves the same as the corresponding `formatDecimal` function.

- Parameters:
  - `value` *(floating point)*: value to format
  - `dp` *(integer)*: number of decimal places (digits after the decmal point)
- Return value
  - *(String)*: formatted string
- Examples:
  - `formatDecimal(PI,0) = "3,"`
  - `formatDecimal(0,10) = ",0000000000"`
  - `formatDecimal(E*10,3) = "27,183"`

`formatDecimal( value, format )`

Turns a floating point value into a formatted string using standard settings. The `format` string is as defined by Java's `java.text.DecimalFormat` (http://docs.oracle.com/javase/6/docs/api/java/text/DecimalFormat.html) class.

- Parameters:
  - `value` *(floating point)*: value to format
  - `format` *(String)*: format specifier
- Return value
  - *(String)*: formatted string

- Examples:
  - `formatDecimal(99, "#.000") = "99.000"`
  - `formatDecimal(PI, "+0.##;-0.##") = "+3.14"`

`formatDecimalLocal( value, format )`

Turns a floating point value into a formatted string using current locale settings. For instance if language is set to French, decimal points will be represented as a comma "," instead of a full stop ".". Otherwise behaves the same as the corresponding `formatDecimal` function.

- Parameters:
  - `value` *(floating point)*: value to format
  - `format` *(String)*: format specifier
- Return value
  - *(String)*: formatted string
- Examples:
  - `formatDecimal(99, "#.000") = "99,000"`
  - `formatDecimal(PI, "+0.##;-0.##") = "+3,14"`

B.1.11 Gaia

Functions related to astrometry suitable for use with data from the Gaia astrometry mission.

The methods here are not specific to the Gaia mission, but the parameters of the functions and their units are specified in a form that is convenient for use with Gaia data, in particular the `gaia_source` catalogue available from http://gea.esac.esa.int/archive/ (http://gea.esac.esa.int/archive/) and copies or mirrors.

There are currently three main sets of functions here:

- position and velocity vector calculation and manipulation
- distance estimation from parallaxes
- astrometry propagation to different epochs

Position and velocity vectors

Functions are provided for converting the astrometric parameters contained in the Gaia catalogue to ICRS Cartesian position (XYZ) and velocity (UVW) vectors. Functions are also provided to convert these vectors between ICRS and Galactic or Ecliptic coordinates. The calculations are fairly straightforward, and follow the equations laid out in section 1.5.6 of *The Hipparcos and Tycho Catalogues*, ESA SP-1200 (https://www.cosmos.esa.int/web/hipparcos/catalogues) (1997) and also section 3.1.7 of the Gaia DR2 documentation (http://gea.esac.esa.int/archive/documentation/GDR2/) (2018).

These functions will often be combined; for instance to calculate the position and velocity in galactic coordinates from Gaia catalogue values, the following expressions may be useful:

```
xyz_gal = icrsToGal(astromXYZ(ra,dec,parallax))
uvw_gal = icrsToGal(astromUVW(array(ra,dec,parallax,pmra,pmdec,radial_velocity)))
```

though note that these particular examples simply invert parallax to provide distance estimates, which is not generally valid. Note also that these functions do not attempt to correct for solar motion. Such adjustments should be carried out by hand on the results of these functions if they are required.

Functions for calculating errors on the Cartesian components based on the error and correlation quantities from the Gaia catalogue are not currently provided. They would require fairly complicated invocations. If there is demand they may be implemented in the future.

Distance estimation

Gaia measures parallaxes, but some scientific use cases require the radial distance instead. While distance in parsec is in principle the reciprocal of parallax in arcsec, in the presence of non-negligable errors on measured parallax, this inversion does not give a good estimate of distance. A thorough discussion of this topic and approaches to estimating distances for Gaia-like data can be found in the papers

- C.A.L.Bailer-Jones, "*Estimating distances from parallaxes*", PASP *127*, p994 (2015) 2015PASP..127..994B (http://adsabs.harvard.edu/abs/2015PASP..127..994B)
- T.L.Astraatmadja and C.A.L.Bailer-Jones, "*Estimating Distances from Parallaxes. II. Performance of Bayesian Distance Estimators on a Gaia-like Catalogue*", ApJ *832*, a137 (2016) 2016ApJ...832..137A (http://adsabs.harvard.edu/abs/2016ApJ...832..137A)
- X.Luri et al., "*Gaia Data Release 2: Using Gaia Parallaxes*", A&A *in press* (2018) arXiv:1804.09376 (https://arxiv.org/abs/1804.09376)

The functions provided here correspond to calculations from Astraatmadja & Bailer-Jones, "*Estimating Distances from Parallaxes. III. Distances of Two Million Stars in the Gaia DR1 Catalogue*", ApJ *833*, a119 (2016) 2016ApJ...833..119A (http://adsabs.harvard.edu/abs/2016ApJ...833..119A) based on the **Exponentially Decreasing Space Density** prior defined therein. This implementation was written with reference to the Java implementation by Enrique Utrilla (DPAC).

These functions are parameterised by a length scale L that defines the exponential decay (the mode of the prior PDF is at r=2L). Some value for this length scale, specified in parsec, must be supplied to the functions as the `lpc` parameter.

Note that the values provided by these functions do *not* match those from the paper Bailer-Jones et al. "*Estimating Distances from Parallaxes IV: Distances to 1.33 Billion stars in Gaia Data Release 2"*, accepted for AJ (2018) arXiv:1804.10121 (https://arxiv.org/abs/1804.10121). The calculations of that paper differ from the ones presented here in several ways: it uses a galactic model for the direction-dependent length scale not currently available here, it pre-applies a parallax correction of -0.029mas, and it uses different uncertainty measures and in some cases (bimodal PDF) a different best distance estimator.

Epoch Propagation

The Gaia source catalogue provides, for at least some sources, the six-parameter astrometric solution (Right Ascension, Declination, Parallax, Proper motion in RA and Dec, and Radial Velocity), along with errors on these values and correlations between these errors. While a crude estimate of the position at an earlier or later epoch than that of the measurement can be made by multiplying the proper motion components by epoch difference and adding to the measured position, a more careful treatment is required for accurate propagation between epochs of the astrometric parameters, and if required their errors and correlations. The expressions for this are set out in section 1.5.5 (Volume 1) of *The Hipparcos and Tycho Catalogues*, ESA SP-1200 (https://www.cosmos.esa.int/web/hipparcos/catalogues) (1997) (but see below), and the code is based on an implementation by Alexey Butkevich and Daniel Michalik (DPAC). A correction is applied to the SP-1200 treatment of radial velocity uncertainty following *Michalik et al. 2014* 2014A&A...571A..85M (http://ukads.nottingham.ac.uk/abs/2014A%26A...571A..85M) because of their better handling of small radial velocities or parallaxes.

The calculations give the same results, though not exactly in the same form, as the epoch propagation functions available in the Gaia archive service.

`polarXYZ( phi, theta, r )`

Converts from spherical polar to Cartesian coordinates.

- Parameters:
  - `phi` *(floating point)*: longitude in degrees
  - `theta` *(floating point)*: latitude in degrees
  - `r` *(floating point)*: radial distance
- Return value
  - *(array of floating point)*: 3-element vector giving Cartesian coordinates
- Examples:
  - `polarXYZ(ra, dec, distance_estimate)`
  - `polarXYZ(l, b, 3262./parallax)` - calculates vector components in units of light year in the galactic system, on the assumption that distance is the inverse of parallax

`astromXYZ( ra, dec, parallax )`

Calculates Cartesian components of position from RA, Declination and parallax. This is a convenience function, equivalent to:

```
polarXYZ(ra, dec, 1000./parallax)
```

Note that this performs distance scaling using a simple inversion of parallax, which is not in general reliable for parallaxes with non-negligable errors. Use at your own risk.

- Parameters:
  - `ra` *(floating point)*: Right Ascension in degrees
  - `dec` *(floating point)*: Declination in degrees
  - `parallax` *(floating point)*: parallax in mas
- Return value
  - *(array of floating point)*: 3-element vector giving equatorial space coordinates in parsec
- Examples:
  - `astromXYZ(ra, dec, parallax)`
  - `icrsToGal(astromXYZ(ra, dec, parallax))`

`icrsToGal( xyz )`

Converts a 3-element vector representing ICRS (equatorial) coordinates to galactic coordinates. This can be used with position or velocity vectors.

The input vector is multiplied by the matrix $\mathbf{A_G}'$, given in Eq. 3.61 of the Gaia DR2 documentation, following Eq. 1.5.13 of the Hipparcos catalogue.

The output coordinate system is right-handed, with the three components positive in the directions of the Galactic center, Galactic rotation, and the North Galactic Pole respectively.

- Parameters:

  - `xyz` *(array of floating point)*: 3-element vector giving ICRS Cartesian components
- Return value
  - *(array of floating point)*: 3-element vector giving Galactic Cartesian components
- Example:
  - `icrsToGal(polarXYZ(ra, dec, distance))`

`galToIcrs( xyz )`

Converts a 3-element vector representing galactic coordinates to ICRS (equatorial) coordinates. This can be used with position or velocity vectors.

The input vector is multiplied by the matrix $\mathbf{A_G}$, given in Eq. 3.61 of the Gaia DR2 documentation, following Eq. 1.5.13 of the Hipparcos catalogue.

The input coordinate system is right-handed, with the three components positive in the directions of the Galactic center, Galactic rotation, and the North Galactic Pole respectively.

- Parameters:
  - `xyz` *(array of floating point)*: 3-element vector giving Galactic Cartesian components
- Return value
  - *(array of floating point)*: 3-element vector giving ICRS Cartesian components
- Example:
  - `galToIcrs(polarXYZ(l, b, distance))`

`icrsToEcl( xyz )`

Converts a 3-element vector representing ICRS (equatorial) coordinates to ecliptic coordinates. This can be used with position or velocity vectors.

The transformation corresponds to that between the coordinates `(ra,dec)` and `(ecl_lon,ecl_lat)` in the Gaia source catalogue (DR2).

- Parameters:
  - `xyz` *(array of floating point)*: 3-element vector giving ICRS Cartesian components
- Return value
  - *(array of floating point)*: 3-element vector giving ecliptic Cartesian components
- Example:
  - `icrsToEcl(polarXYZ(ra, dec, distance))`

`eclToIcrs( xyz )`

Converts a 3-element vector representing ecliptic coordinates to ICRS (equatorial) coordinates. This can be used with position or velocity vectors.

The transformation corresponds to that between the coordinates `(ecl_lon,ecl_lat)` and `(ra,dec)` in the Gaia source catalogue (DR2).

- Parameters:
  - `xyz` *(array of floating point)*: 3-element vector giving ecliptic Cartesian coordinates
- Return value
  - *(array of floating point)*: 3-element vector giving ICRS Cartesian coordinates

- Example:
  - `eclToIcrs(polarXYZ(ecl_lon, ecl_lat, distance))`

`astromUVW( astrom6 )`

Calculates Cartesian components of velocity from quantities available in the Gaia source catalogue. The output is in the same coordinate system as the inputs, that is ICRS for the correspondingly-named Gaia quantities.

The input astrometry parameters are represented by a 6-element array, with the following elements:

```
index  gaia_source name  unit    description
-----  ----------------  ----    -----------
  0:   ra                deg     right ascension
  1:   dec               deg     declination
  2:   parallax          mas     parallax
  3:   pmra              mas/yr  proper motion in ra * cos(dec)
  4:   pmdec             mas/yr  proper motion in dec
  5:   radial_velocity   km/s    barycentric radial velocity
```

The units used by this function are the units used in the `gaia_source` table.

This convenience function just invokes the 7-argument `astromUVW` function using the inverted parallax for the radial distance, and without invoking the Doppler correction. It is exactly equivalent to:

```
astromUVW(a[0], a[1], a[3], a[4], a[5], 1000./a[2], false)
```

Note this naive inversion of parallax to estimate distance is not in general reliable for parallaxes with non-negligable errors.

- Parameters:
  - `astrom6` *(array of floating point)*: vector of 6 astrometric parameters as provided by the Gaia source catalogue
- Return value
  - *(array of floating point)*: 3-element vector giving equatorial velocity components in km/s
- Examples:
  - `astromUVW(array(ra, dec, parallax, pmra, pmdec, radial_velocity))`
  - `icrsToGal(astromUVW(array(ra, dec, parallax, pmra, pmdec, radial_velocity)))`

`astromUVW( ra, dec, pmra, pmdec, radial_velocity, r_parsec, useDoppler )`

Calculates Cartesian components of velocity from the observed position and proper motion, radial velocity and radial distance, with optional light-time correction. The output is in the same coordinate system as the inputs, that is ICRS for the correspondingly-named Gaia quantities.

The radial distance must be supplied using the `r_parsec` parameter. A naive estimate from quantities in the Gaia source catalogue may be made with the expression `1000./parallax`, though note that this simple inversion of parallax is not in general reliable for parallaxes with non-negligable errors.

The calculations are fairly straightforward, following Eq. 1.5.74 from the Hipparcos catalogue. A (usually small) Doppler factor accounting for light-time effects can also optionally be applied. The effect of this is to multiply the returned vector by a factor of

`1/(1-radial_velocity/c)`, as discussed in Eq. 1.2.21 of the Hipparcos catalogue.

Note that no attempt is made to adjust for solar motion.

- Parameters:
  - `ra` *(floating point)*: Right Ascension in degrees
  - `dec` *(floating point)*: Declination in degrees
  - `pmra` *(floating point)*: proper motion in RA * cos(dec) in mas/yr
  - `pmdec` *(floating point)*: proper motion in declination in mas/yr
  - `radial_velocity` *(floating point)*: radial velocity in km/s
  - `r_parsec` *(floating point)*: radial distance in parsec
  - `useDoppler` *(boolean)*: whether to apply the Doppler factor to account for light-time effects
- Return value
  - *(array of floating point)*: 3-element vector giving equatorial velocity components in km/s
- Examples:
  - `astromUVW(ra, dec, pmra, pmdec, radial_velocity, dist, true)`
  - `icrsToGal(astromUVW(ra, dec, pmra, pmdec, radial_velocity, 1000./parallax, false))`

`epochProp( tYr, astrom6 )`

Propagates the astrometry parameters, supplied as a 6-element array, to a different epoch.

The input and output astrometry parameters are each represented by a 6-element array, with the following elements:

```
index  gaia_source name  unit    description
-----  ----------------  ----    -----------
  0:   ra                deg     right ascension
  1:   dec               deg     declination
  2:   parallax          mas     parallax
  3:   pmra              mas/yr  proper motion in ra * cos(dec)
  4:   pmdec             mas/yr  proper motion in dec
  5:   radial_velocity   km/s    barycentric radial velocity
```

The units used by this function are the units used in the `gaia_source` table.

Null values for `parallax`, `pmra`, `pmdec` and `radial_velocity` are treated as if zero for the purposes of propagation. The documentation of the equivalent function in the Gaia archive comments *"This is a reasonable choice for most stars because those quantities would be either small (parallax and proper motion) or irrelevant (radial velocity). However, this is not true for stars very close to the Sun, where appropriate values need to be taken from the literature (e.g. average velocity field in the solar neighbourhood)."*

The effect is that the output represents the best estimates available for propagated astrometry; proper motions, parallax and RV are applied if present, but if not the output values are calculated or simply copied across as if those quantities were zero.

- Parameters:
  - `tYr` *(floating point)*: epoch difference in years
  - `astrom6` *(array of floating point)*: astrometry at time t0, represented by a 6-element array as above (a 5-element array is also permitted where radial velocity is zero or unknown)
- Return value

  - *(array of floating point)*: astrometry at time `t0+tYr`, represented by a 6-element array as above
- Example:
  - `epochProp(-15.5, array(ra,dec,parallax,pmra,pmdec,radial_velocity))` - calculates the astrometry at 2000.0 of gaia_source values that were observed at 2015.5

`epochPropErr( tYr, astrom22 )`

Propagates the astrometry parameters and their associated errors and correlations, supplied as a 22-element array, to a different epoch.

The input and output astrometry parameters with associated error and correlation information are each represented by a 22-element array, with the following elements:

index	gaia_source name	unit	description
0:	ra	deg	right ascension
1:	dec	deg	declination
2:	parallax	mas	parallax
3:	pmra	mas/yr	proper motion in RA * cos(dec)
4:	pmdec	mas/yr	proper motion in Declination
5:	radial_velocity	km/s	barycentric radial velocity
6:	ra_error	mas	error in right ascension
7:	dec_error	mas	error in declination
8:	parallax_error	mas	error in parallax
9:	pmra_error	mas/yr	error in RA proper motion * cos(dec)
10:	pmdec_error	mas/yr	error in Declination proper motion
11:	radial_velocity_error	km/s	error in barycentric radial velocity
12:	ra_dec_corr		correlation between ra and dec
13:	ra_parallax_corr		correlation between ra and parallax
14:	ra_pmra_corr		correlation between ra and pmra
15:	ra_pmdec_corr		correlation between ra and pmdec
16:	dec_parallax_corr		correlation between dec and parallax
17:	dec_pmra_corr		correlation between dec and pmra
18:	dec_pmdec_corr		correlation between dec and pmdec
19:	parallax_pmra_corr		correlation between parallax and pmra
20:	parallax_pmdec_corr		correlation between parallax and pmdec
21:	pmra_pmdec_corr		correlation between pmra and pmdec

Note the correlation coefficients, always in the range -1..1, are dimensionless.

Null values for `parallax`, `pmra`, `pmdec` and `radial_velocity` are treated as if zero for the purposes of propagation. The documentation of the equivalent function in the Gaia archive comments *"This is a reasonable choice for most stars because those quantities would be either small (parallax and proper motion) or irrelevant (radial velocity). However, this is not true for stars very close to the Sun, where appropriate values need to be taken from the literature (e.g. average velocity field in the solar neighbourhood)."*

The effect is that the output represents the best estimates available for propagated astrometry; proper motions, parallax and RV are applied if present, but if not the output values are calculated or simply copied across as if those quantities were zero.

This transformation is only applicable for radial velocities determined independently of the astrometry, such as those obtained with a spectrometer. It is not applicable for the back-transformation of data already propagated to another epoch.

This is clearly an unwieldy function to invoke, but if you are using it with the gaia_source catalogue itself, or other similar catalogues with the same column names and units, you can invoke it by just copying and pasting the example shown in this documentation.

- Parameters:
  - `tYr` *(floating point)*: epoch difference in years
  - `astrom22` *(array of floating point)*: astrometry at time t0, represented by a

22-element array as above

- Return value
  - *(array of floating point)*: astrometry at time t0+tYr, represented by a 22-element array as above
- Example:
  - `epochPropErr(-15.5, array(ra,dec,parallax,pmra,pmdec,radial_velocity, ra_error,dec_error,parallax_error,pmra_error,pmdec_error,radial_velocity_error, ra_dec_corr,ra_parallax_corr,ra_pmra_corr,ra_pmdec_corr, dec_parallax_corr,dec_pmra_corr,dec_pmdec_corr, parallax_pmra_corr,parallax_pmdec_corr, pmra_pmdec_corr))` - calculates the astrometry with all errors and correlations at 2000.0 for gaia_source values that were observed at 2015.5.

`rvMasyrToKms( rvMasyr, plxMas )`

Converts from normalised radial velocity in mas/year to unnormalised radial velocity in km/s.

The output is calculated as `AU_YRKMS * rvMasyr / plxMas`, where `AU_YRKMS=4.740470446` is one Astronomical Unit in km.yr/sec.

- Parameters:
  - `rvMasyr` *(floating point)*: normalised radial velocity, in mas/year
  - `plxMas` *(floating point)*: parallax in mas
- Return value
  - *(floating point)*: radial velocity in km/s

`rvKmsToMasyr( rvKms, plxMas )`

Converts from unnormalised radial velocity in km/s to normalised radial velocity in mas/year.

The output is calculated as `rvKms * plxMas / AU_YRKMS`, where `AU_YRKMS=4.740470446` is one Astronomical Unit in km.yr/sec.

- Parameters:
  - `rvKms` *(floating point)*: unnormalised radial velocity, in mas/year
  - `plxMas` *(floating point)*: parallax in mas
- Return value
  - *(floating point)*: radial velocity in mas/year

`distanceEstimateEdsd( plxMas, plxErrorMas, lPc )`

Best estimate of distance using the Exponentially Decreasing Space Density prior. This estimate is provided by the mode of the PDF.

- Parameters:
  - `plxMas` *(floating point)*: parallax in mas
  - `plxErrorMas` *(floating point)*: parallax error in mas
  - `lPc` *(floating point)*: length scale in parsec
- Return value
  - *(floating point)*: best distance estimate in parsec

`distanceBoundsEdsd( plxMas, plxErrorMas, lPc )`

Calculates the 5th and 95th percentile confidence intervals on the distance estimate using the Exponentially Decreasing Space Density prior.

Note this function has to numerically integrate the PDF to determine quantile values, so it is relatively slow.

- Parameters:
  - `plxMas` *(floating point)*: parallax in mas
  - `plxErrorMas` *(floating point)*: parallax error in mas
  - `lPc` *(floating point)*: length scale in parsec
- Return value
  - *(array of floating point)*: 2-element array giving the 5th and 95th percentiles in parsec of the EDSD distance PDF

`distanceQuantilesEdsd( plxMas, plxErrorMas, lPc, qpoints, ... )`

Calculates arbitrary quantiles for the distance estimate using the Exponentially Decreasing Space Density prior.

Note this function has to numerically integrate the PDF to determine quantile values, so it is relatively slow.

- Parameters:
  - `plxMas` *(floating point)*: parallax in mas
  - `plxErrorMas` *(floating point)*: parallax error in mas
  - `lPc` *(floating point)*: length scale in parsec
  - `qpoints` *(floating point, one or more)*: one or more required quantile cut points, each in the range 0..1
- Return value
  - *(array of floating point)*: array with one element for each of the supplied `qpoints` giving the corresponding distance in parsec
- Examples:
  - `distanceQuantilesEdsd(parallax, parallax_error, 1350, 0.5)[0]` calculates the median of the EDSD distance PDF using a length scale of 1.35kpc
  - `distanceQuantilesEdsd(parallax, parallax_error, 3000, 0.01, 0.99)` returns a 2-element array giving the 1st and 99th percentile of the distance estimate using a length scale of 3kpc

`distanceToModulus( distPc )`

Converts a distance in parsec to a distance modulus. The formula is `5*log10(distPc)-5`.

- Parameters:
  - `distPc` *(floating point)*: distance in parsec
- Return value
  - *(floating point)*: distance modulus in magnitudes

`modulusToDistance( distmod )`

Converts a distance modulus to a distance in parsec. The formula is `10^(1+distmod/5)`.

- Parameters:
  - `distmod` *(floating point)*: distance modulus in magnitudes

- Return value
  - *(floating point)*: distance in parsec

AU_YRKMS
This quantity is A_v, the Astronomical Unit expressed in km.yr/sec. See the Hipparcos catalogue (ESA SP-1200) table 1.2.2 and Eq. 1.5.24.

PC_AU
Parsec in Astronomical Units, equal to 648000/PI.

PC_YRKMS
Parsec in units of km.yr/sec.

C_KMS
The speed of light in km/s (exact).

B.1.12 Json

Functions for working with JSON strings.

Usage of this class to manipulate hierarchical JSON objects is a bit different from the way that most of the other functions in the expression language are used. The main function provided by this class is `jsonObject`, which can be applied to a string value containing a JSON object (legal JSON object strings are enclosed in curly brackets), and returns a *JSONObject* instance. This JSONObject cannot be used on its own in the rest of the application, but various *methods* can be applied to it to extract information from its structure. These methods are applied by writing `jsonObject.method(arg,arg,...)` rather than `function(jsonObject,arg,arg,...)`.

Methods you can apply to a JSONObject include:

- `getString(key)`
- `getInt(key)`
- `getDouble(key)`
- `getBoolean(key)`
- `getJSONObject(key)`
- `getJSONArray(key)`

where `key` is a string giving the name with which the member of the JSON object is associated. The first four of the above methods give you string, numeric, or boolean values that you can use in the application for plotting etc. `getJSONObject` returns another JSONObject that you can interrogate with further methods. `getJSONArray` gives you a JSONArray.

You can apply a different set of methods to a JSONArray, including:

- `getString(index)`
- `getInt(index)`
- `getDouble(index)`
- `getBoolean(index)`
- `getJSONObject(index)`
- `getJSONArray(index)`

where `index` is an integer giving the (zero-based) index of the element in the array that you want.

Using these methods you can drill down into the hierarchical structure of a JSON string to retrieve the string, numeric or boolean values that you need. If you are not familiar with this syntax, an example is the best way to illustrate it. Consider the following JSON string, which may be the value in a column named `txt`:

```
{
   "sequence": 23,
   "temperature": {
      "value": 278.5,
      "units": "Kelvin"
   },
   "operational": true,
   "readings": [12, null, 23.2, 441, 0]
}
```

To obtain the sequence value, you can write:

```
jsonObject(txt).getInt("sequence")
```

to obtain the numeric temperature value, you can write:

```
jsonObject(txt).getJSONObject("temperature").getDouble("value")
```

and to obtain the first element of the readings array, you can write:

```
jsonObject(txt).getJSONArray("readings").getDouble(0)
```

Other methods are available on JSONObject and JSONArray; you can currently find documentation on them at https://stleary.github.io/JSON-java/ (https://stleary.github.io/JSON-java/). Note in particular that each of the JSONObject.`get*(key)` and JSONArray.`get*(index)` methods is accompanied by a corresponding `opt*` method; where the key/index may not exist, this will probably give effectively the same behaviour (generating a blank result) but may be considerably faster.

This class also contains some other utility functions for working with JSONObjects and JSONArrays; see the function documentation below for details.

Note: This class is somewhat experimental, and the functions and methods may change in future. An attempt will be made to retain the functions and methods described in this section, but those described in the external JSON-java javadocs may be subject to change.

`jsonObject( txt )`

Converts the supplied string to a JSONObject which can be interrogated further. If the input string doesn't have the syntax of a JSON object, an empty JSONObject will be returned.

The JSON parsing is currently somewhat lenient, for instance allowing a comma before a closing curly bracket.

- Parameters:
  - `txt` *(String)*: string assumed to contain a JSON object
- Return value
  - *(JSONObject)*: JSON object value on which further methods can be called

`jsonArray( txt )`

Converts the supplied string to a JSONArray which can be interrogated further. If the input string doesn't have the syntax of a JSON array, a zero-length JSONArray will be returned.

The JSON parsing is currently somewhat lenient, for instance allowing a comma before a closing square bracket.

- Parameters:
  - `txt` *(String)*: string assumed to contain a JSON array
- Return value
  - *(JSONArray)*: JSON array value on which further methods can be called

`jsonObjectOpt( txt )`

Converts the supplied string to a JSON object which can be interrogated further. If the input doesn't have the syntax of a JSON object, null will be returned.

For most purposes this will behave the same as the `jsonObject` function, but it may be slower if there are many invalid or empty JSON strings.

- Parameters:
  - `txt` *(String)*: JSON object text
- Return value
  - *(JSONObject)*: JSON object value on which further methods can be called, or null

`jsonArrayOpt( txt )`

Converts the supplied string to a JSONArray which can be interrogated further. If the input string doesn't have the syntax of a JSON array, null will be returned.

- Parameters:
  - `txt` *(String)*: JSON object text
- Return value
  - *(JSONArray)*: JSON array value on which further methods can be called, or null

`jsonToDoubles( jsonArray )`

Converts a JSONArray to an array of floating point values. The result will be the same length as the supplied JSONArray, and any element in the JSONArray which cannot be interpreted as a floating point value is represented by a NaN.

- Parameters:
  - `jsonArray` *(JSONArray)*: JSON Array
- Return value
  - *(array of floating point)*: floating point array the same length as the input array
- Example:
  - jsonToDoubles(jsonArray("[true, \"two\", 3.0, 4, null"])) = [NaN, NaN, 3.0, 4.0, NaN]

`jsonToStrings( jsonArray )`

Converts a JSONArray to an array of string values.

- Parameters:
  - `jsonArray` *(JSONArray)*: JSON Array
- Return value
  - *(array of String)*: string array the same length as the input array
- Example:
  - jsonToStrings(jsonArray("[true, \"two\", 3.0, 4, null]")) = ["true", "two", "3.0", "4", null]

`jsonGetKeys( jsonObject )`
Returns an array giving keys for each key-value pair in a JSON object. The members of a JSON object are not ordered, so no order can be guaranteed in the output array.

- Parameters:
  - `jsonObject` *(JSONObject)*: JSON object
- Return value
  - *(array of String)*: string array containing keys of object
- Example:
  - jsonGetKeys(jsonObject("{\"one\", 1, \"two\", 2, \"three\", 3}")) = ["one", "two", "three"]

B.1.13 KCorrections

Functions for calculating K-corrections.

`kCorr( filter, redshift, colorType, colorValue )`
Calculates K-corrections. This allows you to determine K-corrections for a galaxy, given its redshift and a colour. Filters for GALEX, SDSS, UKIDSS, Johnson, Cousins and 2MASS are covered.

To define the calculation you must choose both a filter, specified as a `KCF_*` constant, and a colour (filter pair) specified as a `KCC_*` constant. For each available filter, only certain colours are available, as described in the documentation of the relevant `KCF_*` constant.

The algorithm used is described at http://kcor.sai.msu.ru/ (http://kcor.sai.msu.ru/). This is based on the paper *"Analytical Approximations of K-corrections in Optical and Near-Infrared Bands"* by I.Chilingarian, A.-L.Melchior and I.Zolotukhin (2010MNRAS.405.1409C (http://adsabs.harvard.edu/abs/2010MNRAS.405.1409C)), but extended to include GALEX UV bands and with redshift coverage up to 0.5 as described in *"Universal UV-optical Colour-Colour-Magnitude Relation of Galaxies"* by I.Chilingarian and I.Zolotukhin (2012MNRAS.419.1727C (http://adsabs.harvard.edu/abs/2012MNRAS.419.1727C)).

- Parameters:
  - `filter` *(KFilter)*: `KCF_*` constant defining the filter for which you want to calculate the K-correction
  - `redshift` *(floating point)*: galaxy redshift; this should be in the range 0-0.5
  - `colorType` *(KColor)*: `KCC_*` constant defining the filter pair for the calculation; check the `KCF_*` constant documentation to see which ones are permitted for a given filter

    - `colorValue` *(floating point)*: the value of the colour
- Return value
    - *(floating point)*: K correction
- Examples:
    - `kCorr(KCF_g, 0.16, KCC_gr, -0.8) = 3.593`
    - `kCorr(KCF_FUV, 0.48, KCC_FUVu, 0.31) = -0.170`

`KCF_FUV`
GALEX FUV filter (AB). Use with KCC_FUVNUV or KCC_FUVu.

`KCF_NUV`
GALEX NUV filter (AB). Use with KCC_NUVg or KCC_NUVr.

`KCF_u`
SDSS u filter (AB). Use with KCC_ur, KCC_ui or KCC_uz.

`KCF_g`
SDSS g filter (AB). Use with KCC_gr, KCC_gi or KCC_gz.

`KCF_r`
SDSS r filter (AB). Use with KCC_gr or KCC_ur.

`KCF_i`
SDSS i filter (AB). Use with KCC_gi or KCC_ui.

`KCF_z`
SDSS z filter (AB). Use with KCC_rz, KCC_gz or KCC_uz.

`KCF_Y`
UKIDSS Y filter (AB). Use with KCC_YH or KCC_YK.

`KCF_J`
UKIDSS J filter (AB). Use with KCC_JK or KCC_JH.

`KCF_H`
UKIDSS H filter (AB). Use with KCC_HK or KCC_JH.

`KCF_K`
UKIDSS K filter (AB). Use with KCC_JK or KCC_HK.

`KCF_U`
Johnson U filter (Vega). Use with KCC_URc.

KCF_B
Johnson B filter (Vega). Use with KCC_BRc or KCC_BIc.

KCF_V
Johnson V filter (Vega). Use with KCC_VIc or KCC_VRc.

KCF_Rc
Cousins Rc filter (Vega). Use with KCC_BRc or KCC_VRc.

KCF_Ic
Cousins Ic filter (Vega). Use with KCC_VIc.

KCF_J2
2MASS J filter (Vega). Use with KCC_J2Ks2 or KCC_J2H2.

KCF_H2
2MASS H filter (Vega). Use with KCC_H2Ks2 or KCC_J2H2.

KCF_Ks2
2MASS Ks filter (Vega). Use with KCC_J2Ks2 or KCC_H2Ks2.

KCC_BIc
Johnson B - Cousins Ic colour.

KCC_BRc
Johnson B - Cousins Rc colour.

KCC_FUVNUV
GALEX FUV - NUV colour.

KCC_FUVu
GALEX FUV - SDSS u colour.

KCC_gi
SDSS g - i colour.

KCC_gr
SDSS g - r colour.

KCC_gz
SDSS g - z colour.

KCC_H2Ks2
2MASS H - Ks colour.

`KCC_HK`
UKIDSS H - K colour.

`KCC_J2H2`
2MASS J - H colour.

`KCC_J2Ks2`
2MASS J - Ks colour.

`KCC_JH`
UKIDSS J - H colour.

`KCC_JK`
UKIDSS J - K colour.

`KCC_NUVg`
GALEX NUV - SDSS g colour.

`KCC_NUVr`
GALEX NUV - SDSS r colour.

`KCC_rz`
SDSS r - SDSS z colour.

`KCC_ui`
SDSS u - SDSS i colour.

`KCC_URc`
Johnson U - Cousins Rc colour.

`KCC_ur`
SDSS u - r colour.

`KCC_uz`
SDSS u - z colour.

`KCC_VIc`
Johnson V - Cousins Ic colour.

`KCC_VRc`
Johnson V - Cousins Rc colour.

KCC_YH
UKIDSS Y - H colour.

KCC_YK
UKIDSS Y - K colour.

B.1.14 Lists

Functions which operate on lists of values.

Some of these resemble similar functions in the `Arrays` class, and in some cases are interchangeable, but these are easier to use on non-array values because you don't have to explicitly wrap up lists of arguments as an array. However, for implementation reasons, most of the functions defined here can be used on values which are already `double[]` arrays (for instance array-valued columns) rather than as comma-separated lists of floating point values.

`sum( values, ... )`
Returns the sum of all the non-blank supplied arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: sum of `values`
- Examples:
  - `sum(1, 3, 99) = 103`
  - `sum(1, 3, NaN) = 4`

`mean( values, ... )`
Returns the mean of all the non-blank supplied arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: mean of `values`
- Examples:
  - `mean(2, 4, 6, 8) = 5`
  - `mean(100.5, 99.5, NaN) = 100`

`variance( values, ... )`
Returns the population variance of the non-blank supplied arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: population variance of `values`

- Examples:
  - `variance(0, 3, 4, 3, 0) = 2.8`
  - `variance(0, 3, NaN, 3, NaN) = 2`

`stdev( values, ... )`
Returns the population standard deviation of the non-blank supplied arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: population standard deviation of `values`
- Example:
  - `stdev(-3, -2, 0, 0, 1, 2, 3, 4, 5, 6) = 2.8`

`min( values, ... )`
Returns the minimum of all the non-blank supplied arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: minimum of `values`
- Example:
  - `min(20, 25, -50, NaN, 101) = -50`

`max( values, ... )`
Returns the maximum of all the non-blank supplied arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: maximum of `values`
- Example:
  - `max(20, 25, -50, NaN, 101) = 101`

`median( values, ... )`
Returns the median of all the non-blank supplied arguments.

- Parameters:
  - `values` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: median of `values`
- Example:
  - `median(-100000, 5, 6, 7, 8) = 6`

`countTrue( values, ... )`

Returns the number of true values in a list of boolean arguments. Note if any of the values are blank, the result may be blank as well.

- Parameters:
  - `values` *(boolean, one or more)*: one or more true/false values
- Return value
  - *(integer)*: number of elements of `values` that are true
- Example:
  - `countTrue(false, false, true, false, true) = 2`

B.1.15 Maths

Standard mathematical and trigonometric functions. Trigonometric functions work with angles in radians.

`sin( theta )`

Sine of an angle.

