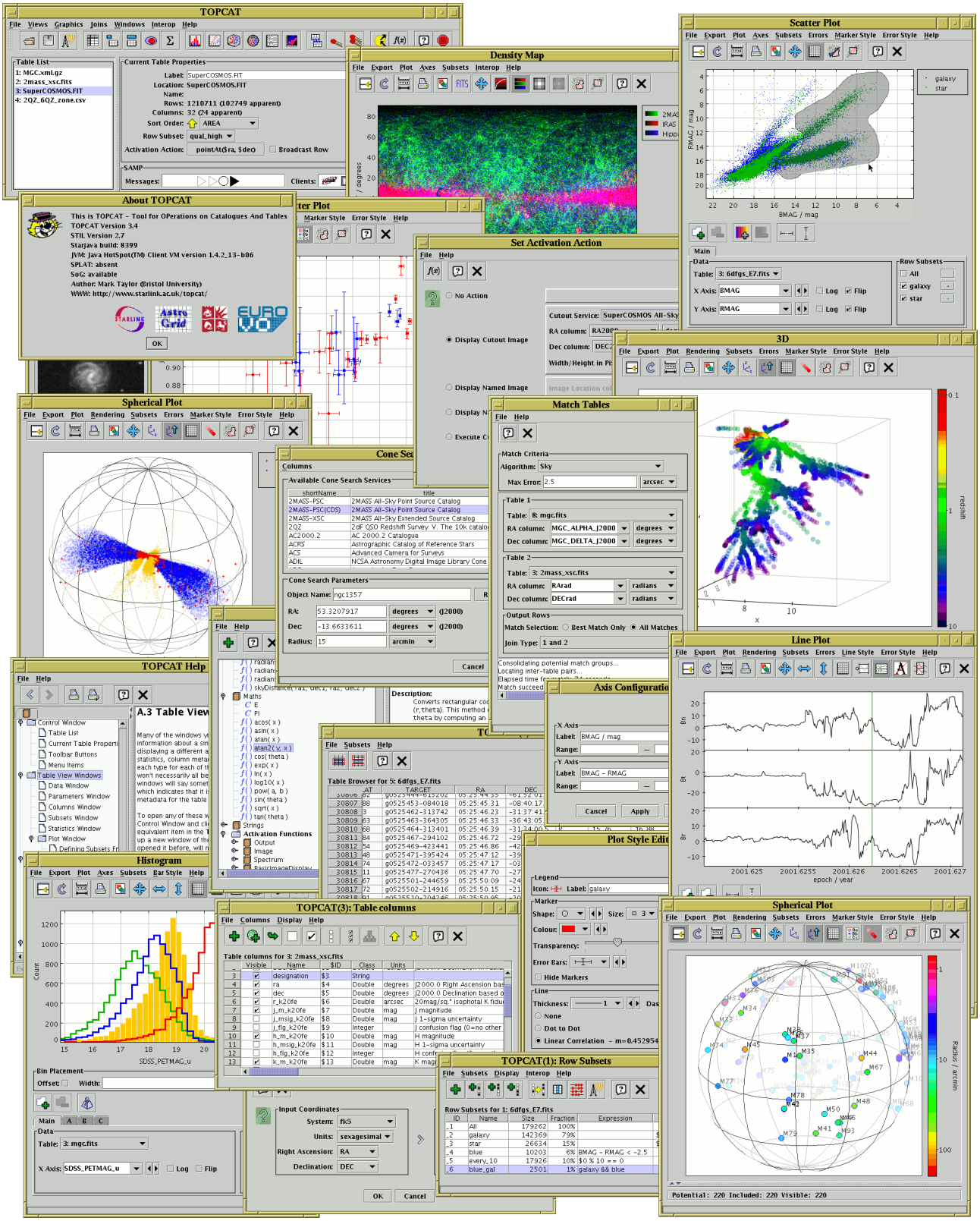
TOPCAT - Tool for OPerations on Catalogues And Tables

Version 3.4-1



Starlink User Note253
Mark Taylor
27 March 2009

$Id: sun253.xml 8609 2009-03-27 11:42:02Z mbt $

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 table using flexible matching algorithms, and visualisation facilities including histograms, 2- and 3-dimensional scatter plots, and density maps. 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.

TOPCAT is written in pure Java 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 **7**
2 Quick Start Guide **9**
3 Apparent Table **11**
3.1 Row Subsets 11
3.1.1 Defining Subsets 12
3.2 Row Order 13
3.3 Column Set 13
4 Table I/O **15**
4.1 Table Formats 15
4.1.1 Supported Input Formats 15
4.1.1.1 FITS 15
4.1.1.2 Column-oriented FITS 16
4.1.1.3 VOTable 16
4.1.1.4 ASCII 17
4.1.1.5 IPAC 18
4.1.1.6 Comma-Separated Values 18
4.1.1.7 Tab-Separated Table 19
4.1.1.8 SQL Database Queries 19
4.1.1.9 World Data Center 20
4.1.2 Supported Output Formats 20
4.1.2.1 FITS 20
4.1.2.2 Column-oriented FITS 21
4.1.2.3 VOTable 21
4.1.2.4 ASCII 22
4.1.2.5 Text 22
4.1.2.6 Comma-Separated Values 22
4.1.2.7 Tab-Separated Table 23
4.1.2.8 SQL Tables 23
4.1.2.9 HTML 23
4.1.2.10 LaTeX 23
4.1.2.11 Mirage Format 24
4.1.3 Custom I/O Formats 24
4.2 Table Locations 24

5 Joins and Matches ... **26**
5.1 Concatenating Tables ... 26
5.2 Matching Rows Between Tables ... 26
5.3 Matching Rows Within a Table ... 28
5.4 Notes on Matching ... 28
6 Algebraic Expression Syntax ... **29**
6.1 Referencing Cell Values ... 29
6.2 Referencing Row Subset Flags ... 30
6.3 Referencing Table Parameters ... 30
6.4 Null Values ... 31
6.5 Operators ... 32
6.6 Functions ... 32
6.6.1 Technical Note ... 34
6.7 Instance Methods ... 34
6.8 Examples ... 35
6.9 Adding User-Defined Functions ... 36
7 Activation Actions ... **38**
7.1 Activation Functions ... 38
8 Tool Interoperability ... **40**
8.1 SAMP control ... 40
8.2 PLASTIC control ... 41
8.3 Messages Transmitted ... 41
8.4 Messages Received ... 43
9 Invoking TOPCAT ... **45**
9.1 TOPCAT Command-line Arguments ... 45
9.2 Java Options ... 48
9.2.1 Class Path ... 48
9.2.2 Memory Size ... 49
9.2.3 System properties ... 49
9.3 JDBC Configuration ... 51
9.4 Tips for Large Tables ... 53
9.5 Examples ... 54
Appendix A: TOPCAT Windows ... **56**
A.1 Common Window Features ... **56**
A.1.1 Toolbar ... 56
A.1.2 Menus ... 56
A.1.3 JTables ... 57
A.1.4 Column Selector ... 58
A.2 Control Window ... **59**
A.2.1 Table List ... 60
A.2.2 Current Table Properties panel ... 60
A.2.3 SAMP Panel ... 61
A.2.4 Toolbar Buttons ... 62
A.2.5 Menu Items ... 64
A.3 Table View Windows ... **65**
A.3.1 Data Window ... 65
A.3.2 Parameters Window ... 67
A.3.3 Columns Window ... 69
A.3.4 Subsets Window ... 72
A.3.5 Statistics Window ... 74
A.4 Graphics Windows ... **77**
A.4.1 Common Features ... 77
A.4.1.1 Dataset Selectors ... 77
A.4.1.2 Axis Configuration and Zooming ... 79
A.4.1.3 Error Bars ... 81

A.4.1.4 Point Labels..... 84
A.4.1.5 Auxiliary Axes..... 86
A.4.1.6 Defining Subsets by Region..... 88
A.4.1.7 Exporting Graphics..... 90
A.4.2 Histogram..... 90
A.4.2.1 Histogram Style Editor..... 94
A.4.3 2D Plot..... 95
A.4.3.1 Plot Style Editor..... 98
A.4.4 Stacked Line Plot..... 101
A.4.4.1 Lines Style Editor..... 105
A.4.5 3D Plot..... 106
A.4.5.1 3D Plot Style Editor..... 110
A.4.6 Spherical Plot..... 112
A.4.7 Density Map..... 116
A.4.7.1 Density Style Editor..... 120
A.5 Load Window..... 121
A.5.1 Filestore Browser..... 122
A.5.2 Hierarchy Browser..... 124
A.5.2.1 Navigation..... 126
A.5.2.2 Table Searches..... 127
A.5.3 SQL Query..... 127
A.5.4 Cone Search..... 128
A.5.5 SIAP/SSAP Query..... 129
A.5.6 GAVO Millennium Database Query..... 131
A.5.7 Example Tables..... 132
A.6 Save Window..... 132
A.6.1 Enter Location..... 134
A.6.2 Filestore Browser..... 134
A.6.3 SQL Output Dialogue..... 135
A.7 Other Windows..... 137
A.7.1 Concatenation Window..... 137
A.7.2 Pair Match Window..... 138
A.7.2.1 Match Criteria..... 140
A.7.2.2 Column Selection Boxes..... 142
A.7.2.3 Output Rows Selector Box..... 142
A.7.3 Internal Match Window..... 144
A.7.3.1 Internal Match Action box..... 146
A.7.4 Activation Window..... 147
A.7.4.1 Image Viewer Applications..... 149
A.7.4.2 Spectrum Viewers..... 151
A.7.4.3 Web Browsers..... 152
A.7.5 SAMP Window..... 153
A.7.6 Help Window..... 156
A.7.7 New Parameter Window..... 157
A.7.8 Synthetic Column Window..... 158
A.7.9 Sky Coordinates Window..... 160
A.7.10 Algebraic Subset Window..... 161
A.7.11 Available Functions Window..... 162
A.7.12 Log Window..... 163
Appendix B: Algebraic Functions..... 165
B.1 General Functions..... 165
B.1.1 Coords..... 165
B.1.2 Arithmetic..... 167
B.1.3 Fluxes..... 168
B.1.4 Arrays..... 169

B.1.5 Tilings..... 170
B.1.6 Strings..... 170
B.1.7 Formats..... 171
B.1.8 Maths..... 172
B.1.9 Times..... 173
B.1.10 Distances..... 174
B.1.11 Conversions..... 176
B.2 Activation Functions..... 176
B.2.1 Browsers..... 177
B.2.2 BasicImageDisplay..... 177
B.2.3 Splat..... 177
B.2.4 SuperCosmos..... 178
B.2.5 Mgc..... 179
B.2.6 Output..... 179
B.2.7 System..... 179
B.2.8 Image..... 180
B.2.9 TwoQZ..... 180
B.2.10 Sdss..... 180
B.2.11 Sog..... 181
B.2.12 Spectrum..... 181
Appendix C: Release Notes..... 182
C.1 Acknowledgements..... 182
C.2 Version History..... 183

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 a given column
- 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
- Write modified tables out in original or different format

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 project, and has subsequently been supported by PPARC grant PP/D002486/1, VOTech and AstroGrid. 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, and should run on any J2SE 1.4 or 1.5 platform. 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. TOPCAT is available under the terms of the GNU General Public License.

2 Quick Start Guide

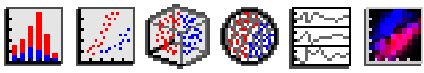
This manual aims to give detailed tutorial and reference documentation on most aspects of TOPCAT's capabilities, and reading it is an excellent way to to learn about the program. However, it's quite a fat document, and if you feel you've got better things to do with your time than read it all, you should 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-lite.jar` (see Section 9 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 and VOTable 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 fall into a few groups:

Load/save etc.

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

Help and information

The **Help** ( ) button appears in most windows - if you click it a help browser will be displayed showing an appropriate part of this manual. As well as the tool bar there are a number of menus along the top - most of the options just repeat those appearing on the toolbar, but a few less common ones may be available as well. 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, or 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 either or both of the `-disk` or `-Xmx` *NNN* `M` flags on startup - see Section 9.2.2.

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 and by all means 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.

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 windows.

3.1.1 Defining Subsets

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

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.

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 6.

Visible plotted points
In some of the graphics windows you can plot columns against each other, and subsequently zoom in and out using the mouse. If you zoom to display only some of the plotted points and then use the **New Subset From Visible** (or) button then a new subset will be created containing only rows represented by points in the field of view of the plot at the time.

Selected plotted points
For more control over which plotted points are to be included in a subset, you can use the **Draw Subset Region** ( ) button in some of the the graphics windows. This allows you to trace out with the mouse a region or regions of any shape, creating a new subset containing only those rows represented by the points within those regions.

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.

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 a selected column. 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 "(none)", 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 from this control, and change the sense by clicking on the arrow. 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 a column 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.

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 order in which columns are displayed in the Data Window and Statistics Window. The Columns Window shows all the known columns, including hidden ones, in Column Set order; whether they are currently visible is indicated by the (leftmost) "Visible" column.

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 box 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). This affects the Column Set, as you can see if you watch the Columns Window while you do it.

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

4.1 Table Formats

TOPCAT supports a wide variety of tabular data formats. In most cases these are file formats for tables stored as single files on a disk or at the end of a URL, but there are other possibilities, for instance a table you have opened could be the result of an SQL query on a database.

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` command line utility in the STILTS package.

The format handling is extensible, so new formats can be added fairly easily. All the table input/output is handled by STIL, the Starlink Tables Infrastructure Library; more detailed descriptions of the I/O capabilities can be found in its documentation (http://www.starlink.ac.uk/stil/).

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 Supported Input Formats

Loading tables into TOPCAT is done either from the command line when you start the program up or using the Load Table dialogue. For FITS and VOTable formats the file format can be detected automatically (note this is done by looking at the file content, it has nothing to do with filename extensions). For other formats though, for instance ASCII or Comma-Separated Values, 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 9 for details. You can always specify the format rather than using automatic detection if you prefer - this can be a good idea if a table appears to be failing to load in a surprising way, since it may give you a more detailed error message.

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.

Note: in some earlier versions of TOPCAT, ASCII format tables could be detected automatically, so you could load them by typing something like "`topcat table.txt`". In the current version, you have to signal that this is an ASCII table, for instance by typing "`topcat -f ascii table.txt`".

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

4.1.1.1 FITS

FITS binary and ASCII table extensions can be read. Unless told otherwise, TOPCAT will display the first TABLE or BINTABLE extension in a given FITS file. If a later extension is required, this is indicated by giving the extension number after a '#' at the end of the table location. 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.

You can select which extension to use more conveniently than by specifying the HDU numbers if

you use the Hierarchy Browser to load the table.

If the table has been written using TOPCAT's "`fits-plus`" output format (see Section 4.1.2.1) then the metadata will be read in from the primary HDU as well.

For normal FITS files, header cards in the table's HDU header will be made available as table parameters (see Appendix A.3.2). 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.

If the table is stored in a FITS binary table extension in a file on local disk in uncompressed form, then the table is 'mapped' into memory - this generally means fast loading and low memory use, even in the absence of TOPCAT's `-disk` flag (Section 9.1).

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 header cards) is the same, namely, the number of rows in the table that is represented. The point of this is that all the cells of a given 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 most non-STIL software will probably not recognise them as 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.

Like normal FITS, two variants are supported; with (`colfits-plus`) and without (`colfits-basic`) metadata stored as a VOTable byte array in the primary HDU.

Colfits format is only available for data which stored as an uncompressed file in the file system (not, for instance, from a URL).

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. TOPCAT is believed to read any table which conforms to the VOTable 1.0 or VOTable 1.1 specification (http://www.ivoa.net/Documents/latest/VOT.html). This includes tables in which the cell data are included in-line as XML elements (VOTable/TABLEDATA format), or included/referenced as a FITS table (VOTable/FITS) or included/referenced as a raw binary stream (VOTable/BINARY). TOPCAT 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.

VOTable documents can have a complicated hierarchical structure, and may contain more than one actual table. Unless told otherwise, TOPCAT will load the first table it finds in the document, so in the (common) case that the document holds exactly one table, giving the filename will load that sole table. To display a table other than the first, you must indicate the zero-based index of the TABLE element in a breadth-first search after a '#' character at the end of the table specification. 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 open the Star Catalogue using the name "cats.xml" or "cats.xml#0", and the Galaxy Catalogue using the name "cats.xml#1".

4.1.1.4 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 TOPCAT 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`
- 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`.

If you understand the format of your files but they don't exactly match the criteria above, the best thing is probably to write a simple free-standing program or script which will convert them into the format described here. You may find Perl or awk suitable languages for this sort of thing.

This format is not detected automatically - you must specify that you wish to load a table in `ascii` format.

4.1.1.5 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. They typically have a filename extension "`.tbl`" and are used for Spitzer data amongst other things. An example looks like this:

```
\title='Animals'
\ This is a table with some animals in it.
|   RECNO |    SPECIES |          NAME |    LEGS | HEIGHT   |
|    char |       char |          char |     int | double   |
|         |            |               |         | m        |
|         |            |          null |         |          |
         1          pig   Pigling Bland         4       0.8
         2          cow           Daisy         4         2
         3     goldfish          Dobbin         0      0.05
         4          ant            null         6     0.001
```

4.1.1.6 Comma-Separated Values

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 that TOPCAT can 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.

This format is not detected automatically - you must specify that you wish to load a table in `csv` format.

4.1.1.7 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. 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]
```

4.1.1.8 SQL Database Queries

With appropriate configuration, TOPCAT can be used to examine the results of queries on an SQL-compatible relational database.

Database queries can be specified as a string in the form:

```
jdbc:driver-specific-url#sql-query
```

The exact form is dependent on the driver. Here is an example for MySQL:

```
jdbc:mysql://localhost/astro1?user=mbt#SELECT ra, dec FROM swaa WHERE vmag<18
```

which would get a two-column table (the columns being "ra" and "dec"), constructed from certain rows from the table "swaa" in the database "astro1" on the local host, using the access privileges of user mbt.

Fortunately you don't have to construct this by hand, there is an SQL Query Dialogue (Appendix A.5.3) to assist in putting it together.

Note that TOPCAT does not view a table in the database directly, but the result of an SQL query on that table. If you want to view the whole table you can use the query

```
SELECT * FROM table-name
```

but be aware that such a query might be expensive on a large table.

Use of SQL queries requires some additional configuration of TOPCAT; see Section 9.3.

4.1.1.9 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 seperated.)
------------------------
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 WDC tables is experimental - it may not be very robust.

This format is not detected automatically - you must specify that you wish to load a table in `csv` format.

4.1.2 Supported 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 - these are described as well.

The program has no "native" file format, but if you have no particular preference about which format to save tables to, FITS 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 (see Section 4.1.2.1). 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 column-oriented FITS ("`colfits`") format can be more efficient for some applications, though this is unlikely to be understood except by STIL-based code (TOPCAT and STILTS). If you want to write to a format which retains all metadata in a portable format, then one of the Section 4.1.2.3 formats might be better.

4.1.2.1 FITS

When saving in FITS format a new file is written consisting of two HDUs (Header+Data Units): a primary one (required by the FITS standard), and a single extension of type BINTABLE containing the table data.

There are two variants of this format:

fits-basic

The primary HDU contains only very minimal headers and no data.

fits-plus
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 mark 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 TOPCAT (or other STIL-based applications) read it however, they 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. Other, more standard schemes exist for combining the benefits of FITS and VOTable, but suffer from some disadvantages: `votable-fits-inline` is hard to process efficiently (in particular the data cannot easily be mapped into memory) and `votable-fits-href` requires that you keep your data in two separate files, which can get separated from each other. If you want to ensure that the metadata are available to other VOTable-aware programs, you should use one of the normal VOTable formats (Section 4.1.2.3).

In general, you can just let TOPCAT detect the format automatically and not worry about which of these variants is being used - if `fits-plus` is being used you just get some hidden benefits.

4.1.2.2 Column-oriented FITS

When saving in column-oriented FITS format a new file is written consisting of two HDUs (Header+Data Units); a primary one (required by the FITS standard) and a single extension of type BINTABLE containing the table data. Unlike normal FITS format however, this table consists of a single row in which each cell holds the data for an entire column. This can be a more efficient format to work with when dealing with very large, and especially very wide, tables. The benefits are greatest when the file size exceeds the amount of available physical memory and operations are required which scan through the table using only a few of the columns (much of TOPCAT's operations, for instance plotting two columns against each other, fit into this category). The overhead for reading and writing this format is somewhat higher than for normal FITS however, and other applications may not be able to work with it (though it is a legal FITS file), so in most cases normal FITS is a more suitable choice.

Like normal (row-oriented) FITS (see Section 4.1.2.1), there are two variants:

colfits-plus
The primary HDU contains an array of bytes which stores the table metadata in VOTable format.

colfits-basic
The primary HDU contains no data.

4.1.2.3 VOTable

When a table is saved to VOTable format, a document conforming to the VOTable 1.0 specification containing a single TABLE element within a single RESOURCE element is written.

There are a number of variants which determine the form in which the table data (DATA element) is written:

votable-tabledata
TABLEDATA element (pure XML)

votable-binary-inline
BINARY element containing base64-encoded data within the document

votable-fits-href
FITS element containing a reference to an external newly-written FITS file (with a name derived from that of the VOTable document)

votable-binary-href
BINARY element containing a reference to an external newly-written binary file (with a name derived from that of the VOTable document)

votable-fits-inline
FITS element containing base64-encoded data within the document

See the VOTable specification (http://www.ivoa.net/Documents/latest/VOT.html) for more explanation of what these variants mean. They can all be read by the VOTable input handler.

4.1.2.4 ASCII

Tables can be written using a format which is compatible with the ASCII input format. It writes as plainly as possible, so should stand a good chance of being comprehensible to other programs which require some sort of plain text rendition of a table.

The first line is a comment (starting with a "#" character) which names the columns, and an attempt is made to line up data in columns using spaces. Here is an example of a short table written in this format:

```
# index Species  Name   Legs Height Mammal
  1     pig      Bland  4    0.8    true
  2     cow      Daisy  4    2.0    true
  3     goldfish Dobbin 0    0.05   false
  4     ant      ""     6    0.0010 false
  5     ant      ""     6    0.0010 false
  6     human    Mark   2    1.9    true
```

4.1.2.5 Text

Tables can be written to a simple text-based format which is designed to be read by humans. No reader exists for this format.

Here is an example of a short table written in this format:

```
+-------+----------+--------+------+--------+--------+
| index | Species  | Name   | Legs | Height | Mammal |
+-------+----------+--------+------+--------+--------+
| 1     | pig      | Bland  | 4    | 0.8    | true   |
| 2     | cow      | Daisy  | 4    | 2.0    | true   |
| 3     | goldfish | Dobbin | 0    | 0.05   | false  |
| 4     | ant      |        | 6    | 0.0010 | false  |
| 5     | ant      |        | 6    | 0.0010 | false  |
| 6     | human    | Mark   | 2    | 1.9    | true   |
+-------+----------+--------+------+--------+--------+
```

4.1.2.6 Comma-Separated Values

Tables can be written to the semi-standard comma-separated value (CSV) format, described in more detail in Section 4.1.1.6. This can be useful for importing into certain external applications, such as some spreadsheets or databases.

There are two variants:

CSV
The first line is a header which contains the column names.

CSV-noheader

No header line is emitted, all lines represent data rows.

4.1.2.7 Tab-Separated Table

Tables can be written to TST format, which is described in more detail in Section 4.1.1.7. This can be useful for communicating with some other astronomical tools such as GAIA.

4.1.2.8 SQL Tables

With appropriate configuration, TOPCAT can write out tables as new tables in an SQL-compatible relational database.

For writing, the location is specified as the following URL:

```
jdbc:driver-specific-url#new-table-name
```

The exact form is dependent on the driver. Here is an example for MySQL:

```
jdbc:mysql://localhost/astro1?user=mbt#newtab
```

which would write the current contents of the browser into a new table named "newtab" in the database "astro1" on the local host with the access privileges of user mbt.

Fortunately you do not have to construct this URL by hand, there is an SQL dialogue box to assist in putting it together.

Use of SQL queries requires some additional configuration of TOPCAT; see Section 9.3.

4.1.2.9 HTML

A table can be written out as an HTML 3.2 TABLE element, suitable for use as a web page or insertion into one.

There are two variants:

HTML
: A freestanding HTML document, complete with HTML, HEAD and BODY tags is output.

HTML-element
: Only the TABLE element representing the table is output; this should normally be embedded in a larger HTML document before use.

4.1.2.10 LaTeX

A table can be written out as a LaTeX `tabular` environment, suitable for insertion into a document intended for publication.

There are two variants:

LaTeX
: The `tabular` element alone is output; this will have to be embedded in a larger LaTeX document before use.

LaTeX-document
: A freestanding LaTeX document, consisting of the `tabular` within a `table` within a `document` is output.

Obviously, this isn't so suitable for very large tables.

4.1.2.11 Mirage Format

Mirage (http://www.bell-labs.com/project/mirage/index.html) is a powerful standalone java tool developed at Bell Labs for analysis of multidimensional data. It uses its own file format for input. TOPCAT can write tables in the input format which Mirage uses, so that you can prepare tables in TOPCAT and write them out for subsequent use by Mirage.

It is also possible in principle to launch Mirage directly from within TOPCAT, using the **Export To Mirage** item on the Control Window's **File** menu; this will cause Mirage to start up viewing the currently selected Apparent Table. In order for this to work the Mirage classes must be on your classpath (see Section 9.2.1) when TOPCAT is run.

There appears to be a bug in Mirage which means this does not always work - sometimes Mirage starts up with no data loaded into it. In this case you will have to save the data to disk in Mirage format, start up Mirage separately, and load the data in using the **New Dataset** item in Mirage's **Console** menu.

Note that when Mirage has been launched from TOPCAT, exiting Mirage or closing its window will exit TOPCAT as well.

4.1.3 Custom I/O Formats

It is in principle possible to configure TOPCAT to work with table file formats 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 and/or output handler for the table format in question.

The steps that you need to take are:

1. Write java classes which constitute your input and/or output handler
2. Ensure that these classes are available on your classpath while TOPCAT is running (see Section 9.2.1)
3. Set the `startable.readers` and/or `startable.writers` system property to the name of the handler classes (see Section 9.2.3)

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 Table Locations

In many cases loading and saving tables will be done using GUI dialogues such as the filestore load and save windows, where you just need to click on a filename or directory to indicate the load/save location. However in some cases, for instance specifying tables on the command line (Section 9.1) or typing pathnames directly into the load/save dialogue windows, you may want give the location of a table for input or output using only a single string.

Most of the time you will just want to type in a filename; either an absolute or relative pathname 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 HTTP resources.

`ftp:`
: Read from anonymous FTP resources.

file:
Read from local files; not particularly useful since you can do much the same using just the filename.

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

myspace:
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:
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:
Used for communicating with SQL-compliant relational databases. These are a bit different to normal URLs - see Section 4.1.1.8 and Section 4.1.2.8.

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.

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 **Concatenate Tables** ( ) button 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.

5.4 Notes on Matching

This section provides a bit more detail on the how the row matching 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 consitute 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.

This means that the matching algorithm is basically 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.

6 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. To define a custom Activation Action (Section 7) in the Activation dialogue.
4. When faced with a column selector for plotting or other purposes, in some cases you can type in an expression rather than selecting or typing a simple column name.

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 intutitive.

The following explanation gives some guidance and examples for writing these expressions. Unfortunately a complete tutorial on writing Java 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 6.9.

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.

6.1 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 6.3); 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 6.3); columns take preference so if a column and a parameter both match the requested Utype, the column value will be used.

There is a special column whose name is "Index" and whose $ID is "$0". The value of this is the same as the row number in the unsorted table (the grey numbers on the left of the grid in the Data Window), so for the first column in the unsorted table it's 1, for the second it's 2, and so on.

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.

6.2 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.

6.3 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 6.1 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 6.1 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.

6.4 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.

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 6.8.

6.5 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)

Numeric Typecasts

`(byte)` (numeric -> signed byte)
`(short)` (numeric -> 2-byte integer)
`(int)` (numeric -> 4-byte integer)
`(long)` (numeric -> 8-byte integer)
`(float)` (numeric -> 4-type 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)
`?:` (conditional switch)
`instanceof` (class membership)

6.6 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:

Coords
Functions for angle transformations and manipulations. In particular, methods for translating between radians and HH:MM:SS.S or DDD:MM:SS.S type sexagesimal representations are provided.

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

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 (http://www.astro.utoronto.ca/~patton/astro/mags.html (http://www.astro.utoronto.ca/~patton/astro/mags.html), citing Frei and Gunn 1995).
- Constants `VEGA_AB_*`, for Vega <-> AB magnitude conversions (Blanton et al., Astronomical Journal *127*, 2562-2578 (2005), eqs.(5)).

Arrays
Functions which operate on array-valued cells. You can only use these functions on values which are already 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.

Tilings
Pixel tiling functions for the celestial sphere.

Strings
String manipulation and query functions.

Formats
Functions for formatting numeric values.

Maths
Standard mathematical and trigonometric functions.

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.

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.

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).

Conversions
Functions for converting between strings and numeric values.

A listing of the functions in these classes is given in Appendix B.1, and complete documentation on them is available within TOPCAT from the Available Functions Window.

6.6.1 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://kinetic.ac.donetsk.ua/JEL/).

6.7 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.

6.8 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 radians

```
hmsToRadians(RA1950)
dmsToRadians(decDeg,decMin,decSec)
```

Conversion from radians to sexagesimal

```
radiansToDms($3)
radiansToHms(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]
```

and 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
skyDistance(ra0,dec0,degreesToRadians(RA),degreesToRadians(DEC))<15*ARC_MINUTE
```

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")
```

Combining subsets

```
(_1 && _2) && ! _3
```

Test for non-blank value

```
! NULL_ellipticity
```

6.9 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 9.2.1
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 9.2.3 *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 Window 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 .`"

7 Activation Actions

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 plot it corresponds to one plotted point.

If you click on plotted point in one of the graphics windows, or on a row in the Data Window, the corresponding table row will be *activated*. When a row is activated, three things happen:

1. If that row is represented by a point in any open 2- or 3-diemensional 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. 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.

The third one can be more complicated. By default, no activation action is set, so nothing else happens, and this may very well be what you want. However, by clicking on the **Activation Action** selector in the Control Window you can bring up the Activation Window which enables you to choose an additional action to take place. There are various options here and various ways to achieve them (see Appendix A.7.4 for more details) but the kinds of actions which are envisaged are to display one or more images or spectra relating to the row you have identified. One of the options available for instance retrieves a postage-stamp image of a few arcminutes around the sky position defined by the row from a SuperCOSMOS all-sky image survey and pops it up in a viewer window. So for instance having spotted an interesting point in a plot of a galaxy catalogue you can click on it, and immediately see a picture to identify its morphological type.

The exact actions you want to perform may be closely tailored to the data you have, for instance you may have a set of spectra on disk named by object ID. It's impossible to cater for such possibilities with a set of pre-packaged options, so you are able to define your own custom actions here. This is done by writing a expression using the syntax described in Section 6. A number of special functions (described in the following subsection) are provided to do things like display an image or a spectrum in a browser (given its filename or URL), or access data from certain data servers on the web, but there is nothing to stop the adventurous plugging in their own external programs so in principle you can configure pretty much anything to happen on the basis of the values in the row that you have activated.

7.1 Activation Functions

When defining custom activation actions in the Activation Window, you enter an expression to be invoked on a row when it is activated. This expression uses the syntax defined in Section 6 and can make use of the functions listed in Appendix B.1. However in this case there is an additional list of functions you can use which cause something to happen (for instance displaying an image) rather than just returning a value. The following classes of functions are available:

Browsers
: Displays URLs in web browsers.

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.

Splat

Functions for display of spectra in the external viewer SPLAT.

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.

Mgc
Specialist functions for use with data from the the Millennium Galaxy Survey.

Output
Functions for simple logging output.

System
Executes commands on the local operating system. These are executed as if typed in from the shell, or command line.

Image
Functions for display of images in a window. Supported image formats include GIF, JPEG, PNG and FITS, which may be compressed. The SoG program (http://www.starlink.ac.uk/sog/ (http://www.starlink.ac.uk/sog/)) will be used if it is available, otherwise a no-frills image viewer will be used instead.

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/).

Sdss
Specialist display functions for use with the Sloane Digital Sky Server.

Sog
Functions for display of images in external viewer SOG (http://www.starlink.ac.uk/sog/ (http://www.starlink.ac.uk/sog/)).

Spectrum
Functions for general display of spectra in a window. Display is currently done using the SPLAT program, if available (http://www.starlink.ac.uk/splat/ (http://www.starlink.ac.uk/splat/)). Recognised spectrum formats include 1-dimensional FITS arrays and NDF files.

A listing of the functions in these classes is given in Appendix B.2, and complete documentation on them is available within TOPCAT from the Available Functions Window.

8 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 (Aladin, GAIA, [DS9 expected soon]) table analysis tools (VisIVO, STILTS, other instances of TOPCAT itself), spectrum analysis tools (SPLAT, VOSpec) scripting languages (SAMPy, Perl SAMP) 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 so PLASTIC is somewhat deprecated in favour of SAMP. At time of writing (December 2008) more tools speak PLASTIC than SAMP, but it is hoped that this will change over the next few months, and that SAMP will come to replace PLASTIC across the astronomical toolbase. It is not necessary for users to understand the differences between the two, but in general a tool using SAMP can communicate with other tools using SAMP, and the same with PLASTIC, but in most cases interoperation between PLASTIC and SAMP tools will not work.

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. A hub can be started from within TOPCAT if required - details of how to do this are given in the SAMP and PLASTIC sections below. Other tools will have their own policies for connecting to the hub, but in general it is a good idea to start a hub first before starting up the tools which you want to talk to it.

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 if a running SAMP or PLASTIC hub can be found, the corresponding mode is used, otherwise it defaults to 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). Facilities for communications monitoring are considerably better in SAMP than in PLASTIC mode.

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.

8.1 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.2.3.

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.

8.2 PLASTIC control

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.

8.3 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.

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 ControlWindow has a **Broadcast Row** checkbox by the **Activation Action** selector. If this is selected, then any time you activate a row (e.g. by clicking on the corresponding point in a plot) other applications who are looking at the same table are invited to highlight that row

in their copy of the table. As for **Transmit Subset** above, this will only have an effect if the other application(s) are displaying the same table.

A similar facility is available as the **Transmit Row** option in the Activation Actions (Section 7) window itself - the checkbox is just a convenience for a commonly-used option.

SAMP MType: `table.highlight.row`; PLASTIC Message ID: `ivo://votech.org/votable/highlightObject`

Transmit Coordinates

Another Activation Action (Section 7) is **Transmit Coordinates**. In this case when you activate a row (e.g. by clicking on the corresponding point in a plot) other applications are invited to point out the sky position corresponding to the row which is activated, for instance by placing a cursor over it. The table columns which correspond to Right Ascension and Declination must be selected for this to work.

SAMP MType: `coord.pointAt.sky`; PLASTIC Message ID: `ivo://votech.org/sky/pointAtCoords`

Transmit Image

The Density Plot (Appendix A.4.7) produces a 2-d histogram which is actually an image. Its **Interop** menu provides **Broadcast Image** and **Send Image** options which will send this (as a FITS image) to other applications which can display it in some way. This is a useful supplement to the facilities of the Density Plot window, since it doesn't have very sophisticated image display features (variable colour maps, contour plots etc).

SAMP MType: `image.load.fits`; PLASTIC Message ID: `ivo://votech.org/fits/image/loadFromURL`

8.4 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` or `table.load.fits`; PLASTIC Message ID: `ivo://votech.org/votable/load` or `ivo://votech.org/votable/loadFromURL`

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 7) 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`

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`

9 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 9.5. 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-lite.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-full.jar or topcat-lite.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-lite.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 9.2.1.

Note that Java Web Start can also be used to invoke the program without requiring any prior download/installation - sorry, this isn't documented properly here yet.

The meaning of the optional `[topcat-args]` and `[java-args]` sequences are described in Section 9.1 and Section 9.2 below respectively.

9.1 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
      -demo          start with demo data
      -disk          use disk backing store for large tables
      -samp          use SAMP for tool interoperability
      -plastic       use PLASTIC for tool interoperability
      -hub           run internal SAMP/PLASTIC hub
      -exthub        run external SAMP/PLASTIC hub
      -soap          start SOAP services
      -noserv        don't run any services (PLASTIC, SAMP or SOAP)
      -stilts <args> run STILTS not TOPCAT

  Optional load dialogue flags:
      -tree      hierarchy browser
      -file      basic file browser
      -sql       SQL query on relational database
      -cone      cone search dialogue
      -gavo      GAVO Millennium run database query
      -registry  VO registry query
      -sia       Simple Image Access Protocol queries
      -ssa       Simple Spectrum Access Protocol queries

  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

  All known formats:
      fits-plus, colfits-plus, colfits-basic, fits, votable, ascii, csv, tst, ipac, wdc

  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       default storage policy
      mark.workaround         work around mark/reset bug
      myspace.cache           MySpace performance workaround
```

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.7.12) for more information about this.

-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.

-disk
If the `-disk` flag is given then the program will use disk backing storage for caching table data that is read in, rather than keeping it in memory. This means that tables much larger than the heap memory assigned to Java can be used. It may lead to slower processing, but usually the performance is not greatly reduced. If you find TOPCAT running out of memory (you see `OutOfMemoryErrors` popping up in windows or on the console) then re-running with the `-disk` flag is a good idea. 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. Note however that uncompressed FITS binary tables on disk are not read into memory in any case (they are *mapped*) so the `-disk` flag may not make much difference with FITS.

-samp
Forces TOPCAT to use SAMP for tool interoperability (see Section 8). This is the default, but PLASTIC mode instead of SAMP will otherwise be chosen if a PLASTIC hub but no SAMP hub is running on startup.

-plastic
Forces TOPCAT to use PLASTIC instead of SAMP for tool interoperability (see Section 8).

-hub
Causes TOPCAT to run an internal SAMP or PLASTIC hub (SAMP by default), if no hub is already running. The hub will terminate when TOPCAT does. See Section 8.

-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 8.

-soap
Controls whether TOPCAT will start up a SOAP server which can accept requests using the SOAP protocol to display tables from other applications. Currently, the default is that it does not start a server. This option may be withdrawn in future releases.

-noserv
Prevents TOPCAT from running any services. Currently the services which it may run are SAMP, PLASTIC and SOAP (see above).

-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>`.

Some of the flags control what load dialogues are visible in the Load Window. In fact all of these load dialogues can be accessed from the Load Window's **DataSources** menu as long as the classes are available, but if you specify these flags on the command line, the corresponding button will appear in the main part of the window, making the option more obvious. The load dialogue flags

are:

-tree
Hierarchy browser (Appendix A.5.2), used for content-sensitive browsing of the filespace (not available with topcat-lite).

-file
Basic file browser. This doesn't do much that the Filestore Browser (which is present by default, and can also access remote filespaces) can't do, but is provided as a fallback.

-sql
SQL query (Appendix A.5.3), used for obtaining tables from an SQL query on a rleational database (only available if JDBC is set up correctly).

-cone
Cone search dialogue (Appendix A.5.4), used for obtaining catalogues of sources in a region of the sky from remote data servers (not available with topcat-lite).

-gavo
GAVO database query dialogue (Appendix A.5.6). Allows queries to the GAVO (German Astrophysical Virtual Observatory) database services. At time of writing, the most generally useful of these are queries to the Virgo Millennium run database.

-sia
Simple Image Access protocol dialogue (Appendix A.5.5), used for obtaining the results of SIA queries on remote data servers (not available with topcat-lite).

-ssa
Simple Spectral Access protocol dialogue (Appendix A.5.5), used for obtaining the results of SSA queries on remote data servers (not available with topcat-lite).

-registry
Registry query dialogue *(experimental)*, used for obtaining the results of a query on a VO resource registry as a table (not available with topcat-lite).

These flags in most circumstances do just the same as adding the relevant dialogue class name to the `startable.load.dialogs` system property (see Section 9.2.3).

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 SQL queries, or perhaps something else. 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.

9.2 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.

9.2.1 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 9. However, for certain things Java might need to find some other classes, in particular for:

- JDBC drivers (see Section 9.3)
- Providing extended algebraic functions (see Section 6.9)
- Launching Mirage (see Section 4.1.2.11)
- Installing I/O handlers for new table formats (see Section 4.1.3)

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-lite.jar:... uk.ac.starlink.topcat.Driver` instead).

9.2.2 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 typically 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`".

Note however: using the `-disk` flag described in Section 9.1 may be a better solution; this makes the program store data from large tables on disk rather than in memory.

9.2.3 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 when using Java 1.5 or later 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`. There are bugs with this feature in Apple's Java 1.4 JRE, so it's set false by default in that case, but you can try setting it true at your own risk if you like.

`java.io.tmpdir`
The directory in which TOPCAT will write any temporary files it needs. This is usually only done if the `-disk` flag has been specified (see Section 9.1).

`jdbc.drivers`
Can be set to a (colon-separated) list of JDBC driver classes using which SQL databases can be accessed (see Section 9.3).

`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 6.9).

`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 6.9).

`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".

`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 4.1.3).

`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 9.1). Other possible values are "`memory`" (the default), "`sideways`" and "`discard`".

`startable.writers`
Can be set to a (colon-separated) list of custom table format output handler classes (see Section 4.1.3).

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

`votable.strict`
Set `true` for strict enforcement of the VOTable standard when parsing VOTables. This prevents the parser from working round certain common errors, such as missing `arraysize` attributes on FIELD/PARAM elements with `datatype="char"`. False by default.

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.

9.3 JDBC Configuration

This section describes additional configuration which must be done to allow TOPCAT to access SQL-compatible relational databases for reading (see Section 4.1.1.8) or writing (see Section 4.1.2.8) 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), described in more detail on Sun's JDBC web page.

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. Set the `jdbc.drivers` system property to the name of the driver class as described in Section 9.2.3
2. Ensure that the classpath you are using includes this driver class as described in Section 9.2.1

These steps are all standard for use of the JDBC system.

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.

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 *can't* currently 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 the Microsoft 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.

Other RDBMSs and drivers may or may not work - please let us know the results of any experiments you carry out. Sun maintain a list of JDBC drivers for various databases; it can be found at http://servlet.java.sun.com/products/jdbc/drivers.

Here are example command lines to start up TOPCAT using databases known to work.