- Parameters:
  - `theta` *(floating point)*: an angle, in radians.
- Return value
  - *(floating point)*: the sine of the argument.

`cos( theta )`

Cosine of an angle.

- Parameters:
  - `theta` *(floating point)*: an angle, in radians.
- Return value
  - *(floating point)*: the cosine of the argument.

`tan( theta )`

Tangent of an angle.

- Parameters:
  - `theta` *(floating point)*: an angle, in radians.
- Return value
  - *(floating point)*: the tangent of the argument.

`asin( x )`

Arc sine of an angle. The result is in the range of -*pi*/2 through *pi*/2.

- Parameters:
  - `x` *(floating point)*: the value whose arc sine is to be returned.

- Return value
  - *(floating point)*: the arc sine of the argument (radians)

`acos( x )`

Arc cosine of an angle. The result is in the range of 0.0 through *pi*.

- Parameters:
  - `x` *(floating point)*: the value whose arc cosine is to be returned.
- Return value
  - *(floating point)*: the arc cosine of the argument (radians)

`atan( x )`

Arc tangent of an angle. The result is in the range of -*pi*/2 through *pi*/2.

- Parameters:
  - `x` *(floating point)*: the value whose arc tangent is to be returned.
- Return value
  - *(floating point)*: the arc tangent of the argument (radians)

`ln( x )`

Natural logarithm.

- Parameters:
  - `x` *(floating point)*: argument
- Return value
  - *(floating point)*: $\log_e(x)$

`exp( x )`

Euler's number *e* raised to a power.

- Parameters:
  - `x` *(floating point)*: the exponent to raise *e* to.
- Return value
  - *(floating point)*: the value e^x, where *e* is the base of the natural logarithms.

`log10( x )`

Logarithm to base 10.

- Parameters:
  - `x` *(floating point)*: argument
- Return value
  - *(floating point)*: $\log_{10}(x)$

`exp10( x )`

Power of 10. This convenience function is identical to `pow(10,x)`.

- Parameters:
  - `x` *(floating point)*: argument
- Return value
  - *(floating point)*: 10^{x}

`sqrt( x )`
Square root. The result is correctly rounded and positive.

- Parameters:
  - `x` *(floating point)*: a value.
- Return value
  - *(floating point)*: the positive square root of `x`. If the argument is NaN or less than zero, the result is NaN.

`square( x )`
Raise to the power 2.

- Parameters:
  - `x` *(floating point)*: a value
- Return value
  - *(floating point)*: x * x

`hypot( xs, ... )`
Returns the square root of the sum of squares of its arguments. In the 2-argument case, doing it like this may avoid intermediate overflow or underflow.

- Parameters:
  - `xs` *(floating point, one or more)*: one or more numeric values
- Return value
  - *(floating point)*: sqare root of sum of squares of arguments
- Examples:
  - `hypot(3,4) = 5`
  - `hypot(2,2,2,2) = 4`

`atan2( y, x )`
Converts rectangular coordinates (`x`,`y`) to polar (`r`,`theta`). This method computes the phase `theta` by computing an arc tangent of `y/x` in the range of *-pi* to *pi*.

- Parameters:
  - `y` *(floating point)*: the ordinate coordinate
  - `x` *(floating point)*: the abscissa coordinate
- Return value
  - *(floating point)*: the `theta` component (radians) of the point (`r`,`theta`) in polar coordinates that corresponds to the point (`x`,`y`) in Cartesian coordinates.

`pow( a, b )`

Exponentiation. The result is the value of the first argument raised to the power of the second argument.

- Parameters:
  - `a` *(floating point)*: the base.
  - `b` *(floating point)*: the exponent.
- Return value
  - *(floating point)*: the value a^b .

`sinh( x )`
Hyperbolic sine.

- Parameters:
  - `x` *(floating point)*: parameter
- Return value
  - *(floating point)*: result

`cosh( x )`
Hyperbolic cosine.

- Parameters:
  - `x` *(floating point)*: parameter
- Return value
  - *(floating point)*: result

`tanh( x )`
Hyperbolic tangent.

- Parameters:
  - `x` *(floating point)*: parameter
- Return value
  - *(floating point)*: result

`asinh( x )`
Inverse hyperbolic sine.

- Parameters:
  - `x` *(floating point)*: parameter
- Return value
  - *(floating point)*: result

`acosh( x )`
Inverse hyperbolic cosine.

- Parameters:
  - `x` *(floating point)*: parameter

- Return value
  - *(floating point)*: result

`atanh( x )`

Inverse hyperbolic tangent.

- Parameters:
  - `x` *(floating point)*: parameter
- Return value
  - *(floating point)*: result

`E`

Euler's number *e*, the base of natural logarithms.

`PI`

Pi, the ratio of the circumference of a circle to its diameter.

`Infinity`

Positive infinite floating point value.

`NaN`

Not-a-Number floating point value. Use with care; arithmetic and logical operations behave in strange ways near NaN (for instance, `NaN!=NaN`). For most purposes this is equivalent to the blank value.

B.1.16 Randoms

Functions concerned with random number generation.

There are two flavours of functions here: index-based (`random*`) and sequential (`nextRandom*`). Briefly, the index-based ones are safer to use, but provide poorer random statistics, while the sequential ones provide decent randomness but are not suitable for use in some/most contexts. They are documented separately below.

Index-based functions

The functions named `random*` all take an `index` parameter which determines the value of the result; the same index always leads to the same output, but there is not supposed to be any obvious relationship between index and output. An explicit index is required to ensure that a given cell always has the same value, since cell values are in general calculated on demand. The quality of the randomness for these functions may not be that good.

In most cases, the table row index, available as the special token `$0`, is a suitable value for the `index` parameter.

If several different random values are required in the same table row, one way is to supply a different row-based index value for each one, e.g. `random(2*$0)` and `random(2*$0+1)`. However, this tends to introduce a correlation between the random values in the same row, so a better (though in some cases slower) solution is to use one of the array-generating functions, e.g. `randomArray($0,2)[0]` and `randomArray($0,2)[1]`.

The output is deterministic, in the sense that the same invocation will always generate the same "random" number, even across different machines. However, in view of the comments in the implementation note below, the output *may* be subject to change in the future if some improved algorithm can be found, so this guarantee does not necessarily hold across software versions.

Implementation Note: The requirement for mapping a given input index deterministically to a pseudo-random number constrains the way that the random number generation is done; most well-studied RNGs generate sequences of random numbers, but that approach cannot be used here, since these sequences do not admit of random-access. What we do instead is to scramble the input index somewhat and use that as the seed for an instance of Java's `Random` class, which is then used to produce one or more random numbers per input index. Some thought and experimentation has gone into the current implementation (I bought a copy of Knuth Vol. 2 specially!) and an eyeball check of the results doesn't look all that bad, but it's still probably not very good, and is not likely to pass random number quality tests (though I haven't tried). A more respectable approach *might* be to use a cryptographic-grade hash function on the supplied index, but that's likely to be much slower. If there is demand, something like that could be added as an alternative option. In the mean time, beware if you use these random numbers for scientifically sensitive output.

Sequential functions

The functions named `nextRandom*` have no arguments, and supply the next value in a global sequence when they are evaluated. These can be used if scanning through a table once (for instance when writing a table using STILTS), but they are not suitable for contexts that should supply a fixed value. For instance if you use them to define the value of a table cell in TOPCAT, that cell may have a different value every time you look at it, which may have disconcerting results. These use the java.util.Random class in a more standard way than the index-based functions and should provide random numbers of reasonable quality.

`random( index )`

Generates a pseudo-random number sampled from a uniform distribution between 0 and 1.

Note: The randomness may not be very high quality.

- Parameters:
  - `index` *(long integer)*: input value, typically row index "`$0`"
- Return value
  - *(floating point)*: random number between 0 and 1

`randomGaussian( index )`

Generates a pseudo-random number sampled from a Gaussian distribution with mean of 0.0 and standard deviation of 1.0.

Note: The randomness may not be very high quality.

- Parameters:
  - `index` *(long integer)*: input value, typically row index "`$0`"
- Return value
  - *(floating point)*: random number

`randomArray( index, n )`

Generates an array of pseudo-random numbers sampled from a uniform distribution between 0 and 1.

Note: The randomness may not be very high quality.

- Parameters:
  - `index` *(long integer)*: input value, typically row index "`$0`"
  - `n` *(integer)*: size of output array
- Return value
  - *(array of floating point)*: `n`-element array of random numbers between 0 and 1

`randomGaussianArray( index, n )`

Generates an array of pseudo-random numbers sampled from a Gaussian distribution with mean of 0.0 and standard deviation of 1.0.

Note: The randomness may not be very high quality.

- Parameters:
  - `index` *(long integer)*: input value, typically row index "`$0`"
  - `n` *(integer)*: size of output array
- Return value
  - *(array of floating point)*: `n`-element array of random numbers

`nextRandom( )`

Returns the next value in a random sequence, sampled from a uniform distribution between 0 and 1.

This function will give a different result every time, hence it is *not* suitable for use in an expression which should have a fixed value, for instance to define a TOPCAT column.

- Return value
  - *(floating point)*: random number between 0 and 1

`nextRandomGaussian( )`

Returns the next value in a random sequence, sampled from a Gaussian distribution with mean of 0.0 and standard deviation of 1.0.

This function will give a different result every time, hence it is *not* suitable for use in an expression which should have a fixed value, for instance to define a TOPCAT column.

- Return value
  - *(floating point)*: random number

B.1.17 Shapes

Functions useful for working with shapes in the (X, Y) plane.

`polyLine( x, xys, ... )`

Function of `x` defined by straight line segments between a specified list of vertices. The vertices are specified as a sequence of X_i, Y_i pairs, for which the X_i values must be monotonic.

The line segment at each end of the specified point sequence is considered to be extended to infinity. If only two points are specified, this is the equation of a straight line between those points. As a special case, if only one point is specified, the line is considered to be a horizontal line (equal to the sole specified Y_i coordinate for all `x`).

By reversing the X_i and Y_i values, this function can equally be used to represent a function `X(y)` rather than `Y(x)`.

If the number of coordinates is odd, or the X_i values are not monotonic, behaviour is undefined.

- Parameters:
  - `x` *(floating point)*: X value at which function is to be evaluated
  - `xys` *(floating point, one or more)*: 2N arguments (`x1`, `y1`, `x2`, `y2`, ..., `xN`, `yN`) giving vertices of an N-point line with monotonically increasing or decreasing X values
- Return value
  - *(floating point)*: Y coordinate of poly-line for specified `x`
- Example:
  - `polyLine(5, 0,0, 2,2) = 5`

`isInside( x, y, xys, ... )`

Indicates whether a given test point is inside a polygon defined by specified list of vertices. The vertices are specified as a sequence of X_i, Y_i pairs.

If the number of coordinates is odd, the behaviour is not defined.

- Parameters:
  - `x` *(floating point)*: X coordinate of test point
  - `y` *(floating point)*: Y coordinate of test point
  - `xys` *(floating point, one or more)*: 2N arguments (`x1`, `y1`, `x2`, `y2`, ..., `xN`, `yN`) giving vertices of an N-sided polygon
- Return value
  - *(boolean)*: true iff test point is inside, or on the border of, the polygon
- Examples:
  - `isInside(0.5,0.5, 0,0, 0,1, 1,1, 1,0) = true`
  - `isInside(0,0, array(10,20, 20,20, 20,10)) = false`

B.1.18 Sky

Functions useful for working with shapes on a sphere. All angles are expressed in degrees.

`skyDistance( lon1, lat1, lon2, lat2 )`

Calculates the separation (distance around a great circle) of two points on the sky in degrees.

This function is identical to `skyDistanceDegrees` in class `CoordsDegrees`.

- Parameters:
  - `lon1` *(floating point)*: point 1 longitude in degrees
  - `lat1` *(floating point)*: point 1 latitude in degrees
  - `lon2` *(floating point)*: point 2 longitude in degrees

  - `lat2` *(floating point)*: point 2 latitude in degrees
- Return value
  - *(floating point)*: angular distance between point 1 and point 2 in degrees

`midLon( lon1, lon2 )`

Determines the longitude mid-way between two given longitudes. In most cases this is just the mean of the two values, but this function copes correctly with the case where the given values straddle the lon=0 line.

- Parameters:
  - `lon1` *(floating point)*: first longitude in degrees
  - `lon2` *(floating point)*: second longitude in degrees
- Return value
  - *(floating point)*: longitude midway between the given values
- Examples:
  - `midLon(204.0, 203.5) = 203.75`
  - `midLon(2, 359) = 0.5`

`midLat( lat1, lat2 )`

Determines the latitude midway between two given latitudes. This simply returns the mean of the two values, but is supplied for convenience to use alongside the `midLon` function.

- Parameters:
  - `lat1` *(floating point)*: first latitude in degrees
  - `lat2` *(floating point)*: second latitude in degrees
- Return value
  - *(floating point)*: latitude midway between the given values
- Example:
  - `midLat(23.5, 24.0) = 23.75`

`inSkyEllipse( lon0, lat0, lonCenter, latCenter, rA, rB, posAng )`

Tests whether a given sky position is inside a given ellipse.

- Parameters:
  - `lon0` *(floating point)*: test point longitude in degrees
  - `lat0` *(floating point)*: test point latitude in degrees
  - `lonCenter` *(floating point)*: ellipse center longitude in degrees
  - `latCenter` *(floating point)*: ellipse center latitude in degrees
  - `rA` *(floating point)*: ellipse first principal radius in degrees
  - `rB` *(floating point)*: ellipse second principal radius in degrees
  - `posAng` *(floating point)*: position angle in degrees from the North pole to the primary axis of the ellipse in the direction of increasing longitude
- Return value
  - *(boolean)*: true iff test point is inside, or on the border of, the ellipse

`inSkyPolygon( lon0, lat0, lonLats, ... )`

Tests whether a given sky position is inside the polygon defined by a given set of vertices. The bounding lines of the polygon are the minor arcs of great circles between adjacent vertices, with an extra line assumed between the first and last supplied vertex. The interior of the polygon is defined to be the smaller of the two regions separated by the boundary. The vertices are specified as a sequence of lon_i, lat_i pairs.

The implementation of this point-in-polygon function is mostly correct, but may not be bulletproof. It ought to work for relatively small regions anywhere on the sky, but for instance it may get the sense wrong for regions that extend to cover both poles.

- Parameters:
  - `lon0` *(floating point)*: test point latitude in degrees
  - `lat0` *(floating point)*: test point longitude in degrees
  - `lonLats` *(floating point, one or more)*: 2N arguments (`lon1`, `lat1`, `lon2`, `lat2`, ..., `lonN`, `latN`) giving (longitude, latitude) vertices of an N-sided polygon in degrees
- Return value
  - *(boolean)*: true iff test point is inside, or on the border of, the polygon

B.1.19 Strings

String manipulation and query functions.

`concat( strings, ... )`

Concatenates multiple values into a string. In some cases the same effect can be achieved by writing `s1+s2+...`, but this method makes sure that values are converted to strings, with the blank value invisible.

- Parameters:
  - `strings` *(Object, one or more)*: one or more strings
- Return value
  - *(String)*: concatenation of input strings, without separators
- Examples:
  - `concat("blue", "moon") = "bluemoon"`
  - `concat("1", 2, 3, "4") = "1234"`
  - `concat("Astro", null, "Physics") = "AstroPhysics"`

`join( separator, words, ... )`

Joins multiple values into a string, with a given separator between each pair.

- Parameters:
  - `separator` *(String)*: string to insert between adjacent words
  - `words` *(Object, one or more)*: one or more values to join
- Return value
  - *(String)*: input values joined together with `separator`
- Examples:
  - `join("<->", "alpha", "beta", "gamma") = "alpha<->beta<->gamma"`
  - `join(" ", 1, "brown", "mouse") = "1 brown mouse"`

`equals( s1, s2 )`

Determines whether two strings are equal. Note you should use this function instead of `s1==s2`, which can (for technical reasons) return false even if the strings are the same.

- Parameters:
  - `s1` *(String)*: first string
  - `s2` *(String)*: second string
- Return value
  - *(boolean)*: true if s1 and s2 are both blank, or have the same content

`equalsIgnoreCase( s1, s2 )`

Determines whether two strings are equal apart from possible upper/lower case distinctions.

- Parameters:
  - `s1` *(String)*: first string
  - `s2` *(String)*: second string
- Return value
  - *(boolean)*: true if s1 and s2 are both blank, or have the same content apart from case folding
- Examples:
  - `equalsIgnoreCase("Cygnus", "CYGNUS") = true`
  - `equalsIgnoreCase("Cygnus", "Andromeda") = false`

`startsWith( whole, start )`

Determines whether a string starts with a certain substring.

- Parameters:
  - `whole` *(String)*: the string to test
  - `start` *(String)*: the sequence that may appear at the start of `whole`
- Return value
  - *(boolean)*: true if the first few characters of `whole` are the same as `start`
- Example:
  - `startsWith("CYGNUS X-1", "CYG") = true`

`endsWith( whole, end )`

Determines whether a string ends with a certain substring.

- Parameters:
  - `whole` *(String)*: the string to test
  - `end` *(String)*: the sequence that may appear at the end of `whole`
- Return value
  - *(boolean)*: true if the last few characters of `whole` are the same as `end`
- Example:
  - `endsWith("M32", "32") = true`

`contains( whole, sub )`

Determines whether a string contains a given substring.

- Parameters:
  - `whole` *(String)*: the string to test
  - `sub` *(String)*: the sequence that may appear within `whole`
- Return value
  - *(boolean)*: true if the sequence `sub` appears within `whole`
- Example:
  - `contains("Vizier", "izi") = true`

`length( str )`

Returns the length of a string in characters.

- Parameters:
  - `str` *(String)*: string
- Return value
  - *(integer)*: number of characters in `str`
- Example:
  - `length("M34") = 3`

`split( words )`

Splits a string into an array of space-separated words. One or more spaces separates each word from the next. Leading and trailing spaces are ignored.

The result is an array of strings, and if you want to use the individual elements you need to use square-bracket indexing, with `[0]` representing the first object

- Parameters:
  - `words` *(String)*: string with embedded spaces delimiting the words
- Return value
  - *(array of String)*: array of the separate words; you can extract the individual words from the result using square bracket indexing
- Examples:
  - `split("211:54:01  +29:33:41")` gives a 2-element array, first element is `"211:54:01"` and second element is `"+29:33:41"`.
  - `split(" cat dog cow ")[1] = "dog"`

`split( words, regex )`

Splits a string into an array of words separated by a given regular expression.

The result is an array of strings, and if you want to use the individual elements you need to use square-bracket indexing, with `[0]` representing the first object

- Parameters:
  - `words` *(String)*: string with multiple parts
  - `regex` *(String)*: regular expression delimiting the different words in the `words` parameter

- Return value
  - *(array of String)*: array of the separate words; you can extract the individual words from the result using square bracket indexing
- Examples:
  - `split("cat, dog, cow", ", *")` gives a 3-element string array.
  - `split("23.0, 45.92", ", ")[0] = "23.0"`
  - `parseDouble(split("23.0, 45.92", ", ")[0]) = 23`

`matches( str, regex )`

Tests whether a string matches a given regular expression.

- Parameters:
  - `str` *(String)*: string to test
  - `regex` *(String)*: regular expression string
- Return value
  - *(boolean)*: true if `regex` matches `str` anywhere
- Example:
  - `matches("Hubble", "ub") = true`

`matchGroup( str, regex )`

Returns the first grouped expression matched in a string defined by a regular expression. A grouped expression is one enclosed in parentheses.

- Parameters:
  - `str` *(String)*: string to match against
  - `regex` *(String)*: regular expression containing a grouped section
- Return value
  - *(String)*: contents of the matched group (or null, if `regex` didn't match `str`)
- Example:
  - `matchGroup("NGC28948b","NGC([0-9]*)") = "28948"`

`replaceFirst( str, regex, replacement )`

Replaces the first occurrence of a regular expression in a string with a different substring value.

- Parameters:
  - `str` *(String)*: string to manipulate
  - `regex` *(String)*: regular expression to match in `str`
  - `replacement` *(String)*: replacement string
- Return value
  - *(String)*: same as `str`, but with the first match (if any) of `regex` replaced by `replacement`
- Example:
  - `replaceFirst("Messier 61", "Messier ", "M-") = "M-61"`

`replaceAll( str, regex, replacement )`

Replaces all occurrences of a regular expression in a string with a different substring value.

- Parameters:
  - `str` *(String)*: string to manipulate
  - `regex` *(String)*: regular expression to match in `str`
  - `replacement` *(String)*: replacement string
- Return value
  - *(String)*: same as `str`, but with all matches of `regex` replaced by `replacement`
- Example:
  - `replaceAll("1-2--3---4","--*","x") = "1x2x3x4"`

`substring( str, startIndex )`

Returns the last part of a given string. The substring begins with the character at the specified index and extends to the end of this string.

- Parameters:
  - `str` *(String)*: the input string
  - `startIndex` *(integer)*: the beginning index, inclusive
- Return value
  - *(String)*: last part of `str`, omitting the first `startIndex` characters
- Example:
  - `substring("Galaxy", 2) = "laxy"`

`substring( str, startIndex, endIndex )`

Returns a substring of a given string. The substring begins with the character at `startIndex` and continues to the character at index `endIndex-1` Thus the length of the substring is `endIndex-startIndex`.

- Parameters:
  - `str` *(String)*: the input string
  - `startIndex` *(integer)*: the beginning index, inclusive
  - `endIndex` *(integer)*: the end index, inclusive
- Return value
  - *(String)*: substring of `str`
- Example:
  - `substring("Galaxy", 2, 5) = "lax"`

`toUpperCase( str )`

Returns an uppercased version of a string.

- Parameters:
  - `str` *(String)*: input string
- Return value
  - *(String)*: uppercased version of `str`
- Example:

  - `toUpperCase("Universe") = "UNIVERSE"`

`toLowerCase( str )`

Returns an lowercased version of a string.

- Parameters:
  - `str` *(String)*: input string
- Return value
  - *(String)*: lowercased version of `str`
- Example:
  - `toLowerCase("Universe") = "universe"`

`trim( str )`

Trims whitespace from both ends of a string.

- Parameters:
  - `str` *(String)*: input string
- Return value
  - *(String)*: str with any spaces trimmed from start and finish
- Examples:
  - `trim(" some text ") = "some text"`
  - `trim("some text") = "some text"`

`padWithZeros( value, ndigit )`

Takes an integer argument and returns a string representing the same numeric value but padded with leading zeros to a specified length.

- Parameters:
  - `value` *(long integer)*: numeric value to pad
  - `ndigit` *(integer)*: the number of digits in the resulting string
- Return value
  - *(String)*: a string evaluating to the same as `value` with at least `ndigit` characters
- Example:
  - `padWithZeros(23,5) = "00023"`

`desigToRa( designation )`

Attempts to determine the ICRS Right Ascension from an IAU-style designation such as "`2MASS J04355524+1630331`" following the specifications in the document https://cdsweb.u-strasbg.fr/Dic/iau-spec.html (https://cdsweb.u-strasbg.fr/Dic/iau-spec.html).

Note: this function should be used with considerable care. Such designators are intended for object identification and not for communicating sky positions, so that the resulting positions are likely to lack precision, and may be inaccurate. If positional information is available from other sources, it should almost certainly be used instead. But if there's no other choice, this may be used as a fallback.

Note also that a designator with no coordsystem-specific flag character (a leading "`J`", "`B`" or "`G`") is considered to be B1950, *not* J2000.

- Parameters:
  - `designation` *(String)*: designation string in IAU format
- Return value
  - *(floating point)*: ICRS right ascension in degreees, or blank if no position can be decoded
- Examples:
  - `desigToRa("2MASS J04355524+1630331") = 60.98016`
  - `desigToRa("PSR J120000.0+450000.0") = 180`
  - `desigToDec("PSR B120000.0+450000.0") = 180.639096`
  - `desigToRa("PN G001.2-00.3") = 267.403`
  - `desigToRa("NGC 4993") = NaN`

`desigToDec( designation )`

Attempts to determine the ICRS Declination from an IAU-style designation such as "`2MASS J04355524+1630331`" following the specifications in the document https://cdsweb.u-strasbg.fr/Dic/iau-spec.html (https://cdsweb.u-strasbg.fr/Dic/iau-spec.html).

Note: this function should be used with considerable care. Such designators are intended for object identification and not for communicating sky positions, so that the resulting positions are likely to lack precision, and may be inaccurate. If positional information is available from other sources, it should almost certainly be used instead. But if there's no other choice, this may be used as a fallback.

Note also that a designator with no coordsystem-specific flag character (a leading "`J`", "`B`" or "`G`") is considered to be B1950, *not* J2000.

- Parameters:
  - `designation` *(String)*: designation string in IAU format
- Return value
  - *(floating point)*: ICRS declination in degrees, or blank if no position can be decoded
- Examples:
  - `desigToDec("2MASS J04355524+1630331") = 16.50919`
  - `desigToDec("PSR J120000.0+450000.0") = 45`
  - `desigToDec("PSR B120000.0+450000.0") = 44.72167`
  - `desigToDec("PN G001.2-00.3") = -28.06457`
  - `desigToDec("NGC 4993") = NaN`

`desigToIcrs( designation )`

Attempts to decode an IAU-style designation such as "`2MASS J04355524+1630331`" to determine its sky position, following the specifications in the document https://cdsweb.u-strasbg.fr/Dic/iau-spec.html (https://cdsweb.u-strasbg.fr/Dic/iau-spec.html).

Obviously, this only works where the *sequence* part of the designation takes one of the family of coordinate-based forms.

Note: this function should be used with considerable care. Such designators are intended for object identification and not for communicating sky positions, so that the resulting positions are likely to lack precision, and may be inaccurate. If positional information is available from other sources, it should almost certainly be used instead. But if there's no other choice, this may be used as a fallback.

Note also that a designator with no coordsystem-specific flag character (a leading "`J`", "`B`" or

`G`") is considered to be B1950, *not* J2000.

- Parameters:
  - `designation` *(String)*: designation string in IAU format
- Return value
  - *(array of floating point)*: 2-element array giving ICRS (RA,Dec) in degrees, or `null` if no position can be decoded

B.1.20 Tilings

Pixel tiling functions for the celestial sphere.

The `k` parameter for the HEALPix functions is the HEALPix order, which can be in the range 0<=k<=29. This is the logarithm to base 2 of the HEALPix NSIDE parameter. At order `k`, there are 12*4^k pixels on the sphere.

`htmIndex( level, lon, lat )`

Gives the HTM (Hierachical Triangular Mesh) pixel index for a given sky position.

- Parameters:
  - `level` *(integer)*: HTM level
  - `lon` *(floating point)*: longitude in degrees
  - `lat` *(floating point)*: latitude in degrees
- Return value
  - *(long integer)*: pixel index
- See Also:
  - HTM web site

`healpixNestIndex( k, lon, lat )`

Gives the pixel index for a given sky position in the HEALPix NEST scheme.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
  - `lon` *(floating point)*: longitude in degrees
  - `lat` *(floating point)*: latitude in degrees
- Return value
  - *(long integer)*: pixel index
- See Also:
  - HEALPix web site

`healpixRingIndex( k, lon, lat )`

Gives the pixel index for a given sky position in the HEALPix RING scheme.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
  - `lon` *(floating point)*: longitude in degrees

    - `lat` *(floating point)*: latitude in degrees
- Return value
    - *(long integer)*: pixel index
- See Also:
    - HEALPix web site

`healpixNestLon( k, index )`

Returns the longitude of the approximate center of the tile with a given index in the HEALPix NEST scheme.

Note: the `index` parameter is 0-based, unlike the table row index special token `$index` (a.k.a. `$0`), which is 1-based. So if the HEALpix index is implicitly determined by the table row, the value `$index-1` should be used.

- Parameters:
    - `k` *(integer)*: HEALPix order (0..29)
    - `index` *(long integer)*: healpix index
- Return value
    - *(floating point)*: longitude in degrees
- See Also:
    - HEALPix web site

`healpixNestLat( k, index )`

Returns the latitude of the approximate center of the tile with a given index in the HEALPix NEST scheme.

Note: the `index` parameter is 0-based, unlike the table row index special token `$index` (a.k.a. `$0`), which is 1-based. So if the HEALpix index is implicitly determined by the table row, the value `$index-1` should be used.

- Parameters:
    - `k` *(integer)*: HEALPix order (0..29)
    - `index` *(long integer)*: healpix index
- Return value
    - *(floating point)*: latitude in degrees
- See Also:
    - HEALPix web site

`healpixRingLon( k, index )`

Returns the longitude of the approximate center of the tile with a given index in the HEALPix RING scheme.

Note: the `index` parameter is 0-based, unlike the table row index special token `$index` (a.k.a. `$0`), which is 1-based. So if the HEALpix index is implicitly determined by the table row, the value `$index-1` should be used.

- Parameters:
    - `k` *(integer)*: HEALPix order (0..29)
    - `index` *(long integer)*: healpix index

- Return value
  - *(floating point)*: longitude in degrees
- See Also:
  - HEALPix web site

`healpixRingLat( k, index )`

Returns the latitude of the approximate center of the tile with a given index in the HEALPix RING scheme.

Note: the `index` parameter is 0-based, unlike the table row index special token `$index` (a.k.a. `$0`), which is 1-based. So if the HEALpix index is implicitly determined by the table row, the value `$index-1` should be used.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
  - `index` *(long integer)*: healpix index
- Return value
  - *(floating point)*: latitude in degrees
- See Also:
  - HEALPix web site

`healpixNestToRing( k, nestIndex )`

Converts a healpix tile index from the NEST to the RING scheme at a given order.

Note: the `nestIndex` parameter is 0-based, unlike the table row index special token `$index` (a.k.a. `$0`), which is 1-based. So if the HEALpix index is implicitly determined by the table row, the value `$index-1` should be used.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
  - `nestIndex` *(long integer)*: pixel index in NEST scheme
- Return value
  - *(long integer)*: pixel index in RING scheme
- See Also:
  - HEALPix web site

`healpixRingToNest( k, ringIndex )`

Converts a healpix tile index from the RING to the NEST scheme at a given order.

Note: the `ringIndex` parameter is 0-based, unlike the table row index special token `$index` (a.k.a. `$0`), which is 1-based. So if the HEALpix index is implicitly determined by the table row, the value `$index-1` should be used.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
  - `ringIndex` *(long integer)*: pixel index in RING scheme
- Return value
  - *(long integer)*: pixel index in NEST scheme

- See Also:
  - HEALPix web site

healpixK(pixelsize)

Gives the HEALPix resolution parameter suitable for a given pixel size. This `k` value (also variously known as order, level, depth) is the logarithm to base 2 of the Nside parameter.

- Parameters:
  - `pixelsize` *(floating point)*: pixel size in degrees
- Return value
  - *(integer)*: HEALPix order `k`
- See Also:
  - HEALPix web site

healpixResolution(k)

Gives the approximate resolution in degrees for a given HEALPix resolution parameter `k` This `k` value is the logarithm to base 2 of the Nside parameter.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
- Return value
  - *(floating point)*: approximate angular resolution in degrees

healpixSteradians(k)

Returns the solid angle in steradians of each HEALPix pixel at a given order.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
- Return value
  - *(floating point)*: pixel size in steradians
- Examples:
  - `healpixSteradians(5) = 0.0010226538585904274`
  - `4*PI/healpixSteradians(0) = 12.0`

healpixSqdeg(k)

Returns the solid angle in square degrees of each HEALPix pixel at a given order.

- Parameters:
  - `k` *(integer)*: HEALPix order (0..29)
- Return value
  - *(floating point)*: pixel size in steradians
- Examples:
  - `healpixSqdeg(5) = 3.357174580844667`
  - `round(12 * healpixSqdeg(0)) = 41253`

`steradiansToSqdeg( sr )`

Converts a solid angle from steradians to square degrees.

The unit sphere is 4*PI steradians = 360*360/PI square degrees.

- Parameters:
  - `sr` *(floating point)*: quantity in steradians
- Return value
  - *(floating point)*: quantity in sqare degrees
- Example:
  - `round(steradiansToSqdeg(4*PI)) = 41253`

`sqdegToSteradians( sqdeg )`

Converts a solid angle from square degrees to steradians.

The unit sphere is 4*PI steradians = 360*360/PI square degrees.

- Parameters:
  - `sqdeg` *(floating point)*: quantity in square degrees
- Return value
  - *(floating point)*: quantity in steradians
- Example:
  - `round(sqdegToSteradians(41253)/PI) = 4`

`htmLevel( pixelsize )`

Gives the HTM `level` parameter suitable for a given pixel size.

- Parameters:
  - `pixelsize` *(floating point)*: required resolution in degrees
- Return value
  - *(integer)*: HTM level parameter

`htmResolution( level )`

Gives the approximate resolution in degrees for a given HTM depth level.

- Parameters:
  - `level` *(integer)*: HTM depth
- Return value
  - *(floating point)*: approximate angular resolution in degrees

`SQDEG`

Solid angle in steradians corresponding to 1 square degree.

B.1.21 Times

Functions for conversion of time values between various forms. The forms used are

Modified Julian Date (MJD)
A continuous measure in days since midnight at the start of 17 November 1858. Based on UTC.

Julian Day (JD)
MJD plus a fixed offset of 2400000.5. The number of days since the notional creation of the universe, midday on 1 Jan 4713 BC.

ISO 8601
A string representation of the form `yyyy-mm-ddThh:mm:ss.s`, where the `T` is a literal character (a space character may be used instead). Based on UTC.

Julian Epoch
A continuous measure based on a Julian year of exactly 365.25 days. For approximate purposes this resembles the fractional number of years AD represented by the date. Sometimes (but not here) represented by prefixing a 'J'; J2000.0 is defined as 2000 January 1.5 in the TT timescale.

Besselian Epoch
A continuous measure based on a tropical year of about 365.2422 days. For approximate purposes this resembles the fractional number of years AD represented by the date. Sometimes (but not here) represented by prefixing a 'B'.

Decimal Year
Fractional number of years AD represented by the date. 2000.0, or equivalently 1999.99recurring, is midnight at the start of the first of January 2000. Because of leap years, the size of a unit depends on what year it is in.

Therefore midday on the 25th of October 2004 is `2004-10-25T12:00:00` in ISO 8601 format, 53303.5 as an MJD value, 2004.81588 as a Julian Epoch and 2004.81726 as a Besselian Epoch.

Currently this implementation cannot be relied upon to better than a millisecond.

`isoToMjd( isoDate )`

Converts an ISO8601 date string to Modified Julian Date. The basic format of the `isoDate` argument is `yyyy-mm-ddThh:mm:ss.s`, though some deviations from this form are permitted:

- The '`T`' which separates date from time can be replaced by a space
- The seconds, minutes and/or hours can be omitted
- The decimal part of the seconds can be any length, and is optional
- The '`mm-dd`' part may be replaced by a 3-digit day of year '`ddd`'
- A '`Z`' (which indicates UTC) may be appended to the time

Some legal examples are therefore: "`1994-12-21T14:18:23.2`", "`1968-01-14`", "`2112-05-25 16:45Z`", and "`1987-172T22:12`".

- Parameters:
  - `isoDate` *(String)*: date in ISO 8601 format
- Return value
  - *(floating point)*: modified Julian date corresponding to `isoDate`
- Examples:
  - `isoToMjd("2004-10-25T18:00:00") = 53303.75`
  - `isoToMjd("1970-01-01") = 40587.0`

`dateToMjd( year, month, day, hour, min, sec )`

Converts a calendar date and time to Modified Julian Date.

- Parameters:
  - `year` *(integer)*: year AD
  - `month` *(integer)*: index of month; January is 1, December is 12
  - `day` *(integer)*: day of month (the first day is 1)
  - `hour` *(integer)*: hour (0-23)
  - `min` *(integer)*: minute (0-59)
  - `sec` *(floating point)*: second (0<=sec<60)
- Return value
  - *(floating point)*: modified Julian date corresponding to arguments
- Example:
  - `dateToMjd(1999, 12, 31, 23, 59, 59.) = 51543.99998`

`dateToMjd( year, month, day )`

Converts a calendar date to Modified Julian Date.