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
```

9.4 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 with tables up to about a million rows at a reasonable speed; 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:

Use the -disk flag

By default, TOPCAT reads the data from tables in most file formats (everything except FITS binary tables) into memory when it loads them. By specifying the `-disk` flag on the command line, you can tell it to use temporary files for storing this data instead. This makes it much less likely to run out of memory. You can achieve the same effect by adding the line `startable.storage=disk` in the `.starjava.properties` in your home directory. This is the best thing to do if you're seeing an out of memory error when loading tables. See Section 9.1, Section 9.2.3.

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 maximum is typically 64Mb. You can increase this by using the `-Xmx` flag, followed by the maximum heap memory, for instance "`topcat -Xmx256M`" or "`java -Xmx256M -jar topcat-full.jar`". Don't forget the "`M`" to indicate megabytes. 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 9.2.2.

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, you may wish 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 `-disk` or `-Xmx` flags as above.

Use column-oriented storage

For really large tables storing them in the colfits 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.

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.

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.

9.5 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:

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.

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:

File menu

Nearly all windows have this menu. At least the following options are available:

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.

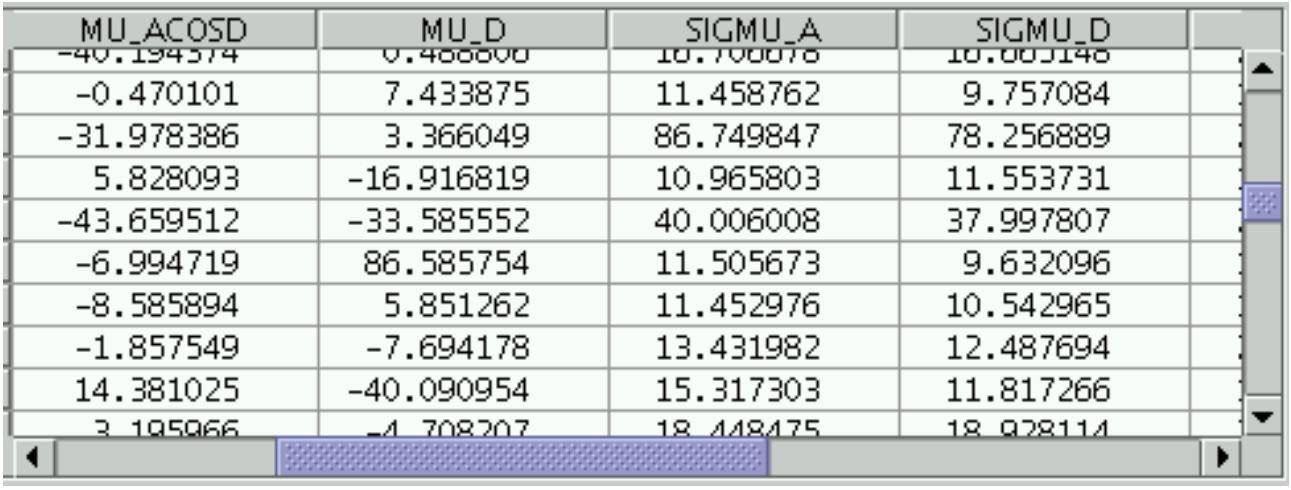
About TOPCAT

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	0.488800	10.700078	10.003148
-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 Java widget 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.

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

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.

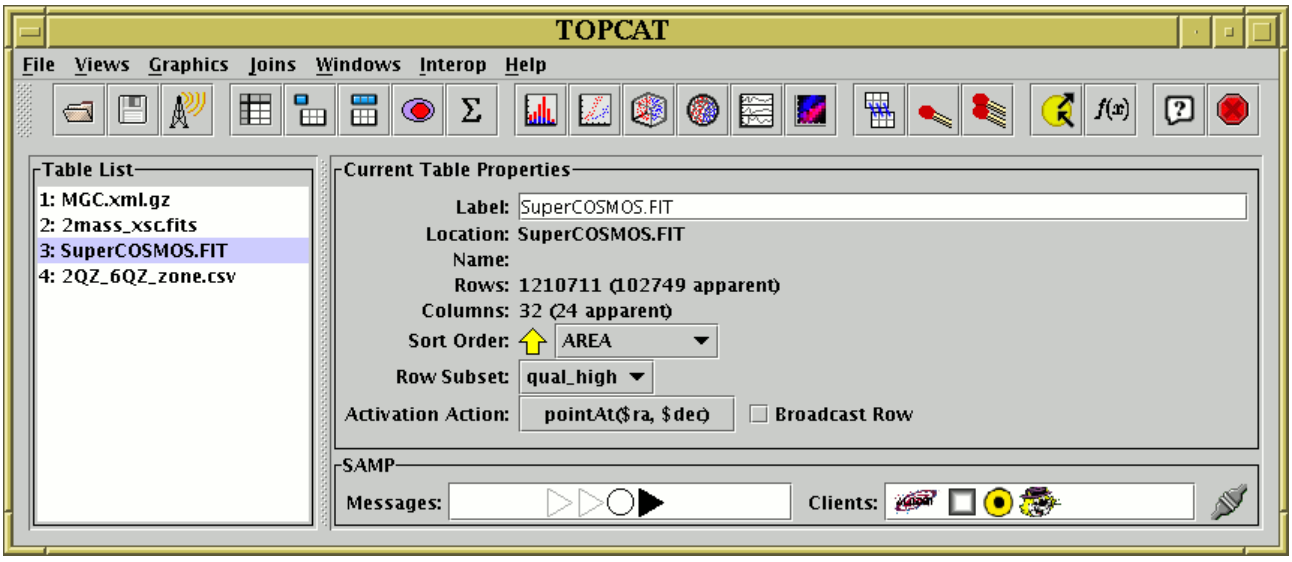
In the simplest case you can simply select the value from the list of columns by clicking on the down-arrow at the right and then selecting one from the drop-down list of columns which is offered. Note that only appropriate columns will be offered - for instance if a numeric value is required, string-valued 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 6. 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.2 Control Window



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, 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, 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.

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.

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 column (if any) which determines the current Row Order. A selector shows the column (if any) on which the rows of the Apparent Table are sorted and allows you to choose a different one. The little arrow beside it indicates whether the sort is ascending or descending.

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 Action
The currently selected Activation Action (Section 7). The action can be changed by clicking on this button to display the Activation Window.

The **Broadcast Row** checkbox to the right is a shortcut for one useful activation action; it corresponds to the Activation Window's option **Transmit Row** to All Listeners - that is it means that any other tools talking to TOPCAT using SAMP or PLASTIC will be informed about a row activation any time it happens. This checkbox will be greyed out if no suitable messaging protocol is in use (see Section 8).

A.2.3 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 8. **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.2.4 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

Pops up the Save Table dialogue which allows you to write out the current Apparent Table.

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.

Broadcast Table

Broadcasts the current **Apparent Table** to any applications registered using the SAMP or PLASTIC protocol. See Section 8.

Send Table

Sends the current **Apparent Table** to a single selected application registered using the SAMP or PLASTIC protocol. See Section 8.

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.

Graphics windows

Histogram

Displays a window for plotting 1-d histograms.

2d Scatter Plot

Displays a window for plotting 2-d scatter plots.

3D Scatter Plot

Displays a window for plotting 3-d scatter plots on Cartesian axes.

Spherical Scatter Plot

Displays a window for plotting 3-d scatter plots on spherical polar axes, with or without a radial coordinate.

Stacked Line Plot

Displays a window for plotting multiple stacked line plots against a single X coordinate

Density Map

Displays a window for plotting 2-d density maps (image-like histograms on a 2-d grid of bins).

Matching and joining (see Section 5)

Concatenation Window

Displays a dialog for joining tables top-to-bottom.

Internal Match Window

Displays a dialog for finding internal matches between the rows of a table.

Pair Match Window

Displays a dialog for joining tables side-by-side by locating rows which match between them.

Note that matches involving more than two tables can be done using the options in the **Joins** menu.

Miscellaneous

SAMP Window

Displays a window which displays detail about the current status, and allows

configuration, of SAMP messaging (see Section 8). Note this button will not be visible if TOPCAT is running in PLASTIC mode.

Available Functions

Displays a window containing all the functions which can be used for writing algebraic expressions (see Section 6).

Help

Displays the help browser, open at the entry on the Control Window.

Exit

Queries for confirmation, then exits the application.

A.2.5 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:

Discard Table

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.

Export To Mirage

Starts up the external Mirage application on the current apparent table. This action is only available if Mirage is on your classpath. See Section 4.1.2.11.

View Log

Pops up the Log Window to display logging messages generated by the application. Intended mainly for debugging.

The **Views** menu contains actions for launching the windows which give certain views of the table metadata. These are all provided as toolbar buttons as well.

The **Graphics** menu contains actions for launching the windows which give various plotting and visualisation options. These are all provided as toolbar buttons as well.

The **Interop** menu contains options concerned with interoperating with other tools using the SAMP or PLASTIC protocol. See Section 8 for more detail.

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 **Joins** menu, as well as containing the actions for table concatenation, internal matching and pair matching which are available from the toolbar, also gives you the option to join three or four tables at once by matching rows. The multi-table match windows work pretty much the same as the Pair Matching Window, but with more tables.

A.3 Table View Windows

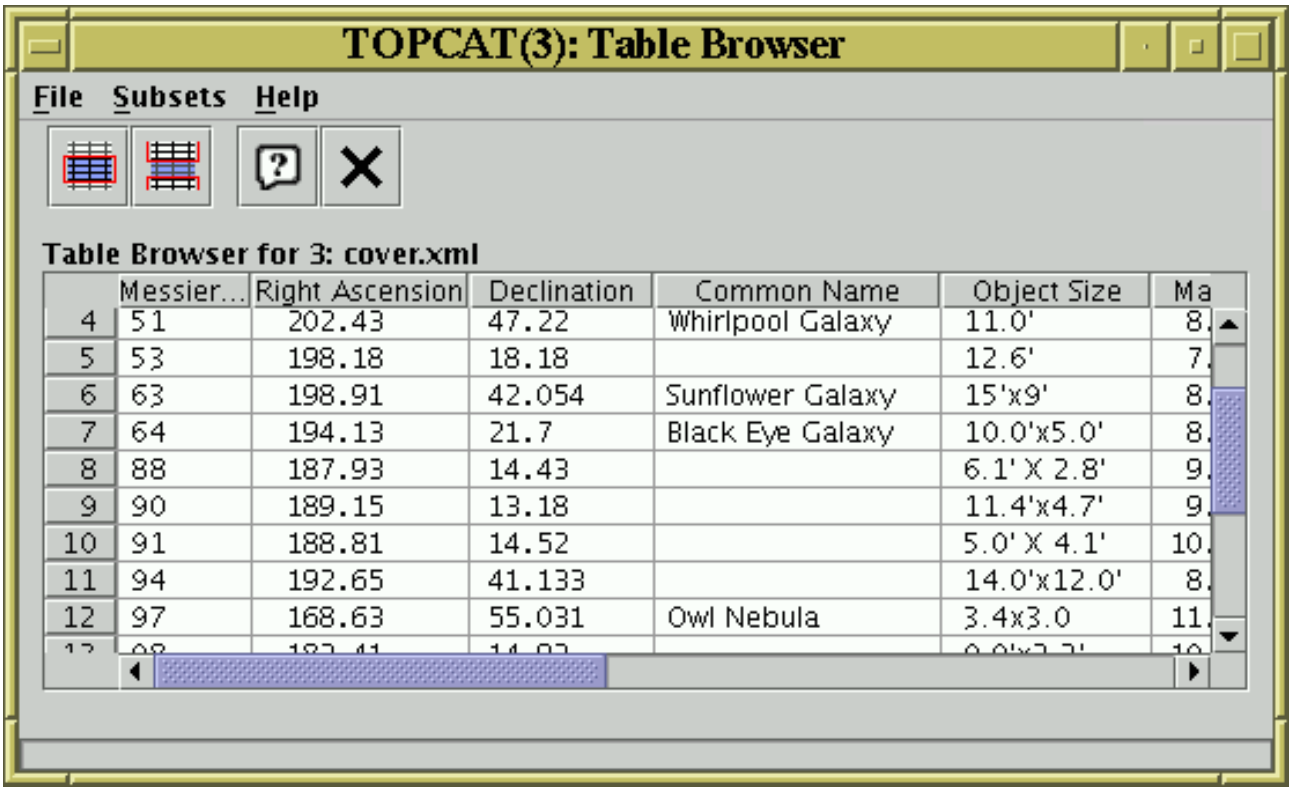
Many of the windows you will see within TOPCAT display information about a single table. There are several of these, each displaying a different aspect of the table data - cell contents, statistics, column metadata etc. There is one of each type for each of the tables currently loaded, though they won't 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.2.5), or the **File|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



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.

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 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 column you are after in the display there. This will have the effect of scrolling the Data Window sideways so that your selected column is visible in the centre of the display here.

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.

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.

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 Columns window.

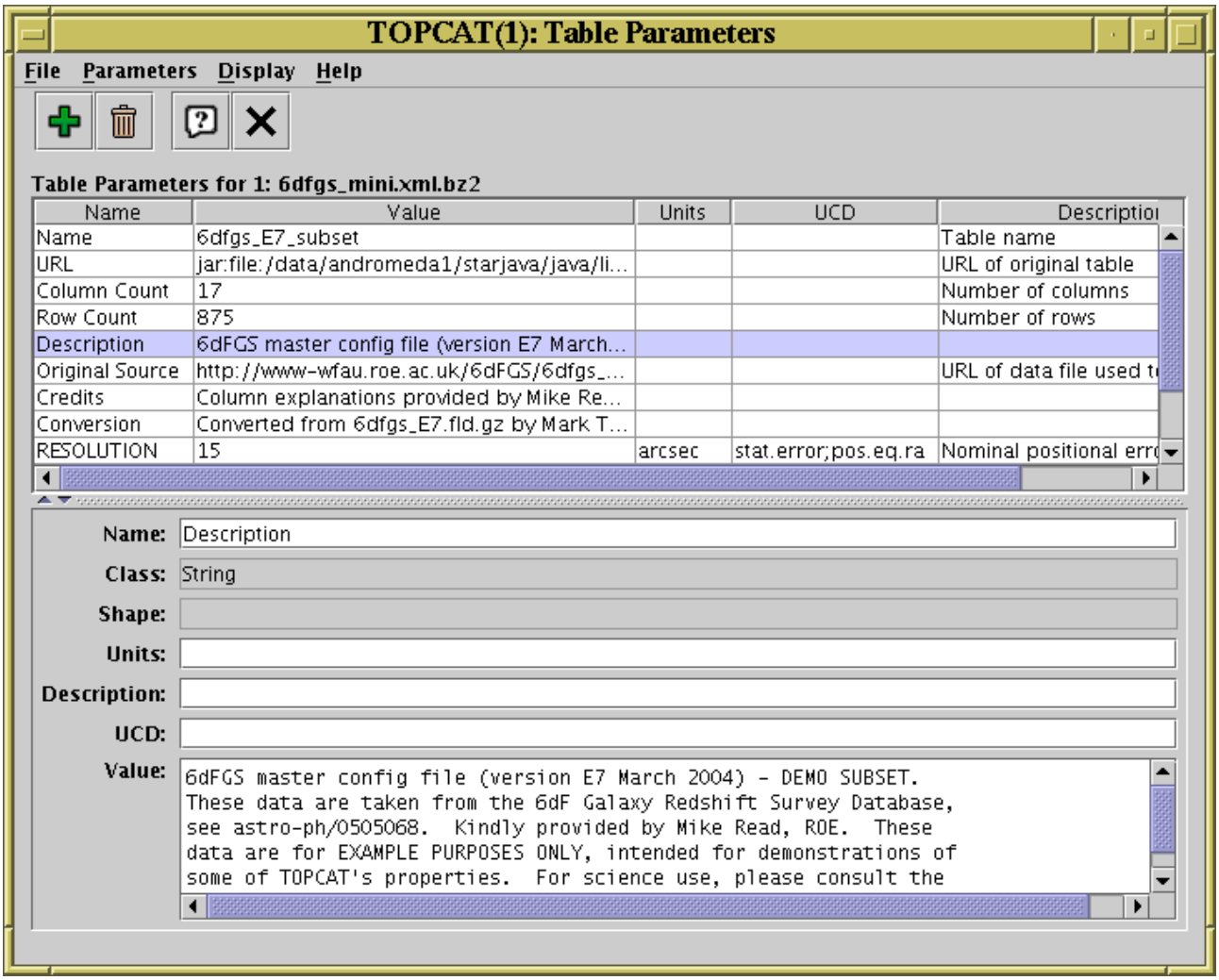
Search Column

For string-valued columns, this option allows you to search for values in a column. If you select it you will be asked to enter a regular expression, and then any row which matches that expression in this column will be selected (highlighted). If there's just one matching column it will be activated as well. The expression obeys normal regular expression syntax, so for instance you'd enter "`.*XYZ.*`" to find all rows which contain the string "XYZ".

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



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 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:

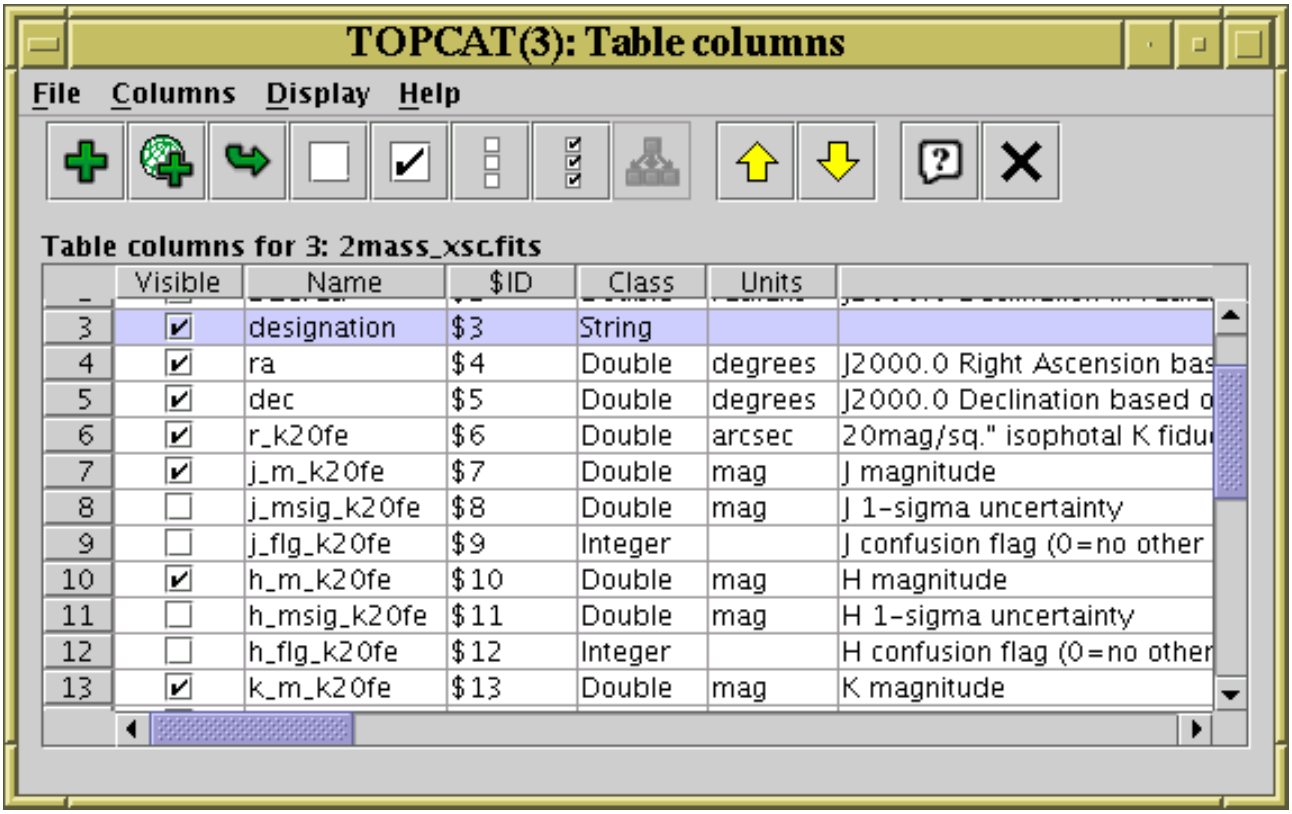
Add Parameter

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



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 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 leftmost column, 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.

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:

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 6) 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.

Units
The units in which quantities in this column are expressed.

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.

There may be other items in the list specific to the table in question.

You can edit column names and some other entries in this JTable by double-clicking on them as usual.

The order in which the rows are presented is determined by the table's current Column Set (Section 3.3), so can be changed by dragging the column headers around in the Data Window.

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.7.9) 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.7.8) 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.

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, 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.

Sort Selected Up

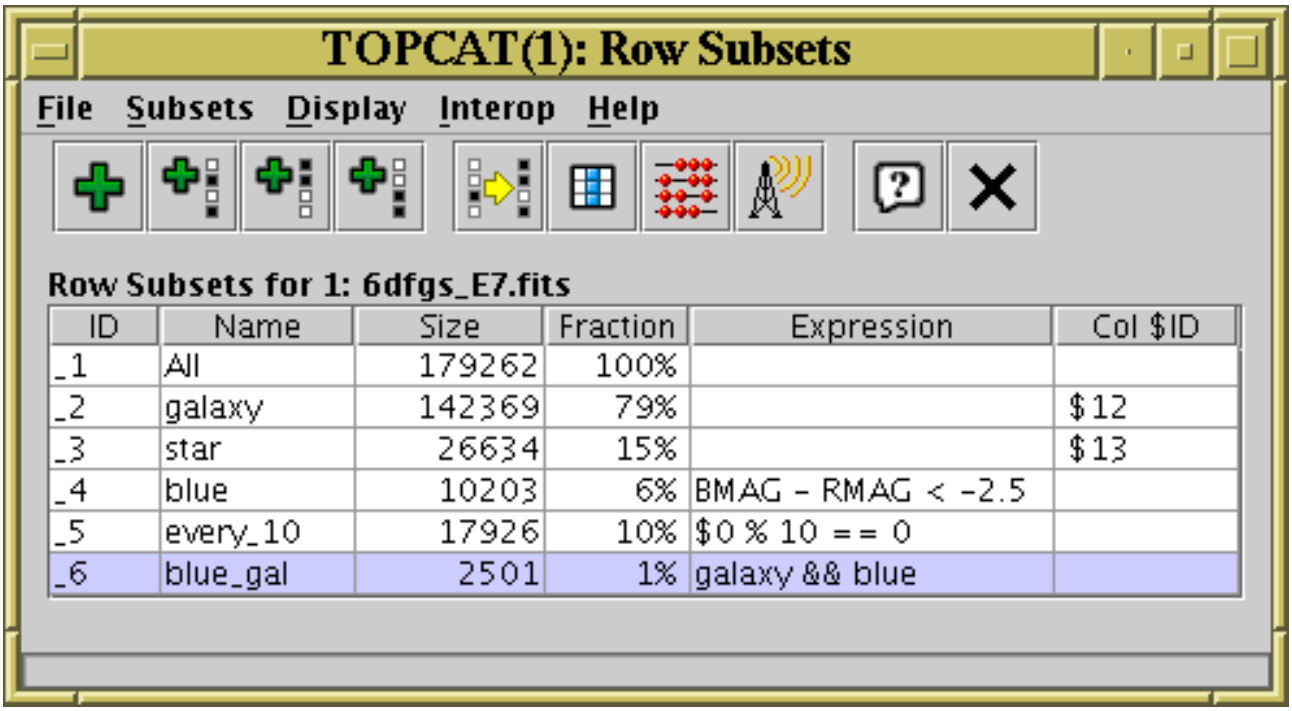
If a single column is selected then the table's current Sort Order will be set to sort ascending on that column. Otherwise this action is not available.

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. A side effect of selecting a single column is that the table view in the Data Window will be scrolled sideways so that the selected 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.

A.3.4 Subsets Window



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.

The subsets are displayed in a JTable widget with a row for each subset. 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 6) 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.

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.

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.

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.

Count Subsets
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 8 for more information about tool interoperability.

The following additional menu items are available:

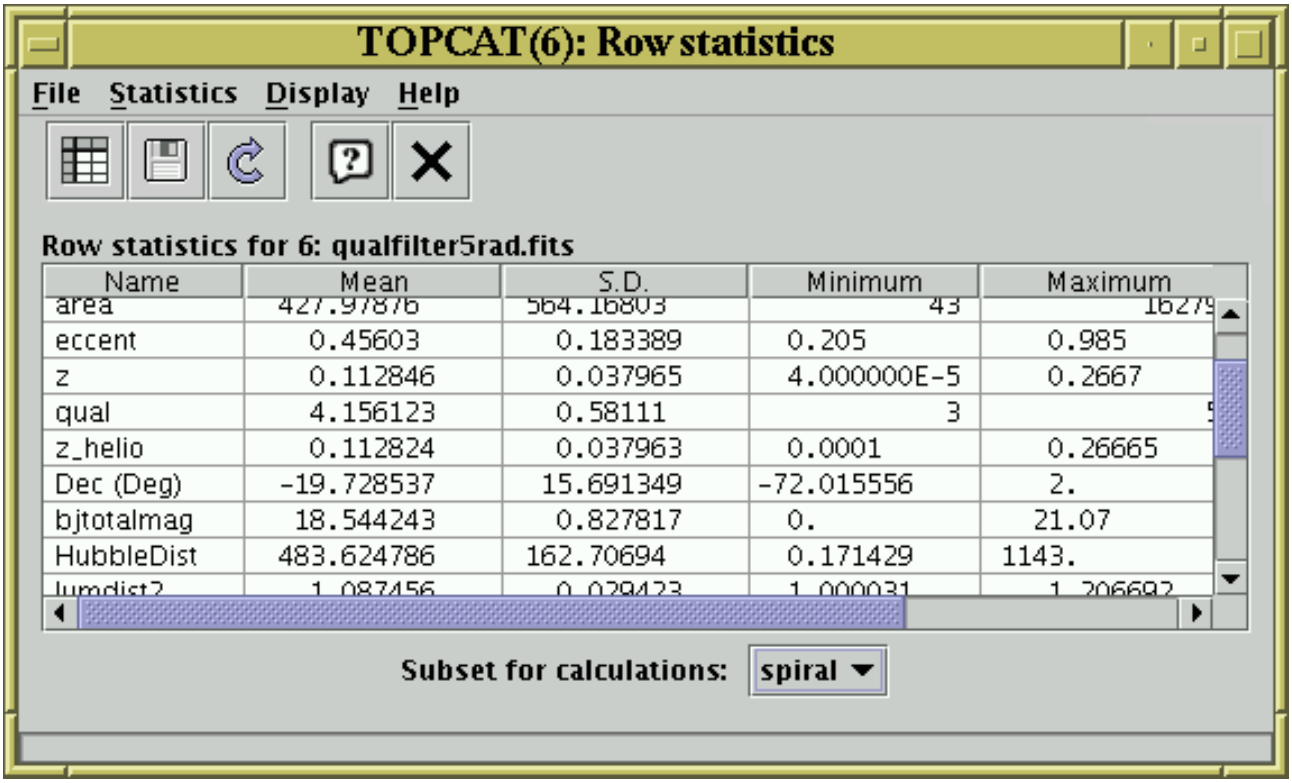
Send subset to ...

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



Statistics Window

The Statistics Window shows statistics for the values in each of the table's columns. You can

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. 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.

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.

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. If you attempt to calculate them for large tables, you may get a message saying that there is insufficient memory.

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.

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.4 Graphics Windows

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.

These visualisation windows are fairly sophisticated, and the plots can exported to vector (EPS) or image (GIF, JPEG, PNG) files for later presentation (see Appendix A.4.1.7). However, at least at present, TOPCAT does not claim to be a full end-to-end system for generating publication quality graphics, and hence lacks facilities for detailed configuration of axis labelling, font control, data annotation and so on. You may well find that you can use it to generate publication quality graphics, but if you need features which are not currently provided you may find it best to use TOPCAT to investigate your data and decide exactly what data you want to present, and then export the data in a form which can be used by a more output-oriented package.

A.4.1 Common Features

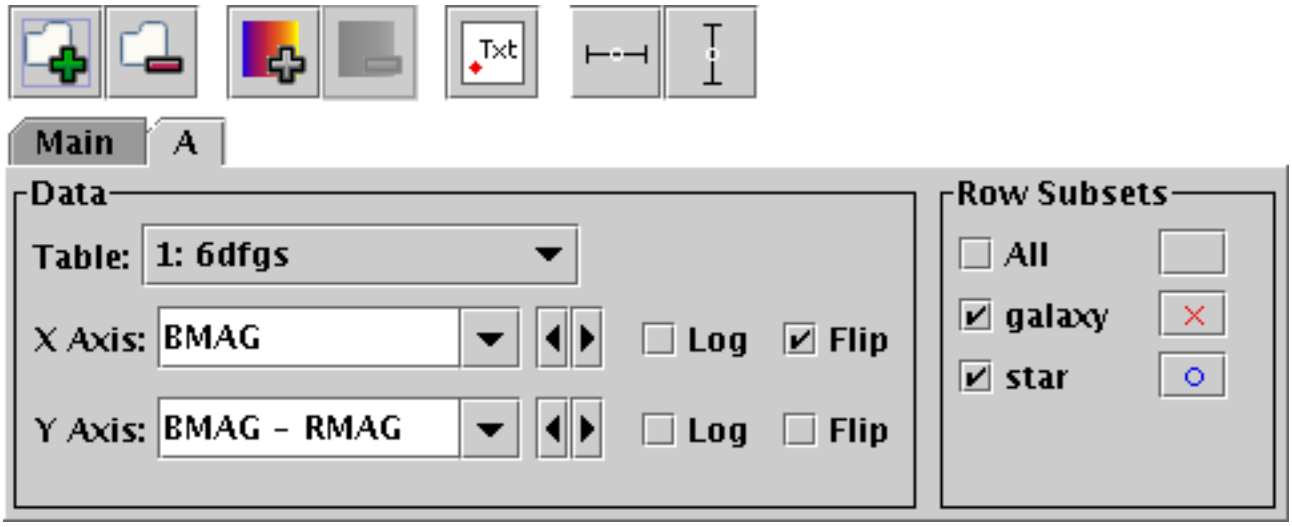
The various types of graphics windows have different characteristics to fulfil their different functions, but they share a common way of doing things. Each window contains a number of controls including toolbar buttons, menu items, column selectors and others. In general any change that you make to any 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 toolbar button in 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 graphics windows.

A.4.1.1 Dataset Selectors

All the 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:



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 6.

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.4.1.5), Point Labels (Appendix A.4.1.4) and Error Bars (Appendix A.4.1.3), 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.4.1.2 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 changes to indicate that zooming can be done. Drag down to zoom in and up 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 changes 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.

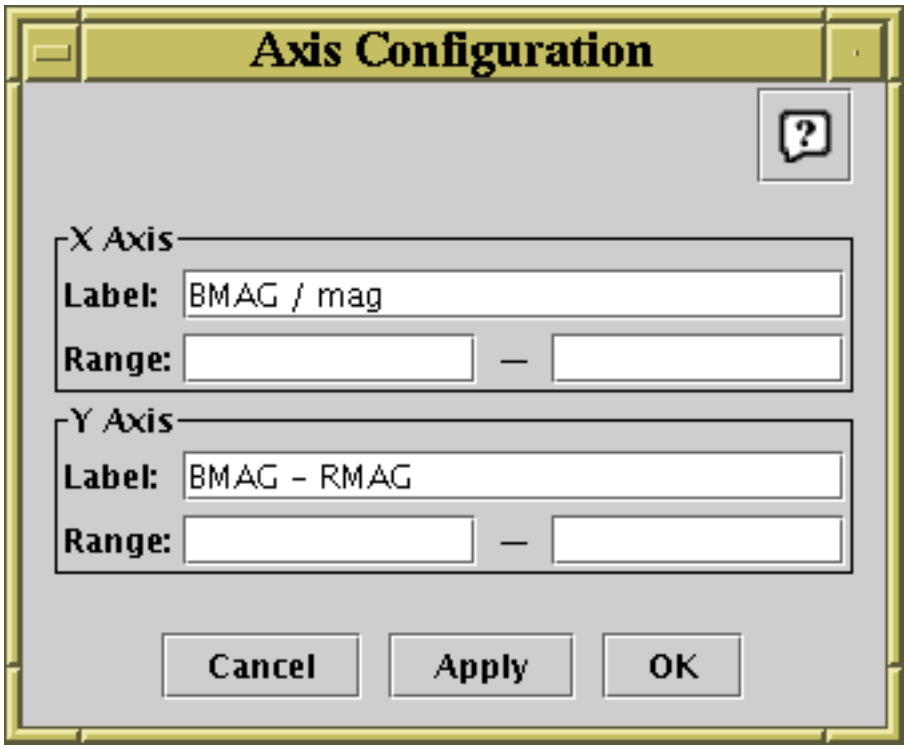
Auxiliary (Colour) Axis Zoom

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** ( ) toolbar button, which will pop up a dialogue like the following:



Axis Configuration Dialogue for 2-d axes

You can fill in these values 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).

A.4.1.3 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:

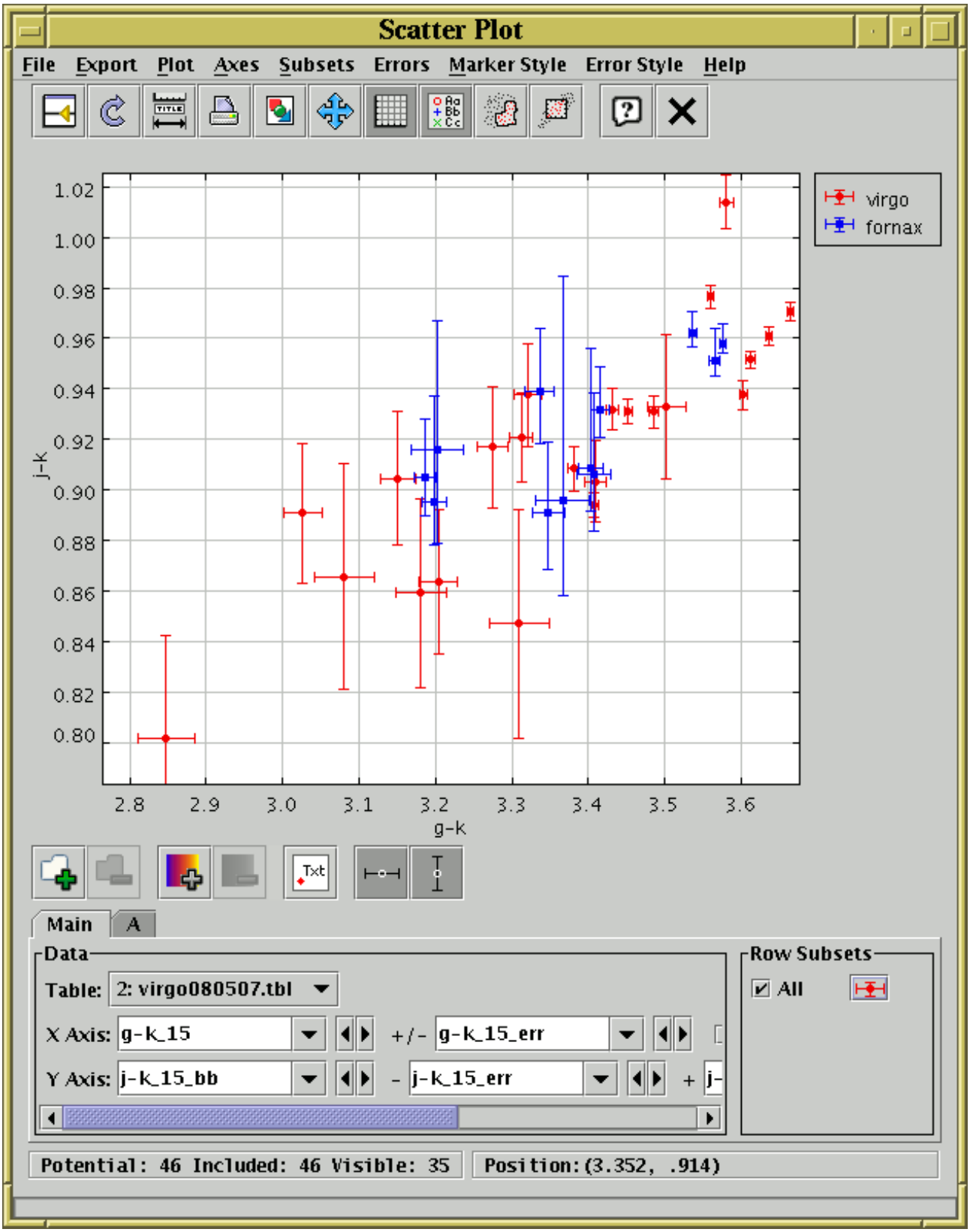
X symmetric errors

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:



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.

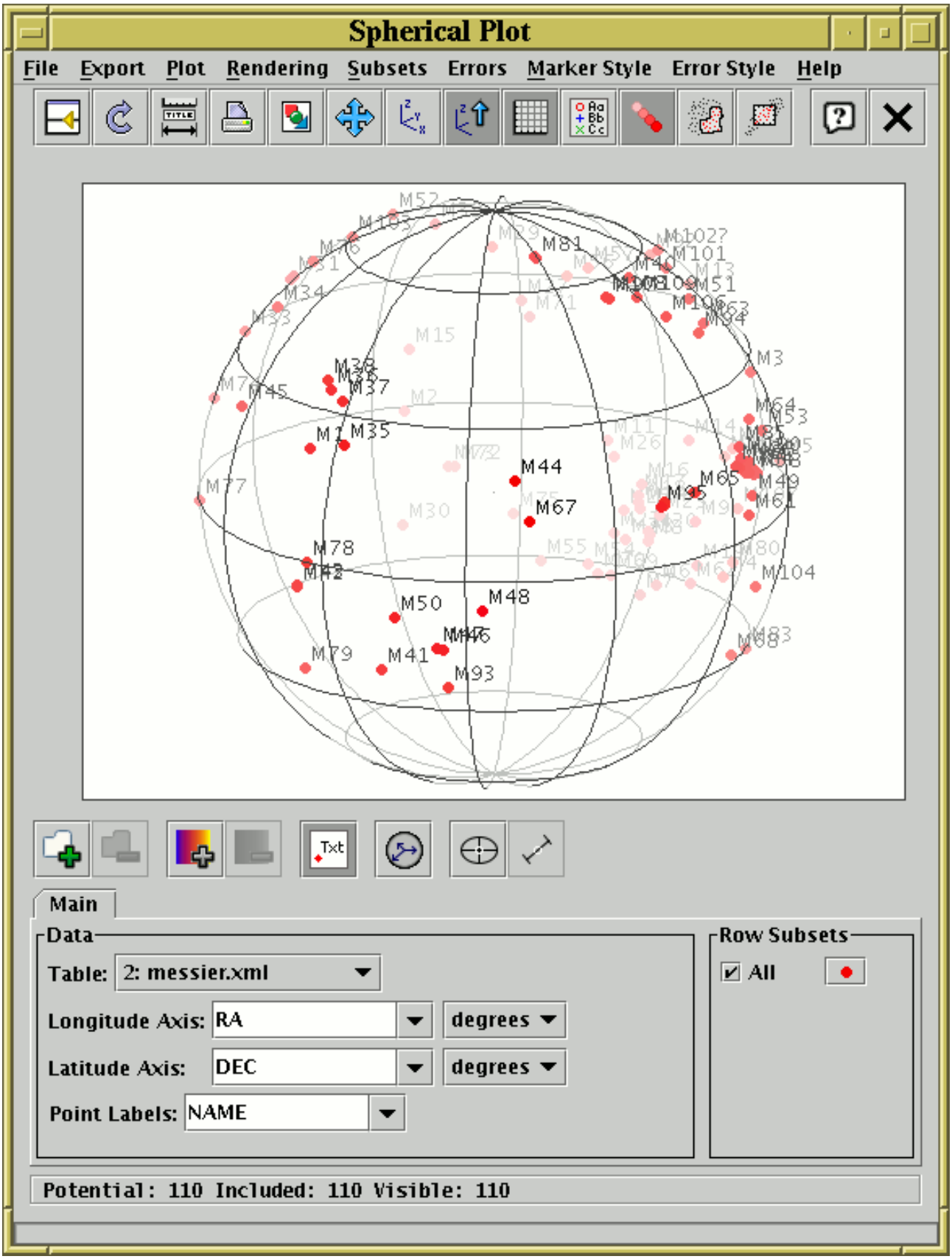
There are many options for the plotting style of one, two and three dimensional error bars, including capped and uncapped bars, crosshairs, ellipses and rectangles. This plotting style is controlled from the plot window's **Style Editor** window (see e.g. Appendix A.4.3.1), which can be viewed by clicking 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.4.1.4 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** (

) 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.



Point Labelling for Messier objects in the spherical plot

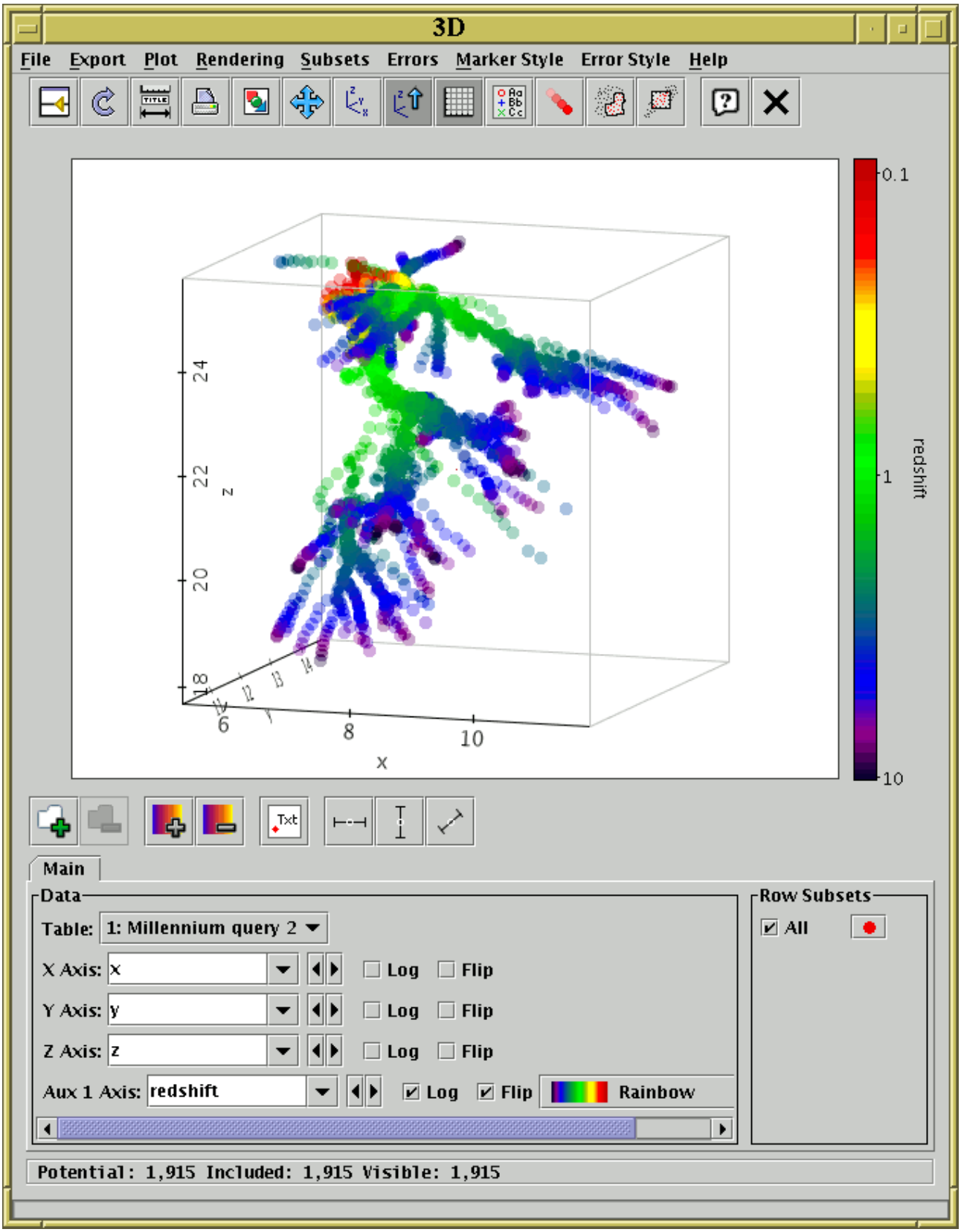
In this example the NAME column has been selected, so that each point plotted (in this case all the Messier objects) is labelled with its name. As you can see, where many labels are plotted near to each other they can get in each others' way. In some cases TOPCAT will omit plotting labels in crowded regions, in others not - but in any case if you have labels too tightly grouped they are

unlikely to be legible.

A.4.1.5 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.



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 9.2.3.

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.

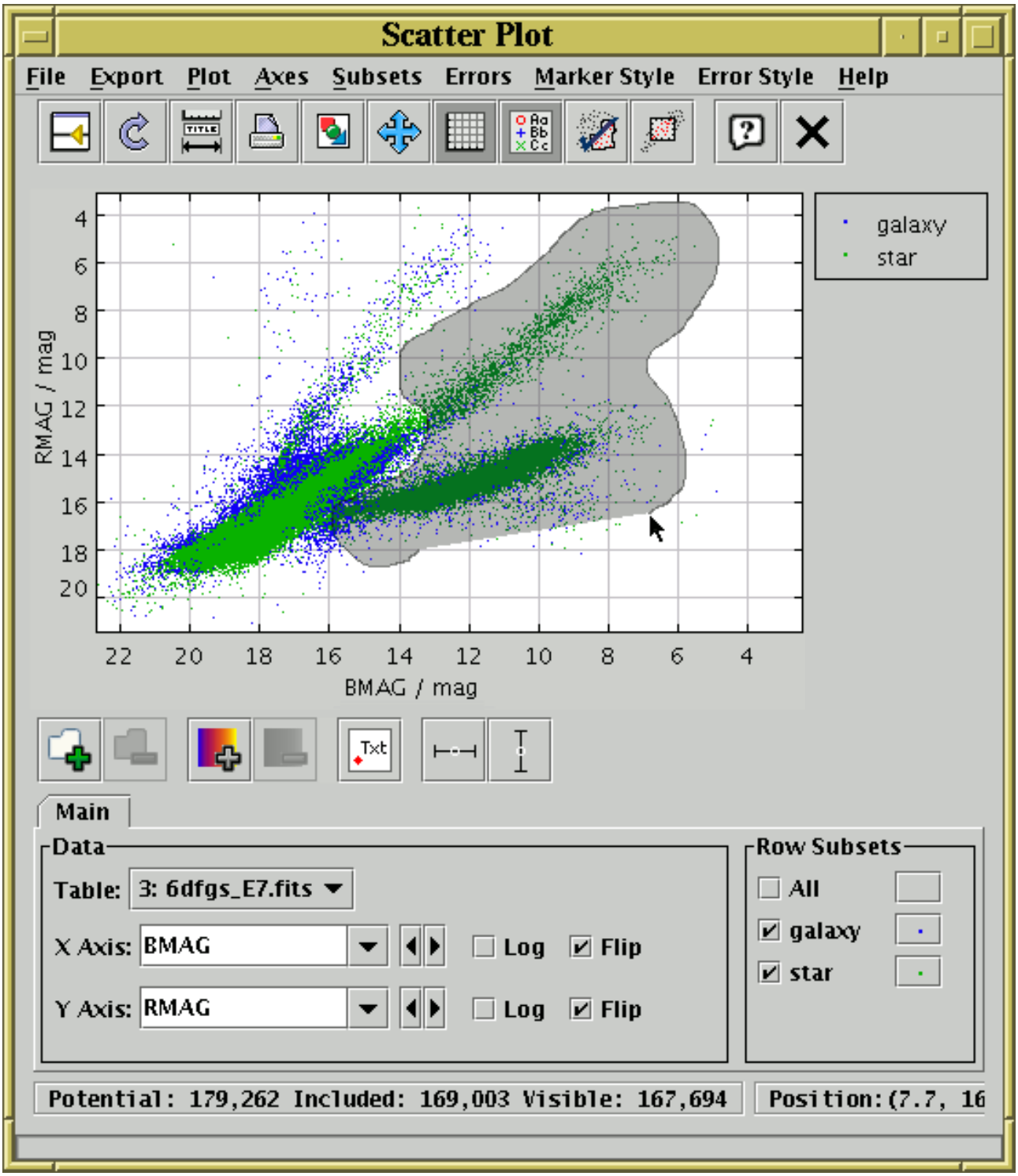
A.4.1.6 Defining Subsets by Region

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.4.1.2) so that only the points you want to identify are visible, and then click the **New 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:



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.4.1.7 Exporting Graphics

All the graphics windows have the following export options in the toolbar:

Export as Encapsulated PostScript

Export as GIF

and additionally, the **Export** menu will probably contain:

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.

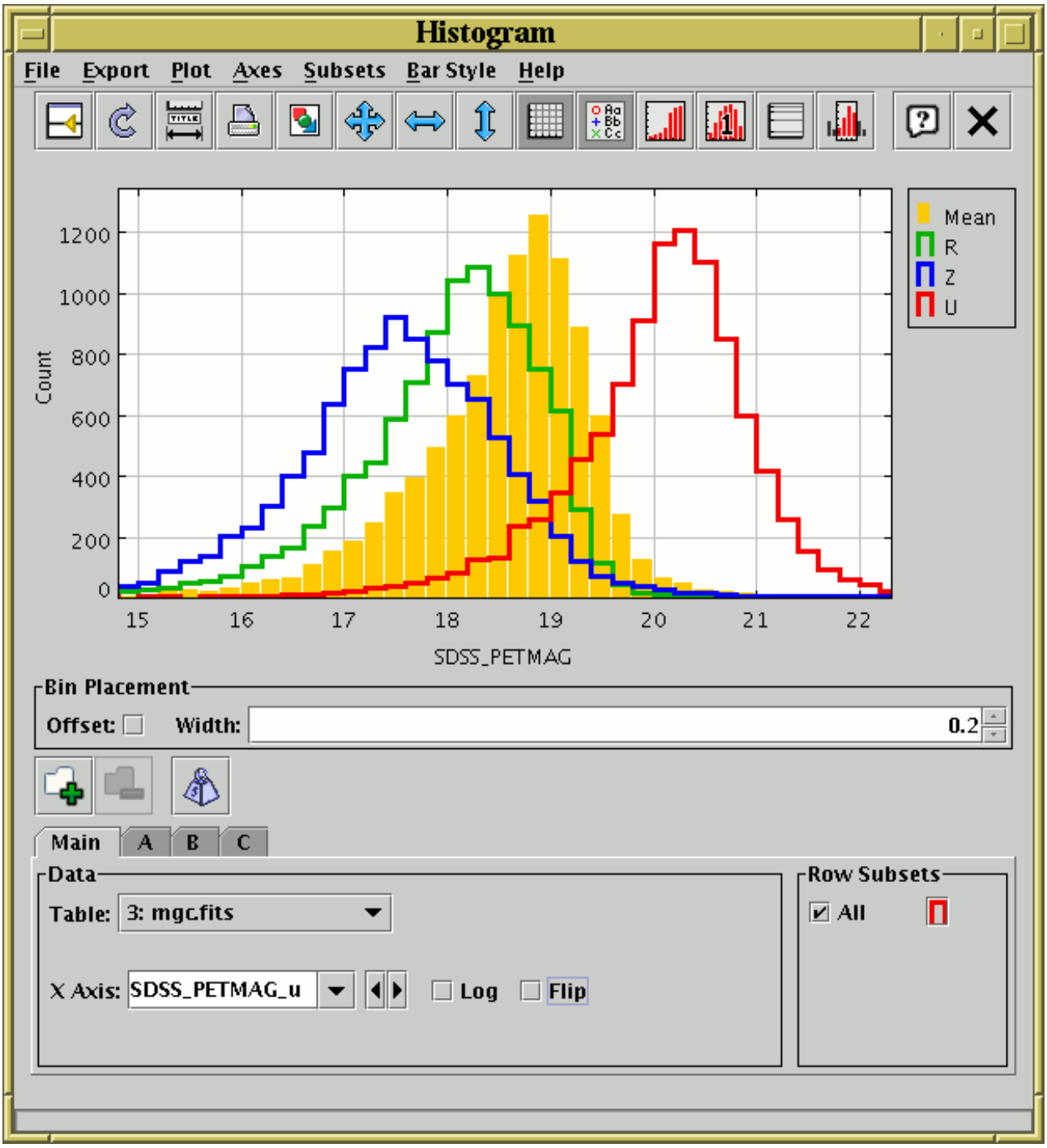
When exporting to Encapsulated PostScript (EPS), which is a vector graphics format, there are a few things to consider:

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. There may be some improvement to this in future releases.

A.4.2 Histogram



Histogram Window

The histogram window lets you plot histograms of one or more columns or derived quantities. You can display it using the **Histogram** ( ) button in the Control Window's toolbar.

You select the quantity or quantities to plot using the dataset selector (Appendix A.4.1.1) 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.4.1.2.

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

Pops up a dialogue to allow manual configuration of axis ranges and labels - see Appendix A.4.1.2.

Export as EPS

Pops up a dialogue which will write the current plot as an EPS 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.4.1.6 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.4.1.1.

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.

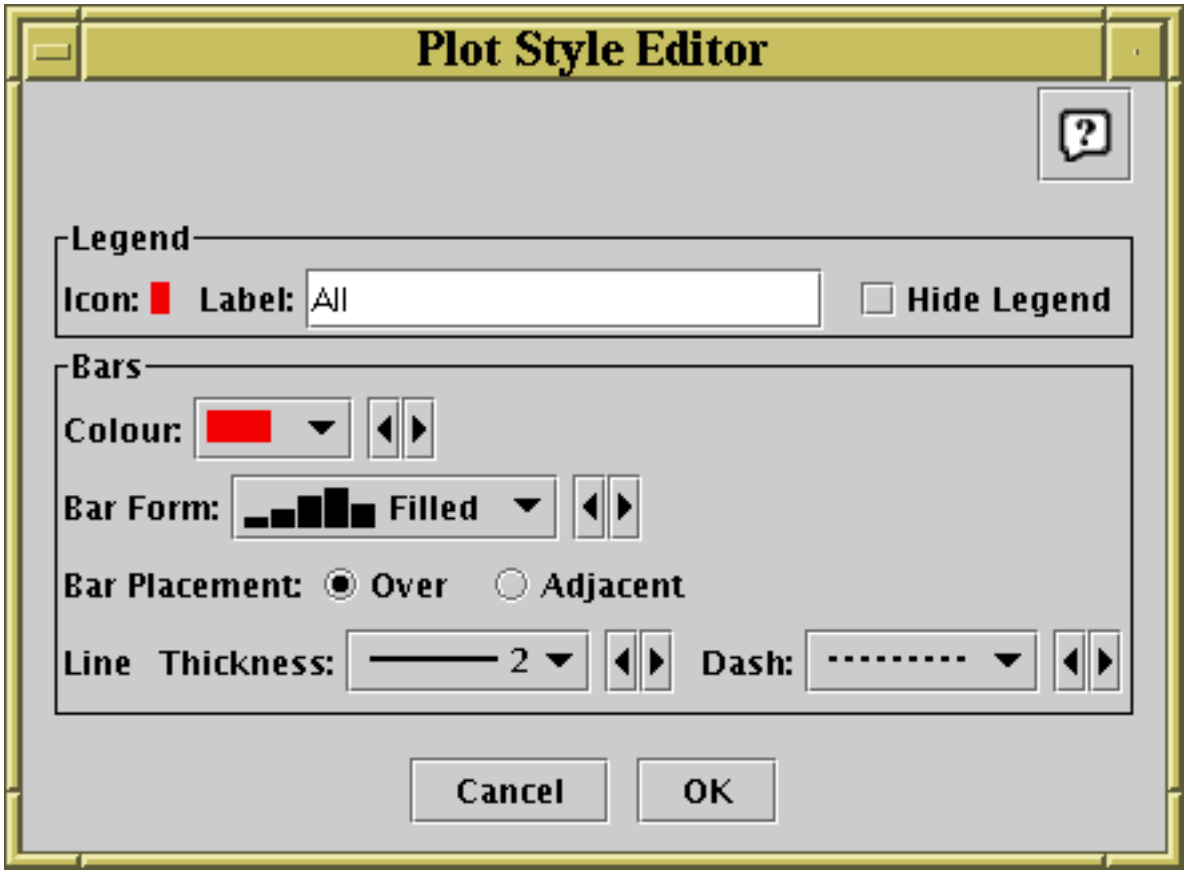
Import as Table

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.4.2.1 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.



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** ( ) 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 **Bars** 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.

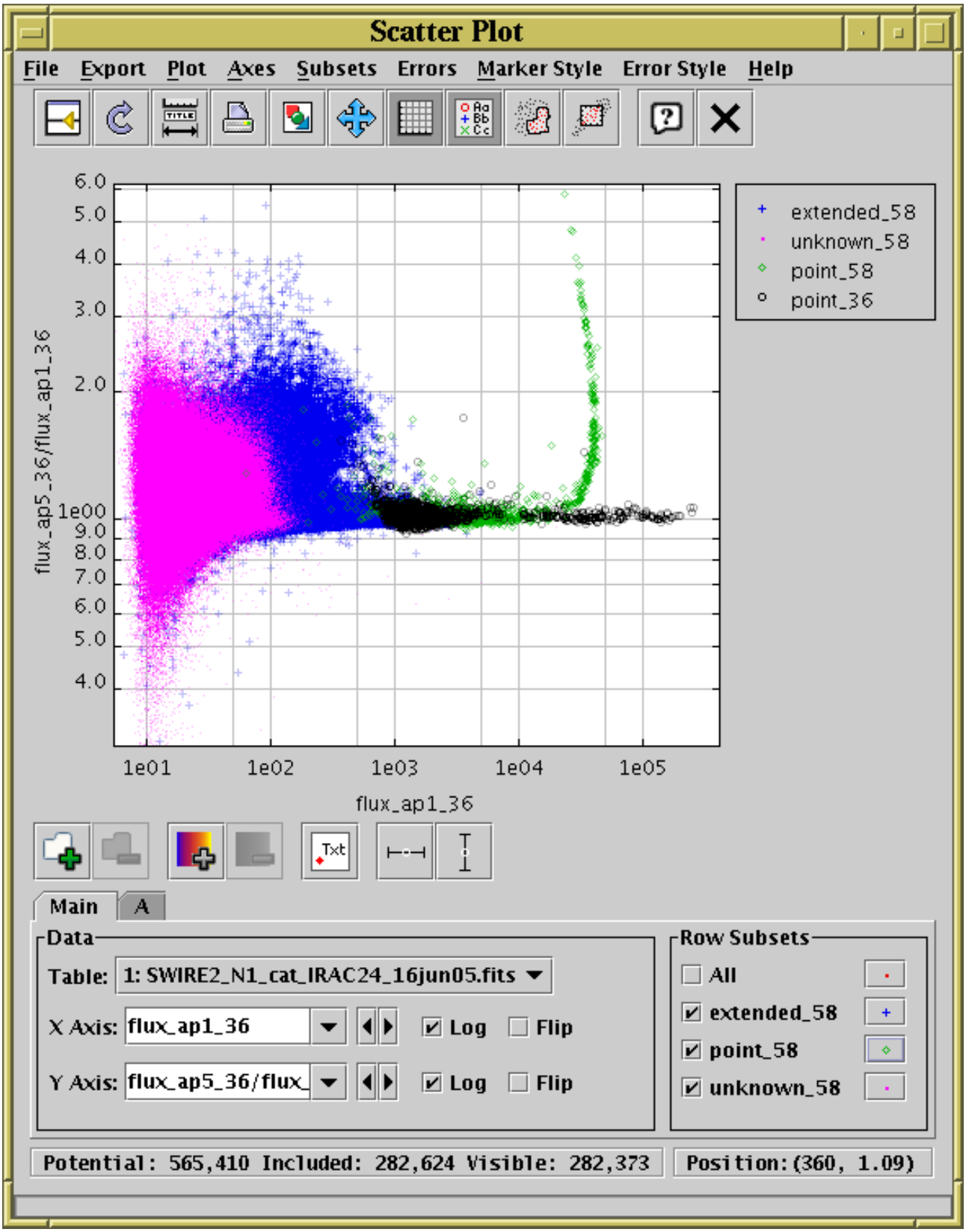
Line Thickness
Determines the thickness in pixels of any lines which are drawn. Only applies to those **Bar Form**s which use lines rather than solid blocks.

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.4.3 2D Plot



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** ( ) button in the Control Window's toolbar.

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.4.1.1) 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.4.1.2.