- Parameters:
  - `year` *(integer)*: year AD
  - `month` *(integer)*: index of month; January is 1, December is 12
  - `day` *(integer)*: day of month (the first day is 1)
- Return value
  - *(floating point)*: modified Julian date corresponding to 00:00:00 of the date specified by the arguments
- Example:
  - `dateToMjd(1999, 12, 31) = 51543.0`

`decYearToMjd( decYear )`

Converts a Decimal Year to a Modified Julian Date.

- Parameters:
  - `decYear` *(floating point)*: decimal year
- Return value
  - *(floating point)*: modified Julian Date
- Example:
  - `decYearToMjd(2000.0) = 51544.0`

`isoToUnixSec( isoDate )`

Converts an ISO8601 date string to seconds since 1970-01-01. The basic format of the `isoDate` argument is `yyyy-mm-ddThh:mm:ss.s`, though some deviations from this form are permitted:

- The '`T`' which separates date from time can be replaced by a space
- The seconds, minutes and/or hours can be omitted
- The decimal part of the seconds can be any length, and is optional
- The '`mm-dd`' part may be replaced by a 3-digit day of year '`ddd`'

- A 'Z' (which indicates UTC) may be appended to the time

Some legal examples are therefore: "`1994-12-21T14:18:23.2`", "`1968-01-14`", "`2112-05-25 16:45Z` and "`1987-172T22:12`".

- Parameters:
  - `isoDate` *(String)*: date in ISO 8601 format
- Return value
  - *(floating point)*: seconds since the Unix epoch
- Examples:
  - `isoToUnixSec("2004-10-25T18:00:00") = 1098727200`
  - `isoToUnixSec("1970-01-01") = 0`

`decYearToUnixSec( decYear )`

Converts a Decimal Year to seconds since 1970-01-01.

- Parameters:
  - `decYear` *(floating point)*: decimal year
- Return value
  - *(floating point)*: seconds since the Unix epoch
- Examples:
  - `decYearToUnixSec(2000.0) = 946684800`
  - `decYearToUnixSec(1970) = 0`

`mjdToUnixSec( mjd )`

Converts a Modified Julian Date to seconds since 1970-01-01.

- Parameters:
  - `mjd` *(floating point)*: modified Julian date
- Return value
  - *(floating point)*: seconds since the Unix epoch

`jdToUnixSec( jd )`

Converts a Julian day to seconds since 1970-01-01.

- Parameters:
  - `jd` *(floating point)*: Julian day
- Return value
  - *(floating point)*: seconds since the Unix epoch

`mjdToIso( mjd )`

Converts a Modified Julian Date value to an ISO 8601-format date-time string. The output format is `yyyy-mm-ddThh:mm:ss`. If the result predates the Common Era, the string "(BCE)" is prepended.

- Parameters:
  - `mjd` *(floating point)*: modified Julian date

- Return value
  - *(String)*: ISO 8601 format date corresponding to `mjd`
- Example:
  - `mjdToIso(53551.72917) = "2005-06-30T17:30:00"`

`mjdToDate( mjd )`

Converts a Modified Julian Date value to an ISO 8601-format date string. The output format is `yyyy-mm-dd`. If the result predates the Common Era, the string "(BCE)" is prepended.

- Parameters:
  - `mjd` *(floating point)*: modified Julian date
- Return value
  - *(String)*: ISO 8601 format date corresponding to `mjd`
- Example:
  - `mjdToDate(53551.72917) = "2005-06-30"`

`mjdToTime( mjd )`

Converts a Modified Julian Date value to an ISO 8601-format time-only string. The output format is `hh:mm:ss`.

- Parameters:
  - `mjd` *(floating point)*: modified Julian date
- Return value
  - *(String)*: ISO 8601 format time corresponding to `mjd`
- Example:
  - `mjdToTime(53551.72917) = "17:30:00"`

`mjdToDecYear( mjd )`

Converts a Modified Julian Date to Decimal Year.

- Parameters:
  - `mjd` *(floating point)*: modified Julian Date
- Return value
  - *(floating point)*: decimal year
- Example:
  - `mjdToDecYear(0.0) = 1858.87671`

`formatMjd( mjd, format )`

Converts a Modified Julian Date value to a date using a customisable date format. The format is as defined by the `java.text.SimpleDateFormat` (http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html) class. The default output corresponds to the string "`yyyy-MM-dd'T'HH:mm:ss`"

Note that the output from certain formatting characters (such as `MMM` for month, `EEE` for day of week) is dependent on your locale (system language settings). The output time zone however always corresponds to UTC.

- Parameters:
  - `mjd` *(floating point)*: modified Julian date
  - `format` *(String)*: formatting patttern
- Return value
  - *(String)*: custom formatted time corresponding to `mjd`
- Examples:
  - `formatMjd(50000.3, "EEE dd, MMM, yy") = "Tue 10 Oct, 95"`
  - `formatMjd(50000.1234, "'time 'H:mm:ss.SSS") = "time 2:57:41.760"`

`jdToMjd( jd )`

Converts a Julian Day to Modified Julian Date. The calculation is simply `jd-2400000.5`.

- Parameters:
  - `jd` *(floating point)*: Julian day number
- Return value
  - *(floating point)*: MJD value

`mjdToJd( mjd )`

Converts a Modified Julian Date to Julian Day. The calculation is simply `jd+2400000.5`.

- Parameters:
  - `mjd` *(floating point)*: MJD value
- Return value
  - *(floating point)*: Julian day number

`mjdToJulian( mjd )`

Converts a Modified Julian Date to Julian Epoch. For approximate purposes, the result of this routine consists of an integral part which gives the year AD and a fractional part which represents the distance through that year, so that for instance 2000.5 is approximately 1 July 2000.

- Parameters:
  - `mjd` *(floating point)*: modified Julian date
- Return value
  - *(floating point)*: Julian epoch
- Example:
  - `mjdToJulian(0.0) = 1858.87885`

`julianToMjd( julianEpoch )`

Converts a Julian Epoch to Modified Julian Date. For approximate purposes, the argument of this routine consists of an integral part which gives the year AD and a fractional part which represents the distance through that year, so that for instance 2000.5 is approximately 1 July 2000.

- Parameters:
  - `julianEpoch` *(floating point)*: Julian epoch

- Return value
  - *(floating point)*: modified Julian date
- Example:
  - `julianToMjd(2000.0) = 51544.5`

`mjdToBesselian( mjd )`

Converts Modified Julian Date to Besselian Epoch. For approximate purposes, the result of this routine consists of an integral part which gives the year AD and a fractional part which represents the distance through that year, so that for instance 1950.5 is approximately 1 July 1950.

- Parameters:
  - `mjd` *(floating point)*: modified Julian date
- Return value
  - *(floating point)*: Besselian epoch
- Example:
  - `mjdToBesselian(0.0) = 1858.87711`

`besselianToMjd( besselianEpoch )`

Converts Besselian Epoch to Modified Julian Date. For approximate purposes, the argument of this routine consists of an integral part which gives the year AD and a fractional part which represents the distance through that year, so that for instance 1950.5 is approximately 1 July 1950.

- Parameters:
  - `besselianEpoch` *(floating point)*: Besselian epoch
- Return value
  - *(floating point)*: modified Julian date
- Example:
  - `besselianToMjd(1950.0) = 33281.92346`

`unixMillisToMjd( unixMillis )`

Converts from milliseconds since the Unix epoch (1970-01-01T00:00:00) to a modified Julian date value

- Parameters:
  - `unixMillis` *(long integer)*: milliseconds since the Unix epoch
- Return value
  - *(floating point)*: modified Julian date

`mjdToUnixMillis( mjd )`

Converts from modified Julian date to milliseconds since the Unix epoch (1970-01-01T00:00:00).

- Parameters:
  - `mjd` *(floating point)*: modified Julian date

- Return value
  - *(long integer)*: milliseconds since the Unix epoch

`MJD_OFFSET`
JD value for MJD = 0 (=2400000.5).

B.1.22 TrigDegrees

Standard trigonometric functions with angles in degrees.

`sinDeg( theta )`
Sine of an angle.

- Parameters:
  - `theta` *(floating point)*: an angle, in degrees
- Return value
  - *(floating point)*: the sine of the argument

`cosDeg( theta )`
Cosine of an angle.

- Parameters:
  - `theta` *(floating point)*: an angle, in degrees
- Return value
  - *(floating point)*: the cosine of the argument

`tanDeg( theta )`
Tangent of an angle.

- Parameters:
  - `theta` *(floating point)*: an angle, in degrees
- Return value
  - *(floating point)*: the tangent of the argument.

`asinDeg( x )`
Arc sine. The result is in the range of -90 through 90.

- Parameters:
  - `x` *(floating point)*: the value whose arc sine is to be returned.
- Return value
  - *(floating point)*: the arc sine of the argument in degrees

`acosDeg( x )`
Arc cosine. The result is in the range of 0.0 through 180.

- Parameters:
  - `x` *(floating point)*: the value whose arc cosine is to be returned.
- Return value
  - *(floating point)*: the arc cosine of the argument in degrees

`atanDeg( x )`

Arc tangent. The result is in the range of -90 through 90.

- Parameters:
  - `x` *(floating point)*: the value whose arc tangent is to be returned.
- Return value
  - *(floating point)*: the arc tangent of the argument in degrees

`atan2Deg( y, x )`

Converts rectangular coordinates (`x`,`y`) to polar (`r`,`theta`). This method computes the phase `theta` by computing an arc tangent of `y/x` in the range of -180 to 180.

- Parameters:
  - `y` *(floating point)*: the ordinate coordinate
  - `x` *(floating point)*: the abscissa coordinate
- Return value
  - *(floating point)*: the `theta` component in degrees of the point (`r`,`theta`) in polar coordinates that corresponds to the point (`x`,`y`) in Cartesian coordinates.

B.1.23 URLs

Functions that construct URLs for external services. Most of the functions here just do string manipulation to build up URL strings, using knowledge of the parameters required for various services.

`urlEncode( txt )`

Performs necessary quoting of a string for it to be included safely in the data part of a URL. Alphanumeric characters and the characters underscore ("`_`"), minus sign ("`-`"), period ("`.`") and tilde ("`~`") are passed through unchanged, and any other 7-bit ASCII character is represented by a percent sign ("`%`") followed by its 2-digit hexadecimal code. Characters with values of 128 or greater are simply dropped.

- Parameters:
  - `txt` *(String)*: input (unencoded) string
- Return value
  - *(String)*: output (encoded) string
- Example:
  - `urlEncode("RR Lyr") = "RR%20Lyr"`
- See Also:

  - RFC 3986

`urlDecode( txt )`

Reverses the quoting performed by `urlEncode`. Percent-encoded sequences (`%xx`) are replaced by the ASCII character with the hexadecimal code `xx`.

- Parameters:
  - `txt` *(String)*: input (encoded) string
- Return value
  - *(String)*: output (unencoded) string
- Example:
  - `urlDecode("RR%20Lyr") = "RR Lyr"`
- See Also:
  - RFC 3986

`paramsUrl( baseUrl, nameValuePairs, ... )`

Builds a query-type URL string given a base URL and a list of name, value pairs.

The parameters are encoded on the command line according to the "`application/x-www-form-urlencoded`" convention, which appends a "?" to the base URL, and then adds name=value pairs separated by "&" characters, with percent-encoding of non-URL-friendly characters. This format is used by many services that require a list of parameters to be conveyed on the URL.

- Parameters:
  - `baseUrl` *(String)*: basic URL (may or may not already contain a "?")
  - `nameValuePairs` *(String, one or more)*: an even number of arguments (or an even-length string array) giving parameter name1,value1,name2,value2,...nameN,valueN
- Return value
  - *(String)*: form-encoded URL
- Example:
  - `paramsUrl("http://x.org/", "a", "1", "b", "two", "c", "3&4") = "http://x.org/?a=1&b=two&c=3%264"`

`bibcodeUrl( bibcode )`

Maps a bibcode to the URL that will display the relevant entry in ADS (https://ui.adsabs.harvard.edu/). If the supplied string does not appear to be a bibcode, null will be returned.

If the supplied string appears to be a bibcode, it just prepends the string "`https://ui.adsabs.harvard.edu/abs/`" and performs any character escaping that is required.

- Parameters:
  - `bibcode` *(String)*: ADS-style bibcode string
- Return value
  - *(String)*: display URL pointing at bibcode record, or null if it doesn't look like a bibcode

- Example:
  - `bibcodeUrl("2018A&A...616A...2L") = "https://ui.adsabs.harvard.edu/abs/2018A%26A...616A...2L"`
- See Also:
  - http://adsabs.harvard.edu/abs_doc/help_pages/data.html

`doiUrl( doi )`

Maps a DOI (Digital Object Identifier) to its dislay URL. If the supplied string does not appear to be a DOI, null will be returned.

If the supplied string appears to be a DOI, it strips any "`doi:`" prefix if present, prepends the string "`https://doi.org/`", and performs any character escaping that is required.

- Parameters:
  - `doi` *(String)*: DOI string, with or without "doi:" prefix
- Return value
  - *(String)*: display URL pointing at DOI content, or null if it doesn't look like a DOI
- Example:
  - `doiUrl("10.3390/informatics4030018") = "https://doi.org/10.3390/informatics4030018"`
- See Also:
  - https://www.doi.org/

`arxivUrl( arxivId )`

Maps an arXiv identifier to the URL that will display its arXiv (https://arxiv.org/) web page. If the supplied string does not appear to be an arXiv identifier, null will be returned.

If the supplied string appears to be an arXiv identifier, it strips any "`arXiv:` prefix and prepends the string "`https://arxiv.org/abs/`".

- Parameters:
  - `arxivId` *(String)*: arXiv identifier, with or without "arXiv:" prefix
- Return value
  - *(String)*: display URL pointing at bibcode record, or null if it doesn't look like a bibcode
- Example:
  - `arxivUrl("arXiv:1804.09379") = "https://arxiv.org/abs/1804.09381"`
- See Also:
  - https://arxiv.org/help/arxiv_identifier

`simbadUrl( sourceId )`

Maps a source identifier to the URL of its SIMBAD (http://simbad.u-strasbg.fr/simbad/) web page. SIMBAD is the astronomical source information service run by the Centre de Données astronomiques de Strasbourg.

The string "`http://simbad.u-strasbg.fr/simbad/sim-id?Ident=`" is prepended to the given id string, and any necessary character escaping is applied. No attempt is made to validate whether the supplied string is a real source identifier, so there is no guarantee that the returned

URL will contain actual results.

- Parameters:
  - `sourceId` *(String)*: free text assumed to represent a source identifier known by SIMBAD
- Return value
  - *(String)*: URL of the Simbad web page describing the identified source
- Example:
  - `simbadUrl("Beta Pictoris") = "http://simbad.u-strasbg.fr/simbad/sim-id?Ident=Beta%20Pictoris"`

`nedUrl( sourceId )`

Maps a source identifier to the URL of its NED (http://ned.ipac.caltech.edu/) web page. NED is the NASA/IPAC Extragalactic Database.

The string "`http://ned.ipac.caltech.edu/byname?objname=`" is prepended to the given id string, and any necessary character escaping is applied. No attempt is made to validate whether the supplied string is a real source identifier, so there is no guarantee that the returned URL will contain actual results.

- Parameters:
  - `sourceId` *(String)*: free text assumed to represent a source identifier known by NED
- Return value
  - *(String)*: URL of the NED web page describing the identified source
- Example:
  - `nedUrl("NGC 3952") = "http://ned.ipac.caltech.edu/byname?objname=NGC%203952"`

`hips2fitsUrl( hipsId, fmt, raDeg, decDeg, fovDeg, npix )`

Returns the URL of a cutout from the Hips2Fits service operated by CDS. The result will be the URL of a FITS or image file resampled to order from one of the HiPS surveys available at CDS.

This function requests a square cutout using the SIN projection, which is suitable for small cutouts. If the details of this function don't suit your purposes, you can construct the URL yourself.

- Parameters:
  - `hipsId` *(String)*: identifier or partial identifier for the HiPS survey
  - `fmt` *(String)*: required output format, for instance "`fits`", "`png`", "`jpg`"
  - `raDeg` *(floating point)*: central Right Ascension (longitude) in degrees
  - `decDeg` *(floating point)*: central Declination (latitude) in degrees
  - `fovDeg` *(floating point)*: field of view; extent of the cutout in degrees
  - `npix` *(integer)*: extent of the cutout in pixels (width=height=npix)
- Return value
  - *(String)*: URL of the required cutout
- See Also:
  - http://alasky.u-strasbg.fr/hips-image-services/hips2fits

B.1.24 VO

Virtual Observatory-specific functions. Some of these are for rather technical purposes.

The UCD parsing functions are based on Grégory Mantelet's library Ucidy (https://github.com/gmantele/ucidy) corresponding to UCD1+ 1.5 (https://www.ivoa.net/documents/UCD1+/20210616/), and the VOUnit parsing functions are based on Norman Gray's library Unity (https://purl.org/nxg/dist/unity) corresponding to VOUnits 1.0 (https://www.ivoa.net/documents/VOUnits/20140523/).

TFCat refers to the Time-Frequency Radio Catalogues format, TFCat 1.0 (https://doi.org/10.25935/6068-8528).

`ucdStatus( ucd )`

Returns a token string giving the category of UCD compliance. UCD1+, UCD1 and SIAv1 ("VOX:")-style UCDs are recognised.

Possible return values are currently:

- "OK": conforms to UCD1+ standard
- "UCD1": looks like a UCD1
- "VOX": is in VOX: namespace introduced by SIAv1
- "BAD_SYNTAX": not a UCD1 and cannot be parsed according to UCD1+
- "BAD_SEQUENCE": UCD words violate UCD1+ sequence rules
- "UNKNOWN_WORD": follows UCD1+ syntax rules but contains non-UCD1+ atom
- "NAMESPACE": follows UCD1+ syntax but contains atoms in non-standard namespace
- "DEPRECATED": contains deprecated UCD1+ words

In the case of non-compliant values, more information can be found using the `ucdMessage` function.

- Parameters:
  - `ucd` *(String)*: UCD string
- Return value
  - *(String)*: "OK" for conformant UCD1+, otherwise some other value

`ucdMessage( ucd )`

Returns a human-readable message associated with the parsing of a UCD. This is expected to be empty for a fully compliant UCD, and may contain error messages or advice otherwise.

- Parameters:
  - `ucd` *(String)*: UCD value
- Return value
  - *(String)*: message text

`vounitStatus( unit )`

Returns a token string giving the category of VOUnits compliance.

Possible return values are currently:

- "OK": conforms to VOUnits syntax

- "UNKNOWN_UNIT": parsed as VOUnit but contains unknown base unit(s)
- "GUESSED_UNIT": parsed as VOUnit but contains unknown, though guessable, base unit(s)
- "BAD_SYNTAX": cannot be parsed as VOUnit syntax
- "PARSE_ERROR": unexpected error during parsing
- "USAGE": violates VOUnit usage constraints
- "WHITESPACE": legal VOUnit except that it contains whitespace, which is not allowed by VOUnits

In the case of non-compliant values, more information can be found using the `vounitMessage` function.

- Parameters:
  - `unit` *(String)*: unit string
- Return value
  - *(String)*: "OK" for conformant VOUnits, otherwise some other value

`vounitMessage( unit )`

Returns a human-readable message associated with the parsing of a unit string. This is expected to be empty for a string fully compliant with VOUnits, and may contain error messages or additional information otherwise.

- Parameters:
  - `unit` *(String)*: unit string
- Return value
  - *(String)*: message text

`tfcatStatus( tfcat )`

Returns a token string giving the category of TFCat compliance.

Possible return values are currently:

- "OK": conforms to TFCat syntax
- "ERROR": parsed as TFCat, but with one or more errors
- "FAIL": could not be parsed as a TFCat object

For non-OK values, more information can be found using the `tfcatMessage` function.

- Parameters:
  - `tfcat` *(String)*: JSON string giving TFCat text
- Return value
  - *(String)*: "OK" for conformant TFCat, otherwise some other value

`tfcatMessage( tfcat )`

Returns a human-readable message associated with the parsing of a TFCat text. This will be empty for a string fully compliant with the TFCat standard, but will contain error messages otherwise.

- Parameters:
  - `tfcat` *(String)*: JSON string giving TFCat text
- Return value

- *(String)*: message text

B.2 Activation Functions

The following functions can be used only for defining custom Activation Actions (Section 8) - they mostly deal with causing something to happen, such as popping up an image display window. They generally return a short string, which will be logged to the user to give an indication of what happened (or didn't happen, or should have happened). The documentation in the following subsections is also available from within TOPCAT in the Available Functions Window.

Some of these items offer rather out of date functionality. They may be withdrawn (replaced by better alternatives) in future releases.

B.2.1 BasicImageDisplay

Functions for display of graphics-format images in a no-frills viewing window (an `ImageWindow`). Supported image formats include GIF, JPEG, PNG and FITS, which may be compressed.

`displayBasicImage( label, loc )`

Displays the file at a given location as an image in a graphical viewer. `label` may be any string which identifies the window for display, so that multiple images may be displayed in different windows without getting in each others' way. `loc` should be a filename or URL, pointing to an image in a format that this viewer understands.

- Parameters:
  - `label` *(String)*: identifies the window in which the image will be displayed
  - `loc` *(String)*: image location
- Return value
  - *(String)*: short log message

B.2.2 Browsers

Displays URLs in web browsers.

`basicBrowser( url )`

Displays a URL in a basic HTML viewer. This is only likely to work for HTML, text or RTF data. The browser can follow hyperlinks and has simple forward/back buttons, but lacks the sophistication of a proper WWW browser application.

- Parameters:
  - `url` *(String)*: location of the document to display
- Return value
  - *(String)*: short log message

`systemBrowser( url )`

Attempts to display a URL in the system's default web browser. Exactly what couts as the default web browser is system dependent, as is whether this function will work properly.

- Parameters:
  - `url` *(String)*: location of the document to display
- Return value
  - *(String)*: short log message

`mozilla( url )`

Displays a URL in a Mozilla web browser. Probably only works on Unix-like operating systems, and only if Mozilla is already running.

- Parameters:
  - `url` *(String)*: location of the document to display
- Return value
  - *(String)*: short log message

`firefox( url )`

Displays a URL in a Firefox web browser. Probably only works on Unix-like operating systems, and only if Firefox is already running.

- Parameters:
  - `url` *(String)*: location of the document to display
- Return value
  - *(String)*: short log message

`netscape( url )`

Displays a URL in a Netscape web browser. Probably only works on Unix-like operating systems, and only if Netscape is already running.

- Parameters:
  - `url` *(String)*: location of the document to display
- Return value
  - *(String)*: short log message

`mozalike( cmd, url )`

Displays a URL in a web browser from the Mozilla family; it must support flags of the type "`-remote openURL(` *url* `)`". Probably only works on Unix-like operating systems, and only if the browser is already running.

- Parameters:
  - `cmd` *(String)*: name or path of the browser command
  - `url` *(String)*: location of the document to display
- Return value
  - *(String)*: short log message

B.2.3 Image

Functions for display of images in a window. Supported image formats include GIF, JPEG, PNG and FITS, which may be compressed.

`displayImage( label, location )`
Displays the file at the given location in an image viewer.

- Parameters:
  - `label` *(String)*: identifies the window in which the imag will be displayed
  - `location` *(String)*: image location - may be a filename or URL
- Return value
  - *(String)*: short log message

B.2.4 Mgc

Specialist functions for use with data from the the Millennium Galaxy Survey.

`imageMgc( mgc_id )`
Displays the postage stamp FITS image for an MGC object in an image viewer.

- Parameters:
  - `mgc_id` *(integer)*: the MGC_ID number for the object
- Return value
  - *(String)*: short log string

`MGC_IMAGE_BASE`
String prepended to MGC_ID for the FITS image URL.

`MGC_IMAGE_TAIL`
String appended to MGC_ID for the FITS image URL.

B.2.5 Output

Functions for writing text to standard output. They will cause output to be written to the console. If you just want values to appear in the activation action logging window, you can just use the expression to report on its own.

`print( str )`
Outputs a string value to system output.

- Parameters:
  - `str` *(String)*: string value to output
- Return value
  - *(String)*: short report message

`print( num )`

Outputs a numeric value to system output.

- Parameters:
  - `num` *(floating point)*: numeric value to output
- Return value
  - *(String)*: short report message

B.2.6 Sdss

Specialist display functions for use with the Sloan Digital Sky Server.

`sdssShowCutout( label, ra, dec, pixels )`

Displays a colour cutout image of a specified size from the SDSS around a given sky position. The displayed image is square, a given number of (0.4arcsec) pixels on each side.

- Parameters:
  - `label` *(String)*: label for display window
  - `ra` *(floating point)*: Right Ascension in degrees
  - `dec` *(floating point)*: Declination in degrees
  - `pixels` *(integer)*: size of displayed image in SDSS pixels
- Return value
  - *(String)*: short log message

`sdssShowCutout( ra, dec, pixels, scale )`

Displays a colour cutout image of a specified size from the SDSS around a given sky position with pixels of a given size. Pixels are square, and their size on the sky is specified by the `scale` argument. The displayed image has `pixels` pixels along each side.

- Parameters:
  - `ra` *(floating point)*: Right Ascension in degrees
  - `dec` *(floating point)*: Declination in degrees
  - `pixels` *(integer)*: size of displayed image in SDSS pixels
  - `scale` *(floating point)*: pixel size in arcseconds
- Return value
  - *(String)*: short log message

`sdssShowCutout( ra, dec )`

Displays a colour cutout image of a default size from the SDSS around a given sky position. The displayed image is 128 pixels square - a pixel is 0.4arcsec.

- Parameters:
  - `ra` *(floating point)*: Right Ascension in degrees
  - `dec` *(floating point)*: Declination in degrees
- Return value
  - *(String)*: short log message

SDSS_DR2_BASE_URL
Base URL for SkyServer JPEG retrieval service, DR2.

SDSS_BASE_URL
Base URL for SkyServer JPEG retrieval service.

B.2.7 SuperCosmos

Specialist display functions for use with the SuperCOSMOS survey. These functions display cutout images from the various archives hosted at the SuperCOSMOS Sky Surveys (http://www-wfau.roe.ac.uk/sss/ (http://www-wfau.roe.ac.uk/sss/)). In most cases these cover the whole of the southern sky.

sssShowCutout(ra, dec, pixels)
Displays a cutout image in one of the available bands from the SuperCOSMOS Sky Surveys. The displayed image is square, and `pixels` pixels in the X and Y dimensions. Pixels are approximately 0.67 arcsec square. Sky coverage is complete.

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image
- Return value
  - *(String)*: short log message

sssShowCutout(ra, dec)
Displays a cutout image of default size in one of the available bands from the SuperCOSMOS Sky Surveys. Sky coverage is complete.

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
- Return value
  - *(String)*: short log message

sssShowBlue(ra, dec, pixels)
Displays a cutout image of default size from one of the blue-band surveys from SuperCOSMOS. Sky coverage is complete.

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image
- Return value
  - *(String)*: short log message

`sssShowRed( ra, dec, pixels )`

Displays a cutout image of default size from one of the red-band surveys from SuperCOSMOS. Sky coverage is complete.

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image
- Return value
  - *(String)*: short log message

`sssShowUkstB( ra, dec, pixels )`

Displays a cutout image taken from the SuperCOSMOS Sky Surveys UK Schmidt Telescope Bj-band survey. The displayed image is square, and `pixels` pixels in the X and Y dimensions. Pixels are approximately 0.67 arcsec square.

Sky coverage is -90<Dec<+2.5 (degrees).

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image
- Return value
  - *(String)*: short log message

`sssShowUkstR( ra, dec, pixels )`

Displays a cutout image taken from the SuperCOSMOS Sky Surveys UK Schmidt Telescope R-band survey. The displayed image is square, and `pixels` pixels in the X and Y dimensions. Pixels are approximately 0.67 arcsec square.

Sky coverage is -90<Dec<+2.5 (degrees).

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image
- Return value
  - *(String)*: short log message

`sssShowUkstI( ra, dec, pixels )`

Displays a cutout image taken from the SuperCOSMOS Sky Surveys UK Schmidt Telescope I-band survey. The displayed image is square, and `pixels` pixels in the X and Y dimensions. Pixels are approximately 0.67 arcsec square.

Sky coverage is -90<Dec<+2.5 (degrees).

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image

- Return value
  - *(String)*: short log message

`sssShowEsoR( ra, dec, pixels )`

Displays a cutout image taken from the SuperCOSMOS Sky Surveys ESO R-band survey. The displayed image is square, and `pixels` pixels in the X and Y dimensions. Pixels are approximately 0.67 arcsec square.

Sky coverage is -90<Dec<+2.5 (degrees).

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image
- Return value
  - *(String)*: short log message

`sssShowPossE( ra, dec, pixels )`

Displays a cutout image taken from the SuperCOSMOS Sky Surveys Palomar E-band survey. The displayed image is square, and `pixels` pixels in the X and Y dimensions. Pixels are approximately 0.67 arcsec square.

Sky coverage is -20.5<Dec<+2.5 (degrees).

- Parameters:
  - `ra` *(floating point)*: right ascension of image centre in degrees
  - `dec` *(floating point)*: declination of image centre in degrees
  - `pixels` *(integer)*: dimension of the displayed image
- Return value
  - *(String)*: short log message

`SSS_BASE_URL`

Base URL for SuperCOSMOS image cutout service.

B.2.8 System

Functions for executing shell commands on the local operating system and other system-level operations.

`exec( words, ... )`

Executes an operating system command composed of a command and one or more arguments.

Each of the `words` values is treated as a single (possibly quoted) word in a shell command. The first argument is the filename (either a full pathname, or the name of a file on the current path) of an executable file. These values can be numeric, or strings, or something else, and are converted automatically to string values.

- Parameters:
  - `words` *(Object, one or more)*: one or more words composing a shell command; first is

command and others are arguments

- Return value
  - *(String)*: short report message
- Examples:
  - `exec("/home/mbt/bin/process_obj.py", OBJ_NAME)`
  - `exec("process_skycoords.py", RA, DEC)`
  - `exec("process_sphericalcoords.sh", RA, DEC, 1.0)`

`exec( line )`

Executes a string as an operating system command. Any spaces in the string are taken to delimit words (the first word is the name of the command).

- Parameters:
  - `line` *(String)*: command line to execute
- Return value
  - *(String)*: short report message
- Example:
  - `exec("do_stuff.py " + RA + " " + DEC)`

`sleepMillis( millis )`

Waits for a specified number of milliseconds.

- Parameters:
  - `millis` *(integer)*: number of milliseconds to wait

`sleep( secs )`

Waits for a specified number of seconds.

- Parameters:
  - `secs` *(integer)*: number of seconds to wait

B.2.9 TwoQZ

Specialist functions for use with data from the the 2QZ survey. Spectral data are taken directly from the 2QZ web site at http://www.2dfquasar.org/ (http://www.2dfquasar.org/).

`image2QZ( name )`

Displays the postage stamp FITS image for a 2QZ object in an image viewer.

- Parameters:
  - `name` *(String)*: object name (NAME column)
- Return value
  - *(String)*: short log message

`jpeg2QZ( name )`

Displays the postage stamp JPEG image for a 2QZ object in an external viewer.

- Parameters:
  - `name` *(String)*: object name (NAME column)
- Return value
  - *(String)*: short log message

`get2qzSubdir( name )`
Returns the name of the subdirectory (such as "ra03_04") for a given 2QZ object name (ID).

- Parameters:
  - `name` *(String)*: ID of object within the 2QZ catalogue (like J120437.7-021003)
- Return value
  - *(String)*: subdirectory name

`TWOQZ_SPEC_BASE`
String prepended to the object NAME for the FITS spectra file URL.

`TWOQZ_SPEC_TAIL`
String appended to the object NAME for the FITS spectra file URL.

`TWOQZ_FITS_IMAGE_BASE`
String prepended to the object NAME for the FITS postage stamp URL.

`TWOQZ_FITS_IMAGE_TAIL`
String appended to the object NAME for the FITS postage stamp URL.

`TWOQZ_JPEG_IMAGE_BASE`
String prepended to the object NAME for the JPEG postage stamp URL.

`TWOQZ_JPEG_IMAGE_TAIL`
String appended to the object NAME for the JPEG postage stamp URL.

C Release Notes

This is TOPCAT, Tool for OPerations on Catalogues And Tables. It is a general purpose viewer and editor for astronomical tabular data.

Author
Mark Taylor (Bristol University)

Email
m.b.taylor@bristol.ac.uk

WWW
http://www.starlink.ac.uk/topcat/

User comments, suggestions, requests and bug reports to the above address are welcomed.

Related software products are

STIL
The Starlink Tables Infrastructure Library, which provides the table handling classes on which TOPCAT is based.

STILTS
The STIL Tool Set, which provides some command-line table manipulation and analysis tools based on STIL. This is a non-GUI counterpart to TOPCAT, providing many of the same facilities (matching, row selection, format conversion, plotting etc) but in a form which can be incorporated into scripts, web services, etc.

See the TOPCAT web page, http://www.starlink.ac.uk/topcat/ for the latest news and releases.

C.1 Acknowledgements

TOPCAT was initially (2003-2005) developed under the UK Starlink project (1980-2005, R.I.P.). Since then it has been supported mainly by the UK's governmentally funded research council STFC and its forerunner PPARC, but it has also been funded by a number of VO-related projects and received interim or targetted support from various other organisations. Specifically:

- Starlink, funded by PPARC
- PPARC, the UK's Particle Physics and Astronomy Research Council, grant PP/D002486/1
- STFC, the UK's Science and Technology Facilities Council, grants ST/H008470/1, ST/I00176X/1, ST/J001414/1 ST/L002388/1, ST/M000907/1, ST/R000700/1, ST/S001980/1, ST/V000454/1, ST/X002969/1 and ST/Y002032/1
- AstroGrid, the UK's VO project, funded by STFC/PPARC
- GAVO, the German Astrophysical Virtual Observatory, BMBF Bewilligungsnummer 05A08VHA
- ESA, the European Space Agency
- VOTech, an EU FP6 virtual observatory project
- AIDA, an EU FP7 virtual observatory project
- GENIUS, an EU FP7 project related to the Gaia mission
- Europlanet-2024 RI, an EU Horizon 2020 planetary science virtual observatory project, grant agreement No 871149

Some more detail is available in the FAQ. All of this support is gratefully acknowledged.

Nearly all of this work has been done in the Astrophysics Group of the School of Physics at the University of Bristol, but the Institute of Astronomy at the University of Cambridge and ARI at the University of Heidelberg have also been involved at some time.

Apart from the excellent Java 2 Standard Edition itself, the following external libraries provide parts of TOPCAT's functionality:

- JEL (Konstantin Metlov, GNU) for algebraic expression evaluation
- cds-healpix-java (F-X Pineau, CDS) for HEALPix manipulation
- Skyview in a Jar (NASA) for sky axis drawing
- JLaTeXMath (Scilab) for LaTeX typesetting in plots
- iText for PDF output
- JFreeSVG for SVG output
- EPSGraphics2D (Jibble) for encapsulated postscript output
- VOLLT ADQL Library (Grégory Mantelet, CDS) for ADQL parsing
- MOC (CDS) Multi-Order Coverage HEALPix maps for footprint manipulation
- Snakeyaml for YAML parsing in ECSV input handler
- Ucidy (Grégory Mantelet, CDS) for UCD parsing
- Unity (Norman Gray, Glasgow) for VOUnits parsing
- JSON-java (Douglas Crockford, org.json) for JSON de/serialization
- JIDE Common Layer (Jidesoft) for double-ended slider controls
- PDS4-JParser (NASA), a few low-level classes used for PDS4 datatype parsing
- IVOARegistry (NVO) for ADQL/X->S translations in registry searches
- HTM (Sloan Digital Sky Survey) for HTM-based celestial sphere row matching (now deprecated within TOPCAT)
- Ptplot (Ptolemy) for some 2D axis plotting
- IVOA FITS Package (Sloan Digital Sky Survey) for simple display of FITS images

Contributed code from the following people is gratefully acknowledged:

- Thomas Boch (CDS), for the VizieR load dialogue.
- Gerard Lemson (MPE, Garching) for the GAVO load dialogue.
- Marco Molinaro (INAF-OATS) for the BaSTI load dialogue.