Clicking on any of the plotted points will **activate** it - see Section 7.

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

Pops up a dialogue to allow manual configuration of axis ranges and labels - see Appendix A.4.1.2.

Export as EPS

Pops up a dialogue which will write the current plot as an EPS 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, it won't come out right since transparency cannot be represented in PostScript; the markers will be rendered as if they were opaque. Currently, if there are many points being plotted, this can result in a rather large output 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 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.4.1.6 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.4.1.6 for more explanation.

The **Dataset Toolbar** contains the following options:

Add/Remove dataset

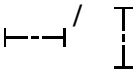
Adds/removes tabs in which the data for extra datasets can be entered. See Appendix A.4.1.1.

Add/Remove auxiliary axis

Adds/removes a selector for entering an auxiliary axis to modify point colours etc. See Appendix A.4.1.5.

Toggle point labelling

Allows text labels to be drawn near plotted points. See Appendix A.4.1.4.



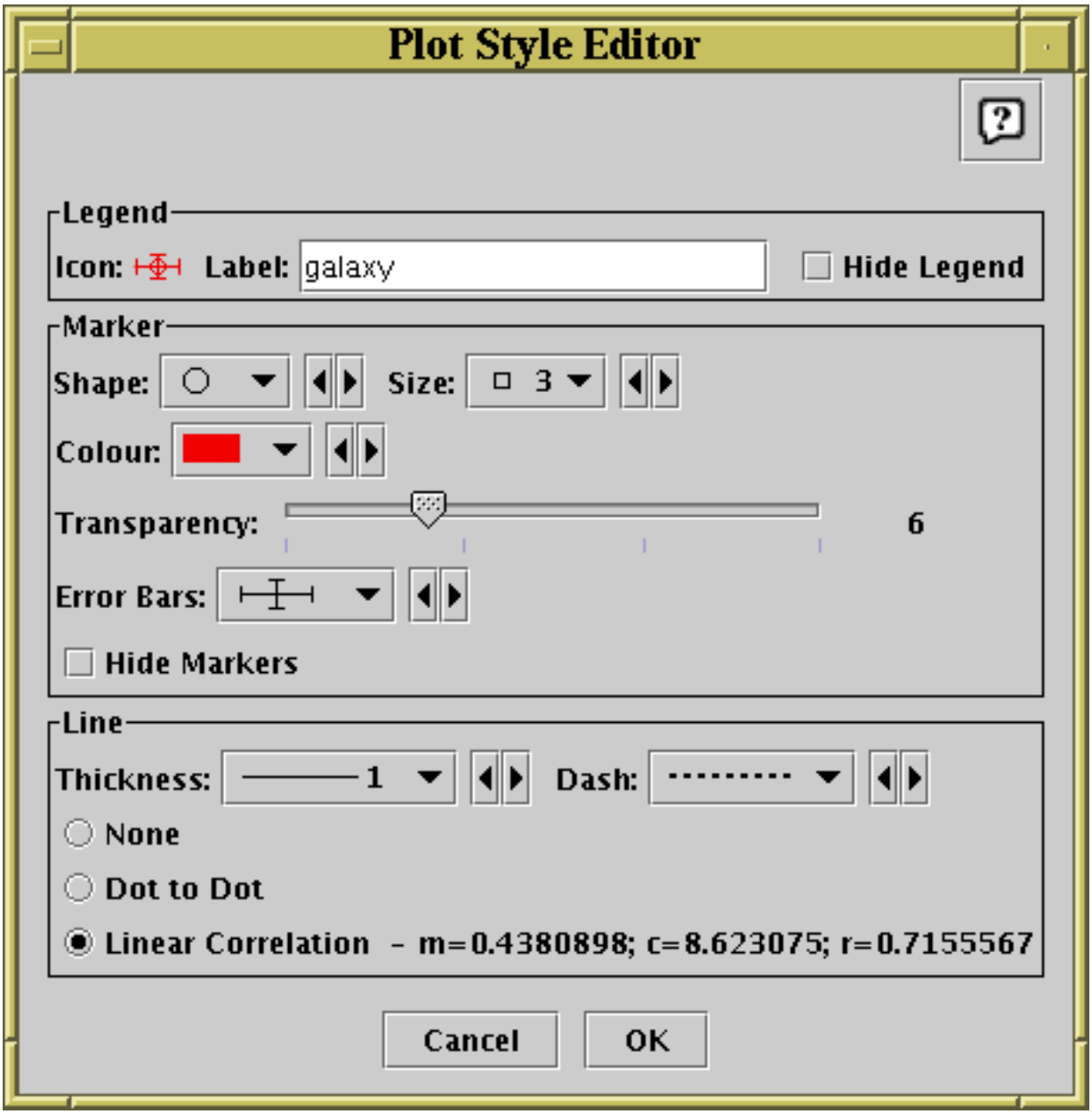
Toggle X/Y error bars

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.4.1.3.

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.4.3.1 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.



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).

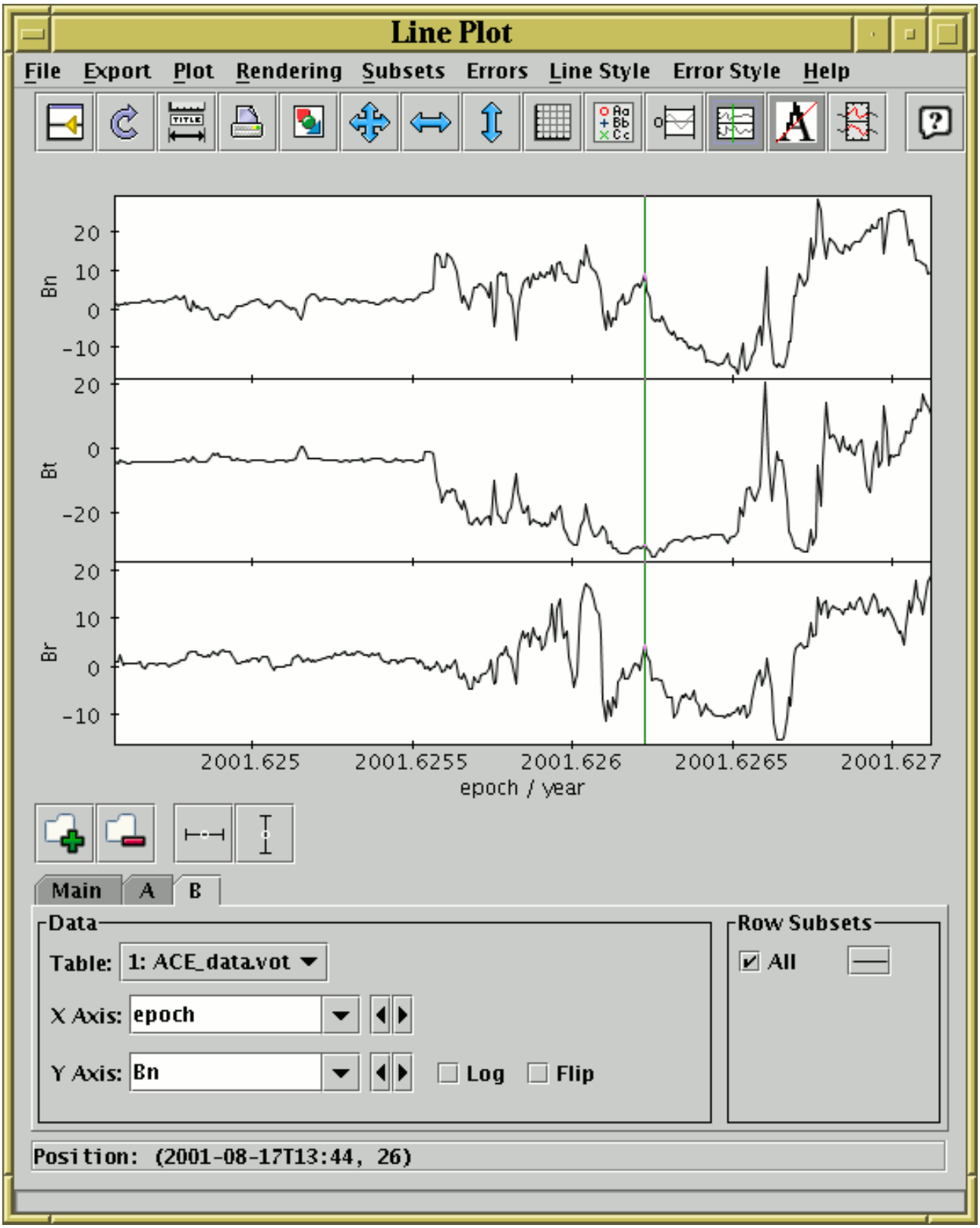
Note that for both the plotted line and the quoted coefficients the data is taken only from the points which are currently visible - that means that if you've zoomed the axes to

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.4.4 Stacked Line Plot



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** ( ) button in the Control Window's toolbar.

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** ( ) 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.4.1.2, 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 7.

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

Pops up a dialogue to allow manual configuration of the range and label for the X axis and each of the Y axes - see Appendix A.4.1.2.

Export as EPS

Pops up a dialogue which will write the current plot as an EPS 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.4.1.3.

You can determine how the data are plotted using lines and/or markers as described in the following subsection.

A.4.4.1 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.

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**.

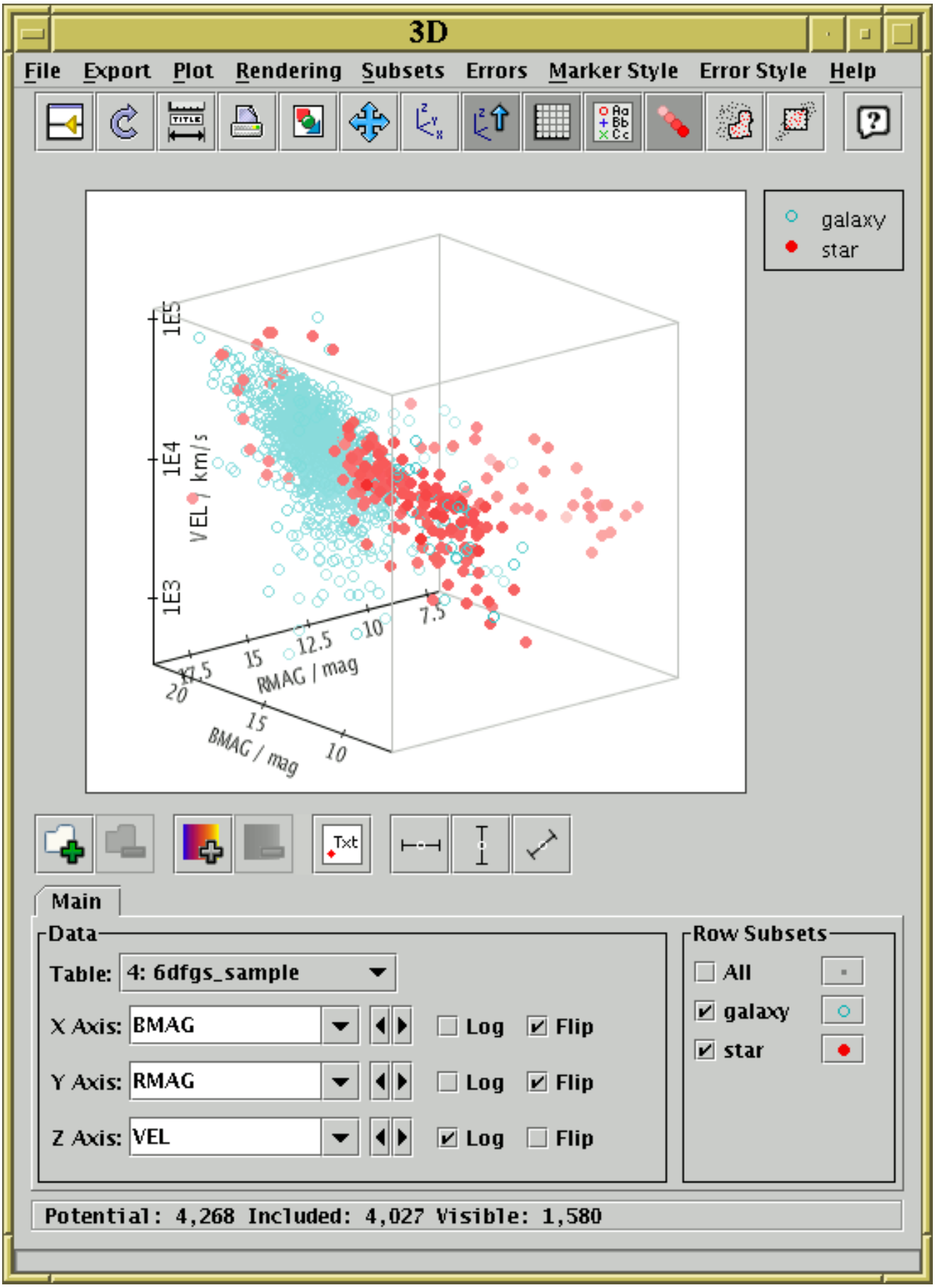
Size
Determines the size of the markers in pixels. If a size of zero is chosen then the shape doesn't matter, the points will be plotted as single pixels.

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.4.5 3D Plot



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** ( ) button in the Control

Window's toolbar.

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.4.1.1) 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. This resembles the **Axis Zoom** in some of the 2-d plots, but in this case the active region is the space to the left of the plot. 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.4.1.6 for more explanation.

Clicking on any of the plotted points will **activate** it - see Section 7.

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

Pops up a dialogue to allow manual configuration of axis ranges and labels - see Appendix A.4.1.2.

Export as EPS

Pops up a dialogue which will write the current plot as an EPS 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, it won't come out right since transparency cannot be represented in PostScript; the markers will be rendered as if they were opaque. Currently, if there are many points being plotted, this can result in a rather large output 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 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.4.1.6 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.4.1.6 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.4.1.1.

Add/Remove auxiliary axis

Adds/removes a selector for entering an auxiliary axis to modify point colours etc. See Appendix A.4.1.5.

Toggle point labelling

Allows text labels to be drawn near plotted points. See Appendix A.4.1.4.

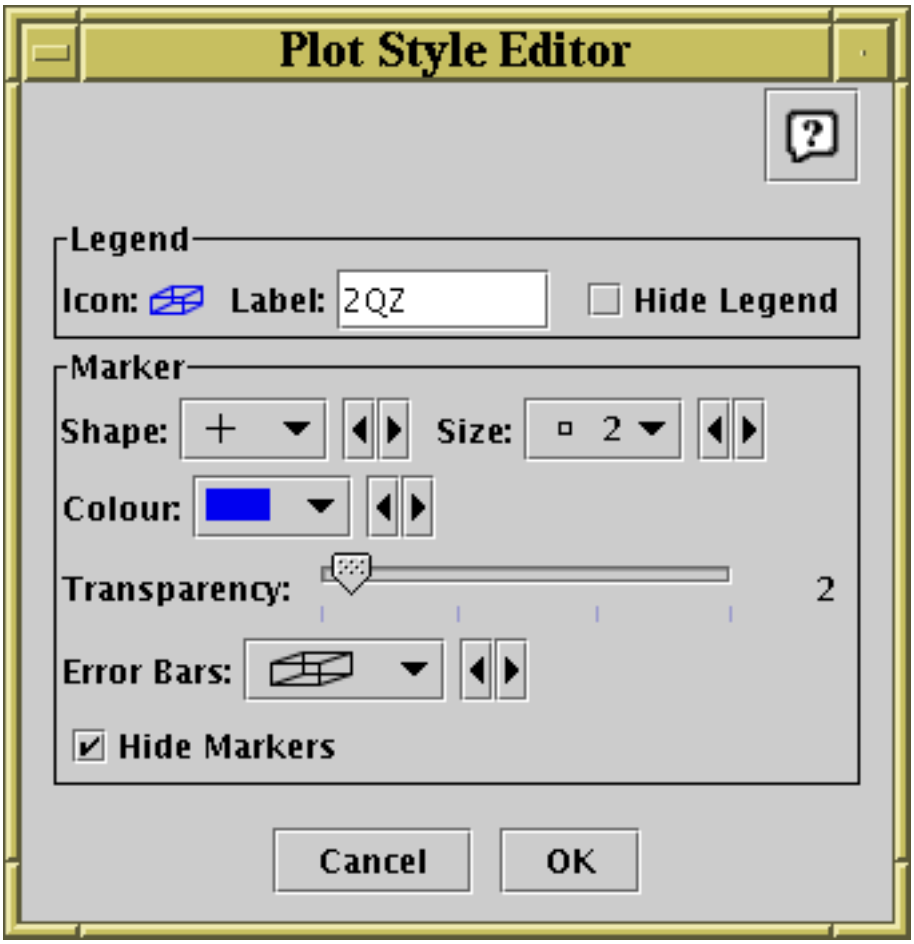
Toggle X/Y/Z error bars

Switches between drawing symmetric error bars and no error bars in the X, Y and Z directions respectively. Other optinos are available in the **Errors** menu. See Appendix A.4.1.3.

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.4.5.1 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.



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** ( ) 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.

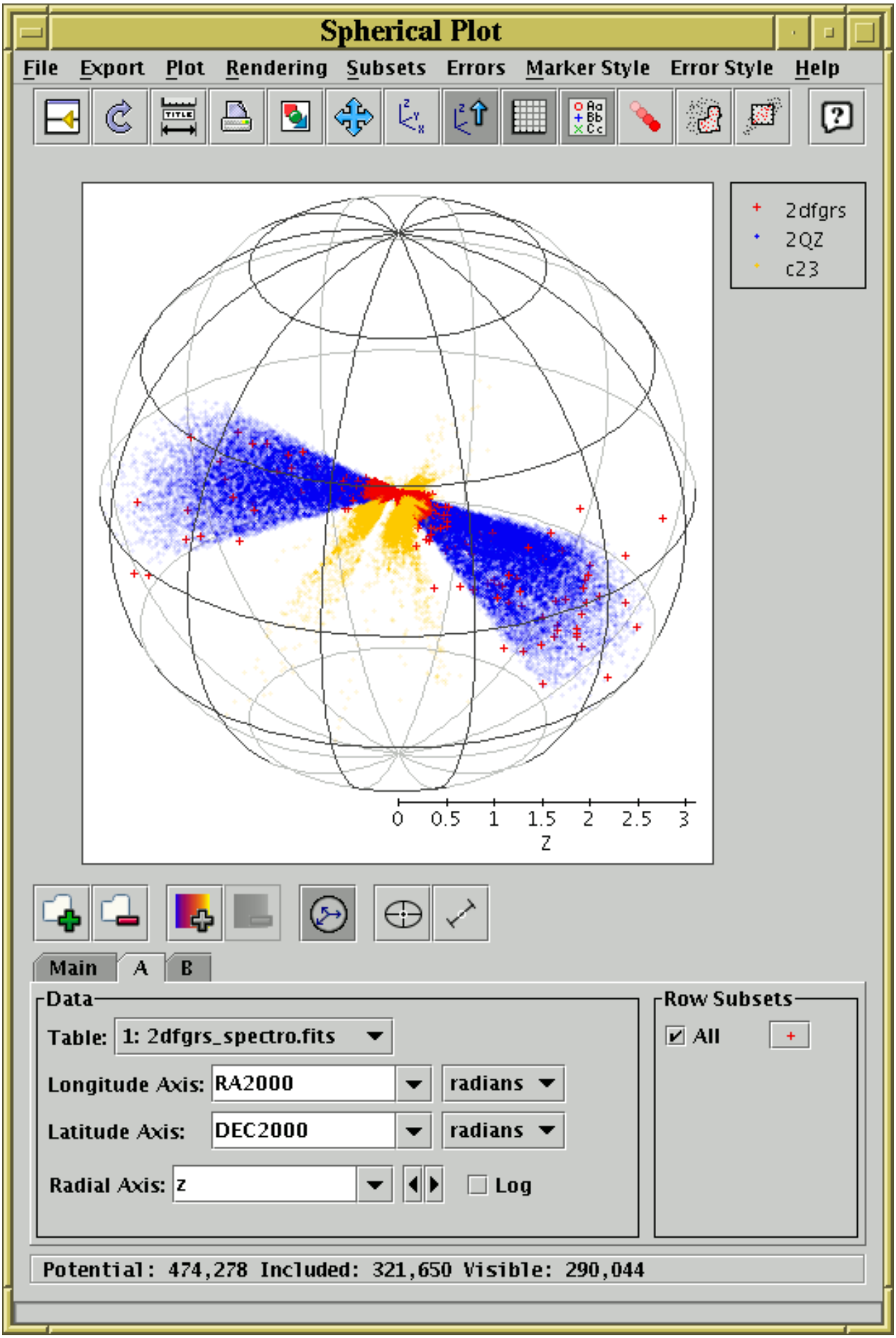
Hide Markers

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.4.6 Spherical Plot


Spherical Plot
File Export Plot Rendering Subsets Errors Marker Style Error Style Help
2dfgrs
2QZ
c23
0 0.5 1 1.5 2 2.5 3
Z
Main A B
Data
Table: 1: 2dfgrs_spectro.fits
Longitude Axis: RA2000 radians
Latitude Axis: DEC2000 radians
Radial Axis: z Log
Row Subsets
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** ( ) button in the Control Window's toolbar.

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.4.1.1) 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 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

Pops up a dialogue to allow manual configuration of axis ranges and labels - see Appendix A.4.1.2. The only configurable axis range is the upper limit of the radial axis.

Export as EPS

Pops up a dialogue which will write the current plot as an EPS 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, it won't come out right since transparency cannot be represented in PostScript; the markers will be rendered as if they were opaque. Currently, if there are many points being plotted, this can result in a rather large output 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 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.4.1.6 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.4.1.6 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.4.1.1.

Add/Remove auxiliary axis

Adds/removes a selector for entering an auxiliary axis to modify point colours etc. See Appendix A.4.1.5.

Toggle point labelling

Allows text labels to be drawn near plotted points. See Appendix A.4.1.4.

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

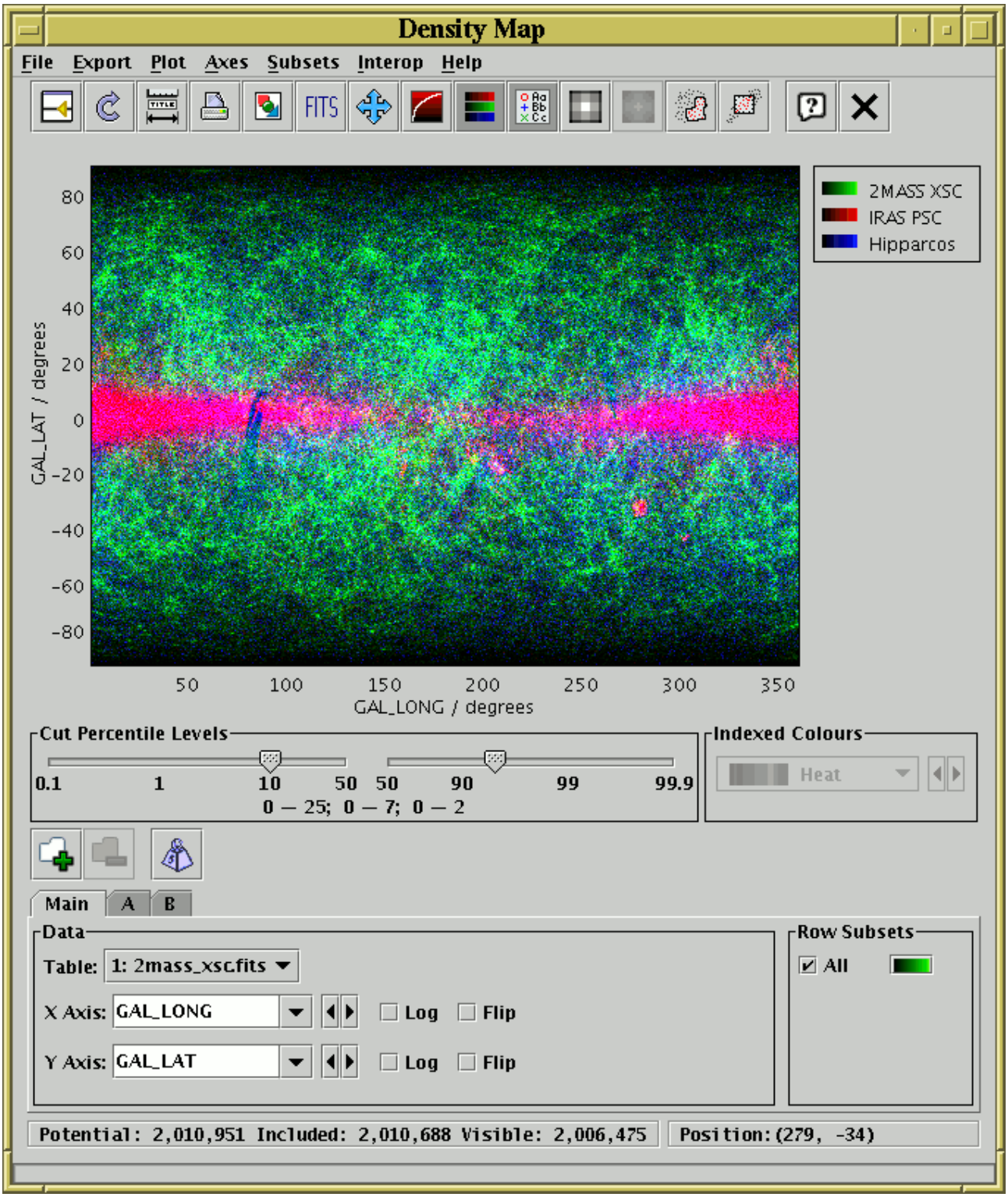
When activated, an additional column selector appears in the dataset panel to the right of the Longitude and Latitude selectors, along with its own unit selector. You can fill this in with an isotropic error value representing the radius of a small circle associated with the selected points, in units 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.4.1.3 for more information.

Toggle radial error bars

Switches radial error bars on and off. See Appendix A.4.1.3. 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.4.5.1.

A.4.7 Density Map



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 9.2.3.

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.4.1.2.

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

Pops up a dialogue to allow manual configuration of axis ranges and labels - see Appendix A.4.1.2.

Export as EPS

Pops up a dialogue which will write the current plot as an EPS 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.4.1.6 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.4.1.6 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.4.1.1.

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 8 for more information about tool interoperability.

How to set the colour channel corresponding to each dataset is explained in the following subsection.

A.4.7.1 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.

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** ( ) is on):

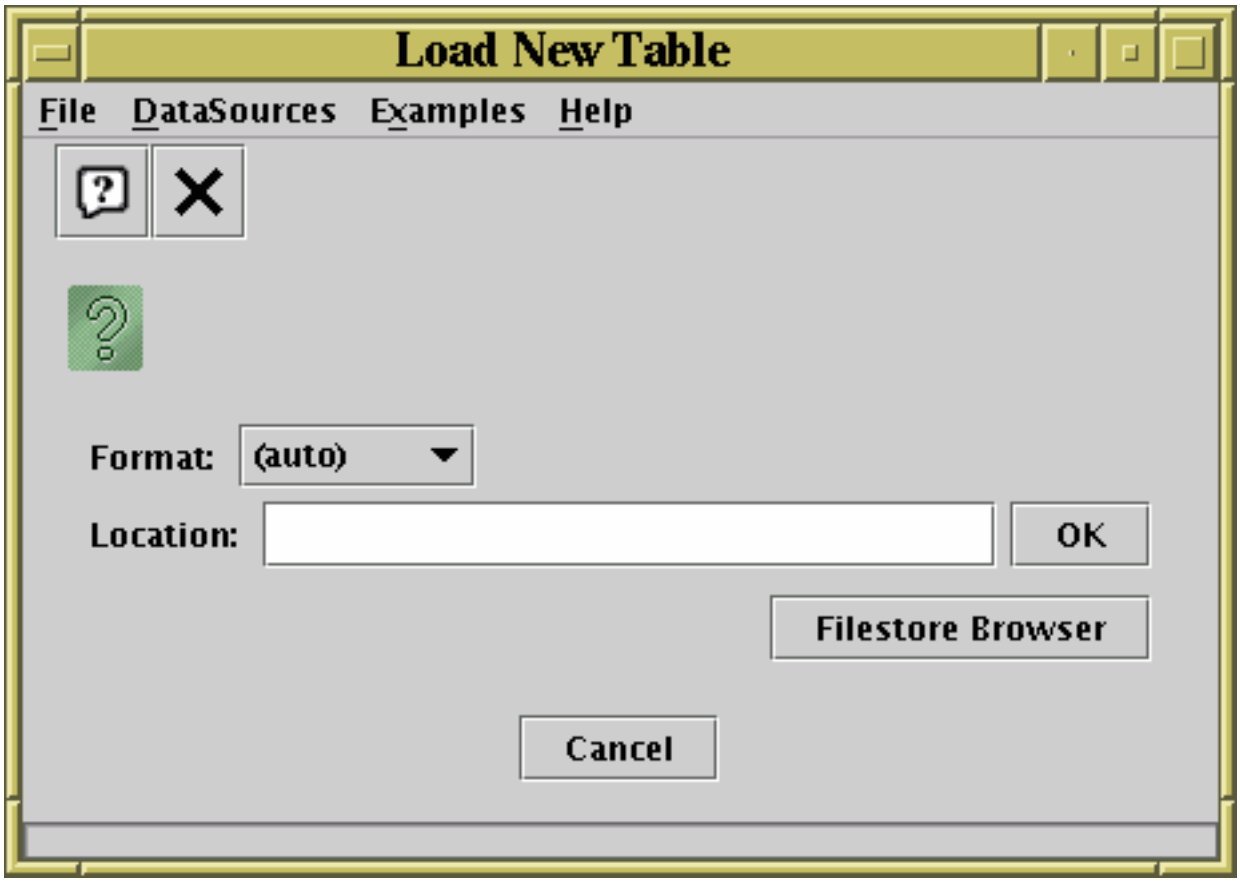
Icon
Displays the icon which will be shown to identify the points in the selected set. Its appearance depends on the selections you make in the rest of this dialogue window.

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.5 Load Window



Load Window

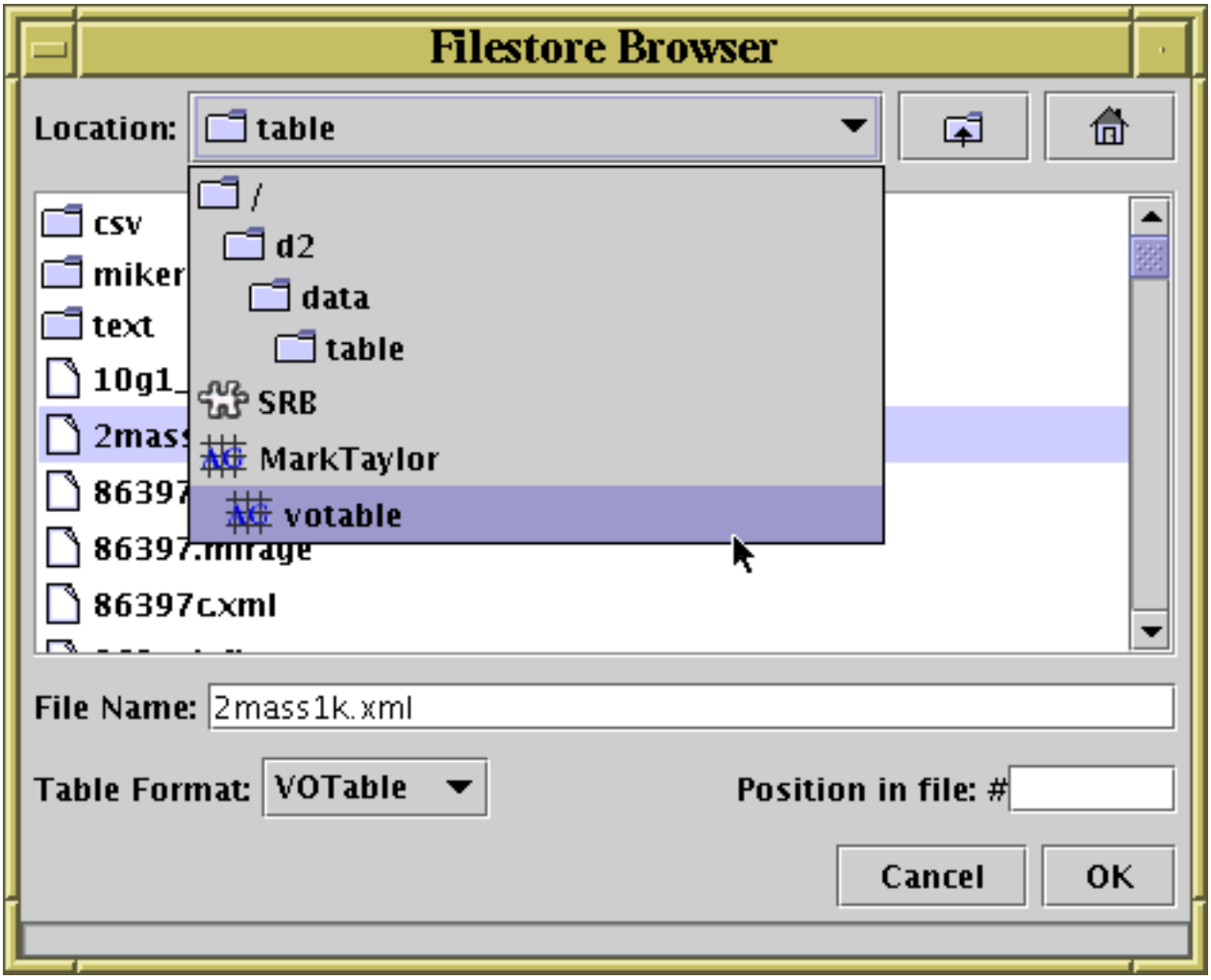
The Load Window is used for loading tables from an external location (e.g. disk or URL) into TOPCAT. It is obtained using the **Load Table** button ( ) in the Control Window toolbar or File menu.

This dialogue allows you to specify a new table to open in several different ways, described below. If you successfully load a table using any of these options, a new entry 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. If you get an `OutOfMemoryError` while loading a table, you will have to run TOPCAT with more memory, as described in Section 9.2.2 or use the `-disk` flag described in Section 9.1.

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 - you can leave it on **(auto)** for loading FITS tables or VOTables, but for other formats such as ASCII or CSV you must select the right one explicitly (again, see Section 4.1.1 for details).