Other contributions to the software are gratefully acknowledged:

- The TOPCAT logo in use since version 4 was kindly drawn by Phil Hall.
- Initial inspiration for some of TOPCAT's features was taken from the pre-existing tools Mirage (Bell labs) and VOPlot (VO-India).
- The MacOS DMG file since v4.9 is built using Install4j.
- The sources for colour maps used in plotting are listed in Appendix A.4.7.
- Some icons were created by or adapted from Smashicons from www.flaticons.com.

TOPCAT has benefitted from, and is largely shaped by, advice, comments and feedback from very many users, testers and developers. Some of the notable contributers are these:

- Alasdair Allan (Starlink, Exeter)
- Mark Allen (CDS)
- Heinz Andernach (Guanajuato)
- Mark Bentley (ESA)
- Gabriel Bihain (MPG)
- Mark Birkinshaw (Bristol)
- Thomas Boch (CDS)
- Laurent Bourges (JMMC)
- Danny Boxhoorn (Kapteyn, Groningen)
- Malcolm Bremer (Bristol)
- Patrick Broos (Penn State)
- Johannes Buchner (MPE)
- Russ Calvert (AstroGrav)
- Baptiste Cecconi (LESIA)

- Igor Chilingarian (CfA)
- Tamara Civera (CEFCA)
- Malcolm Currie (Starlink, RAL)
- Clive Davenhall (Royal Observatory Edinburgh)
- Luke Davies (Bristol)
- Christoph Deil (MPI Heidelberg)
- Markus Demleitner (ARI Heidelberg)
- Sébastien Derriere (CDS)
- Emilio Donoso (ICATE)
- Peter Draper (Starlink, Durham)
- Heinz-Bernd Eggenstein (AEI, Hannover)
- Stéphane Erard (Observatoire de Paris)
- Jeremy Faden (Cottage Systems)
- Pierre Fernique (CDS)
- Boris Gänsicke (Warwick)
- Shashikiran Ganesh (PRL, India)
- Fedrico Garcia (RUG)
- Vincent Génot (IRAP)
- Robin Geyer (TU-Dresden)
- David Giaretta (Starlink, RAL)
- Bertrand Goldman (MPIA)
- Eduardo Gonzalez-Solares (IoA)
- Norman Gray (Starlink, Glasgow)
- Claire Greenwell (Durham)
- Phil Hall (Son of Ken)
- Haggis Harris (Bath)
- Evanthia Hatziminaoglou (ESO)
- Greg Hennessy (USNO)
- Avon Huxor (Bristol/ARI)
- Jonathan Irwin (Cambridge)
- Tim Jenness (JACH)
- JJ Kavelaars (NRC)
- Grant Kennedy (IoA)
- Sergei Klioner (Dresden)
- Christian Knigge (Southampton)
- Marina Kounkel (WWU)
- Gerard Lemson (MPE Garching)
- Jochen Liske (Hamburg)
- Mark Mahabir (Leicester)
- Brian Major (CADC)
- Bob Mann (Royal Observatory Edinburgh)
- Grégory Mantelet (CDS)
- Jean-Baptiste Marquette (Bordeaux)
- Ben Maughan (Bristol)
- Richard McMahon (IoA)
- Guillaume Mella (JMMC)
- Konstantin Metlov (Donetsk)
- Daniel Michalik (Lund)
- David Mills (Bristol)
- Alexey Mints (AIP)
- Douglas Morgan (Bristol)
- Rhys Morris (Bristol)
- August Muench (AAS)
- Simon Murphy (ANU)
- John Murrell

- Ada Nebot (CDS)
- François Ochsenbein (CDS)
- Masato Onodera (NAOJ)
- Clive Page (Leicester)
- Olivia Panzenböck (TU-Wien)
- Menelaus Perdikeas (ESAC)
- François-Xavier Pineau (CDS)
- Roy Platon (RAL)
- James Price (Bristol)
- Paul Price (Princeton)
- Mike Read (ROE)
- Anita Richards (Jodrell Bank)
- Kristin Riebe (AIP)
- Aaron Robotham (Bristol)
- Yannick Roehlly (OAMP)
- Robert Ryans (QUB)
- Henrik Rydberg (Gothenburg drinking water laboratory)
- Juan Carlos Segovia (ESAC)
- Petr Škoda (AIAS, CZ)
- Benjamin Steltner (MPG)
- Ole Streicher (AIP)
- Rachel Taylor (Cotham)
- Mattia Vaccari (Cape Town)
- Juande Sandander Vela (IAA-CSIC)
- Stelios Voutsinas (Rubin)
- Mike Watson (Leicester)
- Ming Yang (Athens)
- Ivan Zolotukhin (SAI MSU)
- Jonathan Zwart (Columbia)

If you use this software in published work, the following citation would be appreciated:

> 2005ASPC..347...29T: M. B. Taylor, "TOPCAT & STIL: Starlink Table/VOTable Processing Software", in Astronomical Data Analysis Software and Systems XIV, eds. P Shopbell et al., ASP Conf. Ser. 347, p. 29 (2005)

C.2 Version History

Releases to date have been as follows:

Version 0.3b (4 June 2003)
First public release

Version 0.4b (8 July 2003)

- Row subset count column in subsets window.
- Fixed and improved broken Parameter window.
- Fixed output of table name and parameters.
- Output to LaTeX `tabular` environment now available.
- SQL access buttons now greyed out when no JDBC drivers are present.
- UCD selection now available from New Synthetic Column dialogue.
- Column metadata display made more logical and flexible.
- Column cardinality now calculated in Stats window.
- Sythetic column expressions and most other column metadata now editable.
- Null value support in evaluated expressions.
- Integral example data provided.

- Hierarchical browser for tables available in load dialogue.

Version 0.4-1b (10 July 2003)

- Fixed a VOTable output bug (not escaping XML special characters).
- Fixed a serious FITS output bug.
- Improved bad value handling for FITS tables.

Version 0.5b (20 October 2003)

- Can now read plain text format tables.
- FITS files of arbitrary size can now be read (they are not loaded into memory).
- TOPCAT can now run without errors in a secure environment (e.g. as an unsigned jar file under WebStart). Of course some actions such as Save Table are unavailable in this context.
- Files compressed with Unix `compress` now work (as well as gzip and bzip2).
- Added hierarchy browser to load dialog.
- Added integral demo data (accessible from load dialog).
- Can now drag tables from Treeview into TOPCAT load dialog (or load button).
- Some bugfixes.

Version 0.5-1 (18 November 2003)

- Rewrite of FITS binary table access for big efficiency improvements.
- It's now possible to plug in user-defined algebraic methods at runtime.
- Improved unit testing leads to some bugfixes.

Version 1.1-0 (21 April 2004)

- User interface redesigned - now based around Control Window not table browser (much easier to work with multiple tables).
- Extensive facilities for table joining by matching rows between multiple tables or within a single one.
- Documentation much improved and available on- or off-line as SUN/253.
- Self-contained table access library STIL now provided as a separate product.
- Tables can be concatenated.
- Better top-level control over window proliferation.
- Columns can now be hidden/revealed not just deleted.
- Additional output formats/variants supported:
  - VOTable using BINARY or FITS encoding, inline or by reference
  - Machine-readable plain ASCII stream
  - HTML <TABLE> element or document
- Hybrid DOM/SAX parsing of VOTables for improved efficiency/memory usage.
- New flag `-demo` starts up with demo data.
- Miscellaneous efficiency improvements and bug fixes.

Version 1.1-3 (5 May 2004)

- Functions provided for radians<->sexagesimal conversion

Version 1.3 (20 October 2004)

This version has introduced many improvements in scalability, efficiency and functionality. TOPCAT is now quite happy with tables of a million rows or more (and hundreds of columns) even on systems with quite modest memory/CPU resources. The main improvements are as follows:

Plotting

- Plotting is much faster and can handle many more points

- Subsets can be selected from plot window by tracing out a non-rectangular region
- You have more choice over plotting symbols (including semi-transparent ones)
- Finally X or Y axes can be flipped!
- Export to encapsulated PostScript is of improved quality (though for many points file sizes can get large)
- Export to GIF format is available
- Regression lines can be plotted and coefficients displayed (experimental capability - could be improved)

Table Formats

- "`-disk`" flag allows use of disk backing storage for large tables
- New 'FITS-plus' format stores rich table/column metadata in a FITS file
- VOTable handler now fully VOTable 1.1 and 1.0 compatible
- VOTable parsing now works with Java 5.0 platform
- Comma-Separated Value format now supported for input and output
- ASCII input handler rewritten to cope with much larger tables
- ASCII handler now understands d/D as exponent letter as well as e/E
- ASCII handler now uses Short/Float not Integer/Double where appropriate to save memory
- ASCII format fixed bug for -0 degrees/hours in sexagesimal angles
- Null handling improved for FITS & VOTable formats
- FITS files store column descriptions in TCOMMx headers
- Better error messages for unparsable tables

Table Joins

- Various efficiency improvements and reductions in memory requirements
- In cases of multiple possible matches, the closest is now chosen rather than picking one at random
- Pair match now adds column containing score for each match (distance between points)
- Units can be selected RA/Dec columns and match errors (so it doesn't need to be all in radians)
- New match types suitable for multivariate matching (anistropic Cartesian, Sky+X, Sky+XY)

Data/Metadata Manipulation

- Can add/remove table parameters
- One-step column replacement dialogue from data or column windows
- Synthetic column expressions now written out to column descriptions

GUI Navigation and Display

- Improved rendering of numbers in tables (esp. Floats)
- Better detection of displayed table column widths
- New Control Window option on File menus
- Better window resizing for some dialogue boxes
- Less confusing error messages in many places

Algebraic Expressions

- All available functions are now fully documented in help document and interactive Method Window
- Many new trig, coordinate, type conversion, string manipulation functions
- Big performance improvements for null values

Activation Actions

- Clicking a point in the plot highlights the corresponding row in the data window and vice versa
- Row selection can trigger display sky cutout region display
- Row selection can trigger user-defined actions on activation

In addition, the following incompatibilities and changes have been introduced since the last version:

- The input format for tables can now be specified in the load window (via a selection box) or on the command line (using the "`-f`" flag). FITS files and VOTables can still be identified automatically (i.e. it's not necessary to specify format in this case) but ASCII tables cannot: you *must* now specify the format when loading ASCII tables. This change allows better error messages and support for more text-like formats.
- Algebraic expressions referencing row subsets now use the underscore character ("_") rather than the hash character ("#") to indicate a subset ID.
- Classes containing user-supplied functions for algebraic expressions are now specified using the properties "`jel.classes`" and "`jel.classes.activation`", not "`gnu.jel.static.classes`".
- The default output format for FITS tables is now the so-called "FITS-plus" format, which has a BINTABLE in its first extension as before, but the text of a VOTable stored in its primary HDU. This can store more metadata for TOPCAT, but should behave just the same for other FITS-compatible applications. The old behaviour can be restored if desired by specifying "FITS-basic" format.

Version 1.3-1 (10 November 2004)

Minor changes:

- 2MASS cutout servers now available from Activation Window
- Added Starlink logo to all windows

Version 1.3-2 (6 Dec 2004)

Bug fix:

- Error in parsing empty VOTable TD elements fixed.

Version 1.4 (4 Feb 2005)

Load Dialogues

The graphical table load dialogue has been overhauled, and now has two main new features. First, it has been rewritten so that the GUI does not freeze during a long load; it is still currently not possible to interact with other TOPCAT windows while a load is taking place, but you can now cancel a load that is in progress.

Secondly, the provision of load dialogues has been modularised, and a number of new dialogues provided. The new ones are:

- Cone Search
- MySpace Browser
- Registry Query
- SIAP Query

If the required classes are present, you can acquire tables from these external sources as well as the traditional methods of loading from disk etc. New command line flags corresponding to each of these have been added to ensure that they are present and make them prominent in the load dialogue. Furthermore it is possible to plug in additional load dialogues at runtime using the `startable.load.dialogs` system property.

The appearance of the **Load Window** has changed; now only the **File Browser** button is visible along with the **Location** field in the body of the window, but the **DataSources** menu can be used to display other available table import dialogues.

Packaging

The program can now be obtained in two standalone forms: `topcat-full.jar` and `topcat-lite.jar`. The former is much larger than before (11 Mbyte), since it contains a number of classes to support custom load dialogues such as the MySpace browser and web service interaction, as well as the SoG classes. The latter contains only the classes for the core functionality, and is much smaller (3 Mbyte).

Explode Array Column action

There is now a new button in the Columns Window which replaces an array-valued column with a scalar column for each of its elements.

Paste'n'Load

You can now load a table by pasting its filename or URL as text into the table list in the Control Window (using the X selection on X-windows - not sure if or how this works on other platforms).

Help message

The result of `topcat -help` is now more comprehensive, describing briefly what each option does and listing system properties as well as arguments/flags proper.

Version 1.4-1 (8 February 2005)

- Added Search Column item to Data Window column popup menu

Version 1.5 (17 March 2005)

File Access

Load dialogues have changed again somewhat, and save dialogues as well. The default file browser in both cases is now a *Filestore Browser*, which is very much like the standard file browser, but can browse files in remote filesystems as well; currently supported are files in AstroGrid's MySpace or on an SRB (Storage Resource Broker) server. You can now save files to these remote locations as well as load from them.

In addition, the save dialogue now displays the current row subset and sort order - this makes it easier to see and/or change the details of the table you're about to save.

BugFixes

A few more minor changes have been made.

- Error display dialogue boxes have been improved in some places
- Various bugs relating to JDBC database access have been fixed
- Some minor issues relating to VOTables with single-character columns have been addressed

Version 1.6 (30 June 2005)

Activation Actions

Some more activation functionality has been added:

- New **View URL as Web Page** option introduced in Activation Window
- New **System** class of activation functions containing `exec` functions which execute commands on the local operating system
- New **Browsers** class of activation functions for displaying URLs in web browsers (external or basic fallback one)

Algebraic Functions

New **Times** class added containing functions for converting between Modified Julian Day and ISO 8601 format epochs.

Sky Matching

The default sky matching algorithm now uses HEALPix rather than HTM for assigning sky pixels to RA,Dec positions. This gives much faster sky matches in most cases, and uses somewhat less memory so can be used on larger tables. It has also fixed a bug which missed out some possible matches. HTM-based matching is currently still provided as an option, but this is mainly for debugging purposes and may be withdrawn in the future.

Logging

The message logging has been tidied up. The main observable consequence of this is that fewer untidy messages are written to the console when TOPCAT is run from a standalone jar file rather than a full starjava installation. By specifying the new `-verbose` (or `-v`) flag one or more times you can get those messages back. The messages (in fact all logging messages at any level) can also be viewed from the GUI by using the new **File|Show Log** menu option from the Control Window.

SOAP Services

TOPCAT now acts as a SOAP server; SOAP requests can now be made to a running instance of TOPCAT to get it to display tables by location or by sending XML for a VOTable direct. Because of limitations in Axis, this latter method won't work for arbitrarily large tables.

Documentation changes

The `tablecopy` tool is no longer covered in this document; it is replaced by the `tcopy` tool in the separate STILTS (http://www.starlink.ac.uk/stilts/) package. There has also been some reorganisation of this document, mainly in the appendices.

Minor changes

- Added `-version` flag
- Added (dummy) **Print** option to Data Window. This just presents a message to the effect that you should save to a printable format.
- Fixed a bug which gave errors when expressions contained a `NULL_` test on the first column of a table.
- Modified one of the demo tables to contain a column with URLs in it.

Version 1.6-1 (7 July 2005)

Bugfixes:

- Work around AstroGrid/Sun bug which prevented loading short VOTables from MySpace.
- Ensure that filestore browsers are kept up to date when dialogues are displayed.

Version 1.7 (30 September 2005)

Crossmatching

There have been major improvements in the flexibility, and minor improvements to performance, of two-table crossmatching.

- New match algorithm **Sky with Errors** introduced. This allows you to specify a column giving the maximum permissible match error (so it can vary per row) rather than a fixed value for the whole table.
- In the case of multiple possible matches between the two tables, instead of automatically giving you only the closest match, you can now select whether you'd like only the closest one or all those which fit your criteria.
- You can now specify which rows you want to see in the output: 1 and 2, 1 or 2, All from 1, All from 2, 1 not 2, 2 not 1, 1 xor 2. This is pretty much all the possibilities

which make sense, and in particular allows you to do 'left outer joins' (1 not 2).
- The match score column which results from most matches now comes (a) in sensible units where possible (e.g. arcseconds not radians) and (b) with metadata which tells you what its meaning and units are.
- More information is available in added columns after the match; as well as the match score, information about matched groups is inserted where appropriate.
- The "Spherical Polar" match algorithm is now rebadged as the hopefully less confusing "Sky 3d".

Similar changes for 1-table and multi-table matches should follow in future versions.

MySpace Access

MySpace I/O has been re-implemented to use the ACR rather than the (now deprecated) CDK classes it was using before. As well as probably being more reliable and less likely to break with future changes in MySpace server protocols, this gives the benefit of single sign on. The effect of this is that you will need to have the AstroGrid desktop running on your machine before you can access MySpace from TOPCAT.

Algebraic functions

- Added Julian Epoch and Besselian Epoch conversion functions to `Times` class.
- Added `RANDOM` special function.

Miscellaneous

- When you select a column in the Columns window, it now scrolls the table in the Data Window so that the selected column is visible. This is a boon when dealing with tables that have very many columns.
- String "`null`" interpreted as a blank value in ASCII tables.
- Added new activation action to launch system default browser.

Bugfixes

- Fixed some relatively harmless bugs to do with actions available when you select the dummy "Index" column. You can now unsort from a popup menu in the table viewer window.
- Believed to work fine with Java 1.5 now (there were previously some issues with MySpace at Java 1.5).
- Fixed bug in ASCII input handler which misidentified blank lines, or DOS-format line ends, as end of file.

Version 1.7-1 (4 October 2005)

Bugfixes:

- Fixed broken MySpace access on MS Windows.

Version 1.8 (13 October 2005)

- Added **Sky Coordinates Window**; it's now easy to add new sky coordinate columns based on old ones in different coordinate systems.
- `roundDecimal` and `formatDecimal` functions introduced for more control over visual appearance of numeric values.
- Now copes with 'K'-format FITS binary table columns (64-bit integers).
- Modifications to JNLP files.

Version 2.0 (3 February 2006)

A major upgrade of TOPCAT's visualisation capabilities has taken place in this release. There are considerable improvements in functionality, flexibility and efficiency over previous versions:

New graphics windows

In addition to the 2-d scatter plot from previous versions, the following visualisation windows are now available:

- Histogram (1-d)
- 3-d Cartesian scatter plot
- 3-d Spherical scatter plot (with optional radial dimension)
- 2-d Density map (2-d histogram)

The new 3-d functionality does *not* require you to install Java3D or any other third-party 3D toolkit to work (nor does it take advantage of any such toolkit which may be present).

Multi-dataset/multi-table plotting

The plot windows are no longer associated with a single table. All of them allow you to display data from different tables, or from different tuples of columns of the same table, on the same plot. You can layer as many plots as you like on the same axes, using different plotting styles for the different datasets. As before, you can still display data from different subsets of the same table and same columns using different styles.

Plotting Styles

All the graphics windows allow you to set the plotting style for each data set individually, using a wide range of options including colour, line width, marker size, (histogram) bar style, etc. This allows you considerable control over the visual details of the plots.

Transparent markers

All the 2-d and 3-d scatter plots allow you to render points using markers of variable transparency. In a crowded plot, this allows you to see much more information than using opaque points, since you can get some idea of how many points (of different data sets) have been drawn at a given point on the plotting surface.

Line drawing & linear correlation

The 2-d scatter plot can now optionally plot lines associated with data sets. It can either draw straight line segments joining all the plotted points in a set, or draw per-dataset linear correlation lines. In the latter case it will report line gradient, intercept and correlation coefficient.

Improved column selection

The selector boxes for selecting which columns to plot are now 'editable' - that is, instead of selecting the column from a drop down list it is now also possible to type an expression into them instead. This may be more convenient if there is a very long list of columns. It also means that you can use an algebraic expression based on the names of one or more columns instead of a simple column name. The selectors also have small arrow boxes next to them which makes it easy to cycle through the list of known columns. These features are also available at some other places in the program where a column value is required.

Manual axis configuration

As well as zooming in and out using the mouse, you can now set the axis limits by typing them into an axis dialogue box. You can also set the text which will form the axis annotation on the plot.

Status line

Most plots now feature a panel at the bottom of the window indicating how many points have been plotted and the current position of the mouse pointer (if any) on the plotting surface.

Performance

Scatter plots of large datasets now use considerably less memory and around an order of magnitude less CPU time than previously (a 2-d million point replot now takes about 1 second - plotting it the first time may be rather longer since it needs to acquire the data which may be I/O intensive).

Some non graphics-related improvements have also been made as follows:

- Selection of a subset in the Control Window now triggers its selection in other windows (plot, statistics, subsets). The same thing doesn't happen the other way around, since that might lead to confusing consequences.
- Boolean columns now display null values distinctly from false ones. Additionally, null/false distinctions are handled more carefully in FITS and VOTable files.
- The Sky Coordinates Window now suggests names for new columns.
- The Filestore Browser now allows you to enter the position in a file of the table to load (e.g. HDU index for FITS or TABLE index for VOTable).
- Added Hide All & Reveal All actions to the Columns window.
- When joining tables, column name comparison to determine whether deduplication is required is now case-insensitive.
- Fixed a problem which was causing TOPCAT to crash when attempting to save an altered copy of a FITS file under the same name.
- The manual has been reorganised somewhat, and a new Quick Start (Section 2) section added.
- There is an experimental implementation of the Aladin interoperability interface. This hasn't really been tested however, so may not work. Improved Aladin interoperability is expected in future releases.
- Fixed a bug in Cartesian crossmatching algorithms which failed to match if the required error in any dimension was zero.

Version 2.1 (7 April 2006)

A number of graphical and other improvements have been made at this release.

Stacked Line Plot

A new Stacked Line Plot (Appendix A.5.4) visualisation window has been added. This is especially suitable for use with time series data.

Asynchronous data reading in graphics windows

All the graphics windows now read data for plotting asynchronously. What this means is that when you change the plot in a way which requires the data to be read or re-read then the GUI will not lock up and you can do other things, including changing the plot in other ways before it has completed drawing. A progress bar at the bottom of the window indicates progress. This is only noticeable for large (slow to read) files.

Axis Zooming

The existing Histogram, 2-D Scatter Plot and Density Map windows, as well as the new Stacked Line Plot, now allow you to do a 1-d zoom by dragging the mouse near the axis, as well as the 2-d zoom by dragging on the plot surface.

PLASTIC Tool Interoperability

TOPCAT now sends and services messages using the PLatform for AStronomical Tool InterConnection protocol. See Section 9.

IPAC Data Format

Data files in the IPAC format defined by the CalTech's Infrared Processing and Analysis Center can now be read. This is how data from the Spitzer satellite amongst others is normally provided.

String-typed coordinate columns

If suitably identified (e.g. by UCD or units), string-valued columns which represent data in certain well-known forms, currently sexagesimal angles and ISO-8601-format dates, can now be treated directly as numeric values in column selector boxes such as the ones in the plotting windows, rather than having to define a new column using a string->numeric converter.

Memory Error Reporting

When the JVM runs out of memory, it now attempts to post a popup window explaining this rather than just writing a terse message to the console. This popup also gives you the opportunity to view a new section added to the documentation: Tips for Large Tables (Section 10.4).

SOAP Service deprecated

TOPCAT no longer by default runs a SOAP server for accepting tables. You can choose to run it by specifying the `-soap` flag on the command line. This facility may be withdrawn in future versions, in view of the fact that the PLASTIC service can provide similar functionality.

Logo fade

The Starlink logo at the top of every window is set to fade out gradually over the period 1-APR-2006 to 1-SEP-2006.

Bug fixes and minor alterations

- Export of GIF and Encapsulated PostScript images from the graphics windows has been improved - unwanted grey backgrounds round the edges are now rendered transparent in GIFs and white in EPS. A section on Exporting Graphics (Appendix A.5.1.7) has also been added to the manual.
- A number of graphical anomalies such as the plot bounds being reset when an axis Flip button was toggled have been ironed out.
- The JDBC behaviour has been improved: some bugs have been fixed and it now no longer asks for an SQL username/password twice.
- Work around intermittent bug when loading small files from MySpace.
- Density maps are now exported into FITS format with better WCS headers where appropriate.
- The SDSS JPEG cutout activation action URL has been updated from DR2 to DR4.
- A bug which caused the first row in a table to be included erroneously in graphical subset selections has been fixed.
- Added an offset selector toggle button to the Histogram window.

Version 2.1-1 (13 April 2006)

Sphere Zoom

You can now zoom in the Spherical Plot.

PLASTIC

There are some improvements to the PLASTIC functionality and documentation.

- In addition to **Broadcast** actions for the various PLASTIC messages transmission types, there are now corresponding **Send** actions which allow you to direct the message to only a single selected listener application. These actions are now enabled/disabled according to whether any suitable PLASTIC listeners are registered.
- You can now pop up a little window from the **Control Window**'s **Interop** menu which displays the currently registered PLASTIC listeners. The functionality of this window may be extended in future releases.
- When TOPCAT receives a `showObjects` message, it now checks if a matching subset exists rather than always creating and adding a new one. If it does, it just sets current the existing one. This can cut down (a bit) on proliferation of Row Subsets.
- The PLASTIC section of the manual has been improved.
- Some of the screenshots have been updated to include PLASTIC-related features.
- URLs using the `file:` scheme sent by TOPCAT in PLASTIC messages now correctly conform to RFC 1738.

MySpace

A workaround has been introduced for MySpace browsing performance problems. Run with `-Dmyspace.cache=true` to speed it up at the expense of accuracy.

Version 2.1-2 (21 April 2006)

This is mainly a bug fix release.

- Added explode columns option to Data Window popup menu.
- Fix bug present since v2.0 which was sometimes drawing wrong numeric labels near zero in 2D plots. A side effect is that exponential multipliers will more often be used on axis labels (as in versions prior to v2.0). Hopefully somewhat improved axis labelling will be present in future versions.
- Fixed bug which was writing unclosed LINK elements in VOTables.
- Fixed some NullPointerException bugs in column selection and 2D plot.
- Fixed bug in scrolling to correct column in Data Window when column is selected in Columns Window.

Version 2.1-3 (11 May 2006)

Bug fixes and some minor enhancements.

- Implements send and receive of PLASTIC `ivo://votech.org/votable/highlightObject` message.
- Added new `csv-noheader` output format.
- Added skew and kurtosis calculations to Statistics Window.
- Improved, though still imperfect, retention of table-wide metadata in VOTables.
- Fixed bug in writing `votable-fits-href` and `votable-binary-href` format tables from the file browser.
- `mark.workaround` system property, see Section 10.2.3.

Version 2.2 (7 July 2006)

Column-oriented storage

New features for permitting column-oriented storage (`colfits` format, `startable.storage` policy "`sideways`") have been introduced. These can provide considerable efficiency improvements for certain tasks when working with very large (and especially wide) tables.

Bug fixes

- Fixed bug which caused some points to be missed out in spherical plots.
- ISO-8601 strings now permit times of 24:00:00 as they should.
- Quoted unit values are examined to determine probable column data types.

Other items

- Added flux conversion functions (Jansky<->magnitude).

Version 2.2-1 (3 August 2006)

- Added new coordinate system ICRS for sky coordinate conversions.
- Eliminated some unsightly but harmless stack dumps that could occur during plotting.
- TUCDnn header cards now used in FITS files to transmit UCDs (non-standard mechanism).
- Added TST (Tab-Separated Table) format input and output handlers.
- Efficiency improvements for column-oriented access.
- Improved responsiveness when viewing table data for extremely large tables.

Version 2.3 (5 October 2006)

Various modifications and improvements:

- Removed Starlink logo from the top right corner of all TOPCAT windows, including screenshots in documentation. New logos (Starlink, Astrogrid, VOTech, Bristol) are added in **About** window.
- Added new **Add Sample Subset**, **Add Head Subset** and **Add Tail Subset** utility tool buttons to the Subset Window.
- Direct MySpace access using `ivo:` or `myspace:` URLs is now provided - see new Section 4.2.
- Added time conversion functions between MJD and Decimal Year.
- ISO-8601 column numerical values (e.g. in plots) are now by default Decimal Year rather than Julian Epoch. The numerical values are quite close, but Decimal Year is probably more what you'd intuitively expect.
- Added `toHex` and `fromHex` numeric conversion functions.
- Added `-J` flag to `topcat` startup script for passing flags directly to Java.
- Added Sample (as opposed to Population) Standard Deviation/Variance calculation options to the Statistics Window.
- Improved listing for functions in Appendix B of user document (subsection for each function class).
- Modified presentation of HTML version of user document using CSS.

Bug fixes:

- Fixed bug (introduced at v2.2-1) which made table display and adding columns slow, and column widths the wrong size.
- Fixed bug (introduced at v2.0) which prevented Exact matches using non-numeric columns. Exact match now also deals with array-valued columns.
- Fixed PLASTIC bug which was ignoring ID argument of `ivo://votech.org/votable/loadFromURL` message.
- Fixed bug in coordinate conversion that caused bad behaviour in the presence of malformed input coords.
- Fixed bug with plotting very large or very small axis labels in 3-d plots.
- Fixed a couple of minor crossmatching bugs (which would not have affected results).

Version 2.3-1 (Starlink Hokulei release)

- Output to MySpace can now be streamed, if running under J2SE1.5 or later.
- Intercept OutOfMemoryErrors during 3D plotting to give an explanatory popup.
- Fixed bug in axis range checking.
- Fixed bug in handling of single quotes in FITS file metadata.
- Concatenation column assignments now smarter for fully compatible input tables.
- Some optimisations for very large files.
- Added Vega<->AB magnitude conversion constants to Fluxes functions.
- Encapsulated PostScript exported from graphics windows is improved. These files should now print when sent directly to a printer, and the image size is now guaranteed sensible for a portrait A4/letter page.
- Added hyperbolic trig functions (`sinh`, `cosh`, `tanh` and inverses) Maths class (sinh, cosh, tanh and inverses).
- Added cosmology distance calculations in class Distances.

Graphics upgrades

- Column selector panels in graphics windows are now in scrollers.

Version 3.0 (5 July 2007)

This release includes major visualisation improvements and some changes to the plotting user interface:

Error bars

Comprehensive plotting of symmetric and asymmetric error bars (and crosshairs, and boxes, and ellipses, ...) in 1, 2 and 3 dimensional plots is now provided -- see Appendix A.5.1.3.

Auxiliary (colour) axes

It is now possible to modify the colour of plotted points according to values in one or more axes additional to the spatial ones -- see Appendix A.5.1.5.

Histogram Weighting

The Histogram and Density Map windows now have the option of weighting the quantity represented, so that the binned values are sums rather than just counts.

Density Map Colour Scale

When the density map is in indexed (non-RGB) mode you can now select from various colour maps to represent bin occupancy. This makes level differences easier to see.

Plot window rearrangement

Buttons which affect data selection are now arranged in their own toolbar below the plot (Appendix A.5.1.1). The size of the plot control area can be manually resized to give more room for the plot itself by using the **Split Window** () button, which is useful on small screens.

Expression language enhancements

Table parameters can now be referenced in expressions using `param$` notation (Section 7.3), and both columns and parameters can be referenced by UCD using `ucd$` notation (Section 7.1).

Statistics & Histogram result re-use

The Statistics and Histogram windows now both provide **Save** and **Import** actions, which take the data presented in the window and allow it to be saved to a file or imported directly into TOPCAT as a new table.

Cartesian 3D zoom

It is now possible to zoom the 3D Plot towards the centre in the same way as the Spherical Plot.

GAVO Millennium simulation database load dialogue

You can now directly query the GAVO Millennium database service (Appendix A.6.11). Thanks to Gerard Lemson of GAVO for contributing code for this.

Row subset improvements

Row Subset names can now be re-used, and when you are asked to select a new subset name you are allowed to choose one from the list of existing ones. When a row selection is received from another application via PLASTIC the subset name (the name of the sending application) is re-used in this way. This helps to keep the number of subsets under control.

Receving a row subset from PLASTIC in this way, and certain other actions, now cause the subset to be shown straight away (and updated if necessary) on any existing plots, which makes this kind of PLASTIC interaction more responsive.

The size of each subset, and also the corresponding percentage of the table it represents, is now calculated automatically and displayed in the Subset Window. The old behaviour of only calculating sizes on request can be reinstated using the **Autocount rows** () menu item if required.

Fix Vertical in 3D plots

A **Stay Upright** () button is provided in the Cartesian and spherical 3D plots which allow you to keep the Z axis/north pole vertical during rotations.

Quantile calculation

The Statistics Window can now calculate quantiles (median, quartiles, .001, .01, .99 and .999) on request.

Minor changes and bugfixes

- JPEG and PNG now plot export available from all graphics windows.
- Dramatic performance improvement for transparent pixel plotting in 3D plots on some platforms.
- Added locale-specific `formatDecimalLocal()` functions in class Formats.
- Added `fluxToLuminosity` and `luminosityToFlux` functions in class Fluxes.
- Improved deduplication of column names when joining tables.
- 3D plots now correctly report number of visible points.
- Improved error message for broken CSV files.
- Improved error reporting in the presence of a deficient JVM (such as GNU `gcj`).
- Worked around a Mac OSX Java bug which caused table display to go funny when columns were deleted.
- Fixed error in output of FITS table `TNULL`*n* header cards - write them as numeric not string values.
- Fixed a number of bugs in responding to PLASTIC messages (threading, failure to respond).
- 3D plots now plot symbols for all subsets not just one, as for 2D plots.
- Fixed bug which prevented non-string parameter values from being reassigned.

Version 3.0-1 (Starlink Puana release)

- Added error bars capped by arrows.
- Fixed bug which caused error bars in legend to draw in the wrong colour in EPS output.

Version 3.1 (29 August 2007)

- You can now annotate plotted points using text labels (Appendix A.5.1.4) in 2d and 3d scatter plots - use the **Draw Label** (Txt) button to assign a column containing label text.
- Radial axis control in sphere plot is now not visible by default - you need to click the **Radial Coordinates** () button in the dataset toolbar to enable it.
- New **Show Legend** (Aa Bb Cc) button in plot windows determines whether a legend will be displayed next to the plot.
- New **Hide Legend** checkbox in style editor windows which allows subset legend entries to be excluded from the displayed legend individually.

Version 3.2 (28 January 2008)

Functionality enhancements

- Considerably improved display of per-table metadata in the Parameter Window. It is now possible to view long or multi-line parameter values.
- FITS header cards in table HDU are now read (but not written) as table parameters.
- Add mouse wheel zoom for 3D plots.
- Add **Write Mode** selector to SQL Output Dialogue, to allow appending and optional drop prior to create of new table in database.
- Now TOPCAT will automatically register with a PLASTIC hub any time one becomes available - the hub does not have to be running when TOPCAT starts. As a corrollary, if you start an internal or external hub from the **Interop** menu, you no longer need to explicitly invoke the **Register** item afterwards.
- Add `-exthub` flag which starts a new external PLASTIC hub.

- Activation window permits selection of different image viewers for **View URL as Image** option. Note this is not currently provided for the **Display Cutout Image** option - this may be addressed in the future.

Bugfixes and minor improvements

- Added `-stilts` convenience flag so you can easily run STILTS from a TOPCAT installation.
- Fix calculation error in `fluxToLuminosity` function.
- Fix some bugs associated with text label plotting.
- Error bars for zero/negative values are now correctly extended to the axis (rather than being omitted) in 2D plots with logarithmic axes.
- Modified the size of some windows - now posts somewhat smaller windows if a small screen is in use.
- System properties specified explicitly on command line now take precendence over those in `.starjava.properties` file.
- Embedded spaces in output ASCII format table column names are now substituted with underscores, which prevents writing ASCII tables that cannot be read.
- Downgraded from WARNING to INFO log messages about the (extremely common) VOTable syntax error of omitting a FIELD/PARAM element's `datatype` attribute.
- Starting an external hub now works more reliably.

Version 3.3 (19 August 2008)

Functionality enhancements:

- Added normalisation option (1) to Histogram Window.
- In overplots using different tables, an attempt is now made to use the same columns/expressions for axis values as for the main table. This may be convenient when overplotting data from several similar tables.
- FITS files with columns using variable-length arrays ('P' or 'Q' data type descriptors) can now be read (in random access mode, i.e. from an uncompressed FITS file on disk, only).
- The **lut.files** system property can now be used to configure custom colour maps for auxiliary axes and density maps - see Section 10.2.3.
- New class Arrays added to algebraic functions.
- Histogram **Offset** control now works for logarithmic X scale as well as linear.

Bug fixes and other changes:

- Efficiency improvements (~25%? in both CPU time and memory usage) for HEALPix-based sky crossmatching. Thanks to Nikolay Kouropatkine at Fermilab for a new version of the PixTools library which provided this improvement (this version also fixed a - minor? - HEALPix bug).
- Sexagesimal field identification for ASCII input files has become more forgiving; it now accepts minutes or seconds equal to 60 without a syntax error.
- Fixed a pair matching performance bug (slower if tables were not given in the right order) introduced at v1.4.
- The `-disk` flag is now honoured when loading tables from JDBC, which makes it possible to input larger datasets from RDBMS.
- Fixed problem which caused the graphics system to hang or fail when drawing Dot-to-dot lines in 2D plot for points a very long way off the screen.
- Fixed bug which caused incorrect plotting of cumulative histograms with small weights.
- Slight modification to spike-style bar drawing in Histogram, now has tidier

appearance when using dashed lines.

Internal

There has been extensive internal reorgamisation of the plotting classes. There should be no user-visible effects of this, but please report anything which seems to be misbehaving.

Version 3.4 (23 December 2008)

SAMP

TOPCAT now uses SAMP as a (preferred) alternative to PLASTIC for inter-tool interoperability. SAMP is now usually the default; the old behaviour can be recovered using the `-plastic` command-line flag. Although much of the messaging behaviour remains the same, various improvements to the GUI accompany this change:

- New SAMP Panel at the bottom of the Control Window summarising status
- New **SAMP Status** ( ) toolbar button in Control Window, which pops up new SAMP Window
- Other slight rerrangements of toolbar buttons and menu items in Control Window
- **Broadcast Row** checkbox added near Activation Action in Control Window

For more discussion see Section 9.

Registry Access

TOPCAT's registry access has finally been updated to use VOResource 1.03 and the Registry Interface 1.01 (it was previously using very out of date protocols). This means that the Cone Search dialogue (as well as the semi-supported SIAP and Registry dialogues) will now pick up a much more up to date set of services. Additionally, you can choose your own registry service, and the registry searches in those dialogues now feature keyword searches rather than picking up all known cone/SIAP services. See Appendix A.9.2.

Other enhancements

- Table columns and parameters can now be referenced by Utype using utype$ syntax in a similar way to using UCDs - see Section 7.1.
- When a new subset is created by whatever method, the user can now elect to transmit it to other applications using SAMP/PLASTIC as an alternative to adding a new subset to the table's subset list.
- Add items to **Help** menu for viewing help in the default web browser rather than TOPCAT's help browser.
- Add a plot output format option to write gzipped PostScript.
- `-version` command line flag now reports subversion revision as well as symbolic version number.

Bugfixes

- Fixed histogram bug (blank plot) for small values when using logarithmic Y axis.
- Fix stacktrace error in 3D plots when no data is available.
- The SIMBAD name resolver is fixed in the cone search window (it was using a no-longer-supported service format).

Version 3.4-1 (27 March 2009)

- Note Sybase ASE works (documentation edit only)
- Fixed bug which caused SIAP queries to fail every time.
- SIAP load dialogue now has name resolution capability.
- Added SSAP load dialogue. Basic functionality only.
- Added documentation for previously-unadvertised SIAP/SSAP load dialogues.

- Fixed bug which caused registry queries (Cone, SIA, SSA) to fail for Java 1.6.
- Fixed bug which caused blank Contact field in registry query results.
- Can now query registry for more registry services.
- Fits BINTABLE TZERO/TSCAL value reading improvements:
    - Columns with integer TZERO values now read as integers rather than floating point values where possible. This includes unsigned longs ('K'), which were previously represented as doubles with lost precision. Unsigned longs which are too large however (>2^{63}) are read as nulls.
    - Byte-valued columns can now be written out by `fits-basic` output handler as signed byte values (TFORM=B,TZERO=-128) rather than signed shorts (TFORM=I).
    - More comprehensive testing.
    - Fixed bug in calculating value scaled double ('D') values.
    - Fixed bug in typing value for scaled float ('E') arrays.
- Added column selector documentation Appendix A.1.4.
- Fix bug when decoding `file:`-type URLs containing `%xx` escapes.
- Fix error reporting bug in registry search.
- Report application name and version in User-Agent header of outgoing HTTP requests.
- Provide aliases `-sia` and `-ssa` for `-siap` and `-ssap` command line flags.
- Fix SIA and SSA queries to avoid exponential notation in parameters; there is currently no standard concerning numeric representations, but this avoids problems with some servers.
- The fixed length Substring Array Convention for string arrays (`TFORMnn=rAw`) is now understood for FITS binary tables.
- Fixed `-noserv` flag.
- Fixed so that attempting to run services under unfavourable firewall conditions does not cause startup to hang.
- Minor SAMP bugs fixed (JSAMP upgraded to 0.3-1).

Version 3.4-2 (17 July 2009)

- Plot windows now allow a plot title to be set from the **Configure Axes and Title** window (Appendix A.5.1.2).
- Loading message now displayed in control window table list when table is being loaded from PLASTIC or SAMP, or during startup.
- Incoming and outgoing SAMP `load.table` messages now use the `name` parameter, which gives a better chance of a useful human-readable name being attached to transmitted tables.
- Sexagesimal delimiters can now be spaces as alternative to colons in Cone/SSA/SIA load dialogues.
- GAVO load dialogue updated in line with modified database service.
- Work around J2SE mark/reset bug when loading table direct from URL.
- Produce null rather than nonsense results from sky coordinate conversions with unphysical latitudes.
- Produce null rather than questionable results from sexagesimal conversions with mins/secs out of range.
- Startup script improved to provide some cosmetic improvements for Mac users: the TOPCAT icon is shown in the dock rather than the generic Java icon, and the menu application name is set to "TOPCAT" rather than the classname. Note these don't take effect if running directly using `java -jar`.
- STILTS run using `topcat -stilts` from standalone jar can now correctly report starjava revision.

Version 3.4-3 (31 July 2009)

- Fixed two bugs related to drag and drop: intermittent failure of drop (Mac only?), and application crash when making a drop that fails.

Version 3.5 (6 November 2009)

There are several improvements in this version, many of them to do with improved functionality and usability for loading or acquiring data both from local files and from various Virtual Observatory services.

Multiple-table load

TOPCAT will now load multiple tables from a single file if multiple tables are present, rather than just the first one. This applies to multi-extension FITS files which contain more than one table HDU, and to VOTable documents which contain more than one TABLE element. It affects the Filestore dialogue, naming files on the command line, and the new VizieR load dialogue.

In previous versions, when you selected a multi-extension FITS file or VOTable from one of the load dialogues or on the command line, only the first table contained in it would be loaded, and any others were silently ignored. Such files often only contain a single table, but not always. Now, if multiple tables are present in the container file, each one will appear as a new table in the tables list. Other table file formats are not currently affected, since there is no mechanism for storing multiple tables in, say, a CSV file.

Multiple Cone, SIA and SSA searches

In previous versions, it was possible to use the Virtual Observatory Cone Search, Simple Image Access and Simple Spectral Access services to locate catalogue records, images and spectra respectively in a single (cone-shaped) region of the sky, with position and radius entered manually. In this version it is possible to execute such a query for each row of an input table, with position and radius obtained from table data in appropriate columns. This effectively gives a join between a local table and a remote one exposed by a cone-like searchable VO interface.

VizieR load dialogue

A new VizieR load dialogue has been introduced to make queries directly to CDS's VizieR service. Thanks to Thomas Boch for contributing code for this.

VO menu

A new menu labelled **VO** (for Virtual Observatory) has been added to the Control Window. This currently contains single and multiple Cone, SIA and SSA queries and the service specific load dialogues, namely the VizieR and Millennium ones. All these features are also available elsewhere in the GUI, but the new menu groups together VO-related functionality for convenience.

Positional search dialogue enhancements

A number of improvements have been made to the Cone Search, SIA and SSA dialogues; these applies both to the old single-region search dialogues and the new multi-region searches.

- The service URL can now be entered (e.g. cut'n'pasted) directly as an alternative to obtaining it from a registry search. This allows use of unregistered services, as well as making it easy to select the service URL for pasting elsewhere.
- The resource table returned from the registry search can now be sorted by column contents, by clicking on the selected column header.
- While searching the registry, some indication of the number of services found so far is displayed.
- Searching the registry is less likely to result in an out of memory error.
- The (not very useful, and potentially confusing) registry **Update** button has been withdrawn; its function is now available only from the **Registry** menu instead.
- The metadata contained in tables returned from a successful search is improved.
- The SIA and SSA dialogues now have a selector for choosing the desired

image/spectrum format.

- The SIA and SSA dialogues now longer require explicit entry of a size parameter (these protocols define default behaviour for when no non-zero size has been specified).
- The SIA and SSA now automatically query the registry for all appropriate services when first displayed. This is sensible behaviour at the moment, since there is at time of writing a reasonable number of them. A directed search may be given instead at the user's option.
- The Cone Search window provides better instructions on use when first displayed.

A new section Section 6 has been added to the manual providing an introduction to the concepts of VO data access, and a new apendix section Appendix A.9 describes the various single- and multiple-query windows.

Storage Policy change

The way that TOPCAT stores large tables has changed. In previous versions, table data was held in memory (the default *Storage Policy* was "`memory`")) unless the `-disk` flag was specified (to use the "`disk`" storage policy), in which case it was stored in temporary disk files. If you didn't put `-disk` on the command line, it was common to run out of memory when working with large tables.

Now, the default storage policy is "`adaptive`". This means that relatively small tables are held in memory, and larger ones are stored in temporary files. The effect should be that most of the time, you don't need to specify any special options, and you can work with small or large tables without running out of memory. Using `-disk` is still permissible, and may help in unusual circumstances. Running out of memory is still a possibility, but should be much less common.

The old behaviour can be restored by using the new `-memory` command-line flag.

The only differences you notice from this change should be good ones, however it is slightly experimental. Please contact the author if you suspect it has introduced any problems.

SAMP-related enhancements

- Spectrum Display activation action now uses SAMP or PLASTIC to display in a suitably compliant external spectrum viewer. The old behaviour was to display the spectrum in an internal SPLAT window, but it didn't work very well.
- The JSAMP library has been upgraded to v1.0, giving some SAMP behaviour changes, enhancements and bugfixes. Control of things like TOPCAT server port and server endpoint hostname are now configurable using `jsamp.*` system properties.
- The `-jsamp` command-line flag has been added for convenience so that the TOPCAT jar file can be used easily as a JSAMP toolkit.

VOTable-related enhancements

- Namespacing of VOTable documents made more intelligent, and configurable using the `votable.namespacing` system property.
- VOTable 1.2 supported.
- The VOTable 1.2 `xtype` attribute is now used to try to identify columns containing ISO-8601 format dates, which allows them to be treated as numeric values for plotting etc.
- As described above, when loading a VOTable document which contains multiple TABLE elements, all the contained TABLEs are now separately loaded into TOPCAT rather than just the first one, which is what used to happen.

Loading changes

- The Load Window now has toolbar buttons as well as items in the **DataSources** menu for more convenient access to the various load dialogues.
- The command-line flags corresponding to load dialogues (`-tree`, `-file`, `-sql`, `-cone`, `-gavo`, `-registry`, `-sia`, `-ssa`) have been withdrawn from use; they were probably not widely used, and are of minimal usefulness with the new load window toolbar buttons.
- While a table is loading an entry is now visible in the Control Window table list. Previously it was not always clear whether a slow load was in progress or not.

Bugfixes and minor enhancements

- There are improvements (and possibly new problems) in the loading dialogue framework. One item is that a failed load less often closes the dialogue.
- HTML table output is now HTML 4.01 by default (includes THEAD and TBODY tags).
- There is a workaround for illegally truncated type declarations in IPAC tables.
- XML, including VOTable, output is handled more carefully; fix VOTable output encoding to be UTF-8, and ensure no illegal XML characters are written.
- Fixed plotting bugs - caused intermittent GUI corruption (e.g. missing toolbar buttons), especially on MacOS.
- Bug fixed in crossmatching output: entries which should have been null were sometimes written as non-null (typically large negative numbers) in FITS and in non-TABLEDATA VOTable output. This affected cells in otherwise non-nullable columns where the entire row was absent. The previous behaviour is not likely to have been mistaken for genuine results.
- `parse*` string->numeric conversion functions now cope with leading or trailing whitespace.
- The list of example queries in the GAVO load dialogue has been extended to match those in its web interface.
- Fixed an obscure bug which could under rare circumstances cause truncation of strings with leading/trailing whitespace read from text-format files.

Version 3.5-1 (21 December 2009)

This version has one significant new feature:

- Plot windows can now export to PDF as well as other formats.

Other than that, is is mostly performance and usability enhancements and bugfixes:

- Fixed a bug which caused an error when attempting to load multiple files from the command line.
- Improve documentation of 3D windows, including zoom options.
- Fixed a bug which caused an error when attempting to do cone/SIA/SSA search from directly supplied service URL.
- Improve error reporting from multi-cone (etc) window.
- You can now cut and paste from the linear correlation coefficient display in the 2-d plot style editor.
- Faster display of load window and VO menu options.
- Fixed internationalisation bug which could cause cone searches etc to fail in locales that use "," for a decimal point.
- Registry searches are now faster and less likely to run out of memory. Registry-search-based operations (cone search dialogues etc) now run in topcat-lite configuration as well as topcat-full.
- Now warns about non-active resources returned from registry searches (which shouldn't happen).
- Improved helpfulness of error messages in multi-cone etc windows.

- Performance improvements when loading from VOTable documents which contain many small tables, both in general and in the special case of SIA/SSA results. This makes a big difference in some cases.

Version 3.5-2 (24 March 2010)

- Add memory monitor indicator in control window.
- Considerable performance and scalability improvements to the crossmatching algorithms. For several common regimes, using default settings, memory use has been decreased by a factor of about 5, and CPU time reduced by a factor of about 3.
- Add optional tuning controls (**Tuning Parameters** and **Full Profiling**) to the crossmatching windows. Experimentation with these can lead to significant performance improvements for given matches.
- Fixed a crossmatch bug; it was giving a possibility of suboptimal "Best Match Only" match assignments when pair matching in crowded fields. Crossmatch results thus may differ between earlier versions and this one. Both are reasonable, but the newer behaviour is more correct. In non-crowded fields, there should be no change.
- Add new functions to Arrays: `array` functions for constructing arrays, and new aggregating functions `median` and `quantile`.
- Added **Verbosity** selector in multi-cone window.
- Improved logging of registry searches.
- Adjusted the way that data types are read from JDBC databases. Date, Time and Timestamp type columns will now be converted to Strings which means they can be written to most output formats (previously they were omitted from output tables).
- Match Window documentation extended and somewhat reorganised (they now have their own appendix subsection).
- MacOS icons improved: thanks to Simon Murphy (ANU) for this contribution.
- Memory management adjusted further - default (Adaptive) storage policy now uses direct allocation (=`malloc()`) for intermediate-sized buffers to avoid running out of java heap space.
- Further performance improvement for VOTable documents with very many TABLEs.
- Fix bug in code for handling very large mapped FITS files. This was causing fatal read errors in some cases.
- Fix issue when sending some SAMP messages (unhelpful reuse of message tags).
- Fixed minor error when a directory in the file browser got deleted while visible.
- Fix minor scrolling bug in table and column windows.
- Fix histogram save/export bug: the exported table now has the same X bounds as the currently displayed histogram.
- Withdraw embedded SPLAT application. SPLAT could previously be used in some circumstances for in-JVM display of spectra, but this facility didn't work well and was presumably little used. You can still view spectra by configuring an Activation Action to talk to an external spectrum viewer (SPLAT or some other choice) via SAMP or PLASTIC.
- Withdraw SOAP server functionality. This has been deprecated since v2.1. External control is better handled using SAMP (or PLASTIC). If anybody misses this functionality, get in touch and I will consider a similar but less SOAPy alternative.

Version 3.6 (6 August 2010)

This version contains some new features and usability enhancements, and a number of bugfixes.

Load/Save Dialogue improvements:

- There is a new **System Browser** option when loading and saving. This does essentially same job as the old **Filestore Browser** (still available), selecting a file to load or save, but it uses the system-default GUI to do it. For Mac and Windows users, this may present a more familiar and functional interface to the file system (for

Linux users, it is probably not an improvement). Both options are available, so you can choose whichever you prefer.
- You can now save multiple tables at once. The Save Window now offers three ways of saving: **Current Table**, **Multiple Tables** and **Session**. Current Table saves the current Apparent Table as before. Multiple Table saves some or all of the loaded Apparent Tables to a container file (typically Multi-Extension FITS or multi-TABLE VOTable). Session saves some or all tables as well, but additionally saves many aspects of the TOPCAT state, such as table subsets, sort order, hidden columns etc.

Other significant enhancements:

- Finally - you can delete and rename row subsets in the Subset Window! Apologies to those who've been asking that this has taken so long to implement.
- There are new actions in the Control Window File menu to change the order of loaded tables in the table list (move them up and down). You can use ALT-up/down keys to do the same thing. This may be useful in conjunction with the new session/multi-table save facility.
- Utype metadata items are now visible in the Parameter and Column windows.
- The JSAMP library has been upgraded to version 1.1. This means that the hub will appear in the "System Tray" where possible (when using java 1.6+, and when using a suitable display manager). It also recognises the SAMP_HUB environment variable for non-standard hub locations.
- There are a couple of improvements in the VizieR load dialogue: it has options to display sub-tables within catalogues and to include or exclude obsolete catalogues.

Minor enhancements:

- The current table can now be discarded by hitting the **Delete** key in the Control Window table list.
- SDSS image cutout downloads updated to DR7 (was DR4).
- The unofficial column type "`long`" is recognised in IPAC format tables.
- An efficiency warning is now issued for large compressed FITS files.
- Format row counts are displayed with group separators (e.g. thousands separated by commas) in some places for better readability.

Fixes for bugs and misfeatures:

- Table parameter values in algebraic expressions now evaluate to their current value, rather than the value when the expression was entered.
- Tables in TOPCAT now have private copies of column and table metadata. This means that changing a column name or parameter in one table will not affect other tables. It should also prevent a related bug that allowed the column headings in the table browser to get out of step with those in the columns window and elsewhere.
- Receipt of a SAMP `table.highlight.row` message will no longer cause a similar message to be sent back (and similarly for the PLASTIC `ivo://votech.org/votable/highlightObject` message). The previous behaviour was an unwanted implementation side effect which could sometimes cause problems in interaction with other tools.
- Fixed bug in FITS-plus metadata output (table parameters were getting lost).
- Better behaviour (warn + failover) when attempting to read large files on 32-bit OS or JVM.
- Corrected literature references in Fluxes conversion class documentation (thanks to Mattia Vaccari).
- Fix memory usage issues (unnecessarily large output graphics files, possible out of memory errors) for density plots with large pixel sizes.
- Fixed bug in CSV file parsing that could ignore header row in absence of non-numeric columns.

- Fix some minor bugs with Cancel operation during table save.
- Reinstate help buttons from Save Window toolbar (erroneously removed since version 2.0).
- Make sure that failed load of table with no rows reports as such rather than failing silently.
- Loading tables by typing their name directly in the Load Window now loads multiple tables if present not just the first one, in common with most of the other load dialogue types.
- Upgraded PixTools HEALPix library to 2010/02/09 version. This fixes a bug that could theoretically cause deficient crossmatch results, though I haven't managed to produce such errors.
- Fixed (I hope) "Table no longer loaded" error when sending tables via SAMP.

Finally, from this release TOPCAT requires version 1.5 (a.k.a. 5.0) of the Java J2SE Runtime Environment; it will no longer run on version 1.4, which is now very old. I don't expect this to cause compatibility issues for anyone, but I'm interested to hear if that's not the case.

Version 3.7 (23 December 2010)

This version contains some significant enhancements and a number of minor improvements and bugfixes.

Load Dialogues:

Table loading has had a major overhaul. Load dialogues are no longer *modal*, that is you can now interact with the rest of the application while they are open. This has a number of benefits:

- A slow load doesn't prevent you from doing other things.
- You can be loading several different tables at once, either from the same or from different places.
- You can use the Help system while a load dialogue is open (load dialogues now have the usual Help button like other TOPCAT windows).

It has also enabled some related changes:

- Progress is reported for each table load - you can see how many rows have loaded and, if known, how many there are in total
- Cancelling a load in progress is more straightforward and works more reliably than in earlier versions

A few related bugs and idiosyncracies have been corrected at the same time.

Registry Queries:

The registry search interface used in single and multiple VO access windows has been improved:

- When specifying a keyword search for Cone, SIA or SSA services, you can select which registry record fields to match (ShortName, ID, Title, Publisher, Subjects etc). This makes it much easier to get the records you're interested in.
- The Subjects field is now displayed for retrieved records.
- Lists of registry records (i.e. Cone, SIA or SSA services) can now be sent to/received from other tools via SAMP (`voresource.loadlist{,.cone,.siap,.ssap}` MTypes).
- Sky coordinates can be received from other tools via SAMP and used to set Cone, SIA or SSA position (`coord.pointAt.sky` MType).

BaSTI Load Dialogue:

A new load dialogue is available for accessing the INAF-OATS BaSTI (Bag of Stellar Tracks and Isochrones) service -- code contributed by Marco Molinaro.

Searchable Help:

The Help browser now has a Search tab as well as the Table of Contents tab, so you can do text searching on the extensive help document.

Minor Enhancements:

- Storage management improvements; removed restriction on large (>2Gb) non-FITS datasets in some circumstances.
- Fix so FITS tables >2Gb can be used in 32-bit mode (though slower than 64-bit).
- All windows have a new Scrollable option in their File menu. It is not generally recommended to use this option, since in general the windows are arranged so that resizing them will resize sensible parts of them, but it may be useful if using some of the larger windows on an unusually small screen.
- FITS files now store table names in EXTNAME (and possibly EXTVAR) header cards.
- Window placement should now behave a bit more like platform norms, rather than sitting in the top left corner.
- When a table is discarded, it is now deselected from any table selector (for instance in a plot, match or concatenation window).
- HTML table output now writes cell contents which look like URLs in HTML <A> tags.
- Basic authorization (`http://user:pass@host/path`) on table URLs handled.
- Logs current version and whether it is up to date with latest release on startup. This behaviour can be controlled by the [-no]checkversion flag.
- Belatedly added STFC logo to About window.
- Add recommendation to use 64-bit java in large table tips section.

Bug Fixes:

- Fix bug in registry result table which displayed resource identifier instead of publisher in the Publisher column.
- Fix SAMP table load bug which tripped over "+" characters in URLs.
- Fix regression bug from v3.6 - loading jdbc tables from command line or text entry now works again.
- Fixed file pointer `int` overflow bug in FITS MultiMappedFiles.
- Fixed a couple of bugs relating to VOTable strict mode.
- Possibly fixed a not-well-characterised bug in registry search.

Version 3.8 (9 May 2011)

TAP

A new TAP load dialogue (Appendix A.9.8) supporting data retrieval using the Table Access Protocol (TAP) has been introduced. This provides SQL-like free-form access to any remote databases exposed with the TAP protocol.

SAMP changes

An internal SAMP hub is now by default started when TOPCAT starts up if no existing hub is apparently running, so under normal circumstances the SAMP panel at the bottom right of the Control Window should be active all the time. This means that you don't need to do any explicit preparation to communicate with other SAMP-aware applications. If you want to return to the previous behaviour (hub only started by explicit user action), use the `-nohub` flag on the command line. You can connect and disconnect with the hub by clicking the connection logo at the right of the SAMP panel.

The JSAMP library is upgraded to JSAMP v1.2, which fixes one or two bugs and supports the experimental Web Profile, though Web Profile support is not switched on by default.

Other enhancements

- Random Groups HDUs are now tolerated, though not interpreted, within FITS files.
- Add quintuple match option to multi-table match window (someone requested it!).
- You can now re-activate a row in the table viewer window by deselecting and then re-selecting it.
- Expressions in column selectors are now left (and coloured grey) rather than deleted in the case of a syntax error.
- Improve text rendering in Available Functions window.
- GAVO load dialogue now uses fixed-width font for SQL entry.
- Registry search now includes Publisher field by default.

Bug fixes and workarounds

- JDBC table input handler now effectively downcasts BigInteger/BigDecimal types to Long/Double. The PostgreSQL JDBC driver seems to use the Big* types routinely for numeric values (which I don't think it used to do).
- Add workaround for J2SE bug #4795134, which could cause errors when reading compressed FITS files.
- Fix FITS character handling bug which could cause corrupted FITS files on output in presence of non-ASCII characters.
- Fix (some) JDBC connection leaks.
- Attempt case-sensitive matching before case-insensitive for column names.

Version 3.9 (27 October 2011)

Crossmatching:

- Additional asymmetric match options have been added to the Pair Match window - as well as the old symmetric Best Match option, you can now choose to do a match which identifies the best match for each row of one or the other input table (see Output Rows Selector Box). They correspond to finding the best match in table B for each row in table A, and in crowded fields often provide more intuitive semantics than the previous symmetric `best` option (in non-crowded fields there is generally no difference). This replicates the matching performed by some other tools, including Aladin.
- New Match Algorithms, **2-d Cartesian Ellipses** and **Sky Ellipses**, have been introduced to permit matching of general elliptical, rather than just circular, regions in both planar and sky coordinates. Another, **N-d Cartesian with Errors**, has been added for dealing with per-object errors in Cartesian coordinates (previously per-object errors could only be handled in sky coords).
- Fixed a significant bug in sky crossmatching. If all points in a table were on one side of the RA=0 line, but the error radius extended across that line, matches on the other side could be missed. Matches could also be missed if different tables used different conventional ranges for RA (e.g. -180..180 in one case and 0..360 in another). This fix may in some, but not most, cases result in slower matching than previously.
- Semantics of the **Sky With Errors** match algorithm have changed slightly.

Notable usability improvements:

- You can now change the column order of a table by dragging the rows up and down in the Columns Window.
- Algebraic functions involving angles are now mostly available using degrees as well as radians. The `Coords` class has been replaced by `CoordsDegrees` and `CoordsRadians` classes providing sky coordinate functions, and a new class `TrigDegrees` provides normal degree-based trigonometric functions alongside the radian-based versions in `Maths`. The functions in the various activation action classes

now take degrees and not radians. Some of the old function names have changed to make clear that they use radians and not degrees.

- The **File|Discard Table** action in the Control Window can now remove multiple tables at once if more than one is selected.
- ADQL parsing and syntax highligting has been added to the TAP load dialogue, thanks to Grégory Mantelet's ADQL library.
- TAP queries are now synchronous by default (the **Synchronous** checkbox is now checked by default).
- The SQL Query Dialogue (finally) has a multi-line entry field for SQL query text.
- Add **To Browser** button to Help Window - this displays the currently displayed help page in your normal web browser.
- New convenience button added to Available Functions window which is a shortcut to display of the expression syntax in the Help browser.

Other upgrades and enhancements:

- JSAMP has been upgraded to version 1.3-1. The main change is that the Web Profile is now enabled by default in the SAMP hub that TOPCAT launches. There are also more options in the Hub menus, and it will overwrite any moribund `.samp` file at startup, which *should* eliminate annoying persistent "404 No handler for URL" warnings.
- Add **Delete On Exit** checkbox to Running Jobs tab of TAP load dialogue. Jobs are now created by default with Delete On Exit set true (you can change the default with the Deletion menu).
- You can now run JyStilts from the `topcat-*.jar` jar files in the same way as from `stilts.jar`.
- Add experimental system properties `star.basicauth.user` and `star.basicauth.password`.
- Added the experimental `topcat.exttools` system property to allow custom tools to be added to the main toolbar at runtime.
- Improve resilience of multi-cone operation in the presence of unreliable or inconsistent DAL services.
- Added new constants to expression language `POSITIVE_INFINITY`, `NEGATIVE_INFINITY` and `NaN`.
- Add `-running` flag which loads tables specified on the command line into a existing instance of TOPCAT if one is already running.
- Add new `join` function to Arrays class to combine array elements into a string.
- TAP load dialogue uses modified upload ID as per most recent TAPRegExt draft. This may cause some TAP services incorrectly to appear not to support uploads if they have not made a similar update.

Bug fixes:

- Improved some issues (reporting wrong row subset counts and membership) related to changing subset definitions.
- Fix cone search verbosity parameter so that VERB=3 is not erroneously ignored.
- Fix bug introduced at v3.7 related to labelling loaded files (improper handling and propagation of `LOAD_SOURCE` table parameter).
- A PARAMref element with no referent in a VOTable no longer causes an uncaught NullPointerException.
- Fix a small bug related to enabledness of buttons in control window and hub connection.
- Work round an obscure java misfeature which could cause the wrong cell to be edited if a sort interrupts an edit.

Version 4.0b (28 March 2013)

TOPCAT version 4 includes a complete rewrite of the plotting (see below for details). The new plotting functionality is however experimental and changes may be made to the GUI following user feedback. For this reason, the classic plot windows remain available from their usual toolbar buttons and the new plot windows are currently hidden away in the **Graphics** menu. The "b" designation of this version acknowledges the experimental nature of the visualisation changes, but other new features are considered stable.

Logo:
I finally bid an affectionate farewell to the Top Cat Hannah-Barbera cartoon graphic. Never having had any legal right whatsoever to use this logo my conscience has eventually got the better of me and TOPCAT now has an excellent new logo kindly drawn by my friend Phil Hall. It's still a yellow cat.

Plotting:
Four new "Layer plot" windows are available from the **Graphics** menu: Plane, Sky, Cube and Sphere layer plots. These feature a new more powerful (and, yes, more complicated) user interface, and are fully documented in Appendix A.4. New functionality includes:

New Sky Coordinate Plot

- Choice of projection: Sin (rotatable), Aitoff, Plate Carrée
- Data and view sky coordinate systems selected separately: options are equatorial, galactic, supergalactic, ecliptic
- Sky coordinate grid labelled and visible at all zooms

New data plot options

- vectors
- ellipses (with position angle)
- pair, triple, ... data point lines/polygons
- contours
- variable size markers

Improved interactive response

- In 2d and sky plots mouse wheel zooms around cursor position
- In 2d and sky plots you can drag the plot around
- In 3d plots right mouse button recentres cube on selected point
- In 3d plots zooming zooms data in the cube rather than enlarging the cube wireframe itself
- Many controls are sliders which update the plot as you slide

New plot colouring modes
Density colour coding for all plot types, with density colour map based either on dataset colour or chosen from a fixed set. Flat, transparent and aux colour coding still available as before.

Better support for large datasets
Several features have been introduced to provide more meaningful visualisation of large datasets. Improved density-like plots and contours give you better ways to understand plots containing many more points than there are pixels to plot them on. There is separately some improvement in scalability: up to roughly 10 million points is currently feasible depending on available memory etc, though it depends what you're doing. However, I hope to improve this limit in future.

Improved axis labelling

- Choice of font size and style
- Option of LaTeX input for non-ASCII characters etc
- Log axes labelled better

- Minor tick option

Legend options
External or manually positioned internal placement.

Analytic function plotting in 2D
Plot functions of X or Y coordinate using TOPCAT expression language.

(Note this plot change list was missing from the v4.0 release and retrospectively added at v4.0-1).

Other new capabilities:

- Use MOC footprints to speed up multi-cone searches, including footprint icon display in multicone window.
- Add IPAC output format.
- Add new class KCorrections to the expression language, containing a method for calculating K-corrections following the method of Chilingarian and Zolotukhin.
- VOTable input and output are now supported for version 1.3 of the VOTable standard.
- The version of the VOTable format used for VOTable output can now be selected, by using the system property `votable.version`. Output version is VOTable 1.2 by default. VOTable output no longer includes `schemaLocation` attribute.
- You can now reference tables in multi-extension FITS files by name (EXTNAME or EXTNAME-EXTVER) as an alternative to by HDU index.

Other enhancements:

- ADQL parsing in the TAP window can now highlight all unknown symbols not just one of them. This is possible by upgrading to the official v1.1 of Grégory Mantelet's ADQL library, which improves some other aspects of ADQL parsing as well.
- Add new function `hypot` (=sqrt(x*x+y*y)) to the `Maths` class in expression language.
- Add new `split` functions for string splitting to the `Strings` class in expression language.
- Some changes to the `toString` function: it now works on non-numeric values, gives the right answer for `Long` integers and character values, and returns a blank value rather than the string "null" or "NaN" for blank inputs.
- Sexagesimal to numeric angle conversion functions now permit the seconds part of the sexagesimal string to be missing.
- Changes to the IPAC format definition are accommodated: the "long"/"l" type, which is apparently now official, no longer generates a warning, and headers may now use minus signs instead of whitespace.
- Fixed SAMP table loads to honour "name" parameter.
- PNG graphics output no longer has transparent background.
- Upgrade JSAMP library to version 1.3-3.
- Work around change in VizieR output so that VizieR catalogue searching works again. This may be a temporary change.
- Update URLs for GAVO Millennium database service at request of Gerard Lemson.
- Add new class Coverage to the expression language containing MOC-related functions (currently, just `inMoc`).
- Add explanatory section to manual on Multi-Object Matches.
- Add `-debug` flag to manage logging messages better.

Bug fixes:

- Fix serious and long-standing bug (bad TZERO header, causes subsequent reads to fail) for FITS output of boolean array columns.
- Fix small but genuine sky matching bug. The effect was that near the poles matches

near the specified threshold could be missed. The bug was in the PixTools library, fixed at the 2012-07-28 release.

- Fix bug which failed when attempting to read FITS files with complex array columns (`TFORMn=rC/rM`).
- Fix failure when loading very large sequential tables.
- Fixed table concatenation so that column datatype array size etc is consistent with both input tables not just the first one.
- Adjust SQL writer to avoid type error for MySQL.
- Fix plotting bug that might have caused mysterious failures to update the plot. Or it might not.
- Fix ADQL parsing in TAP window so that TAP_UPLOAD tables are treated correctly.
- Fix bug which could cause truncation of strings in FITS and possibly VOTable output when tables were hand-edited to add strings longer than previously-declared length.
- Fixed bug in multi-table matches (>2 tables) which could result in output rows with columns from only a single table, i.e. not representing an inter-table match.
- Fix bug in HMS sexagesimal formatting: minus sign was omitted from negative angles. Now the output is forced positive.
- Fix minor bug associated with deleting the current row subset (this now causes All to become current).
- Cope with 1-column CSV files.
- Fix (some, though probably not all) possible bug(s) related to running on Java 1.7 on Mac.
- Use the correct form "`rows`"/"`bytes`" rather than "`row`"/"`byte`" for TAP capability unit values.
- Infinite floating point values are now correctly encoded in VOTable output ("`+Inf`"/"`-Inf`", not "`Infinity`"/"`-Infinity`" as in previous versions).
- Fixed some layout problems which could lead to zero-width text entry fields for RA/Dec.
- Fixed bug when attempting to explode a hidden array column.
- Fix error bar rendering bug which could result in diagonal lines being offset near the edge of plots.
- Fix problem with GUI locking up when plotting outsize histograms.
- Fixed bug in stacked line plot which caused scribbly drawing in the presence of null X axis values.
- Fixed problem with row highlight messages bouncing for ever between applications when the Transmit Row activation action is in use.
- Improve behaviour when deleting a parameter in the Parameter Window.

Version 4.0-1 (1 July 2013)

New Functionality

- Add read-only support for CDF (NASA Common Data Format) files. Currently, no attempt is made to present time-like values in a human-friendly way, but this may improve in a future release. The SAMP `table.load.cdf` MType is also supported for receiving CDF files.
- Improve handling of HTTP basic authorization. HTTP 401s now pop up a user/password dialogue window, unless the `star.basicauth.*` system properties have been set up.
- Add options to calculate the (scaled or unscaled) Median Absolute Deviation in the Statistics window.