There are many other ways of loading tables however, described in the following subsections. The **Filestore Browser** button is always visible below the location field. Depending on startup options, there may be other buttons here. In any case, you can look in the **DataSources** menu to see other table load dialogues. Exactly which ones are available will depend on your setup (some may be absent or greyed out, and additional ones may be available). The following subsections describe some of the options which may be available.

A.5.1 Filestore Browser



Filestore Browser window

By clicking the **Filestore Browser** 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. Access to remote filesystems is dependent on certain optional components of TOPCAT, and it may not be available if you have the topcat-lite configuration.

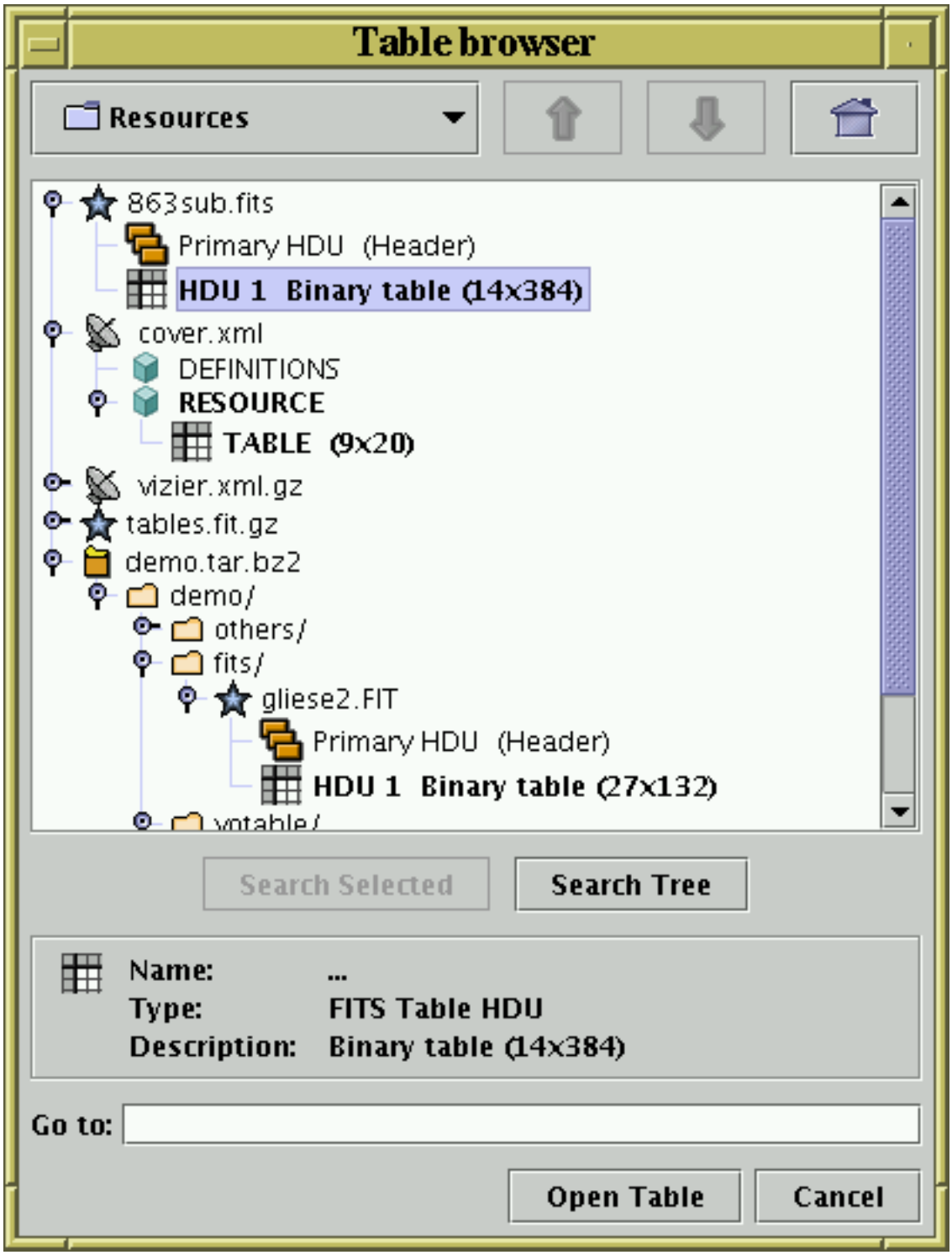
The browser initially displays the current directory, but this can be changed by typing a new directory into the **File Name** field, or moving up the directory hierarchy using the selector box at the top, or navigating the file system by clicking the up-directory button or double-clicking on displayed directories. The initial default directory can be changed by setting the `user.dir` system property (Section 9.2.3).

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 of the HDU containing the table you're after (the first extension after the primary HDU is numbered 1), 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 the first table in the file in question - many file formats only allow one table per file in any case. For a more table-aware view of the file system, use the Hierarchy Browser (Appendix A.5.2) instead.

A.5.2 Hierarchy Browser



File load Hierarchy Browser window

By selecting the **Hierarchy Browser** option from the Load Window's **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 9.2.3)). 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.5.7.

Note that this window requires certain optional components of the TOPCAT installation, and will not be available if you have the topcat-lite configuration.

A.5.2.1 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.

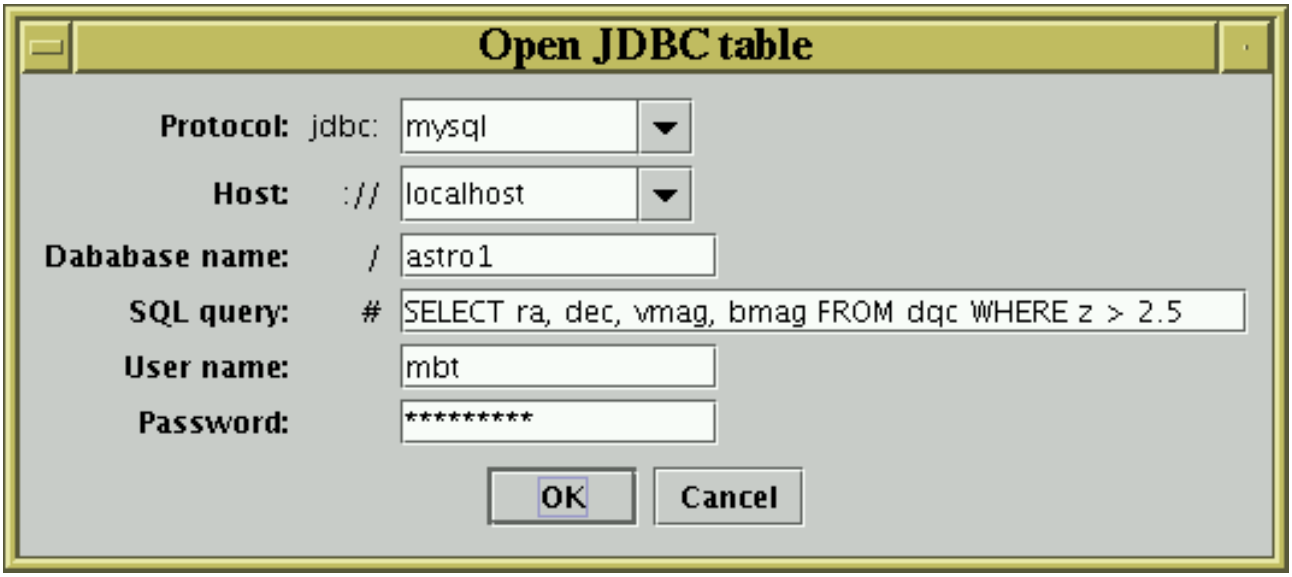
(In fact the above navigation options are not restricted to changing the root to a new directory, they

can move to any node in the tree, for instance a level in a Tar archive.)

A.5.2.2 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.5.3 SQL Query



SQL Query Dialogue

If you want to read a table from an SQL database (Section 4.1.1.8), you can use a specialised dialogue to specify the SQL query by selecting **SQL Query** option from the Load Window's **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.

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

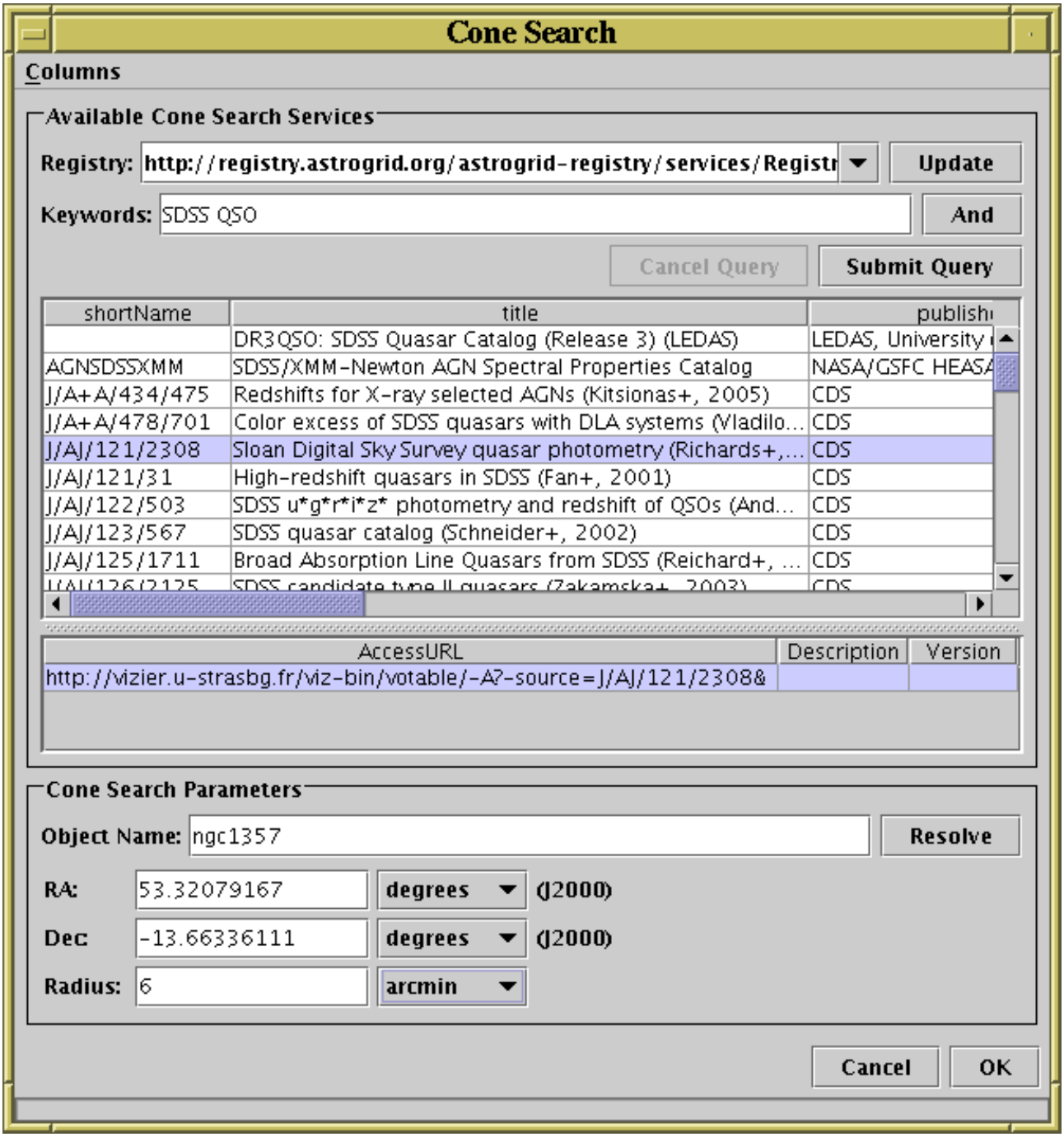
is permitted will be defined by the JDBC driver you are using.

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.

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 9.3. If you don't take these steps, this dialogue may be inaccessible.

A.5.4 Cone Search



Cone search table import dialogue

By selecting the **Cone Search** option from the Load Window's **DataSources** menu, you can obtain a dialogue which allows you to query one of a number of external web services for a catalogue of objects known in a given region of the sky.

There are a lot of cone search services out there, so the first thing is to decide which one you want to query. When this dialogue first appears you must fill in the **Registry** and **Keywords** fields at the top of the window. The Registry value defaults to the AstroGrid registry, but you can choose another if you prefer. In general, since the various registries harvest resources from each other, they can be expected to have similar content, but there will be exceptions. The **Update** button searches the currently selected registry for more registry services, so gives you a longer list of registries to choose from. The Keywords value is a space-separated list with no wildcarding. Each keyword is matched against the title, description, identifier, shortname and some other attributes to find matching services. By default a resource must match all of these to be selected, but you can toggle it to match any of them by clicking the **And/Or** button to the right of the selector.

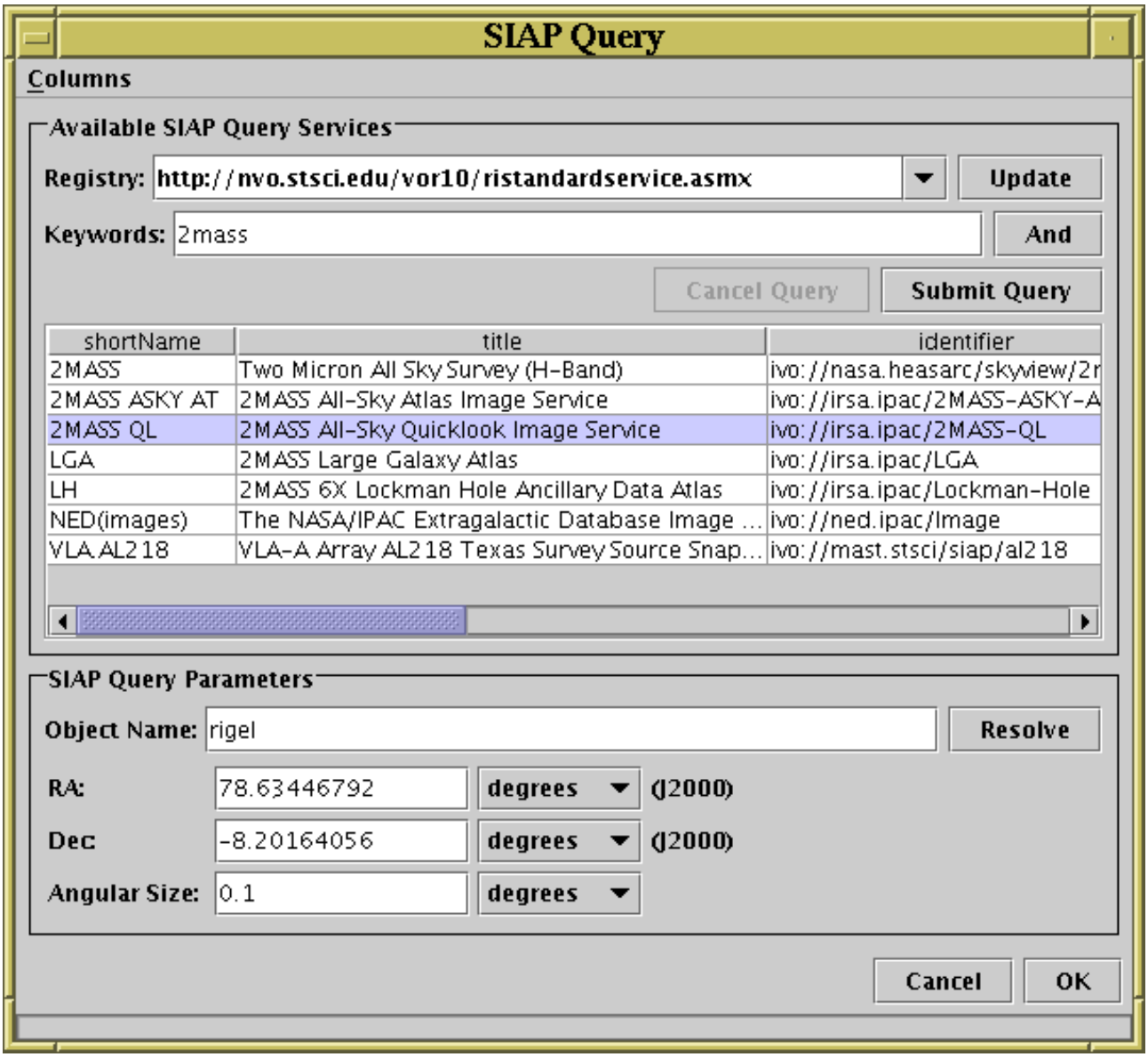
Once these fields are filled in, hit the **Submit** button, and TOPCAT will query the selected registry for a list of the services which contain the chosen keywords. The services found will be displayed in the table. For more information about each one, use the **Columns** menu to select what information, such as publisher, reference URL etc is displayed in the table. You can scroll up and down this table and select the one which you want to query by clicking on it. The lower table lists the actual cone search services associated with each registry resource found. In most cases there is exactly one cone search service per resource, but if there is more than one you will need to select the one you want by clicking on it (it defaults to the first in the list).

Having selected one of the cone search services from the table, you need to specify the sky region in which you are interested. If you enter the name of an astronomical object into the **Object Name** field and hit the **Resolve** button, the coordinates will be entered into the **RA** and **Dec** fields below. Alternatively you can type the coordinates in directly, choosing either degrees or sexagesimal coordinates using the unit selector boxes. Enter the search radius too.

Having done this, hit the **OK** button. This will send the query to the service you selected and, if successful, load into TOPCAT a table containing all the objects in the region of the sky you have specified. The exact format of the returned table will depend on the service you have selected, but it will contain at least columns representing Right Ascension and Declination.

Note that this window requires certain optional components of the TOPCAT installation, and will not be available if you have the topcat-lite configuration.

A.5.5 SIAP/SSAP Query



SIAP table import dialogue. The SSAP one looks very similar

By selecting the **SIAP** or **SSAP** option from the Load Window's **DataSources** menu, you can obtain a dialogue which allows you to perform a Simple Image Access or Simple Spectral Access protocol query to one of the servers on the network which supports the Virtual Observatory protocol in question.

The purpose of these protocols is to allow you to locate image or spectral data from virtual observatory servers. The basic idea is that you decide which server you want to query, specify constraints such as the region of sky in which you are interested, and then make the query. The server responsds with a table (a VOTable) which contains a row for each image/spectral data set matching the query. The load dialogues described here allow you to locate suitable servers, make simple queries along these lines, and view the returned tables in TOPCAT. This isn't the most sensible use of these protocols - typically they would be used by an image/spectrum viewer which would take care of acquiring the actual image/spectrum data described in the return table - so these dialogues are of somewhat limited use. Moreover, they only provide a subset of the options specified by the protocols (especially in the case of SSAP, which defines many possible constraints beyond sky position). However, they're here if you want to use them for whatever reason.

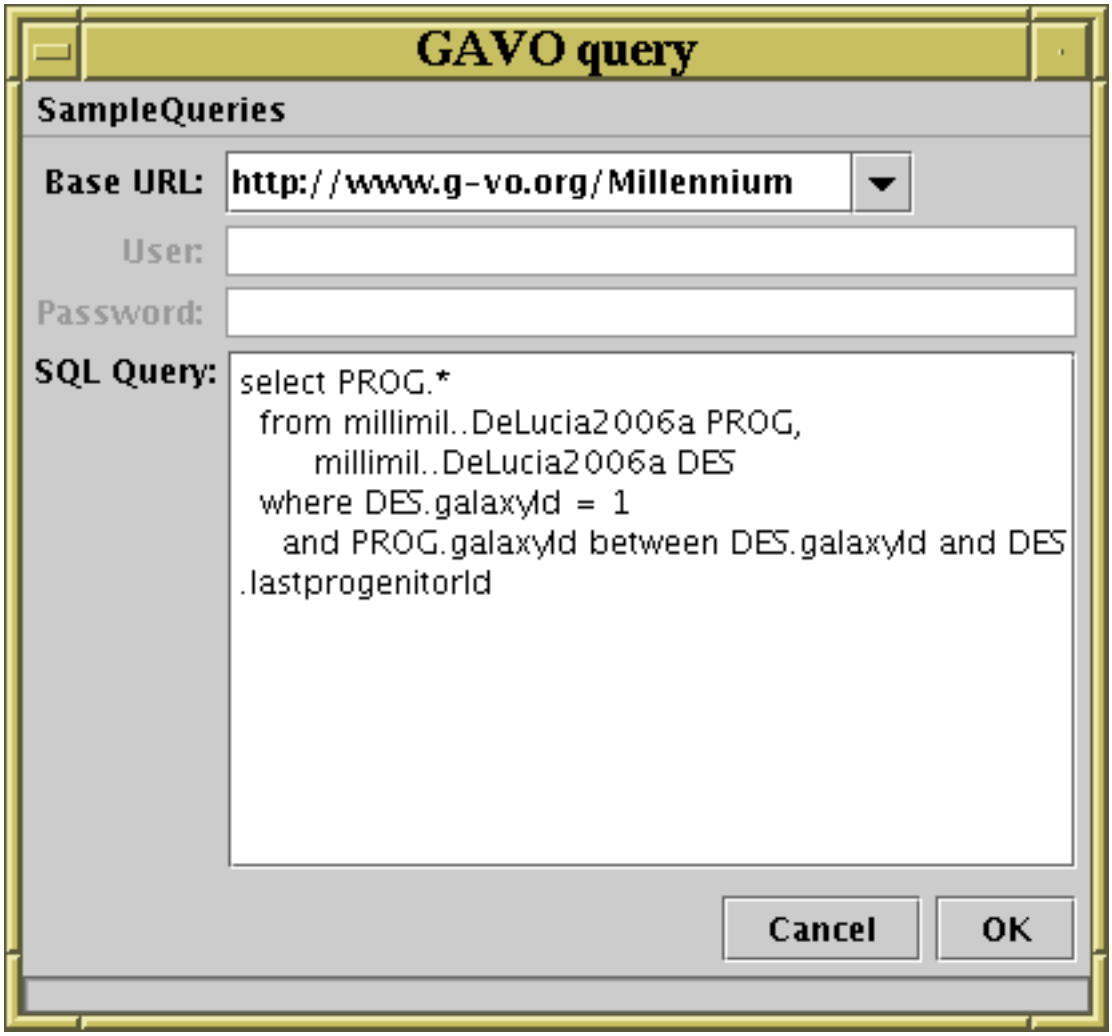
Operation is quite similar to the Cone Search dialogue (see Appendix A.5.4 for slightly more detail on this). First you need to select the server to use. Fill in the **Registry** and **Keywords** fields at the

top of the window to restrict which SIAP/SSAP servers you want to search for, and hit the **Submit Query** button. If you leave the **Keywords** field blank, all suitable servers will be listed; at the time of writing, there are not very may SSAP servers in total, so it's not an unreasonable thing to do. Then select the server you want by clicking on its row in the table.

Having selected the server, select the region of sky in which you are interested. Either fill in the **RA** and **Dec** fields manually, or fill in the **Object Name** field to get it resolved to a position using Sesame. Then enter the region size - semantics of this differ slightly between protocols, see the tooltip for details. Finally, hit **OK** and the query will be performed, loading the result into TOPCAT as a new table.

Note that this window requires certain optional components of the TOPCAT installation, and will not be available if you have the topcat-lite configuration.

A.5.6 GAVO Millennium Database Query



GAVO load dialogue with an example query on the milli-Millennium database