Minor enhancements

- Now subscribes to SAMP `voresource.loadlist.tap` MType, accepting resource list in TAP load dialogue "Select Service" tab.
- The Columns window now makes a better attempt at displaying non-standard per-column metadata of unusual types, in particular array values.
- Line plotting improved in layer plots (including better dashing, new antialias option).
- Implemented fixes to reduce the chance of users inadvertently overloading external Cone/SIA/SSA services with multicone-like queries. First, fix it so that abandoned queries are properly terminated, rather than continuing to hit the server until completion or JVM shutdown. Second, implement a sensible default maximum value for the **Parallelism** field in the multi-cone (etc) panel (though this may be adjusted with a system property).
- Source code is now managed by git and not subversion. The format of the "Starjava revision" string reported by the `-version` flag and the Help|About menu item has changed accordingly.

Bug fixes

- Fix CSV regression bug introduced at v4.0b - CSV files now work again with MSDOS-style line breaks.
- TAP example queries now quote table and column names where necessary. This fixes a bug that was particularly evident with the VizieR TAP service for which nearly all table names are not legal ADQL identifiers.
- Quoting behaviour has changed when generating SQL to write to RDBMS tables. This ought to reduce problems related to mixed-case identifiers. However, it is possible that it could lead to unforseen new anomalies.
- Fix bug with reading session files containing 1-column tables.
- Fixed FITS output bug which could result in badly-formed string-valued header cards (no closing quote).
- Turn off layer plot optimisation that could result in lost precision for double values.
- Fix layer plot bug that broke the plot if both handles of a range slider were dragged to the same position.
- Fixed it so that layer plot axis range settings are reset when the Rescale button is used.
- Fixed a bug that caused an exception when a table in a layer plot was deleted from the application.
- Fixed a bug when zooming way out from a sky plot.

Version 4.1 (7 March 2014)

In this version, the new plotting windows appear for the first time in the main Control Window toolbar, and are no longer considered experimental. There have been many changes to their user interface and functionality since the last release. The old plotting windows are still available from the Control Window **Graphics** menu, but are now considered somewhat deprecated, and will not be developed or fixed further.

Navigation changes to the new plotting windows:

- In 2d plots, right-button drag (or ctrl-drag) does a 2-d anisotropic zoom - you can stretch or squash the plot in both directions.
- In 3d plots, right-button drag (or ctrl-drag) does a 2-d anisotropic zoom along the 2 axes most face-on.
- In 3d plots, middle-button drag (or shift-drag) slides the plot along the 2 axes most face-on.
- In 2d plots, the pan/zoom actions (wheel, pan-drag, zoom-drag) can be done outside the plot to affect just one axis.
- Most navigation mouse actions now show visual feedback on the screen.

- A new tab **Navigation** is added to the Axis controls. This lets you configure zoom/pan actions, for instance selecting zoom factor and X/Y/XY zoom options for when mouse wheel is used.
- A hints panel is added below the plot showing current mouse navigation actions.
- Removed zoom in/out buttons in the toolbar of the layer plot windows; there are better ways to zoom.

Other changes to the new plotting windows:

- Histogram plot type added. You can also add histograms to a normal plane plot if you want to.
- Layer plot windows now have a progress bar at the bottom for loading data, and optionally for other potentially slow operations like replotting, identifying a point, turning blob-selected points into a subset etc.
- For very large/slow plots, intermediate subsampled plots are optionally displayed while panning/zooming to improve responsiveness.
- Toolbar buttons rearranged; the buttons for adding and removing layer controls are now in the layer control panel not the main plot window toolbar.
- When a pair crossmatch completes successfully, you get the option to plot the results, which shows you exactly how the located matches relate to the input positions (pairs of points are plotted).
- Text label plotting now has crowding configuration options: according to configuration, text labels are only drawn if there aren't too many close together.
- New grid and label colour configuration options for plane and sky plots.
- More colour maps added.
- Antialiasing option added for grid line and label drawing.
- New auto transparency shading mode.
- Add some padding around auto-ranged plot limits.
- Changing table in a plot now tries to retain the coordinate values if they still make sense.
- Axis tick labelling improved, including avoiding overlapping tick labels.
- Sky plot re-ranging improved: new data set now re-ranges unless lock button is set.
- Clip selector added for density and aux colour maps, which means you can now use just part of a colour map, giving you much more control over colouring.
- Sky plot position formatting now honours sexagesimal setting.
- Some minor threading/performance improvements.
- Experimental time plot type added. This plot type is considered experimental and appears in the Graphics menu but not currently on the main toolbar. It can plot time series and spectrograms. May be improved in future versions.
- The control for plotting pairs of points works differently (now has its own layer control not just a Form). The triple-point form (which didn't work properly) is withdrawn for now.

Other enhancements:

- A new experimental SAMP message `table.get.stil` is supported, allowing SAMP clients to retrieve tables from TOPCAT.
- URL selector fields in Activation Action window are now editable.
- Add cuboid match algorithm to match windows.
- The Exact matcher now considers scalar numeric values equal if they have the same numeric value; they are no longer required to have the same type.
- Array-valued per-column metadata items are now displayed properly.
- The `toString` function now works for byte and boolean values as well as other data types.
- Replace the (slightly dangerous, also undocumented) Apply Subset option in the table viewer window with a new Highlight Subset option. Also add a new Highlight

Subset option in the Subsets window.
- In the old-style histogram plot, exporting it now provides cumulative counts if the visible histogram is currently in cumulative mode.
- Keyboard focus is more helpfully positioned by default in some dialogue windows (it starts off in the first text field).

Bugfixes etc:

- Fix a registry access bug related to namespaces. It is now finally possible to query the STSci/NVO/VAO registry, as well as the AstroGrid and Euro-VO registries, from the Cone/SIA/SSA/TAP VO windows. Queries to the STSci registry have not been working since mid-2010 as a consequence of bugs in (mostly) the TOPCAT client and (partially) the STSci registry service.
- Work around bug in MacOS text rendering in plots that painted text in the wrong direction when not horizontal (e.g. letters going backwards when running vertically). The workaround involves defaulting to antialiased text when drawing axis grids for MacOS. As well as getting the text the right way round this does look a bit nicer, but it is also perceptibly slower. If speed is more important than correctness you can turn it off using the **Antialias** checkbox in the **Axes-Grid** tabs of the various plot windows.
- Withdraw **Labels** tab in the Sphere plot, since it didn't do anything.
- Rename the **File** menu in most windows to **Window**, which is much more appropriate.
- Fix bug when concatenating tables whose data cells may have been edited by hand.
- Fix service selection bug (NullPointerException) in multi-SIA/SSA windows.
- Fix bug which prevented access to long integer array elements from expression language.
- Fix bug in Statistics window so that Boolean columns correctly report mean values (true/true+false proportion).
- Fix TST input handler so TST files with fewer than 3 columns can be read.
- Upgrade JSAMP to version 1.3-4 (hub now supports web clients with `https` URLs).
- Moved text documenting `table.load.cdf` and `table.load.stil` SAMP MTypes from the transmit section (wrong) to the receive section (right) of this document.
- Minor improvements to CDF support.
- Fix some plot threading issues.
- The reported point count is improved in plots.
- Better handling of OutOfMemoryErrors during plotting.
- Better handling of LaTeX errors in axis labelling.

Version 4.2 (4 July 2014)

Significant new functionality and changes:

- Add the new CDS X-Match Upload Window. In most cases (matching with a table that can be found in VizieR) this window can and should be used instead of the old Multi-Cone window - it is *much* faster. The button for this window replaces the MultiCone button in the main Control Window toolbar. MultiCone is still available from the VO and Joins menus, but is now for most purposes deprecated.
- Windows with a Registry Query Panel can now talk to the registry using either RI1.0 (the only option in previous versions) or RegTAP. RegTAP is generally much better (faster, less bandwidth, more reliable), and is the default choice, but a selector allows you to revert to RI1.0 if preferred.
- The "frame"-style zooming that was used in old-style plots is now available in the new-style plots - drag the mouse using the middle button or shift key.
- The TAP Window button replaces the Concatenate window button on the toolbar. The Concatenate window is still available from the Joins menu. This is not new

functionality, but the UI change reflects the growing usefulness of TAP services.

Minor improvements:

- Form configuration options are now preserved, rather than reset, when a new table is selected in one of the plot windows.
- Registry search panels now display resource count at the bottom of the resource list.
- Update the endpoint for the NVO/VAO RI1.0 registry (now deprecated in favour of RegTAP), on advice from Theresa Dower.
- Fixed TAP load dialogue so that by default it stays open after a table is loaded (the **Stay Open** option is set true by default).
- Improve TAP window Running Jobs display. The Job URL is now always shown even when the other job details scroll off screen, and the details scroll to the top not the bottom when a job is selected or refreshed.
- Upgraded Gregory Mantelet's ADQL library to version 1.2. Better ADQL parsing.
- Modify the way antialiasing is controlled in plot windows. The Grid antialiasing option, where present, now just covers the actual grid lines. Text antialiasing is now selected with a new "Antialias" option in the (now possibly misnamed) "Text Syntax" item in Font configuration. This now affects the Aux axis and Legend labelling as well as the grid labelling.
- Added new graphics output format `png-transp` to generate PNG files with transparent backgrounds.

Bug fixes:

- Set antialiasing on by default for OSX - get it right this time.
- Fixed some bugs related to TAP table uploads.
- Fix out-by-one error in reporting of Row of min/max values in Statistics window.
- Add missing geometric reserved words to ADQL reserved word list. Has a slight effect on ADQL validation in TAP window.
- No longer disable "View URL As ..." options in Activation window when no suitable columns are present.
- Fixed bug in which cube plot navigation help button gave you sphere plot help and vice versa.

Version 4.2-1 (13 November 2014)

New functionality:

- Add (experimental) read-only support for Gaia/DPAC GBIN format.
- Add special variables `$ncol` and `$nrow` to the expression language to refer to the column and row counts in the underlying table. The special variable `index` is also deprecated in favour of `$index` or `$0`.
- Support viewing tables up to 2^{31} (2 billion) rows in the table viewer window. The previous limit was 2^{27} (134 million) rows. This does not necessarily mean that using TOPCAT on Gigarow tables is a good idea!

Minor improvements and bugfixes:

- Change the default RegTAP registry endpoint to `http://reg.g-vo.org/tap`, which should have good reliability, since it can point to different RegTAP services as required.
- Add some more colour maps for aux/density shading.
- Fix regression bug (introduced at v4.2) which caused authentication to fail when using the SQL Query load dialogue.
- Attempting to write FITS tables with >999 columns now fails with a more helpful error message.

- Somewhat improved Unicode handling in VOTables. If you load a VOTable with columns marked `datatype="unicodeChar"` and save it again, the columns now remain `unicodeChar` instead of getting squashed to type `char`. Some lurking Unicode-related issues remain.
- Fix time plot to work for ISO-8601 times. This change also breaks use of ISO-8601 times in non-time plots, but that didn't work particularly well before. Please complain if this change of functionality causes a problem.
- Fix problem with activation action image viewer selection box.
- Update JSAMP to v1.3.5.

Version 4.2-2 (6 February 2015)

Plotting enhancements:

- New Linear Fitting form for Plane plot. This reinstates behaviour available from old plot windows, but it provides some more options, like point weighting.
- New SizeXY form, which allows plotting (optionally autoscaled) markers with horizontal and vertical extents independently determined by input data.
- New Frame fixed control in plot windows with two tabs, **Size** and **Title**. Size allows explicit setting of output plot dimensions and alignment in pixels - can be useful when exporting plots. Title lets you add a text title at the top of the plot.
- New **Save**/**Import** options in Histogram window allow you to export histogram bin data as a new table (this reinstates behaviour available from old plot windows).
- More flexibility when assigning colour maps, in the Aux Axis control, Density shading mode, and Spectrogram layer control. These now provide a **Scaling** option allowing use of Square and Sqrt mappings as well as Linear and Logarithmic, and a **Quantise** slider allowing quantisation of the colour map to discrete levels.
- **Rescale Y** button added to Histogram window.
- Histogram normalisation option adjusted so that total area under bars, rather than total height of bars, is fixed.
- Tick Crowding configuration option added to Aux Axis control Ramp tab. Also changed default for Aux axis to use fewer ticks.
- Some "dart" options (fixed-base open or filled triangles) added for plotting vectors.
- Some "triangle" options (variable-base open or filled triangles) added for plotting ellipses.
- Max Marker Size option replaced with with Autoscaling option in plot Size form. You can now optionally turn off autoscaling and specify marker size in pixels instead.
- Every slider control in plot windows now has a little button on the right that resets it to its default value.
- The text labels for configuration options and coordinate entry fields in the plot windows now provide a tooltip of the form "`name=value`". These strings give the STILTS parameter assignment corresponding to the current setting.

Other enhancements

- New Column Classification Window lets you add mutually exclusive subsets based on column contents.

FITS I/O:

- Reworked part of the FITS input implementation, in particular adjusting the way memory mapping is done to reduce resource requirements on some platforms. If you notice any difference, it should be reduced virtual and perhaps resident memory usage, and some (~10%?) performance improvements, when reading large FITS/colfits files. However, please report anything that appears to be working worse than before.

- Colfits files can now be accessed from streams, not just uncompressed disk files (though that's not necessarily a good idea).

MacOS packaging

Some changes have been made to the MacOS DMG file/application bundle. I believe these are at worst harmless, but pre-release testing of platform-specific features is patchy, so problems are possible - please report if you find any, or if the following items don't seem to be working.

- OSX application bundle adjusted so that the DMG file *should* work on any MacOS with either legacy Apple Java 6 or Oracle/OpenJDK Java 7+ installed.
- OSX application bundle adjusted (I think) so that application working directory is user's home directory.

Bugfixes and workarounds:

- Fixed a query bug (missing `REQUEST=queryData` parameter) in the Multiple SSA Query window. This long-standing bug would have stopped this window working at all with well-behaved SSA services.
- Fixed error in fits-var output (PCOUNT header card did not include block alignment gap).
- Graphics coordinates are now calculated in floating point rather than as integers. This fixes problems that could cause scaled vectors, ellipses etc to be drawn with shapes or orientations badly wrong due to rounding errors. It also improves plotting of analytic functions, especially to vector contexts (PDF/EPS).
- Fix some problems to do with zooming to very large/small plot axis ranges.
- Hide error bars (etc) that would extend to negative values on logarithmic axes; previously they were being drawn in anomalous places.
- Fix Aux axis positioning for 3D plots so that the numeric labels don't get snipped off at top and bottom.
- Add a hack that allows LDAC FITS tables to be treated sensibly in auto-format-detection mode.
- Make VOTable handling more robust against unknown (illegal) datatypes.

Version 4.2-3 (14 April 2015)

Histogram enhancements:

- New Kernel Density Estimate plotting option variants added in the Histogram and Plane plot windows: KDE, KNN and Densogram.
- More flexible normalisation options for histogram/KDE plots: normalise by area, maximum bar height, or total bar height.
- The Bars fixed control is now relabelled Bins, and has separate tabs for Histogram, KDE and General controls.
- The existing Bin Size slider, and new kernel Smoothing sliders, now display current width in data units as the slider is moved.
- More bar styles are available for Histogram plots; **Semi Filled** (the new default) and **Semi Steps** options make it easier to understand multiple overplotted data sets. Note Semi Steps does not currently export very nicely to PDF/EPS.
- Sketch mode is switched off by default.

Minor improvements:

- More slider controls have options allowing explicit entry of values from the keyboard (colour map quantisation, marker/vector/ellipse scaling values).
- More slider controls have text value readout displays alongside the slider (colour map quantisation, marker size/errorbar scaling values).

- Column data read in as unsigned bytes will now be written out as unsigned bytes where the output format permits; previously they were forced to 16-bit signed integers. This affects FITS, VOTable and CDF I/O handlers.
- The Subset Styles panel in table data layer controls now has a **Visible** checkbox indicating whether the subset undergoing subset-specific configuration is currently visible in the plot.
- Be less strict about recognising colfits files (tolerate implicit TDIMn headers).
- Tables specified on the command line may now use syntax that reads data from a unix pipeline.
- OSX application bundle fixed to run application in user's home directory (broken in previous version).

Bug fixes:

- Fix minor ADQL parse failure when using TAP tables named without explicit schema prefix by service.
- Fix regression bug (from v4.2-2) that caused problems in Histogram and Spectrogram layers: *New subset from visible* action wasn't working, and the displayed point count was always zero.
- Work round FITS read bug that could cause problems for VOTables using inline FITS serialization, and possibly elsewhere.
- Fix bug that caused trouble when auto-ranging a plot with a single sky position.
- Catch and report the error properly if the user enters a bad explicit axis range (max value <= min value).

Version 4.3 (17 August 2015)

TAP client enhancements:

The TAP client GUI and implementation in the TAP Load Window have been much improved. As well as being generally more capable and easy to use, the GUI is now quite usable for services providing very large datasets, for instance TAPVizieR (~30,000 tables). In detail:

- TAP service discovery is now done by default by requesting search terms relating to the *tables*, rather than the *services*, that you're interested in. The old (by service) GUI is still available in a different tab.
- The TAP metadata display has been reorganised; schemas and tables are displayed in a searchable tree on the left of the panel, with details of the currently highlighted schema/table in tabs on the right. Organisation of tables at the schema level is now visible. Much more information about the schemas and tables is available, and it's displayed better.
- The Columns and Foreign Keys metadata display tables can now be sorted by clicking on column headers.
- Much more information about the TAP service is displayed, including a full service description and listing of available geometry functions, user-defined functions and data models (if the service/registry provides this information).
- For large services, metadata acquisition is now done as required on a table-by-table basis, rather than attempting to download all the metadata up front. This works much better for services with very many tables.
- The query entry panel has many new features, including multiple query entry tabs, full query text Undo/Redo, and limited support for pasting table and column names that have been selected in the GUI.
- A new **Quick Look** mode has been introduced alongside the Synchronous and Asynchronous modes, to execute a query whose result is displayed but not loaded into TOPCAT.
- The **Examples** menu has been extended to include various sub-menus with more

ADQL query examples appropriate to the current service, including examples provided by the service itself for services supporting the (somewhat) standard `/examples` endpoint.

- The Examples menu GUI has been modified to facilitate browsing, and a button is provided to link to web pages describing the selected example in more detail.
- The **TAP** menu has new options for configuring **Metadata Acquisition**, **Response Format**, **Upload Format**, **Service Discovery**, and **HTTP-level compression**. The old Deletion menu has been moved under this TAP menu as **Job Deletion**.
- Where supported by the service, the BINARY2 VOTable serialization is used by default to retrieve TAP results, which should reduce bandwidth and (marginally) improve handling of null values.
- For upload queries, tables are now by default uploaded to the service using the TABLEDATA rather than BINARY VOTable serialization. BINARY ought to work, but at least some services (e.g. CADC) currently fail with BINARY uploads. The default can be changed using the **TAP|Upload Format** menu item.
- There is a change in the way that TAP service metadata is interpreted: table/column names are now expected to be delimited if required. An attempt is made to fix up names that not delimited when they obviously should be. May cause some ADQL examples to work incorrectly in the presence of non-compliant service table metadata reports.
- The ADQL syntax highlighting is now sensitive to User-Defined Functions declared by the service (if any are declared), so that misuse of UDFs or use of non-existent UDFs can signalled. This is thanks to an upgrade to Grégory Mantelet's ADQL parser library.
- Improve error reporting in the face of non-VOTable TAP error responses. In many cases this makes it much easier to see what's going wrong with a TAP query.
- As a diagnostic tool, when making TAP queries, a log message giving a roughly equivalent `curl(1)` command is now issued at the CONFIG level (visible using flags `-verbose -verbose`).

Bugfixes:

- Fix a serious bug in processing of FITS bit vector (`TFORMn='rX'`) columns. Values read from these columns are presented as a `boolean[]` array. In all previous versions of STIL the bits have appeared in that array in the wrong sequence (LSB..MSB per byte rather than the other way round). Apologies to anyone who may have got incorrect science results from this error in the past, and thanks to Paul Price for helping to diagnose it.
- Fix a less serious bug with `TFORMn='rX'` processing; attempting to read a single-element bit vector column (`TFORMn=1X` or `X`) previously resulted in an error making the file unreadable. Values read from such columns are now presented as Boolean scalars.
- Fix a VOTable reading bug relating to bit vector columns (`datatype="bit"`) appearing in BINARY/BINARY2 serializations. This one was more obvious, it would usually generate an error when attempting to read the file.
- Fix serious bug in time conversion for CDF TIME_TT2000 data types. This was giving completely wrong (deep future) labels when using TIME_TT2000 columns in time plots.
- Fix sky plot vector bug: delta latitude and delta longitude values were being used the wrong way round.
- Upgrade JEL to v2.0.2. Fixes problem with evaluating void-typed expressions (especially user-supplied activation actions), and possibly some other obscure bugs.
- Prevent submission of multi-cone/SIA queries for which the Search Radius parameter is blank.

Minor enhancements and behaviour changes:

- Added some functions to the Arrays class that return array-valued results from array-valued parameters: `add`, `subtract`, `multiply`, `divide`, `reciprocal`, `condition`.
- HTTP-level compression is now requested (`Accept-Encoding gzip` request headers included) in most places where it makes sense. This should substantially reduce required bandwidth in communications with HTTP services that support this.
- When clicking on the headers of columns to sort the rows in displayed JTables that show registry services and VizieR catalogues, clicks now cycle between three options (up, down, none) rather than just two (up, down).
- Minor improvement to version reporting (reports java specification version, no longer issues warning for absent revision string).
- Update JCDF library to v1.1 (minor changes to do with leap seconds).
- Now only runs in PLASTIC mode if the `-plastic` flag is set explicitly (otherwise runs in the default SAMP mode). Previously it ran in PLASTIC mode if a `.plastic` lockfile was present.
- Activation action summary messages are now written via an INFO log message rather than just printing to standard output. To see them you now need to run the application with the `-verbose` flag. The various `exec` functions however do now write their output direct to stdout/stderr rather than including them in a return value.
- Custom Code activation action invocation behaviour has subtly changed when one of the values in the expression is from an integer or boolean column, and that column has a blank value. In that case, it's not possible to represent the column value properly to the function (since integer/boolean values can't be null). Previously, the function was invoked with a zero/false value where the null would have been. Now, the function is not invoked at all. Note this only applies to Activation Actions (which are expected to have side-effects); expressions evaluated as column values etc are, as previously, executed but the result is ignored.
- Modify the heuristics that determine whether the first row of a CSV file is a header.

Version 4.3-1 (22 October 2015)

New features:

- A new **Hints** tab is added to the TAP load dialogue, with useful tips to jog your memory about ADQL syntax.
- The expression language has a new way of referring to a column; if you use the form "`Object$` *<column-id>*" you get the value as an Object not a primitive. This is a special-interest measure for user-defined activation functions that need to see null numeric values.
- Adjustments to the GBIN input handler: avoid descending into Class-typed members of gbin list objects, and add logging for object->column translations.

Bugfixes

- Fix error reporting bug when a non-VOTable response is received from a TAP service.
- Fix TAP service location bug that caused an error if searching in By Tables tab with only Service field selected.
- Fix bug in ADQL parsing that failed to recognise column names that are delimited identifiers when qualified by table names.
- Bugfix updates to ADQL library: disallow use of spurious schema names for tables with no schema name; disallow table-qualified names in ORDER/GROUP BY clauses.
- Upgrade to JCDF v1.2 - fixes a bug when reading large (multi-Gb) CDF files.

Version 4.3-2 (27 October 2015)

Crossmatching bug fix

Fix a long-standing crossmatch bug relating to range restriction during pre-processing. This could have caused missed associations (but not false positives) near the edge of coverage regions when using per-row errors, if the scale of the errors differed (especially differed significantly) between the matched tables. It affected match algorithms **Sky with Errors**, **<n>-d Cartesian with Errors**, **2d_ellipse** and **Sky Ellipses** only. Thanks to Grant Kennedy (IoA) for reporting this bug.

Density plots

Some more options for making weighted density plots have been added. Since v4.0 the Density shading mode has let you see the density of plotted points, but this lacked some features. Three new ways to do density plots are added; these all give you the option of weighting by an additional coordinate (like the Aux mode), choosing the combination method (mean, median, sum, max, ...), and displaying the quantitative value-colour mapping on the shared colour ramp (previously aux axis) beside the plot. The new density plots are:

- Weighted shading mode, using shaped marker kernels on the screen pixel grid, available for all plot types
- SkyDensity form, using HEALPix bins on the celestial sphere, for the Sky plot
- Density form *(later replaced by Grid form)* using square N*N screen pixel bins, for the Plane plot

The details are somewhat experimental and may undergo some adjustments in future releases (feedback welcome).

Colour maps

There are various changes affecting selection and display of colour maps used for density and aux axis shading:

- The default colour map for Aux mode, and other layers using the shared colour map, is no longer Rainbow! It's Inferno. Rainbow colour maps are much hated by visualisation experts. Of course you can still choose Rainbow if you like.
- Add some new colour maps: *Viridis*, *Inferno*, *Magma* and *Plasma* from Matplotlib 1.5, the *SRON* rainbow variant developed by Paul Tol, some diverging maps (*HotCold*, *RdBu*, *PiYG*, *BrBG*) and a qualitative constant chroma/luminance map *HueCL*.
- The options for Density and Aux shading are now mostly the same as each other except where there's good reason to differ. Previously they were different in haphazard ways. The order of the options is now a bit more rational too (similar maps are grouped together, more common ones near the start).
- An attempt is made to give the default form of each colour map a sensible name, without leading minus signs.
- Fix it so that the whole range of each map is distinguishable from white. This is a good idea when you're plotting symbols on a white background, which is common in topcat. Perhaps there are cases it's not such a good idea; if you think so, complain and I may change it back.
- Try to fix it so that all the colour maps go in the same direction (light->dark) where applicable.
- Throw out a couple of particularly useless colour maps.
- Colour map ramp display is now different for non-absolute maps; their effect is shown on a selection of base colours, not just for one base colour.

Minor items

- Try harder to identify epoch columns (suitable for time plot), in particular look for VOTable `xtype` of JD or MJD, and `units` of year.
- Add some functions to the Tilings class to do with solid angles (`healpixSqdeg`,

healpixSteradians, steradiansToSqdeg, sqdegToSteradians, SQDEG).

- Tweaked topcat startup script to work better if it's invoked from a symbolic link.
- Fix the description in SUN/253 document of Sky ellipse and vector plot forms, previously they were not distinguished from the Plane variants of those forms.
- Fix plot bug; titles were painted in white for pixel export formats.
- Rationalise plot report logging. Some more diagnostic information about plots is now logged at the INFO level (visible if topcat is run with the -verbose flag).

Version 4.3-3 (10 June 2016)

Expression Language

The JEL library underlying the expression language parser has been upgraded to v2.1.1 (thanks to Konstantin Metlov), now supporting variable-length argument lists among other things. This allows the following improvements:

- New functions that support any number of arguments are provided: array, intArray, stringArray in class Arrays; concat, join in class Strings; sum, mean, variance, stdev, min, max, median, countTrue in new class Lists; and exec in activation class System.
- Some old lists of similarly-named functions with fixed numbers of arguments have been replaced by single functions that take an arbitrary number of arguments (e.g. array(x1), array(x1,x2), ... array(x1,x2,x3,x4,x5,x6,x7,x8) replaced by array(values...)).
- The 2-argument min/max functions in class Arithmetic have been renamed minNaN/maxNaN to avoid confusion, but in most cases existing expressions involving min/max will work as before.
- Some functions that used to require string arguments will now auto-convert numeric types (e.g. exec("do_stuff.py",toString(RA),toString(DEC)) can now be written exec("do_stuff.py",RA,DEC)).
- You can now implement user-defined functions with variable numbers of arguments.
- Writing large (>=2**31) literal integers used to fail with and inscrutable error message. Now the message tells you to append the "L" character.

Time plot

The Time Plot window (available from the Control window **Graphics** menu) has been enhanced so that it can make multi-zone plots - multiple plots stacked vertically that share the same horizontal (time) axis but have independent vertical axes. The time plot itself and this multi-zone feature are currently experimental; in future versions they may be improved or changed, and the multi-zone feature may be extended to other plot types. Some other changes have gone along with this:

- Plotted points (Mark form) in the time plot now supports shading modes as for other plot types.
- The spectrogram layer now uses the (per plot, or more precisely per-zone) Aux colour map rather than a colour map specific to the layer. This means that the colour ramp is now displayed alongside the plot, so you can see quantitatively what colour corresponds to what value. It also means that to adjust the colour map, you now need to use the Aux fixed control rather than controls in the spectrogram's **Style** tab.
- The plot window tries to guess the time coordinate and fill it in automatically.
- The Function layer control now works in the experimental Time plot, rather than throwing an error. However, it doesn't work very well, since the time coordinate is in unix seconds rather than something more user-friendly.
- Fixed a serious bug in ISO-8601 axis labelling in the time plot. In some cases axis labels were being drawn at positions badly different from the correct position.

Miscellaneous enhancements and changes

- This and subsequent releases target **Java SE 6**, so will no longer run under the (now very ancient) Java 5 runtime.
- Add Fill plotting mode for Plane and Time plots.
- The TAP window now uses blocking HTTP requests rather than repeated polls to wait for asynchronous TAP job completion from services that declare themselves UWS 1.1 compliant.
- Improve sky plot border painting.
- Clean up noisy Cubehelix colour map.
- New function `countTrue` in class Arrays.
- VERBose selector added to Cone Search dialogue.
- UType matching is now case-insensitive when identifying likely columns in Activation window.
- Replace Opaque Limit with Transparency config option for plane and sky density plots.
- Update list of VizieR mirrors.
- Changed implementation of GIF exporter for plots, from Acme to ImageIO. Shouldn't be any noticable difference.

Bug fixes

- Fix the sky join syntax example in the TAP window Hints panel, which was incorrect (comma and parenthesis misplaced).
- Fix bug in cumulative histogram calculation.
- Fix read failure for FITS files with non-blank TDIM for zero-length columns.
- Fix bugs that led to timezone-dependent results when reading ISO-8601 or decimal year time columns.
- Fix numeric field truncation bug in LaTeX table output.
- Fix bug in which expression metadata was sometimes not revealed in Columns Window display following addition of a new synthetic column (regression bug since v4.0b).
- Don't use white as one of the auto-selected plotting colours, since it's likely to be invisible (this was a regression bug introduced at v4.3-2).

Version 4.3-4 (13 September 2016)

Visualisation and HEALPix

- Add options (from layer control Report panel or plot window Export menu) to export the HEALPix density map calculated by a SkyDensity plot; either save it to disk or import it as a new table.
- New Healpix layer control in the sky plot lets you plot tables that represent healpix maps (e.g. as exported from `SkyDensity` plot).
- Add some HEALPix-related functions to the Tilings class: conversions from pixel index to sky position and conversions between ring and nested schemes.
- Colour ramp **Clip** control (e.g. for the Aux Axis) now has a new **Default** checkbox. When checked, a default clip is applied, which avoids very light colours, and when unchecked you can use the double-slider over the whole range. Previously, the double-slider range was pre-clipped, so very light colours were not available.
- Improve performance of some plot types. In weighted shading mode you can now adjust the colour map without appreciable delay. SkyDensity plots are somewhat improved.

TAP/VO enhancements

- The TAP window now honours the (TAP 1.1) `column_index` column if it is present in the `TAP_SCHEMA.columns` table. This means that columns can be displayed in a predefined, rather than random, order in the column metadata display tab if the service supports that.

- The TAP window also now acquires and displays non-standard, as well as standard, table and column metadata items if provided by the TAP service. These come from non-standard columns in the `TAP_SCHEMA` `.tables` and `.columns` tables, or from attributes in custom namespaces in the TableSet document, depending on metadata acquisition policy. Services can use this if they wish to report things like table row counts or column data limits.
- Add VOSI 1.1-style `detail`-sensitive `/tables` endpoint query options to the **TAP|Metadata Acquisition** menu in the TAP window.

Bug fixes and minor enhancements

- The GBIN input handler can now pick up more metadata from the classpath. For suitable tables, metadata included in datamodel classes if present can be interrogated to provide table and column descriptions and UCDs. There are still some deficiencies of this functionality (no column order, utypes and units missing, large file "temp.xml" written to current directory) dependent on issues in the upstream Gaia libraries and ICD.
- Fix bug that caused read failures for large (>0.5Gb) FITS files outside of the current directory on 32-bit JVMs. This was a regression bug since v4.2-2.
- Fix long-standing bug that failed to release file descriptors when opening FITS tables (could cause an error if very many FITS files were opened).
- Improve the Concatenation Window. You can now use algebraic expressions for the appended table columns, and changes to the appended table no longer reset the window state.
- Fix minor border-painting bug in sky plot window (sky outline painted on GUI components).
- Combiner option `stdev` replaces `variance` in density, sky density and weighted plot layers.

Version 4.3-5 (23 September 2016)

- Add **Lock Aux Range** action to plot window toolbars and Aux Axis Control **Range** tab. This lets you prevent the display from updating the Aux colour ramp range during navigation. It also affects some dynamic Auto-Scale range calculations in a similar way.
- Fix aux ranging bug in some plot types (SkyDensity, Healpix, Density, Weighted) that used an aux data range too small to show colour variations when Aux scaling was logarithmic and negative data values were present.
- Add Registry menu to TAP load window, allowing list of RegTAP registries to be updated (as for other VO windows).
- Improved performance (better memory use, faster) of some plot types (SkyDensity, Healpix, Density, Weighted).

Version 4.4 (8 March 2017)

Visualisation improvements

- The GUI has changed in the **Form** tab of plot window Data Layer Control panels. To add (and remove) forms there is now a ➕ **Forms** button that pops up a menu rather than the toolbar with form icons that used to be there. This change was necessitated because there were getting to be too many icons to fit in the default width of the toolbar. But it's probably an improvement anyway since you can now see the name of each available form as well as its icon.
- Improved the documentation of plot Forms and Shading Modes in the user document - each option now has a screenshot showing it in action.

- The colour selector widget now comes with a new **Free colour chooser** button, giving you various ways to select any RGB colour you like.
- New Grid plot form, to plot a 2-d weighted histogram. This replaces the *Density* form, which has been withdrawn (Grid can do all the same things and more, except specify bin size in screen pixels).
- New Quantile plot form, which can (e.g.) plot median lines through noisy data.
- New Gaussian plot form for Gaussian fits to histograms.
- New normalisation (scaling) option **unit** for histogram, KDE, and KNN layers.
- New normalisation (scaling) options **per_day** etc for histogram, KDE and KNN layers in the Time plot window only.
- The Histogram control is now available from the Time as well as the Plane plot window, so you can now plot histograms with a temporal horizontal axis.
- Various tweaks to the details of how plots are positioned on the screen or in output graphics files.

Miscellaneous enhancements

- Fixed the match score (distance measure) for the **Sky+X** and **Sky+XY** matching algorithms. Previously, the score was a linear sum of the unscaled distances for the constituent matchers, which meant a *Best* match was pretty meaningless. Now, it adds scaled distances in quadrature, so Best matching should give you a somewhat sensible result.
- The **Sky with Errors** and **N-dimensional Cartesian with Errors** matching algorithms now report output separations as scaled (dimensionless) values rather than in physical units. This means the results are more comparable, so *Best* match results will make more sense.
- The Maths function `hypot` now takes an arbitrary number of arguments (instead of exactly two).
- Added STScI service to default list of RegTAP registry endpoints.
- Minor improvements to the logic in the Concatenation Window that tries to match up columns between the two tables.
- Update JCDF library to v1.2-2 (2017-01-01 leap second).

Bug fixes

- Fix KDE/KNN plotting bug that could get scaling badly wrong for normalised cumulative plots.
- Fix nasty Concatenation Window bug; the current subset and sort order of the Appended table were not properly taken account of in the output. This was a regression bug introduced at v4.3-4.
- Fix a regression bug (since v4.3-2) in the Frame layer control; values entered were being ignored for image export or display or both.
- Fixed a bug that could show Error information for the wrong job in the TAP Running Jobs tab.
- Fix TAP window bug that prevented job status from being updated in Running Jobs tab.
- Fix small bug in Plane/Time plot windows; horizontal axis label was sometimes off the bottom of the plot by a few pixels.
- Fix subpixel offset of colour ramp frame in PDF/PostScript graphics output.
- Remove spurious padding from EPS graphics output.

Version 4.5 (29 September 2017)

New functionality

- New STILTS Control in plot windows. This displays a STILTS plotting command that can regenerate the currently visible plot.

- Saving and loading sessions now preserves algebraic expressions used to define columns and row subsets (previously only the fixed values were saved and the expressions were lost and could not be edited on reload). Note that sessions saved by this version cannot be loaded by older versions (though the other way round is OK).
- New SkyGrid Layer Control can draw multiple sky system coordinate axis grids on sky plot.
- New plot forms XYCorr and SkyCorr for error ellipses rotated as specified by Gaia-style correlation values.
- Add click-header-to-sort action for a number of displayed JTables: the Columns, Statistics, Parameters and Subsets windows. This makes it easier to deal with very wide tables.
- Add position angle calculation functions `posAngDegrees` and `posAngRadians` to expression language.

Table I/O changes

- It is now possible to write and re-read tables with >999 columns to FITS or colfits format. The new limit is 2^31 columns. This uses a non-standard convention; software that is not aware of the convention (e.g. CFITSIO or earlier STILTS versions) will only be able to use the first 998 columns of tables written in this way.
- For VOTable columns that reference `COOSYS` elements, the relevant information is now visible (`CoosysSystem`, `CoosysEpoch`, `CoosysEquinox`) in the Columns Window.
- Any columns referencing `COOSYS` elements read from VOTable-based formats (VOTable or FITS-plus) will now be written out to VOTable-based formats with equivalent COOSYS references included. Currently not table parameters though.
- The default version for output VOTables is now VOTable 1.3. New output formats `votable-binary2-inline` and `votable-binary2-href` are now offered alongside the five previously available VOTable variants.
- Slight changes to the FITS-plus output handler VOTable formatting in the primary HDU; now uses default output VOTable version rather than VOTable 1.1.
- FITS keywords using the ESO HIERARCH convention can now be read as table parameters rather than ignored when reading FITS tables.

Minor enhancements and changes

- Added Durham Galaxy Formation Catalogue URLs to the Virgo-Millennium Simulation Query window (which has been slightly renamed from the *GAVO Millennium Run Query* window).
- The Euro-VO RegTAP Registry is now included in the hard-coded list of options in VO window registry selectors.
- Updated experimental **TAP|Service Discovery|Reg Prototype** menu option to match v1.1 of Data Discovery Note.
- Replace option **autoscale** with **unit** in SkyVector and SkyEllipse sky plot forms.
- Change autoscaling defaults for plot forms Vector, SkyVector, XYEllipse, SkyEllipse; autoscaling is now turned *off* by default. Apologies for this change in behaviour, but it's better for consistency with new layers XYCorr and SkyCorr, and presents less danger of misinterpreted plots.

Bug fixes

- Update PixTools (HEALPix) library to 2017-09-06 version (https://github.com/kuropat/eag-HEALPix, `447a7be073876dba32`). This fixes a bug in `healpixRingIndex` that could give the wrong value for small values of longitude near zero. It seems possible that this might have led to very infrequent missed associations when crossmatching in these regions, but tests appear to indicate that no such errors would actually have resulted.

- Fix misfeature in SkyVector and SkyEllipse plot forms; these now preserve orientation on the sky even when the View and Data Sky Systems differ. Previously the **Delta Longitude/Latitude** and **Position Angle** parameters were always interpreted in the View, rather than the Data, sky system.
- TAP examples are now picked up from the `/examples` document using more lenient rules (no requirements for a containing `vocab` attribute).
- TAP Examples display line now copes better with long and multi-line names from service-provided examples.
- Long fields (>10240 characters) in output CSV files are no longer truncated.
- If a table is received from SAMP with a `table-id` that has already been seen, it will not be loaded (a SAMP error response is returned).
- Fix bug in plot title placement.
- Fix bug when saving tables generated from plots (e.g. SkyDensity HEALPix maps) that sometimes saved data from the wrong plot.
- Fix bug in Statistics Window table export/import actions; it was failing if the Sum statistic was included in the presence of boolean columns.

Version 4.5-1 (7 November 2017)

This is mainly a bugfix release, because of the issue with metadata bloat listed below. You are advised to update if you have v4.5.

New expression language functions

- New functions `desigTo*` for extracting positions from IAU-style object designations (like `2MASS J04355524+1630331`) - use with care.
- New functions `polarDistanceDegrees` and `polarDistanceRadians`; these calculate the distance in 3d space between two positions specified in spherical polar coordinates.
- New `phase` functions to help with phase folding given a known period. A new modulus function `mod`, whose output is always positive (unlike the `%` operator) is also added.

Bug fixes and workarounds

- Fix nasty bug introduced at v4.5 with STILTS integration; viewing plots added large numbers of table parameters with names "`uk.ac.starlink.topcat.plot2.TopcatLayer*`". This led to useless metadata bloat (potentially very many `INFO` elements) when saving previously plotted tables to VOTable or FITS-plus format. As well removing this bug, at this version any such metadata is discarded when loading tables into TOPCAT, so a load/save cycle with this version will get rid of it.
- Upgrade JEL to v2.1.2: now you can use functions in the expression language that have the same name as table columns.
- Modify the way that synthetic column expressions are represented in column description metadata.

Version 4.6 (24 April 2018)

Activation and DataLink

There has been a complete overhaul of the Activation Action system, partly motivated by the need to work with the DataLink standard, and some related DataLink functionality introduced:

- A new Activation Window replaces the old one, with more flexibility, better configurability, better reporting, and better extensibility.
- You can now configure multiple activation actions at once.

- The result (success/failure plus detail message) is visible for each action performed.
- New general-purpose actions available: Load Table, Run System Command, Send VOTable.
- New DataLink-related actions: Invoke Service, Invoke Datalink Row, View Datalink Table.
- The new framework is also much more extensible, so other actions will be added in the future, and it's also possible to plug in your own at runtime.
- A new Datalink Window is added for viewing {links}-response tables and invoking their links.
- DataLink-style Service Descriptor RESOURCEs in input VOTables (and FITS-plus tables) now become table parameters; they can be viewed in the Parameter Window with class `ServiceDescriptor`, and will survive a VOTable Save/Load cycle.

Note at this release the Activation Window configuration is not persisted, so you have to set it up every time you load a table. In future releases, it is planned to save configuration state with table sessions.

New expression language functions

- Add class Gaia to the expression language. This contains functions for estimating distances from parallaxes and propagating astrometric parameters and errors to different epochs. These astrometric functions are not specific to data from the Gaia astrometry satellite, but are presented in a form convenient for use with the Gaia DR2 source catalogue.
- New array manipulation functions `slice` and `pick` added to class Arrays.
- New utility function `square` added to class Maths.

Plotting enhancements

- Various improvements to the Contour plotting form. You can now contour quantities weighted by a given coordinate, rather than just point density. The smoothing from weighted point samples to the grid being contoured can be performed using sum, mean and other combination methods, it now uses a Gaussian kernel rather than a square top hat, and performance is considerably improved especially at large smoothing widths. The contour levels used are now *reported*, so can be seen at the bottom of the Form tab in the control panel. The contours are now plotted correctly right up to the edge of the visible plot, and various bugs have been fixed.
- The Histogram plot now offers the **Combine** option (sum, mean, median, min, max, stdev, etc) for weighted histograms. This means that you can plot e.g. the mean value of a given quantity per interval on the X axis rather than just the summed quantity. A corresponding (though somewhat less well-defined) option is also provided when plotting KDEs and Densograms.
- The **Combine** option that configures how values are binned in various histogram-like plots (SkyDensity, Healpix, Grid, and since this version also Contour, Histogram, KDE) now has two new options, **sum-per-unit** and **count-per-unit**. These work like the existing sum and count options, but scale the combined values by the relevant unit (e.g. X axis unit or solid angle). Where these units are physical, a **Per Unit** parameter is also provided for scaling in convenient units: second, day, year etc for time (Time plot), and square degree, arcminute, arcecond etc for solid angle (SkyDensity, Healpix). In some cases, the default Combine values have changed.
- The Histogram plot form now lets you export the calculated binned data per plot layer for histograms plotted in the Plane and Time plot windows as well as in the Histogram window (from the **Report** panel or **Export|Layer Data** menu).
- The Linear Fit form is now available in the Time Plot window as well as the Plane Plot.

Minor behaviour changes

- The width of table selector widgets is now limited, so tables with very long labels no longer result in very wide windows.
- Improvements in documentation of the expression language functions: in the Function documentation section classes are now listed in alphabetical order, and the documentation of each function is more complete, including descriptions of all parameters and return values, with examples in some cases. Some readability improvements have also been made in the Available Functions Window.
- The Available Functions window icon is now a bit less ugly.
- The IPAC table reader now matches data type specifications case-insensitively.
- Adjust subset name assignment heuristics in Column Classification Window.
- Updated URLs in Messier demo table.
- Rearrange **TAP|Metadata Acquisition** menu options in the TAP window to reflect current practice and standards. The default behaviour (Auto) now uses VOSI-1.1-style scalable tables endpoint rather than VOSI-1.0 for relatively small services (this is still VOSI-1.0 compatible), the CADC option is withdrawn, and some of the other options are re-ordered.

Bugfixes

- Update JCDF to v1.2-3; fixes some CDF reading bugs.
- Stop the right hand side of the subsets stack being obscured by the scrollbar when there are too many subsets to see at once.
- Fix a bug in the Data Window **Subset From Unselected Rows** action - it was including all unselected rows rather than just the visible ones, as per documentation. This was a regression bug introduced at v4.2-1.
- Prevent MultiCone window from sending cone search queries with `sr=NaN`.
- Fix bug in VOSI 1.1 TAP metadata acquisition detail request URLs (spuriously doubled '/' character).

Version 4.6-1 (18 May 2018)

Activation Action improvements

There have been several improvements to the Activation Window. In most cases these were intended parts of the new activation framework introduced in v4.6, but didn't get completed before the (Gaia DR2-imposed) release date for that version.

- Persistence: all activation state in the Activation Window is now preserved when you save a session.
- Multiple activation: buttons are now provided to invoke actions on every (apparent) table row in turn, rather than just on a selected row.
- Download URL activation action added.
- Plot Table activation action added, as well as a corresponding URL invocation action in the Datalink Window.
- Add/Remove buttons: the Activation Window now has toolbar buttons for adding and removing actions in the displayed list (previously these were only available as menu items).
- `sleep` and `sleepMillis` functions added to System class.
- Import Parameters option added to Load Table activation action.

Bugfixes

- Fix SoG internal image viewer to work with non-HTTP URLs.
- Fix SAMP-related activation actions so they work with filenames as well as URLs for locations (as documented).
- Fix a bug in the TAP client that caused a read error for result VOTables longer than

2Gb.

- Introduce a fallback RangeSlider for use in plot windows when the JIDE-OSS one is unavailable. That cures some circumstances in which plot windows could fail, e.g. (apparently?) Java 10 on OSX or use of Nimbus Look&Feel. It does mean that range sliders may look ugly in at least some OSX environments, but at least it doesn't crash.
- Improvements and bugfixes related to guessing activation service and resource types.
- Fix minor GUI issue with selecting TAP services by double-clicking on their name in the selection tree.
- Fix range bug; caused plot failure when plotting very large numbers with no variation (e.g. gaia_source solution_id).

Minor upgrades and changes

- Upgrade Grégory Mantelet's ADQL parsing library to v1.4.
- Remove dependency on external library BrowserLauncher2 (`edu.stanford.ejalbert.BrowserLauncher`); now use `java.awt.Desktop` functions instead. This may improve browser interactions on some platforms. Obsolete browser options Firefox and Mozilla have been removed from the View in Web Browser activation action.
- VOTable output tweaked: single quotes now used for attribute values if attribute contains double quotes.
- Set the JVMVersion setting in the OSX DMG Info.plist to 1.8+ (was previously 1.3+). Although it will run on Java 1.6, some problems (scrambled JTables) have been reported with 1.6, apparently fixed at 1.8. However, I'm not sure whether the OSX runtime honours this setting; quite possibly it will make no difference. If you notice this either fixes or breaks anything when running on a Mac, I'd be interested to know!

Version 4.6-2 (2 November 2018)

Plotting:

- New Draw Subset Polygon action added to Plane Plot. This lets you draw a polygonal shape with mouse clicks, and constructs the corresponding algebraic expression to define subsets.
- New Sphere Grid layer control, for plotting a spherical net around the origin, added to the Cube and Sphere plot windows.
- New Line3d plot form, for plotting lines joining points in 3d, added to the Cube and Sphere plot windows.
- New **Sort Axis** option, for joining out-of-sequence points, added to Line plot form.
- Provide independent grid crowding controls for longitude and latitude in the SkyGrid layer control. Previously there was one control that determined crowding in both directions. This change may have introduced slight changes to sky axis grid line spacing at low crowding levels.
- Contour level calculations improved; in some cases this previously didn't work well at low point density, resulting in missing contours. The zero point slider function has also changed slightly, it is now not phase folded.
- The sphere plot now centers axes on zero if the center would otherwise be near zero.
- Add option to set bin phase by text entry as well as by slider for Histogram and Grid plot layers.
- Address long-standing plot auto-ranging issue; when auto-ranging resulted in the bottom X limit or right-hand Y limit being exactly equal to zero, the corresponding zero-valued results were not plotted. Auto-ranging has been slightly adjusted to avoid that. This results in pixel-level changes to plot appearance in some cases; these can be avoided interactively adjusting or explicitly setting ranges, e.g. in the Axis

Control **Range** tab.

Expression Language:

- New functions in class Gaia: `polarXYZ`, `astromXYZ`, `astromUVW`, `icrsToGal`, `galToIcrs`, `icrsToEcl`, `icrsToEcl`. These can calculate Cartesian position and velocity components from (e.g. Gaia) astrometric parameters.
- New Shapes class added for working with polygons in the X,Y plane, with functions `isInside` and `polyLine`. This is used by the Draw Subset Polygon action.
- Rearrange expression language documentation slightly: there is a new section listing Special Tokens. This includes "`$random`", which was previously undocumented and named "`RANDOM`".
- Special evaluation tokens `$index0`/`$00`, `$nrow0` and `$ncol0` added; these operate on the apparent, rather than the base table.
- New functions `urlEncode` and `urlDecode` added to Strings class.
- New convenience function `exp10` in class Maths.

Activation Actions:

- New **Download URL** action added to the list of options in the Invoke Service activation action and the DataLink Window.
- TAP example queries now use recently-activated sky positions where appropriate.
- New Display Image Region activation action added.
- Security measures put in place to defend against loading maliciously configured session files.

Miscellaneous Improvements:

- Column Search Window introduced (replaces and upgrades similar action from column popup menu).
- The **Columns** metadata browser in the TAP Window now shows arraysize information if present in the **Type** column, and includes a new **Xtype** column (TAP 1.1).
- Add a status line at the bottom of the Data Window displaying total, visible and selected row counts.

Minor enhancements and behaviour changes:

- Default value of `votable.strict` system property is now `true`. That means that by default, missing `arraysize` attributes on character-valued VOTable FIELD/PARAM elements are taken seriously (indicating a single character) rather than being interpreted as `arraysize="*"`. **Note this could change VOTable interpretation:** columns that previously omitted an arraysize declaration when they shouldn't have done may now be interpreted as 1-character rather than multi-character columns. You can restore the old behaviour with `-Dvotable.strict=false` if required.
- Introduce a couple of measures to reduce the likelihood of unintentional service overload from multicone-type queries (Multiple Positional Search Panel): add a progressive delay for *Retry* error handling modes, and decrease parallelism default value to 3 (from 5) and soft maximum to 5 (from 10).
- GBIN read fix to work around changed behaviour in recent GaiaTools (versions 19.4.*, >=20.1.0 and >=21.0.0) that caused GBIN table reading to fail.
- Load progress reports now format long numbers to include separators (e.g. commas every three digits).
- Modified `topcat` and `stilts` startup script behaviour for OS X. The scripts are now duplicated in `TOPCAT.app/Contents/Resources/bin` in the DMG file, which may be a more conventional location(?) and may mean that Homebrew installation works better(?).
- Slight improvements to the JDBC Configuration section of this manual.

- Tables generated by an internal match now have a name like `match(ID)`, indicating their origin.
- Slightly improved reporting of Service Descriptor descriptions in Parameter Window.
- Small change to MOC handling that might possibly avoid some network-related performance issues.

Bugfixes:

- Fix function plotting so that NaN values are omitted rather than interpolated.
- Fix bug that meant *View Image Internally* datalink invocation option failed in absence of JAI/SoG.
- Fix bug that failed to update the table row header when the content of a DataLink panel changed.
- Fix i18n bug that caused slider text values to be reported with unwanted precision in certain (e.g. fr, de) locales.
- Fix minor bug in TAP Resume Job tab that continued unnecessary polling of ERROR/ABORTED status jobs.
- Fix bug that reported wrong column name in column window for renamed synthetic columns loaded from a session file.
- Fix bugs in `topcat` startup script particularly for OS X, relating to following symbolic links, locating icons and suppressing unwanted output on stderr.

Version 4.6-3 (9 May 2019)

Plotting enhancements

- Introduced new colour scaling options `histogram` and `histolog` for the **Scaling** selector in the Aux Axis Control and certain other places. This can make it much easier to see structure in quantities that do not vary smoothly over their min-max range.
- Improvements to the Draw Algebraic Subset action (renamed from Draw Subset Polygon, and icon slightly adjusted):
  - New shape modes **Box** and **Ellipse** in Plane Plot.
  - Now also available in Sky Plot, with shape modes **Circle** and **Polygon**.
  - Visual feedback improved (little squares mark click position and indicate drawing mode).
  - Algebraic expression reported continuously at bottom of plot during drawing.
  - On completion, Multi Algebraic Subset Window now displays inclusion expression in TOPCAT and (where possible) ADQL formats; ADQL can be pasted into TAP queries.
  - New similar Algebraic Subset From Visible action added to Plane and Cube plots; TOPCAT and ADQL expressions displayed.
- New options for measuring distances on plots:
  - The Sky Plot window now by default draws a small scale bar at the bottom left corner, indicating the scale of the plot in degrees, minutes or seconds. It is controlled by the **Draw Scale Bar** control in the Sky Axes Control Grid tab.
  - A new Measure Distance action is available in the Plane, Histogram, Sky and Time plot windows. This lets you use the mouse interactively to measure the distance and/or vector components between two points.
- Options to plot polygons:
  - New Quad Position Layer Control in all plot types, with new plot forms Poly4 and Mark4 for plotting quadrilaterals (or triangles).
  - New Polygon form for polygons with arbitrary numbers of vertices (may be

improved or adapted in future).

- Improvements to the Healpix plot.
  - HEALPix levels up to 20 can now be plotted (previous maximum was 13).
  - Autoranging now works properly, so the plot is correctly centered on the data if it only covers part of the sky.
  - Graphical subset selection and point activation now works for (the centers of) HEALPix pixels in the same way as for markers.
  - Memory management is improved for fairly large maps.
- The Line plot form now has an **Aux** coordinate that can vary the colour of the line along its length according to some third quantity.
- Improve defaulting of coordinate values in plot windows. Column names and UCDs are now used in more cases to choose default values for plotting.
- New projection option **Car0** available in the Sky Axes Control Projection tab. This is like Car (Plate Carrée) but has longitude=0 at the left/right edge rather than the center of the plot.

HEALPix-FITS support

Various changes to support the semi-standard HEALPix-FITS serialization convention. Available information about HEALPix encoding (level, index column, ordering scheme, coord sys) can now be stored in custom table parameters (of the form `STIL_HPX_*`), and is used by FITS output handlers to insert the relevant FITS headers. The existing FITS handlers do this where it's not disruptive, and the new fits-healpix output handler will additionally move and rename columns where the convention requires. This metadata is round-tripped by FITS and VOTable I/O handlers. It is added automatically when exporting HEALPix maps from the SkyDensity plot form. FITS support is not perfect: the `BAD_DATA` FITS keyword is ignored, and the 1024-element array-valued column variant is not understood. When the Sky Plot encounters a table tagged with this HEALPix metadata, it now makes use of that information to improve defaulting in HEALPix layer controls, and in some cases by defaulting the initial plot layer to be of type HEALPix rather than scatter plot.

VOTable 1.4 support

Support is introduced for version 1.4 of the VOTable format and its new `TIMESYS` element.

- For VOTable columns that reference TIMESYS elements, the relevant information is now visible (`TimesysTimeorigin`, `TimesysTimescale`, `TimesysRefposition`) in the Columns Window.
- Any columns referencing TIMESYS elements read from VOTable-based formats (VOTable or FITS-plus) can now be written out to VOTable-based formats with equivalent TIMESYS references included, so TIMESYS round-tripping for columns works; however this will only be done if the VOTable output format is set to version 1.4. By default (at least as long as 1.4 is not finalised) the output version is 1.3. To enable this TIMESYS output, set the system property `-Dvotable.version=1.4`. Currently this TIMESYS output works only for table columns (FIELDs) not parameters (PARAMs).
- The `timeoffset` attribute of a TIMESYS element referenced by a VOTable column is used to make sense of column data when interpreting it as an absolute time. Currently, the only use of this is in time plots.
- `FIELD/@ref` attributes are no longer imported as "`VOTable ref`" column aux metadata items, since they often interfere with TIMESYS references. Doing this was probably always a bad idea since the referencing is not kept track of within the application, so withdrawing this functionality makes sense, but beware that it might change or break some existing behaviour.

Other enhancements

- Overflow or error flags indicated by DALI `<INFO name="QUERY_STATUS"/>` elements in loaded VOTables are now reported in the **Rows** line of the Control Window properties panel.
- You can now edit the **Shape** metadata item in the Columns window by double-clicking on it, e.g. to specify a fixed length for unknown-length array columns. This can be useful when preparing metadata for output and also to enable the Explode Array Column action.
- New functions `indexOf` in class Arrays to find position of a given value in an array.
- New functions `parseInts` and `parseDoubles` in class Conversions for extracting array values from strings (experimental).
- If an expression with bad syntax is entered in the Function Layer Control Function Expression field, it is now greyed out like in column selectors rather than deleted. This makes it easier to enter function expressions.
- Fix synthetic column and subset definition GUIs so that users are prevented from supplying an algebraic expression which references itself. Such recursive definitions cannot be evaluated and lead to bad behaviour of the application.
- Fix expressions so that `$ID` column references referring to nonexistent columns are rejected when entered rather than causing trouble during later evaluation.
- A sequence number is now appended to plot window titles, which may make it easier to keep track of plots from the window manager or desktop.

Bug and misfeature fixes

- Healpix and Spectrogram layer controls now default to initially displaying only the current subset if one is defined, rather than the whole dataset.
- Vizier queries now use the output flag "`-out.meta=Dhul`" rather than "`-out.meta=DhuL`"; this means that some link columns get meaningful content that did not before.
- Added bumper to Vizier table selection control in CDS Upload Match window.
- Fix some bugs relating to plotting values close to the limits of the double precision range.
- Fix bug that prevented the STILTS control working when the plot is empty.
- Fix bug/misfeature in CDF table parameter construction: CDF global attributes were ignored (with a "WARNING: Omitting complicated global attribute" message) if they contained any null entries. Now such entries are just ignored and the table parameter is constructed from the global attribute using the non-null entries.
- Fix a problem with the Link2 plot form that caused short lines that should span the antimeridian to appear as long lines crossing the whole sky. Such links are now just not drawn.
- Improve colour ramp quantisation (e.g. **Shader Quantise** control in Aux Axis Control); the full colour range is now included.
- Worked round a performance issue that meant in some environments (OpenJDK?) creating a large subset could lock up the GUI for a long time.
- Downgrade log messages about UDFs with unparsable signatures when parsing TAPRegExt documents; now INFO not WARNING.
- Use period not comma as decimal separator for non-sexagesimal sky plot axis labels regardless of Locale; also avoid trailing comma sometimes erroneously present.

Version 4.7 (18 November 2019)

Runtime environment: Java 8

From this release, **TOPCAT requires Java 8** (a.k.a. Java 1.8) or greater to run, rather than Java 6 as for previous releases. Java 8 has been around since 2014, so it should be available on all but very ancient platforms. If startup fails with a `java.lang.UnsupportedClassVersionError` then you need to upgrade.

New or enhanced functionality:

- Most of the plotting, though not the intial data preparation, will now run in parallel for large (>1e5 row) datasets. This should make interacting (e.g. pan, zoom, change settings) with slow plots faster on multi-core machines. (In rare cases this multi-threading might cause problems with memory usage; it can be effectively turned off if required by using the system property `java.util.concurrent.ForkJoinPool.common.parallelism`.)
- New activation actions Display HiPS Cutout and Send HiPS Cutout; these make it easy to access imagery for many surveys using the excellent hips2fits service from CDS.
- New class URLs contains expression language utility functions for constructing certain service URLs: `hips2fitsUrl`, `bibcodeUrl`, `doiUrl`, `arxivUrl`, `simbadUrl` and `nedUrl`. Existing functions `urlEncode` and `urlDecode` have been moved to `URLs` from class Strings.
- Rework the View in Web Browser activation action so that it can show web pages for not only URLs and filenames, but things like DOIs, bibcodes and arXiv identifiers too.
- New **JavaFX** browser option in the View in Web Browser activation action. This is like the existing Basic Browser, but more capable, e.g. it can render JavaScript. Note JavaFX is not available with all Java installations.
- Add **Pause Sequence** action to Activation Window.
- Activation Window sequence actions now update activated row (display highlight in table viewer and plots) as they progress.
- New Delay activation action added as a convenience for use with "slideshows".
- Add tiny **Show**/**Hide All** buttons at the bottom of the subset list in the Data Layer Control Subsets tab.
- Add new **Use Position** configuration option to polygon plot form to toggle inclusion of reference position in polygon vertex list.
- Add new **Fix Errors** button for common ADQL errors in the TAP Window.
- Modified behaviour for offset (e.g. unsigned) longs in FITS files. 64-bit integer columns (`TFORMn='K'`) with non-zero integer offsets (`TSCALn=1`, `TZEROn<>0`) are now represented internally as Strings; previously they were represented as Long integers, but values out of the possible range appeared as null (with a warning written through the logging system). Such columns are most commonly seen representing unsigned long values. If written back out to FITS, the offset long value will be reinstated, but other output formats cannot encode unsigned longs, so they will stay as strings.
- New functions `parseBigInteger` and `parseBigDecimal` in class Conversions.

Minor enhancements and behaviour changes:

- VOTable output now writes VOTable version 1.4 by default (was 1.3).
- Modified plot legend display so that small markers are displayed a bit bigger in the legend than on the plot for readability.
- Add new colour map Cividis.
- Upgraded ADQL parser from 1.4 to VOLLT 1.5-1. Should result in minor improvements to TAP ADQL syntax highlighting.
- Permit FITS and VOTable files with zero-length string columns. Previously all-null or zero-length string columns were sometimes forced to single-character values.
- Fix one RegTAP example query in accordance with RegTAP-1.0 Erratum #1.
- Add more detail (response content) to certain bad response TAP error messages.
- Update mapped file unmapping implementation to work for java9+.
- Minor changes to behaviour when querying TAP 1.1 services (REQUEST parameter is omitted).
- Deprecate Display Cutout Image activation action; Display HiPS Cutout is usually a

better choice.
- The front page screenshot montage (which was very out of date) is replaced by one with up-to-date screenshots.
- Registry queries for standard services now search for `intf_role='std'` instead of `intf_type='vs:paramhttp'`, which is the recommended pattern for RegTAP 1.1, and believed to work for current contents of known searchable registries.
- Eliminate unnecessary "(no position)" text in Display Image activation action log message.
- JSAMP to version 1.3.7.

Bugfixes:

- Avoid sometimes losing precision when reading ASCII/CSV values in the range +/-(1e-45..1e-38).
- Fix TAP window to avoid locking up GUI in case of slow loading service logo image.
- Fix Execute Code activation action so that the Synchronous option is set false by default (this was a bug - true can cause responsiveness problems for slow commands).
- JEL bug fix update, to avoid unwanted debugging output for String function null returns.
- Fix Quantile plot form bug (no non-JIDE RangeSlider fallback) that stopped it working in some environments such as Java 10 on OSX and Nimbus L&F.
- Fix load error encountered when specifying a Cone Search service by URL rather than selecting from a registry query list. This was a regression error since v4.6-2.
- Fix GUI updates when current subset expression is edited: data window view and apparent row count are now updated accordingly.
- Fix plot axis ranging bug: padding was not always applied properly for logarithmic axes.

Version 4.7-1 (5 June 2020)

File Formats:

- The **ECSV** (Enhanced Character Separated Values) storage format is now supported for input and output.
- The **Feather** storage format is now supported for input and output.

Performance:

A number of implementation changes have been made which may improve performance, particularly for crossmatching (typical improvements for large sky crossmatches are a factor of 2, though YMMV). These should have no effect on the results, but if anybody notices crossmatching behaviour which is changed since previous versions or otherwise suspicious, please report it.

- The HEALPix implementation has been replaced; all HEALPix manipulation is now done using the excellent cds-healpix-java library written by François-Xavier Pineau from CDS, which speeds up sky crossmatching considerably. Many thanks to François-Xavier and to CDS for providing this library and for assistance with its use; thanks also to Nikolay Kuropatkin from FermiLab whose PixTools library served this purpose in TOPCAT up till now.
- Rows are now binned during crossmatches using a HashSet rather than a TreeSet.
- Evaluations of the `arcsin` function in Sky matches now use the (Apache via cds-healpix) FastMath implementation rather than the standard J2SE version.
- Performance is improved when reading long String values from FITS files.

Visualisation:

- New Area Layer Control, offering forms Area, Central and AreaLabel, can plot region data supplied as area coordinates in the form of STC-S (e.g. from ObsCore/EPN-TAP `s_region`), DALI polygon/circle/point or (ASCII) MOC columns on Sky, Plane or Sphere plots.
- The Time Plot window is no longer labelled *experimental*, and now appears on the main control window toolbar. When selecting its time coordinate, you can now also select how input values are mapped to time (as MJD, JD, ISO-8601 etc), which makes it much easier to plot time quantities from input tables with insufficient metadata.
- The default initial layer in the Time Plot is now Line, not Mark.
- Scatter plot marker size now defaults to a value dependent on the number of rows in the table, so by default large tables have small markers and small tables have larger markers.
- Provide more options in the **Shape** selector for the Mark plot form: versions of circle, cross etc with thicker lines.
- Maximum marker **Size** in Mark plot form is increased from 5 to 9.
- Add line thickness control to the contour plot.
- Improved plot axis labelling in LaTeX mode, e.g. in LaTeX write "3x10^6" not "`3e6`".
- The Sky Plot position layer control now automatically selects lon/lat coordinates for EPN-TAP tables (`midLon(c1min,c1max)`, `midLat(c2min,c2max)`) where available. Similarly, `c{1,2}{min,max}` values are automatically used for the sky plot Quad Position Layer Control, so that EPN-TAP-style bounding boxes are displayed by default.
- Add replacement marker functionality (Minimal Size/Shape options) to Poly4 form, so that small polygons aren't too small to see.
- Improve accuracy when drawing large HEALPix tile boundaries in some cases for SkyDensity and Healpix sky plot layers.
- The Sort Axis option for the Line plotter in the Time plot window now presents the option Time rather than X (or Y).
- Add Center option to the Label plot Anchor selector.

Other minor enhancements/changes:

- Basic support for SIA version 2 as well as version 1 in the SIA Query and Multiple SIA Query windows.
- It is now permitted to provide blank RA/Dec entries in the SSA Query window. This is legal according to the SSA standard and reasonable for e.g. theory services.
- Remove **SAMP Window** button from main Control Window toolbar for reasons of space; this option is still available in the **Interop** menu.
- Add new functions `midLon` and `midLat` to Sky class.
- Add new conversion functions `*ToUnixSec` to Times class.
- Adjust coordinate labels for SkyVector plot form, to make it more obvious that cos(lat) premultiplication is required for Delta Lon.
- Reorder registry service URLs for the (largely obsolete) RI1.0-mode registry search in the Registry Search Panel. Default is now Euro-VO, which still offers a working RI1.0 service; the (moribund?) AstroGrid services are demoted.

Bug and misfeature fixes:

- Fix Ellipse mode in the Plane Plot's Draw Algebraic Subset action; the ellipse expression was previously covering an area $2^{-0.5}$ smaller than the drawing.
- Fix polygon plot form in Sphere Plot; this combination was failing to plot anything, now it works.
- FITS ASCII table extensions with TFORM values of `In` are now treated as 64-bit integers for `n>=10` rather than `n>10`.
- Fix regression bug since last release that refused to make dataless plots.

- Slightly improve plot axis labelling for small numbers; remove unnecessary decimal point in scientific notation in some cases.
- Fix fairly harmless NullPointerException report when displaying a blank plot with too few auto-selected axes (regression).

Version 4.7-2 (24 August 2020)

New Functionality:

- SVG (Scalable Vector Graphics) is now one of the supported graphic output formats in the Plot Export Window.

Bug fixes and minor enhancements:

- Fix plot failure when supplying nonexistent columns for optional coordinates. This was sometimes causing significant knockon problems elsewhere in the GUI.
- Fix Healpix trouble at large angles, e.g. sky crossmatch failure with match radius >6 degrees.
- Fix regression bug at v4.7-1 which failed to update subset content in plots when subset name was reused.
- Fix some regression bugs at v4.7-1 related to selection boxes with user-entered values: user-entered *VizieR Table ID/Alias* values in the CDS Upload X-Match window didn't work; user-entered *Maximum Row Count* values for the VizieR load dialogue caused the load to fail; and user-entered *UCD* values in the Synthetic Column window were ignored. These are all now fixed.
- Fix ECSV output bug: encoding was incorrect for metadata scalars with certain non-alphanumeric first characters, leading to invalid YAML.
- Remove some unhelpful per-column metadata items from ECSV output.
- Improve seeding of the expression language `$random` special value; it should now be less dependent on JVM details.
- TAP `curl(1)` equivalent logging flag error fixed (write `--compressed` not `--compress`).
- Improve HiPS Survey selection menu in the View/Send HiPS cutout activation actions so that the submenus are not too tall for the screen.

Version 4.7-3 (23 October 2020)

Packaging

- The build of the .dmg file for MacOS has been reworked. It should no longer get confused by versions of Java 6 installed alongside Java 8; it may also avoid security issues in accessing some local directories. This should make things work better for (some) MacOS users, but please report if there are new problems.

Minor enhancements

- The Position Layer Control for the Plane Plot now has an **X<->Y** button for convenience, which lets you switch the contents of the X and Y coordinate value fields.
- Minor fixes to match window help text.
- Improve guesses about default coordinates in Area plots.

Bug fixes

- Upgrade cds-healpix-java library to v0.28_1; avoid occasional sky crossmatch failures (termination with error).
- Fix minor plotting bug that could cause white points to be invisible.

Version 4.8 (11 January 2021)

In this version, the table handling library STIL has been upgraded to v4.0, which enables some enhancements including multithreading and new I/O handler features.

Notable new functionality:

- Auto file format detection now examines filenames to help guess format; this means that e.g. when loading CSV files named with the "`.csv`" extension, it is no longer necessary to select "CSV" in the Load Window **Format** selector.
- Scheme specifiers can now be entered in the Load Window to load tables not corresponding to external files. Options currently provided include simple sequence, simulated sky and strange attractor data.
- Add more example tables (using skysim and attractor schemes) to the Example Tables menu.
- Table input/output handlers specified in the table Save/Load Windows can now in some cases take parenthesised options, e.g. you can type in "**votable(version=V12,format=BINARY)**" instead of just selecting "**votable**". For examples see individual handler documentation, e.g. the VOTable output handler and ECSV input handler.
- Improved performance when multiple threads are accessing synthetic columns or subsets concurrently.
- Statistics calculations in Statistics Window now execute in parallel.
- New **Approximate Quantile Calculation** option in Statistics Window that runs in limited memory.