This dialogue permits direct queries to the services provided by GAVO, the German Astrophysical Virtual Observatory. The main databases of general interest available through these services are the **Millennium Simulation** results, documented at http://www.g-vo.org/Millennium/Help.

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://www.g-vo.org/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://www.g-vo.org/MyMillennium`)
Full, protected simulation results database. The username and password must be filled in for these queries. Registration is by application to GAVO.

Other
Other GAVO-provided services 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 **SampleQueries** menu provides 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 and descriptions of the database schemas, is available on the GAVO website, at http://www.g-vo.org/Millennium/Help.

A.5.7 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.

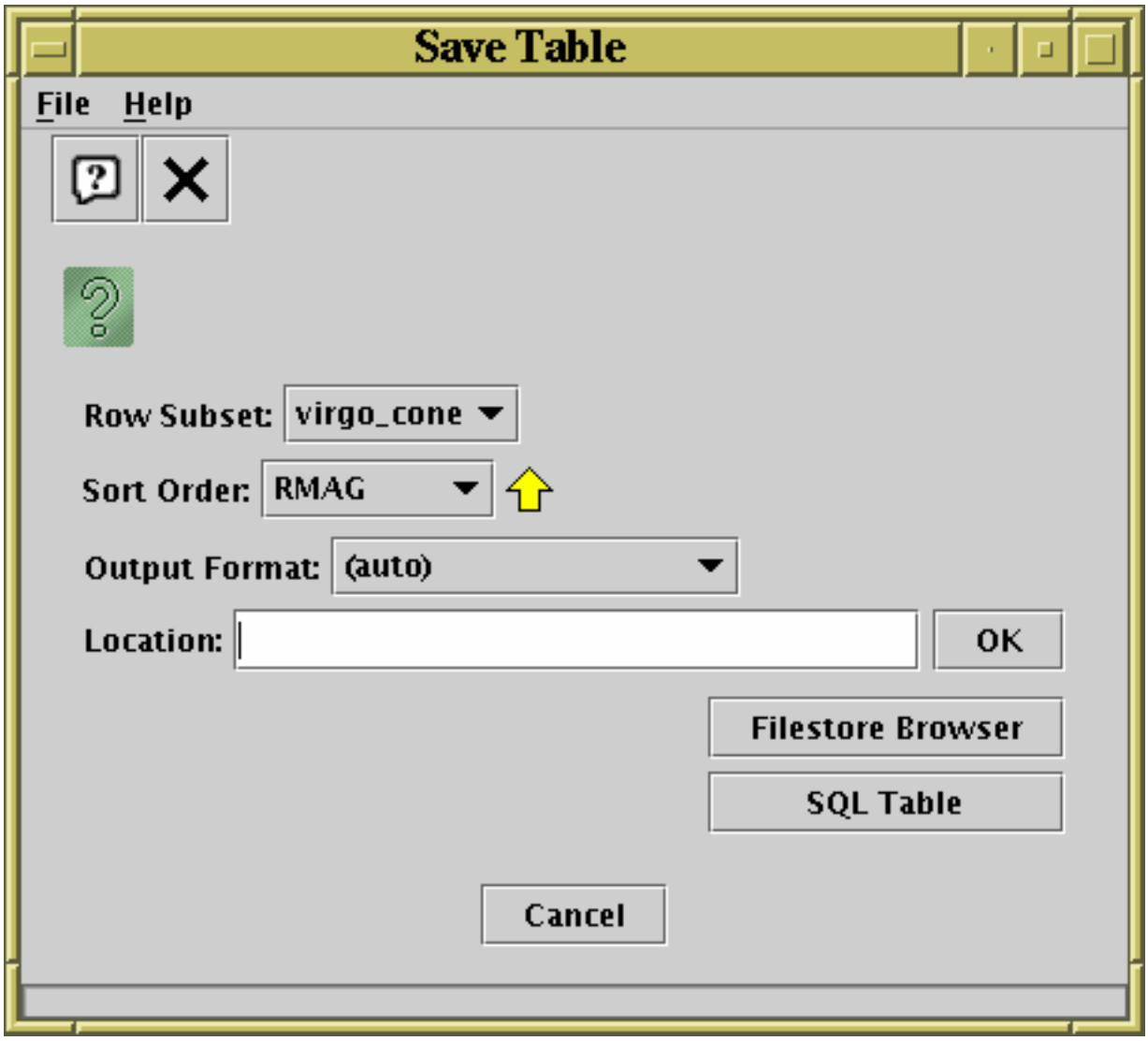
You can also load examples in from the **Examples** menu in the Load Window however. This contains the following options:

Load Example Table
Loads in a single example table.

Browse Demo Data
Pops up a Hierarchy Browser (Appendix A.5.2) 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.6 Save Window



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. Any table in the Control Window's table list can be written at any time; what is written is the Apparent Table corresponding to the currently selected table, which takes into account any modifications you have made to its data or appearance this session. The current **Row Subset** and **Row Order** are displayed in this window as a reminder of what you're about to save; if you modify the values in these selectors you will be modifying the Apparent Table in the usual way.

Any Row Subsets which have been defined on the table in the current session will not be saved themselves, but you can save information about subset membership by creating new boolean columns based on subsets using the "To Column" button ( ) from the Subsets Window.

You can use the **Table Output Format** selector box to pick the format in which the table will be written from one of the supported output formats (Section 4.1.2). There is no default format, and it won't automatically save to the same format it was loaded from, but if you leave it on "(auto)" it will try 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.

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.6.1) as a filename or a URL as described in section Section 4.2
- Use the **Filestore Browser** button to get a browser (Appendix A.6.2) that shows you local and remote files
- Use the **SQL table** button to get the SQL Output Dialogue (Appendix A.6.3)

In some cases, saving the table to 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.

There is no option to compress files on output (though you can of course compress them yourself once they have been written).

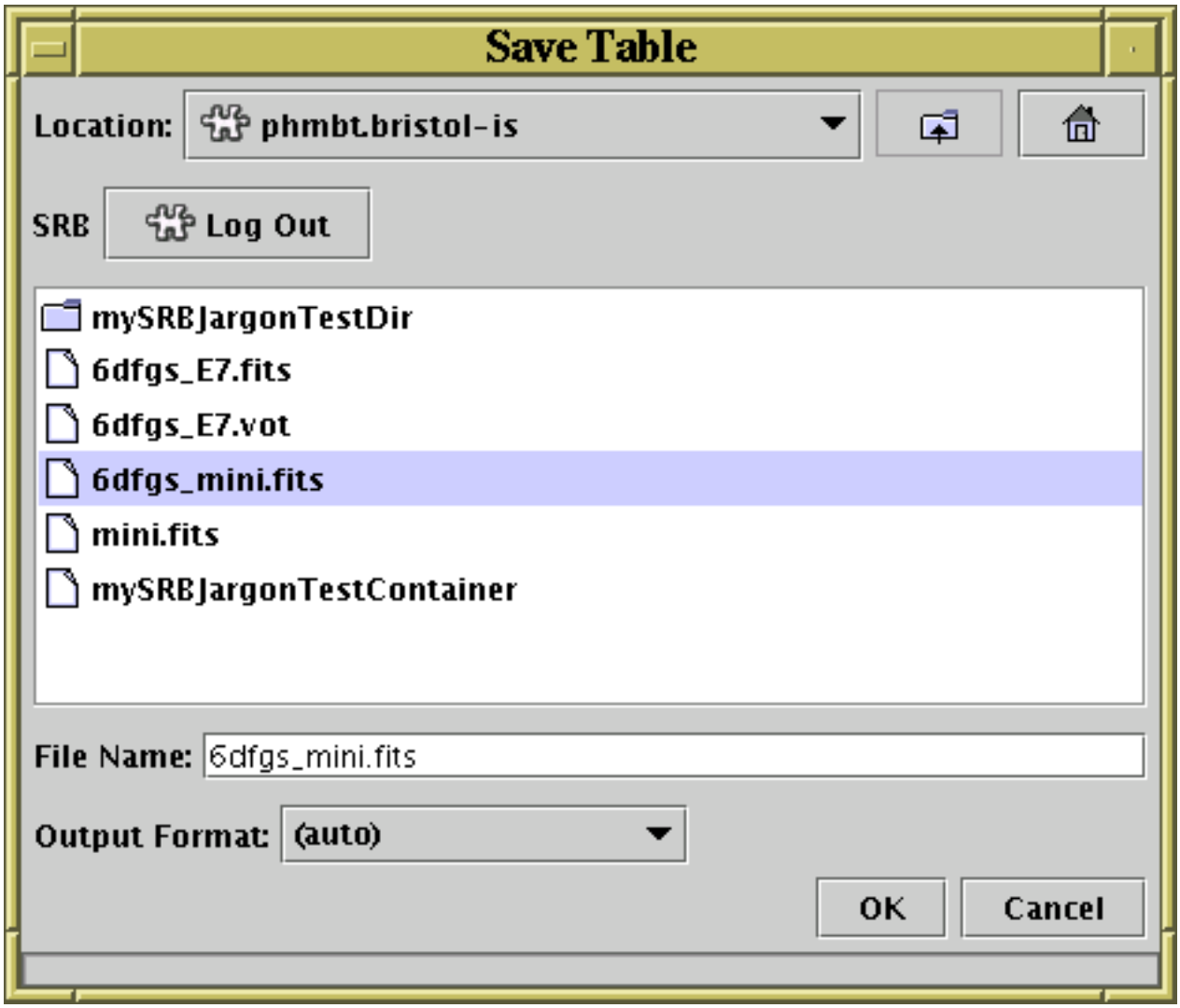
If the table is large, a progress bar indicating how near the save is to completion will appear. It is not advisable to edit the table during a save operation.

In some cases, when saving 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. For instance a WDC table can contain data which represent epoch (date), and this cannot be stored in a FITS table. In this case the table may be written with such columns missing. Some message to this effect may be output in this case.

A.6.1 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.6.2 Filestore Browser



Filestore Browser for table saving

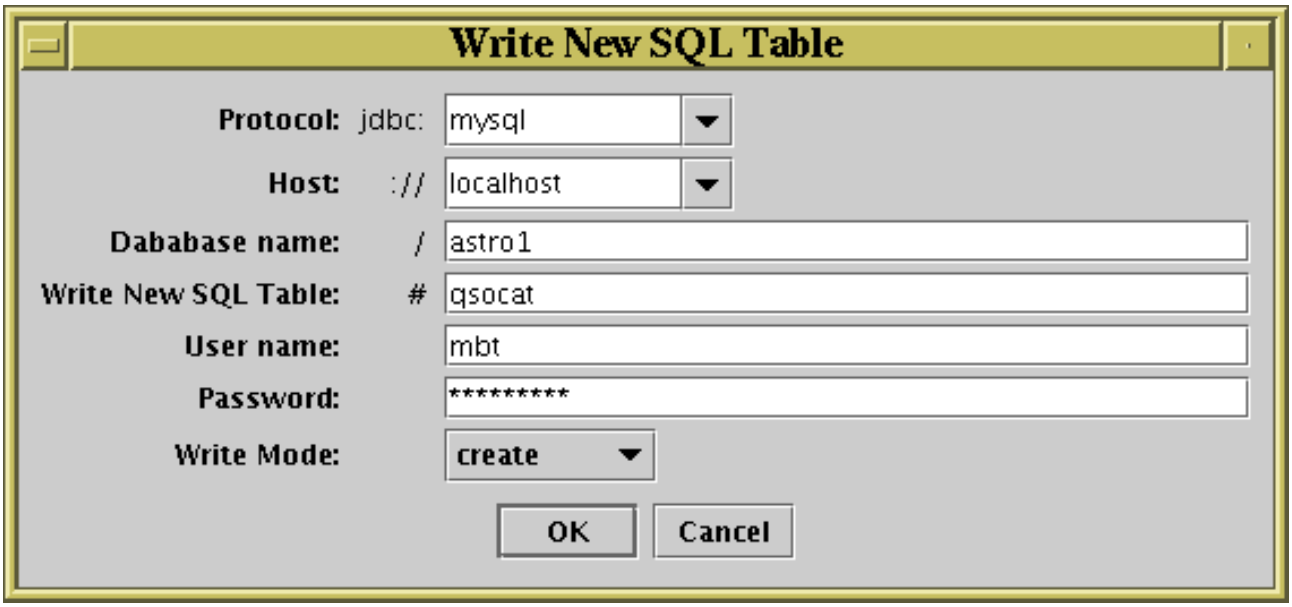
By clicking the **Browse Filestore** button in the **Save Table** window, you can obtain a browser which will display the files in a given directory.

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 9.2.3).

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.5.1) 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 Section 4.1.2 for discussion of output formats).

A.6.3 SQL Output Dialogue



SQL table writing dialogue

If you want to write a table to an SQL database (Section 4.1.2.8), 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.

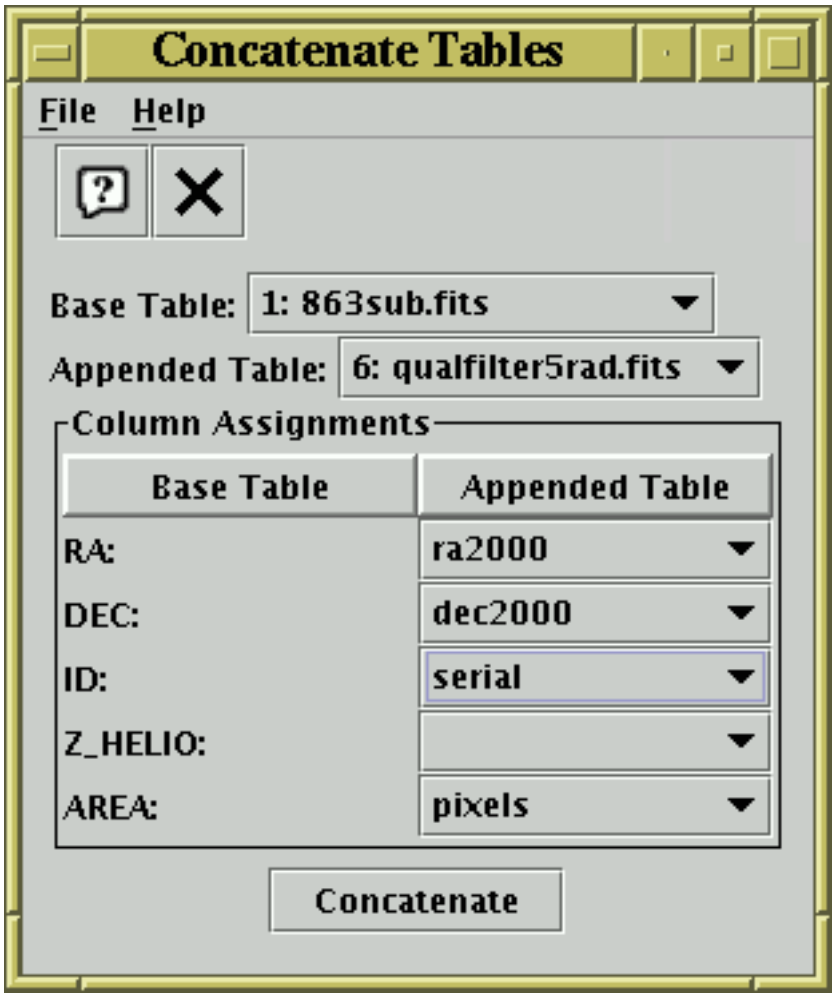
dropcreate
A new table is created for the output. If one with the given name already exists, it is dropped first.

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 9.3). If you don't take these steps, this dialogue may be inaccessible.

A.7 Other Windows

A.7.1 Concatenation Window



Concatenation Window

The Concatenation Window allows you to join two tables together top-to-bottom. It can be obtained using the **Concatenate Tables** button ( ) in the Control Window toolbar or Joins menu.

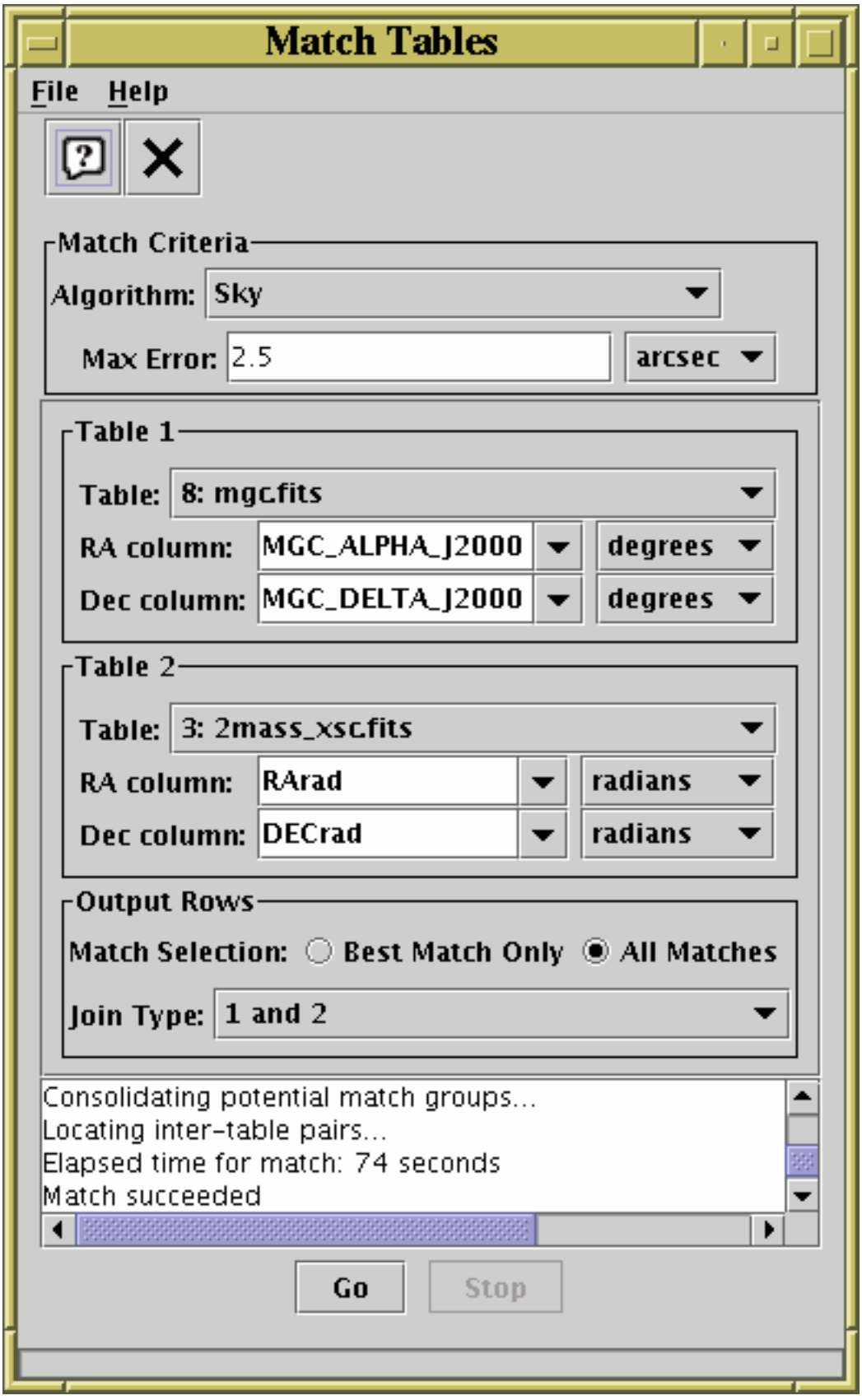
When two windows are concatenated all the rows of the first ("base") table are followed by all the rows of the second ("appended") table. The result is a new table which has a number of rows equal to the sum of the two it has been made from. The columns in the resulting table are the same as those of 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 which column in the appended table corresponds to it. You may select a blank for this, in which case the column in question will have all null entries in the resulting table. 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. Only suitable columns are available for choosing from these column selectors; in most cases this means numeric ones.

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.7.2 Pair Match Window



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.7.2.3.

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).

The following sections describe some of these components in more detail.

A.7.2.1 Match Criteria

The match criteria box allows you to specify what counts as a match between two rows. The selection you make in this box will determine which columns you have to fill in for the table(s) being matched in the rest of the window. In most cases what you are selecting here is the coordinate space in which rows will be compared against each other, and a numerical value or values to 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 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) can be given for each row in the input tables, rather than just a single value for the whole match. You need to specify a single Max Error value, which gives the global maximum separation applying to all matches, and for each of the input tables, 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 smaller than *both* the Max Error value *and* the sum of the two Error values for the corresponding rows. If either of the per-row Error values is blank, then any separation up to the Max Error is considered to match. According to these rules, you might decide to set the Max Error to an arbitarily large number so that only the sum of per-row Errors will determine the actual match criteria. However please *don't* do this, since the Max Error also functions as a tuning parameter for the matching algorithm, and ought to be reasonably close to the actual maximum acceptable separation - if necessary use the Statistics Window to determine the actual maximum uncertainty.

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 along the line of sight 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.

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 radius for each of these dimensions. Points P1 and P2 are considered to match if P2 falls within the ellipsoid defined by the error radii 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?

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.

Sky + XY

Comparison of positions on the celestial sphere with an additional 2-d anisotropic Cartesian constraint. 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 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 and redshifts.

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.

A.7.2.2 Column Selection Boxes

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 6.

A.7.2.3 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 (somewhat interlinked) choices to make about which rows are produced.

The **Match Selection** selector allows you to choose what happens when a given row in one table can be matched by more than one row in the other table. There are two choices:

Best Match Only
Only the best match is chosen, and the other correct but inferior matches are ignored. The best match is usually the "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.

All Matches
Any pairs which meet the match criteria are retained in the output table. This means that you may have data from some of the same input rows appearing more than once in the output.

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.

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.

All from 2
As for **All from 1** but the other way round.

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**.

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**.

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 you choose the **All Matches** option and some of the rows match more than once, 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
```