Minor enhancements:

- COOSYS and TIMESYS attributes are now preserved during VOTable I/O for table PARAMs (as well as for FIELDs, which was already the case). However, they are not currently displayed in the application GUI for table parameters, only for columns.
- The Table I/O section has been somewhat reorganised and tidied up.
- Add new options `acos` and `cos` for **Scaling** in the Aux Axis Control etc; these provide linear-like stretch functions with steeper/flatter ends, which may be useful for shading by quantities with most variation near to/far from the middle of the range.
- Perform part of the Sort operation in parallel.
- Slightly improved documentation for CDS Upload X-Match Window.
- Add **Plot|Parallel Caching** menu option in plot windows; this is experimental and not generally recommended.

Bugfixes:

- Upgrade JEL to v2.1.2: fixes a bug that could cause trouble when reloading saved sessions with certain boolean-valued synthetic columns: load could fail or values could be inverted. Also some efficiency improvements in object creation.
- Fix histogram ranging bug (plot failure under certain circumstances).
- Avoid inserting `NULL_VALUE` custom metadata entries into ECSV output.
- Fix annoying column selector bug that sometimes required an extra click.
- Fix up MacOS application packaging issue so that (hopefully) `topcat`/`stilts` startup scripts work on Mac M1 architecture.
- Cope better with infinite values in aux plot coordinates.

Version 4.8-1 (10 June 2021)

File formats:

- Apache Parquet format is now supported for input and output (note not available in all configurations).
- AAS Machine-Readable Table (MRT) format is now supported for input.
- ECSV format input and output handlers are upgraded to version 1.0 of the ECSV format, meaning they can now read and write array-valued columns.
- Add configuration option `header` for CSV input handler, to indicate whether header line is present.
- Add configuration option `maxSample` for CSV and ASCII input handlers to reduce 2-pass read time.
- Variable-length array-valued columns in FITS tables (`P`/`Q` descriptors) can now be read even in compressed or streamed FITS files.

Other new functionality:

- Add new XYArray Layer Control with associated plot forms Lines, Marks, YErrors and XYErrors forms for plotting array-valued columns such as per-row spectra and time series.
- Add new functions to class Arrays for working with array values: `arrayFunc`, `intArrayFunc`, `sequence`, `constant`.
- Add new SAMP Message activation action for sending custom SAMP messages.
- The **Cumulative** option in histogram-like plots can now take the values **none**/**forward**/**reverse**, not just true/false.
- Introduce the **Aitoff0** projection alongside **Aitoff** in sky plots. In the same way as **Car**/**Car0**, this gives an Aitoff projection but with the longitude=0 line at the left/right edge instead of the center.
- The **Table** tab in the TAP window now reports approximate row count if provided by VODataService 1.2 `nrows` element (TableSet-VOSI metadata acquisition mode only).
- Add new table scheme test.

Minor enhancements and workarounds:

- Maximum line thickness in line plots raised from 5 to 9.
- Collapse whitespace in some metadata display items, such as column/parameter Description entries.
- The fits-var output handler now avoids use of the THEAP keyword (no pre-heap gap is written). Heap padding is legal FITS, but bugs in other FITS software mean that some third party components (including `fverify` in FTOOLS e.g. v3.14-3.50) have problems with such files.
- JDBC output no longer attempts to create VARCHAR(0) columns.
- Space-delimited ECSV files now write empty fields quoted.
- Unknown or unsupported column datatype values in ECSV files are now treated like `string` rather than causing table read failure.
- Empty strings in FITS 1-character columns are now returned as blank values rather than ASCII NUL ('`\0`').
- Undersized, including zero-length, strings written to FITS columns are now by default terminated with an ASCII NUL rather than in some cases padded with spaces.
- Modify Example RegTAP "TAP accessURLs" query in TAP window to avoid aux resources.

Bugfixes:

- Fix regression bug in previous release (v4.8) that broke use of `jdbc:` URLs and SQL Query window.
- Fix regression bug in previous release (v4.8) that meant table drag'n'drop wasn't working.

- Fix regression bug in previous release (v4.8) that failed to re-read Byte-typed synthetic columns from saved sessions in some circumstances.
- Upgrade cds-healpix-java library to v0.29.3; this avoids occasional sky crossmatch failures (termination with error) introduced at TOPCAT v4.7-1.
- Fix bug in **FOV** tab of sky plot axes control that could point to the wrong position for some projections such as **Car0**.
- Fix TAP metadata column display so that numeric non-standard items are actually shown; previously the columns appeared but with blank values.
- Fix problem with evaluation of a synthetic column/subset referencing a second synthetic column whose expression gets edited to yield a different type.
- Fix issue with cumulative histograms; bars beyond the last sample are now displayed with total value not zero.
- Fix bug that meant writing long (>2Gb) fits-var files could output incorrect/corrupted FITS.
- Fix long-standing file caching bug; mostly seemed to affect large (>2Gb) streams.
- Fix long-standing logic error in ASCII/CSV input handler that could misidentify column types and cause read failures.
- Fix plot failure when trying to use Histogram aux scaling and aux limits set explicitly outside of data range.
- FITS TZERO headers are now written correctly with numeric values rather than string values.

Version 4.8-2 (15 October 2021)

New functionality:

- Ellipse selection in the Draw Algebraic Subset action can now be rotated rather than just aligned with the axes ("Ellipse" is renamed "Aligned Ellipse", and there is a new option "Rotated Ellipse").
- A new **Thickness** control has been added to several plot forms to enable lines thicker than a single pixel when drawing error bars, arrows, outlines, pair links etc: `Vector`, `Error Bars`, `SkyVector`, `SkyEllipse`, `SkyCorr`, `XYEllipse`, `XYCorr`, `SizeXY`, `Area`, `Polygon`, `Poly4`, `Link2`, `YErrors`, `XYErrors`.
- New Area type **UNIQ** to plot single HEALPix tiles e.g. from MOC files in the Area Layer Control.
- The Log Window, which has been broken since v4.7-1, and not much use before that, has been rewritten. It now presents logging information, including things like external service access details, in a useful way.
- New functions in class VO to check syntax of UCD and VOUnit strings.
- Add **Auto-Invoke** option to View Datalink Table activation action's Datalink display.
- Copy/paste is now possible from the Activation Window **Results** panel.
- Add Julian Day manipulation functions `jdToMjd` and `mjdToJd` to class Times.

Minor enhancements and behaviour changes:

- Replace function `julianToUnixSec` in class Times, which didn't do what it said it did, with `jdToUnixSec`.
- The `mjdToIso` and `mjdToDate` functions in class Times now prepend the string "`(BCE)`" to dates before the common era.
- Guess meaning for some non-standard COORDSYS values in HEALPix-FITS files, e.g. allow "GALACTIC" instead of "G".
- Revert to sequential processing in some cases for HEALPix plots to reduce resource usage.
- Columns that are all blank in ASCII-like tables (CSV, ASCII, TST) are now interpreted as String not boolean.

- Files compressed using multi-stream bzip2 compression (e.g. `pbzip2` output) are now supported alongside single-stream bzip2.
- Simplify (aligned) ellipse expression in algebraic subset; it doesn't need a square root.
- The Fast config option has been withdrawn from polygon-plotting forms `Area`, `Polygon`, `Poly4`; the shape-filling algorithm has been improved, and it's no longer necessary to choose between speed and accuracy.

Bugfixes:

- Fix serious threading bug that could return nonsense values from fixed-length string fields during parallel processing, for instance statistics calculation, of large cached or randomised tables.
- Fix behaviour for a couple of activation actions (Invoke Service and Invoke Datalink Row), that invoked the wrong row for tables with sorted rows.
- Fix aux ranging bug that meant supplying a fixed Maximum Aux value was ignored without a fixed Minimum Aux value. This was a regression bug introduced at v4.6-3.
- Upgrade of cds-healpix-java library to v0.30.2; this fixes some HEALPix cell plotting bugs.

Version 4.8-3 (31 January 2022)

New functionality:

- The PDS4 (NASA's Planetary Data System v4) file format is now supported for input tables.
- The SAMP MType `table.load.pds4` is now supported alongside some other format-specific table.load.* messages.
- Add new plane plot forms StatLine and StatMark; these can plot per-element averages etc of fixed-length-array-valued columns such as spectra.
- Plots using Grid Form now allow export of the calculated density map to a table.
- New per-element quantities available in Statistics Window for fixed-length-array-valued columns (Array Means, Array Sums etc).
- New class Randoms contains pseudo-random number generation functions. Special token `$random` is now deprecated.
- Provide quantile options for some Combine selectors.

Minor enhancements and behaviour changes:

- Add new colour map Painbow.
- Improve the way that Auto-Invoke works for View/Invoke Datalink Row activation types; these now work properly with the sequence auto-play feature.
- Modify ADQL sent to DaCHS RegTAP services; should speed up registry queries in some cases, especially when using the Description field.
- Subject keyword matching in registry searches is relaxed; a keyword can match part of a subject term, it doesn't have to match the subject term exactly.
- Follow HTTP->HTTPS 3xx redirects when parsing VOTables in a few places that it didn't work before.
- The test for whether a newly-defined row subset adds an entry to the subset list or replaces an existing entry is now based on case-insensitive, rather than case-sensitive, name matching.
- Subset row count is no longer recalculated when synthetic subset expression is selected/highlighted but not changed.
- Improve identification of TIME_TT2000 columns as time values in certain CDF files.
- Improved auto-ranging for some histogram-like plotters; changing style attributes

like normalisation will now reset the plot view as appropriate.
- Improved auto-ranging for Gaussian plot.
- Make a couple of adjustments to SVG graphics output: output is now to a bare `svg` element (no XML or DOCTYPE declaration), and a `viewBox` attribute is included which may improve scaling behaviour in some contexts.
- Permit leading 's' in ASCII-MOC area specifications, following MOC 2.0.

Bug fixes:

- Fix Session Save/Load bug; row subsets created using the Invert Subset action applied to an algebraic subset were not being loaded correctly (they always contained all rows).
- Fix bug in parsing `MOC-ASCII` strings in Area plot; trailing depth specifier was interpreted as cell index and some cells near end of MOC were omitted/misshapen.
- Fix failure when attempting to read unsigned 32-bit integer values from parquet files.
- Bugfix update of JCDF to v1.2-4.

Version 4.8-4 (6 April 2022)

Performance

- Substantial I/O performance improvements, mainly for FITS and VOTable formats e.g.: writing to FITS 2x, reading FITS from a stream 2x, reading VOTable with inline BINARY/BINARY2 4x, writing VOTable with inline BINARY/BINARY2 2x, writing VOTable with TABLEDATA 1.5x.
- A new **Parallel Execution** toolbar button is present in the Match Windows. This should speed up matching for large tables on multi-core machines. It should make no difference to the results, but since it is less well tested than the sequential mode, at this release it is turned off by default.
- Other matching performance improvements.
- FITS I/O is now all done internally, there is no longer a dependency on the nom.tam.fits package.

New functionality

- New class Bits with bit manipulation functions `bitCount`, `hasBit`, `toBinary`, `fromBinary`.
- Add RMS Deviation report to LinearFit plot output.
- Add Select All and Deselect All buttons to the Multiple Table Save and Session Save panels.
- Add configuration options `compact` and `encoding` to VOTable output handler. By default thin (<=4 column) TABLEDATA VOTables are now written in "compact" mode, using reduced whitespace.
- FITS BINTABLE headers used as table parameters now support FITS 4.0 long-string syntax (CONTINUE records).
- FITS header values of the form "(a,b,c,...)" are now interpreted where possible as numeric arrays; this works for long-string values (CONTINUE records) as well.

Bug fixes and workarounds

- Fix bug that broke plots (and other things?) using expressions that generated RuntimeExceptions. This was a regression bug since TOPCAT 4.8.
- Upgrade JEL to v2.1.3-pre1. This fixes a bug that caused evaluation failure when comparing a String against null.
- Defend against OutOfMemoryErrors in the presence of very deep MOCs.
- ECSV format now preserves table name.
- Upgrade Unity library to 1.1 pre-release and improve VOUnits validation reporting

(new status "GUESSED_UNIT").
- FITS BINTABLE reader now copes with (illegal?) embedded spaces in TDIMn headers.
- Adjust MRT null handling; "-" in a single-character field no longer interpreted as null.

Version 4.8-5 (10 June 2022)

New Functionality

- Add new plane plot form ArrayQuantile. This can be used to e.g. to plot medians of multiple spectra that are not aligned on the same wavelength grid.
- When using the XYArray Layer Control, you can now omit either the X or Y array value, and a suitable linear sequence will be assumed.
- Add X/Y Offset options for Label plot form.

Performance

- Reduce number of file mapping calls by FITS readers.

Bug Fixes

- Fix FITS parsing issue that could result in StackOverflowError for long array-valued headers.
- Fix bugs in StatLine/StatMark forms that incorrectly treated blank array values and negative array values on logarithmic axes.
- Fix bug in multi-threaded read of string columns from colfits files.
- Table columns in Data Window are now always wide enough for column name.
- Fix legend positioning bug for Label form.
- Aitoff sky projections are now more correctly documented as Hammer-Aitoff (though not renamed in the UI).

Version 4.8-6 (8 July 2022)

Enhancements

- The View Datalink Table activation type now provides an alternative option for giving Datalink table location; you can supply *either* the full URL *or* the {links} endpoint and an ID value.
- Adjust datalink table row-matching heuristics to work with less constrained datalink metadata; this provides smoother operation for (e.g.) Gaia DR3 datalink activation actions.
- Modify functions `add`, `subtract`, `multiply`, `divide` in class Arrays; these now all take either two array arguments or an array and a scalar in either order.
- New function `dotProduct` in class Arrays.
- Provide `value*` functions for column references to strangely-named columns.
- Tweaks to default plot setup to avoid using identifiers as coordinates in some cases.

Bugfix

- Fix FITS output so it doesn't fail when attempting to write metadata with non-ASCII Unicode characters.

Version 4.8-7 (5 October 2022)

New functionality

- Add new **Activated** Row Subset to each table; this can be used to highlight plotted rows on activation.

- Add new plot form Handles to mark reference positions for X/Y array data, thus allowing activation and graphical selection with XYArray plots.
- New function `sequence(n,start,step)` in class Arrays.
- New functions `tfcatStatus` and `tfcatMessage` in class VO for validating instances of the Time-Frequency Radio Catalogue format.
- Provide new option TFCAT for Area type in Area Layer Control, giving partial support for TFCat shape descriptions.
- Semi-standard `nrows` column from `tap_schema.tables` table is now recorded and displayed as TAP table row count if present.

Performance:

- Crossmatching performance improvements: sky matching does better pre-selection of potential matches based on sky region, post-processing row sorting etc accelerated, various other steps parallelised. Large matches might typically be about twice as fast as before.
- Matching is now done in parallel by default; sequential execution can be selected using the Parallel Execution Tuning control if preferred.

Minor changes and usability improvements:

- Improve Row Subset redefinition behaviour for plotting: when a subset name is re-used, any plot configuration (colours etc) set up for the old subset is retained for the new one, and it keeps its position in the displayed stack.
- Improve formatting of some sections of this document.
- Extend ranges a bit beyond (0-1) for axis range sliders.
- Prevent image display window from grabbing focus during e.g. Display HiPS Cutout activation.
- When a new column is added, the Column and Table viewer window displays are now scrolled to ensure that the new entry is visible.
- When a new Row Subset is added, the Subset window display is scrolled to ensure that the new subset is visible.
- Blob drawing doesn't disappear until after new subset is defined.
- Add new section listing Colour Maps to this document.
- Add some new colour maps from CMasher: Cosmic, Ember, Gothic, Rainforest, Voltage, Bubblegum, Gem, Chroma, Neon, Tropical (sequential); Guppy, Iceburn, Redshift, Pride (diverging); Infinity (cyclic).

Bugfixes and workarounds:

- Improve behaviour of some Axis Control tabs; in particular the **Range** and **View** tabs of the Cube and Sphere plot windows, and the **FOV** tab of the Sky plot window, which have long been somewhat broken, now work properly.
- Prevent annoying lines appearing in plot window before plot is displayed.
- Fix VOTable reader so BINARY/2 VOTables with no columns don't read forever.
- Fix PDS4 reader to accept columns of type `ASCII_Numeric_Base16` without the read operation failing.

Version 4.8-8 (20 April 2023)

Axis drawing improvements and changes:

- Secondary X and Y axes can now be drawn in the Plane and Histogram Plots; see Plane Axes control Secondary tab and example plot.
- Secondary T and Y axes can now be drawn in the Time Plot: see Time Axes control Secondary tab.

- Matching ticks are now plotted by default on all four sides of Plane, Histogram and Time plots, rather than just on primary (bottom/left) axes. This can be configured with the new **Shadow Ticks** option in the Axes Control Grid tab.
- Reduce frequency of minor ticks.
- Major tick marks on axes now extend only inside the plot bounds not outside.
- Grid lines plotted in the Plane, Time and Sky plots are now all partially transparent and plotted over the plot content (previously they were opaque and the Sky grid was drawn over, while the Plane and Time lines were drawn under, the plot content). Grid line transparency is controlled by the new **Grid Transparency** slider in the Axes Control Grid tab. Grid line colour can now be controlled in Time plots as for Plane and Sky.
- Add new options **ExternalSys** and **InternalSys** to **Label Positioning** selector of Sky Axes Control Grid panel, to display lon/lat axis names alongside axis values.

Other new functionality:

- You can now draw **Ellipse** shapes when using Draw Algebraic Subset with the Sky Plot.
- New function `inSkyEllipse`.
- Upgrade ADQL support in the TAP window to accommodate ADQL 2.1, thanks to new version of Grégory Mantelet's VOLLT ADQL library. Syntax validation and highlighting, and help from the ADQL Examples menu and Hints tab, are now done according to the ADQL version (2.0 or 2.1) shown in the Service Capabilities panel, and to the declared capabilities. ADQL 2.1 nominally corresponds to version PR-ADQL-2.1-20230418. The TAP Service metadata display tab now also has two additional headings to display service-specific characteristics: ADQL 2.1 Optional Features and Non-Standard Language Features.
- The Columns Window and Subsets Window have new **Edit Column/Subset** actions respectively, that open a popup window to edit name, expression, etc. These are now activated automatically if you enter an invalid algebraic expression in the metadata JTables, rather than just wiping out your new expression and reverting to the previous one.
- New action **Collapse Columns to Array** in the Columns Window.
- The **Examples** menu in the TAP Window now presents Service-Provided examples in a hierarchical menu if the service `/examples` endpoint uses DALI-examples continuation documents.
- Make use of DataLink metadata introduced at DataLink 1.1. Values `content_qualifier` and `local_semantics` are now displayed in the DataLink Window and if present these and the Service Descriptor ContentType are used to make better decisions abut manipulating links.
- Add **Sort Axis** config option to Lines plot form.
- Add configuration options `readMeta` and `hierarchicalNames` to the GBIN input handler.
- VOTable service descriptor I/O now preserves `contentType` and `exampleURL` PARAMs, introduced in DataLink 1.1

Workarounds and minor behaviour improvements:

- Modify column width determination in text-like output formats (`text`, `ascii`, `ipac`) to avoid occasional unwanted truncation of formatted values. Tables are now read in two passes, the first to establish column widths and the second to write the data. By default all rows are sampled, but the `sampledRows` option can be configured so that only some rows are sampled, which is more like the old behaviour.
- Update Ucidy library to v1.3, corresponding to UCDList 1.5 EN.
- If only one single point is visible in a Sky Plot using the (default) sin projection, the default view is now zoomed out to the whole sky rather than zoomed in to a few

milliarcsec.
- Improve ECSV reader performance, especially for Gaia DR3 bulk download files (which use semi-standard "`null`" token).
- Write empty string not semi-standard "`nan`" token for NaN in ECSV writer.
- Table parameters, as well as columns, named in column selectors now include their units metadata.
- Avoid oversized service icons in TAP window Service tab.
- Recognise "`smoc`" as an alternative to "`moc`" when examining xtype values for area format defaults.
- Improve reporting and defaulting of activation action URL invocation.
- Better titling of Activation Action windows.
- The JDBC input scheme should now read columns that are array-valued in the database as array values that can be used in TOPCAT; previously they were read as opaque `Array` objects.
- Make FITS and VOTable output handlers robust against input tables that declare incorrect row counts.
- The PDS4 reader now reads `Ascii_Numeric_Base16/8/2` fields as numeric not string (updated pds4-jparser library code).
- Slight change to RESOURCE structure of Primary HDU metadata in multi-table FITS-plus output. This fixes a problem in which saved Service Descriptors could end up associated with the wrong tables.
- HTTP redirects with response code 308 (Permanent Redirect) are now handled in the same way as 307 (Temporary Redirect).
- Withdraw **Antialias** config option from Lines plot form, since it didn't really work.
- Improve handling of illegal sky coordinates (latitude out of range) in sky matching; in v4.8-7 only, bad positions caused match failure, now they are just ignored.
- Improve transparency rendering in SkyGrid plot.

Bug fixes:

- Fix a nasty bug that could plot SkyVector, SkyEllipse and SkyCorr forms incorrectly. The numeric value of the shape size coordinates (ellipse radii, vector extents) was interpreted in degrees before being rescaled as appropriate, so that values of a few tens or larger resulted in signficant distortions. Now fixed.
- Fix stability issue when using the **Best Match, symmetric** selection option (which should be used with caution) in the Pair Match Window. Results may be different in crowded regions, and may also have differed between v4.8-6 and v4.8-7, but are not obviously more or less correct.
- Fix Cartesian matcher overflow issue; for very large ratio between coordinate extent and match radius this could cause pathologically slow, though not incorrect, crossmatching.
- Fixed bug which failed to plot some markers of multi-marker shapes like Mark2 when multithreaded (i.e. for large tables). Regression bug since introduction of multithreaded plotting in TOPCAT v4.7.
- Fix line drawing bug that meant dotted and dashed lines were often not displayed correctly. This was a regression bug introduced in TOPCAT 4.6-3.
- Fix auto-range and export bugs that ignored final bar for forward cumulative histograms.
- Fix Basic image viewer in Display Image/Display Image Region activation types to display (uncompressed) FITS images, by re-including nom.tam.fits library in standalone jar files; this was a regression bug, broken since v4.8-4.
- Avoid some plot failures related to oversized legends and undersized plot windows.
- Adjust colour ramp (aux axis) painting to avoid faint white stripes seen when vector graphics (e.g. PDF) output was rendered in some external viewers.
- Fail with an error rather than silently reading a broken table when encountering GaiaTools/zStd-jni bug during GBIN input.

- Empty/invalid fields encountered by the PDS4 reader no longer cause the table read to fail.

Version 4.9 (1 November 2023)

New functionality:

- New Corner Plot Window added.
- The authentication handling has changed, and it is now possible to log in to TAP services that comply with the (draft) "SSO_next" proposal for advertising authentication methods in the VO. The `star.basicauth.user`/`star.basicauth.password` system properties can no longer be used to set authentication information globally for the application (this was a rather insecure practice anyway).
- Column metadata export actions (**Import as Table**, **Save as Table**) added to Columns Window.
- You can now Search for columns by metadata in the the Columns Window. This is available as the **Search Column** action in the toolbar and Columns menu, and also using the new popup menu available from the metadata display JTable.

Minor enhancements and behaviour changes:

- MacOS topcat-full.dmg file should now work for both Intel and Apple Silicon architectures.
- Add new **Sky + X with Errors** Match Algorithm option.
- Update hard-coded Registry URLs for Euro-VO and STSci to use https not http protocol.
- Avoid annoying popup in Cone/SIA/SSA windows when position is resolved before radius is filled in.
- Flag ConeSearch 1.03-style error result tables.
- Column names with embedded carriage returns are now recognised in column selectors.

Bugfixes:

- Fix a bug in the Pair Match and Multiple Match windows which assigned the created `matched`/`match<N>` subsets to completely the wrong rows. These subsets are only created for some match types, and everything else about the matches was unaffected. STILTS was not affected. This was a regression bug since v4.8-4. Thanks to Claire Greenwell (Durham) for reporting this.
- Fix bug which ignored some points when painting multi-threaded, multi-dataset or multi-subset plots. This bug was introduced in v4.7 and was supposed to be fixed at v4.8-8 but wasn't.
- Fix TAP window to select ADQL-2.0 not ADQL-2.1 if service declares no ADQL versions.
- Fix TAP capabilities parsing to identify language features with non-standard capitalisations.
- Fix fencepost error in reverse cumulative histogram plotting.
- Fix missing secondary X axis bug in stacked Time plots.
- Fix bug in histogram plotting that could cause crashes for small ranges far from the origin.
- Fix aux axis/legend positioning issue in sky plot (regression bug introduced at v4.8-8).
- Fix bug in ADQL syntax highlighting related to delimited identifiers (quoted table names). This bug was only introduced with the VOLLT upgrade in the last release (TOPCAT v4.8-8).
- Fix issues to do with sky coordinate system choice that sometimes caused SkyGrid

and Healpix layers to generate broken STILTS commands for export.

Version 4.9-1 (29 February 2024)

TAP window improvements:

- ADQL functions and declared features, including UDFs, are now listed in a separate **ADQL** tab with the TAP metadata.
- Schemas and tables are now listed by default in service-defined order in the TAP window metadata tree. A new **Sort** control can switch back to alphabetic.
- Initially selecting, or subsequently double-clicking, a table in the TAP Select Service tab will now select it for display in the Use Service tab metadata panel.
- The TAP Use Service tab now initially displays the Columns not the Schema metadata tab.
- Choose better columns for TAP window ADQL examples under some circumstances.
- Use TAP 1.1-specific communications (avoid `REQUEST=doQuery`, use `RESPONSEFORMAT` not `FORMAT`) in data queries to TAP services known to declare TAP 1.1 compliance.
- Improve scrolling in TAP window Service, Schema and Table tabs.

Other new functionality:

- Add support for HAPI time-series services: query window, input handler and scheme.
- The Cube Plot Window has a new **Coordinates** selector that lets you specify positions as Cartesian 3-vectors (XYZ) or spherical polar (Lon, Lat, Radius) coordinates as alternatives to the normal Cartesian components (X, Y, Z).
- Add Circle mode for the Plane Plot in the Draw Algebraic Subset action.
- The spectrogram plot now tries to plot spectra on a spectral axis, as controlled by the new **Scale Spectra** style control.
- New array functions `loop` with arguments `(start,end)` or `(start,end,step)`.
- The first argument of the `inMoc` and `nearMoc` Coverage functions can now be an ASCII MOC string as well as a MOC file location or VizieR table identifier.

Minor enhancements and behaviour changes:

- The gap between vertically stacked plots in the Time Plot window can now be adjusted using the Frame Control Spacing tab.
- The Time Axis label can now be configured in the Time Plot.
- Add instructions in the Time Plot documentation about abusing it to stack non-temporal plots.
- The Function layer control is now available from the Corner Plot Window.
- ISO-8601 conversion functions in Times now accept a trailing "`Z`" even for date-only specifications, and now accept `YYYY-DDD` format for dates as well as `YYYY-MM-DD`.
- The **CoosysRefposition** metadata item (VOTable 1.5) is now listed if present in the Columns Window.
- Some fixes/improvements to the `topcat` startup script. It now works better in the MacOS DMG image, it's more robust against pathnames with embedded spaces, and it will use `topcat-extra.jar` if present in preference to `topcat-full.jar`.
- Update list of VizieR mirrors servers in VizieR Query window.
- Add new colour map Sunset.
- Use authentication and HTTP-level compression for external (href-referenced) VOTable STREAM data.
- Improve error reporting for corrupted/truncated FITS files.
- Upgrade VOLLT adqlLib to official 2.0-beta release.
- Upgrade Unity to v1.1, which corresponds to REC-VOUnits-1.1. Only minor

functional differences expected since previous version (1.1-b1).
- Upgrade snakeyaml library (used for ECSV headers) from 1.25 to 2.2. No change in behaviour (or security) expected, but prevents vulnerability warnings in some circumstances.

Bugfixes:

- Fix sky match failure in case of very large error radius (>22 deg).
- Fix bug in legend/aux axis text style configuration for Time Plot. This was a regression bug present only in TOPCAT v4.9.
- Fix STILTS control bug that did not include all settings for the Corner Plot.
- Fix FITS multi-table read bug that sometimes generated a spurious error popup when reading basic FITS files.
- Cope better with out of range pixel indices in healpix plots.
- Some spectrogram/time plot bug fixes.
- Fix some problems with images in the PDF version of this document.
- Update JSAMP to v1.3.8. This may improve hub Web Profile browser compatibility.

Version 4.10 (7 August 2024)

Source code

At this version the source code and build system have been substantially tidied up so that the application can be built straightforwardly using modern versions of Java (8, 11, 17, 21) with a minimum of warnings. This should be mostly invisible to users, but a few behaviour changes may be observed:

- Some changes have been made to URL handling. Syntactically invalid URIs are now mostly rejected. This should not be noticeable in most cases, but questionable usages like embedded spaces in URLs may need to be replaced by their `%`-encoded equivalents.
- The SoG internal image viewer has been withdrawn, since the required JAI classes are now unlikely to be available.
- The `topcat-lite` configuration is no longer supplied, since it's no longer significantly smaller than `topcat-full`.
- WebStart is no longer supported, since it has been removed from more recent versions of Java.

I/O handler improvements

- Documentation for I/O handler config options now mostly reports their default values.
- Add to this document detailed documentation of FITS-plus and Wide FITS conventions.
- FITS output handling improved and reorganised to provide more flexible configuration options; this is now all documented under the FITS heading, there is no longer a separate `colfits` section in the documentation.
- Xtypes are now written into FITS headers using the non-standard header card `TXTYPnnn`.
- Non-standard FITS headers `TUCDnnn` and `TUTYPnnn` are now written with comment parts, space permitting.
- Upgrade parquet support libraries to parquet-mr 1.13.1. This means that Snappy compression is now supported in parquet for MacOS ARM, as well as other, architectures.
- Add support for LZ4_RAW compression to Parquet I/O handlers.
- Decrease size of parquet output files in most cases when writing NaNs and empty strings/arrays.
- Parquet input handler now copes with some additional variants of array-valued

columns.
- New config option `date` for VOTable output handler.
- New config options `compression`, `usedict` for Parquet output handler.
- New config option `tryUrl` for Parquet input handler, set false by default to avoid cryptic error messsages when trying to open remote parquet files.
- New config option `useFloat` in MRT input handler, set false by default to avoid a rare bug that could read large values as infinite.

Minor enhancements and behaviour changes

- Treatment of null values in the astrometric epoch propagation functions `epochProp` and `epochPropErr` has changed; null values for parallax and proper motion are now treated as if zero rather than invalidating the propagation. This (partly) follows behaviour of corresponding functionality in Gaia and VO ADQL propagation UDFs, and is more likely what you want to see.
- Add Corner Plot Table option to DataLink Window Action selector.
- MacOS DMG now bundled with Java 11 not Java 8, fixes poor rendering of selected tab label in JTabbedPane. This also avoids runtime warnings on some MacOS versions about secure coding.
- About application button on MacOS should now display custom About TOPCAT window.
- Improve metadata in tables created by Match windows; they now record match parameters such as max error.
- Column XType is now displayed in the Columns Window.
- Avoid writing `arraysize="1"` in `FIELD`/`PARAM` elements of VOTables with version >=1.3, in accordance with VOTable 1.3 Erratum #3.
- Avoid non-VOUnit units in output VOTables in some cases.
- New `test` scheme options "`g`" and "`w`".
- Upgrade JSAMP to v1.3.9.

Bugfixes

- Fix annoying bug where text pasted into the TAP URL selector got mixed up with the placeholder text.
- Fix some bugs to do with authentication and metadata acquisition in the TAP window.
- Fix issue with display of multi-line text in TAP window ADQL tree on MacOS.
- Fix bug in TAP window ADQL tab tree population in presence of illegal syntax for UDF form.
- Fix ADQL Hints tab in TAP window which failed to load for some versions of Java.
- Initial highlight of selected table in TAP window now works as well for delimited-identifer table names.
- Bugfix update of CDS HEALPix library to v0.30.3, prevents occasional plot failures.
- Fix parquet input handler bug related to compression.
- Fix broken STC-S BOX parsing for Area plots.
- Fix VOTable output bug that wrote infinite floating point array elements to TABLEDATA as "`+/-Infinity`" rather than "`+/-Inf`".
- Avoid stack overflow errors for long ASCII MOCs in Coverage functions.
- Fix authentication bug to do with redirects.
- Fix column search bug.

Version 4.10-1 (6 November 2024)

New functionality

- New STILTS Window button available from some windows: Match, TAP, CDS Upload X-Match, single cone, SIA, SSA, and multiple cone, SIA, SSA.

- New **Tick Label Angles** configuration option in Grid tab of most plot type axis controls; allows angled labels that can accommodate more major ticks on crowded axes. Cube/Sphere (sometimes) and Corner plots now use angled axis labels by default.
- New config option **Sideways** in Histogram Bins control General tab. This allows histogram-like plots to be accumulated on the vertical, not horizontal, axis.

Minor enhancements

- STC-S encoding in Area-like plots now copes with UNIONs of all shapes, not just of POLYGONs.
- More units (seconds with SI prefixes) are now properly handled in VOTable `TIMESYS`-referencing fields.
- Cope with whitespace round input URLs better than previous version (regression).
- Update JSON-java library to 20240303. Behaviour is not expected to change in interesting ways, but some vulnerabilities are addressed.

Bugfixes

- Fix regression bug (since v4.9) in HTTP 3xx redirect handling that failed to cope with relative Location field values.
- Fix regression issue (since v4.10) in URL construction that meant (at least) use of the `-running` command-line argument didn't work properly with relative filenames.
- Fix bug in TAP curl logging.

Version 4.10-2 (8 November 2024)

Bugfix

- Fix failure when using STILTS window in Match windows for most matcher types other than Sky.

Version 4.10-3 (7 March 2025)

New functionality

- The Sort Order can now be defined using multiple columns or expressions, not just a single column.
- Add **Isometric** option to Cube Axis Control.
- Add new Private Auxiliary and Private Weighted plot shading modes PAux and PWeighted.
- Upgrade CDS MOC library to v6.31; now reads MOC 2.0 compliant FITS and ASCII representations.
- New MOC handling functions added to class Coverage: `mocUniq`, `mocUniqToOrder`, `mocUniqToIndex`, `mocSkyProportion`, `mocTileCount`.
- New class Json contains expression-language functions for use with JSON text (experimental).

File formats (Parquet and CDF)

- Parquet input and output handlers now support the VOParquet convention for embedded metadata.
- The parquet I/O handler in `topcat-extra.jar` now supports ZSTD as well as some other compression algorithms.
- Fix some significant parquet I/O bugs related to array-valued columns.
- The parquet reader can now read data with the un-annotated BINARY (BYTE_ARRAY) type; it is interpreted as a byte array.
- The CDF input handler can now extract multiple tables from a single CDF file.

- Update JCDF to v1.2.5. Significant improvements to CDF read performance.

Minor enhancements

- Improve placement of filled polygons in plot forms Area, Polygon and Poly4. Filled polygons no longer need outlines round them to avoid annoying gaps between tiles.
- Opening Sky Plot Window when a MOC table is selected now automatically plots it like a MOC.
- MOC-ASCII and UNIQ shapes can now be plotted in the Area layer control of the Sphere Plot Window.
- TOPCAT now subscribes to the SAMP MType `coverage.load.moc.fits`. The MOC is loaded as a normal table.
- New TAP window menu option **TAP|Service Discovery|RegTAP 1.2**.
- Add **Load Tables** option alongside **Load Table** in the Action selector for the Invoke Service and Datalink Window.
- Table row counts can now be given with embedded underscores or exponential notation for Table Schemes skysim, attractor, test and loop.
- STScI RegTAP endpoint updated.

Bugfixes

- Restore a lot of content (boxed items) that had gone missing from the documentation available from the Help Window.
- Fix time format reselection in Time plot window when changing tables.
- Fix regression bug since v4.10 that failed to pick up Service Descriptors from VOTables loaded directly (mostly didn't affect tables loaded from services).
- Fix the Function layer to appear in only a single zone of the Time Plot.
- Fix regression bug since v4.10 that caused broken COUNT column in Import as Table result from Histogram Plot.
- Retain column ordering when selecting multiple columns at once for the **Insert Columns** action in the TAP window.