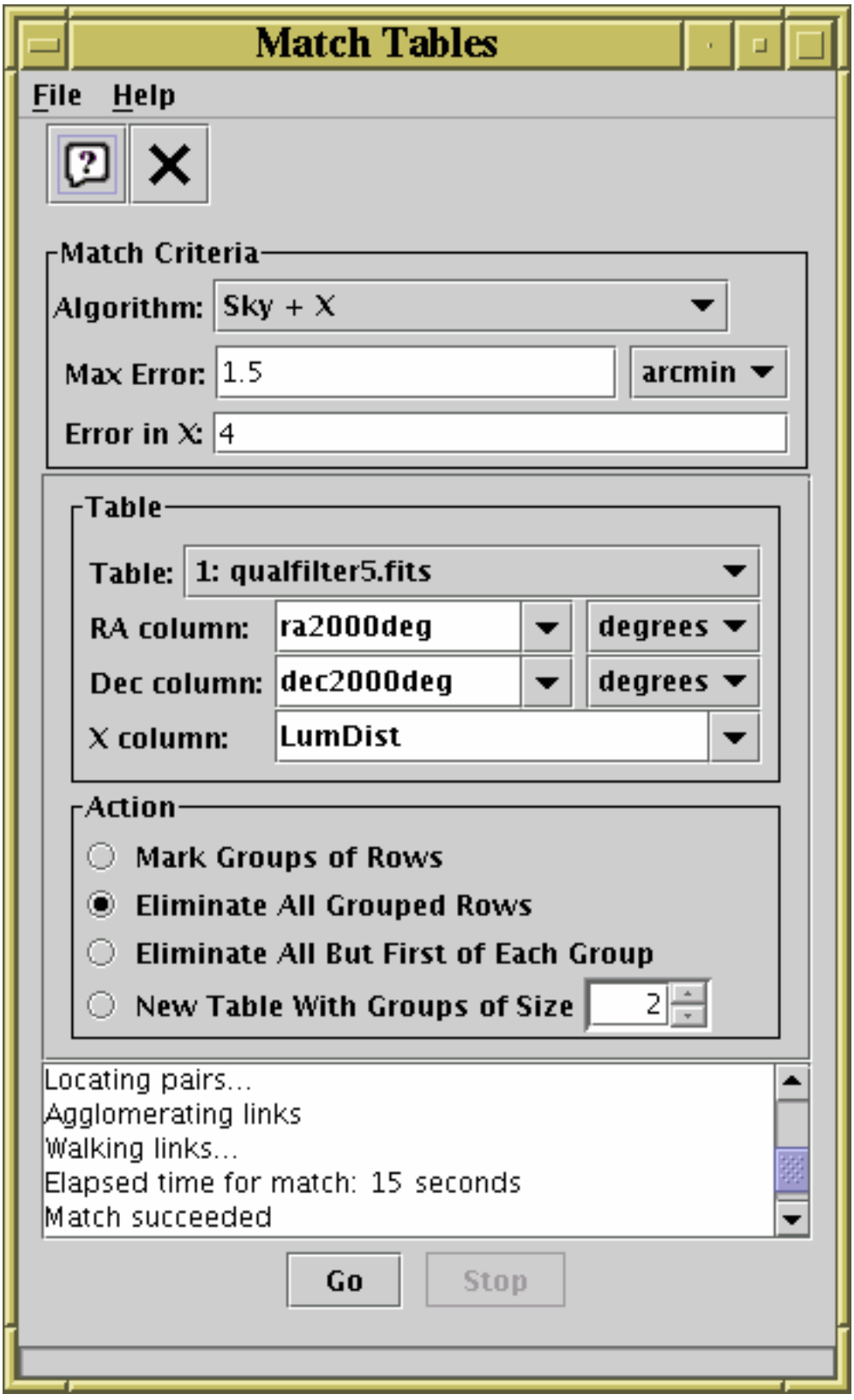
using **All from 1** would give you this:

```
    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.7.3 Internal Match Window



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** ( ) button in the Control Window toolbar or **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 (the same as for pair matching).

Column Selection box
Allows you to select which table to operate on and which columns supply the matching coordinates (the same as for pair matching).

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).

A.7.3.1 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.

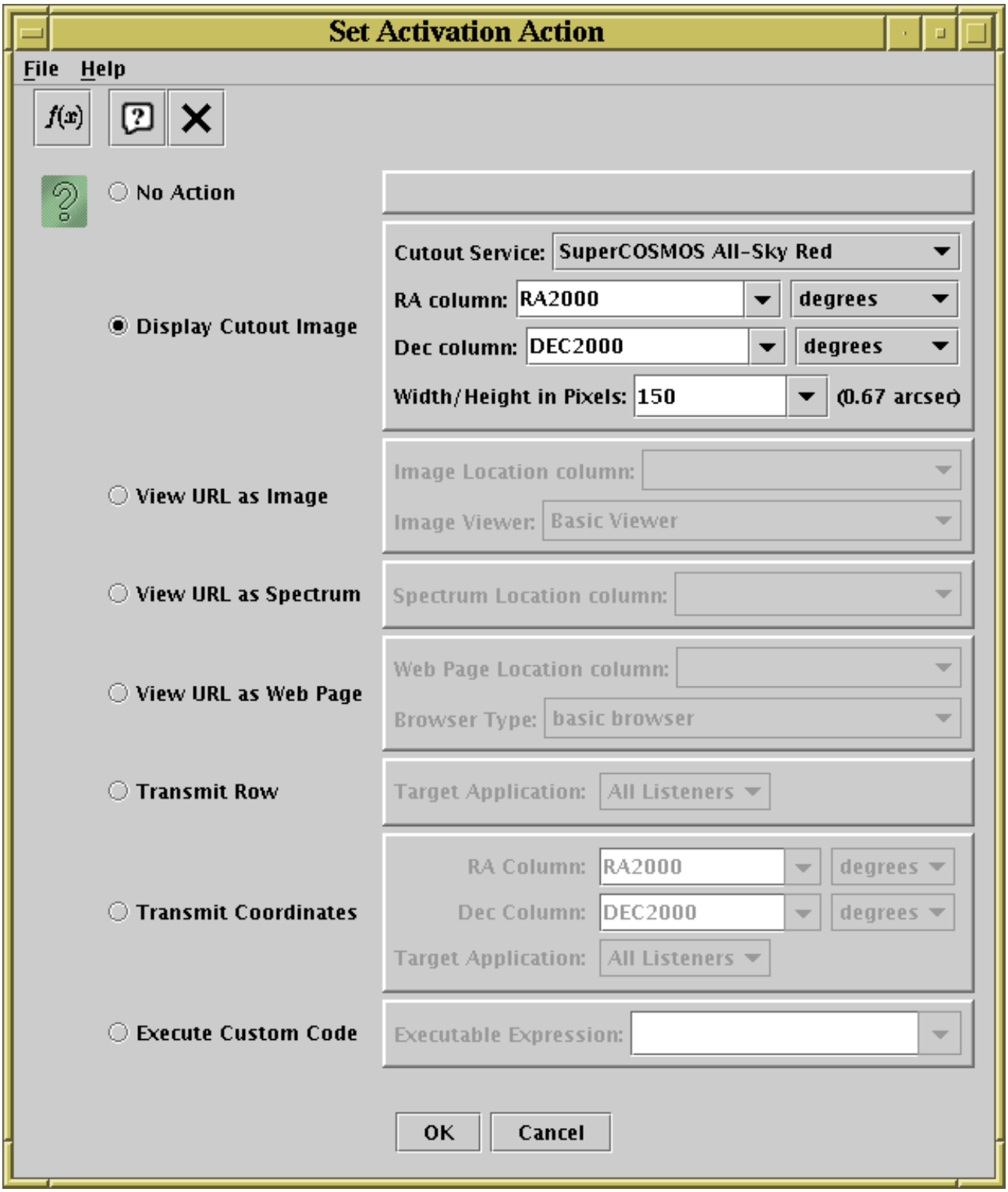
Eliminate All Grouped Rows
The result is a new table containing only "single" rows, that is ones which don't match any other rows in the table according to the match criteria. Any rows which match are thrown out.

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.7.4 Activation Window



Activation Window

The Activation Window allows you to configure an action to perform when a table row is activated (Section 7) by clicking on a row in the Data Window or a point in the Plot Window. It can be obtained by clicking the **Activation Action** selector at the bottom of the properties panel in the Control Window.

You have various options for how to define the action. On the left of the window is a list of options; you have to choose one of these to determine what kind of action will take place. When you click on one of these options the corresponding controls on the right hand side will become enabled: use these to select the details of the action and then click the **OK** button so that subsequent activation events will cause the action you have defined (or **Cancel** so that they won't). When you click OK the **Activation Action** in the control window will indicate the action you have configured.

The available options are as follows:

No Action
If this is selected, no special action will take place when a row is activated. This is the default.

Display Cutout Image
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. You need to select the columns in your table which represent Right Ascension and Declination, including the units in which they are entered in the table (TOPCAT may be able to guess some or all of this information based on column names, UCDs and unit values, in which case it will enter its guesses in the selectors for you to accept or change). You also need to select the size in pixels of the image you want to see and the name of the survey which will supply the image from one of the listed ones:

- 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)

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 an image viewer window.

View URL as Image
This option is suitable if one of the columns in your table gives the location (filename or URL) of an image file. The image may be in FITS, GIF, JPEG or PNG format, optionally compressed using gzip, Unix compress, or bzip2 format. Select the column which contains the location, and activating a row will pop up an image viewer to display it. The **Image Viewer** selector allows you to choose one of the options in Appendix A.7.4.1 or, if you are registered with a SAMP or PLASTIC hub, any SAMP/PLASTIC image viewer which is also registered - see Section 8. Unfortunately you cannot currently select image or spectrum viewers for the other options in this window at present.

View URL as Spectrum
This option is suitable if one of the columns in your table gives the location (filename or URL) of a spectrum file. The spectrum may be in FITS or text format, optionally compressed. Select the column which contains the location, and activating a row will try to pop up a spectrum viewer to display it. Note this may not be possible if some components are not installed - see Appendix A.7.4.2 for information about spectrum viewers.

View URL as Web Page
This option is suitable if one of the columns in your table gives the location (filename or URL) of a web page; this should normally be in HTML or plain text, but depending on what browser you use other kinds of document may be supported. Select the column which contains the location and the browser which you would like to use for display, and activating a row will try to pop up a browser window to display it. See Appendix A.7.4.3 for more information about

browsers.

Transmit Row
This option causes the row number of an activated row to be transmitted to one or more other applications using an appropriate SAMP or PLASTIC message. This will only work if TOPCAT is registered with a SAMP/PLASTIC hub, and so are one or more other applications which understand that message, and the other application(s) know about the table in question, for instance because it has previously been sent to/from TOPCAT. You may specify that either a single application, or all appropriate ones, will receive the messages. See Section 8 for more explanation.

Transmit Coordinates
This option causes the sky position of an activated row to be transmitted to one or more other applications using an appropriate SAMP or PLASTIC message. This will only work if TOPCAT is registered with a SAMP/PLASTIC hub, and so are one or more other applications which understand that message. An example might be a sky viewing application such as Aladin which can point to a particular region of sky whenever you activate a point. You need to specify the columns which represent (J2000) Right Ascension and Declination, and optionally a particular listener to receive the messages (otherwise all registered ones will). See Section 8 for more explanation.

Execute Custom Code
This option must be used if none of the others (which are fairly restrictive) do what you want. It is highly flexible, but not so easy to use. What you have to do is to write an expression following the rules in Section 6 involving some of the column names which will be invoked when a row is activated. This expression will typically have the effect of popping up an image or a spectrum in a viewer, but, especially if you link in your own functions (see Section 6.9) it can do pretty much anything.

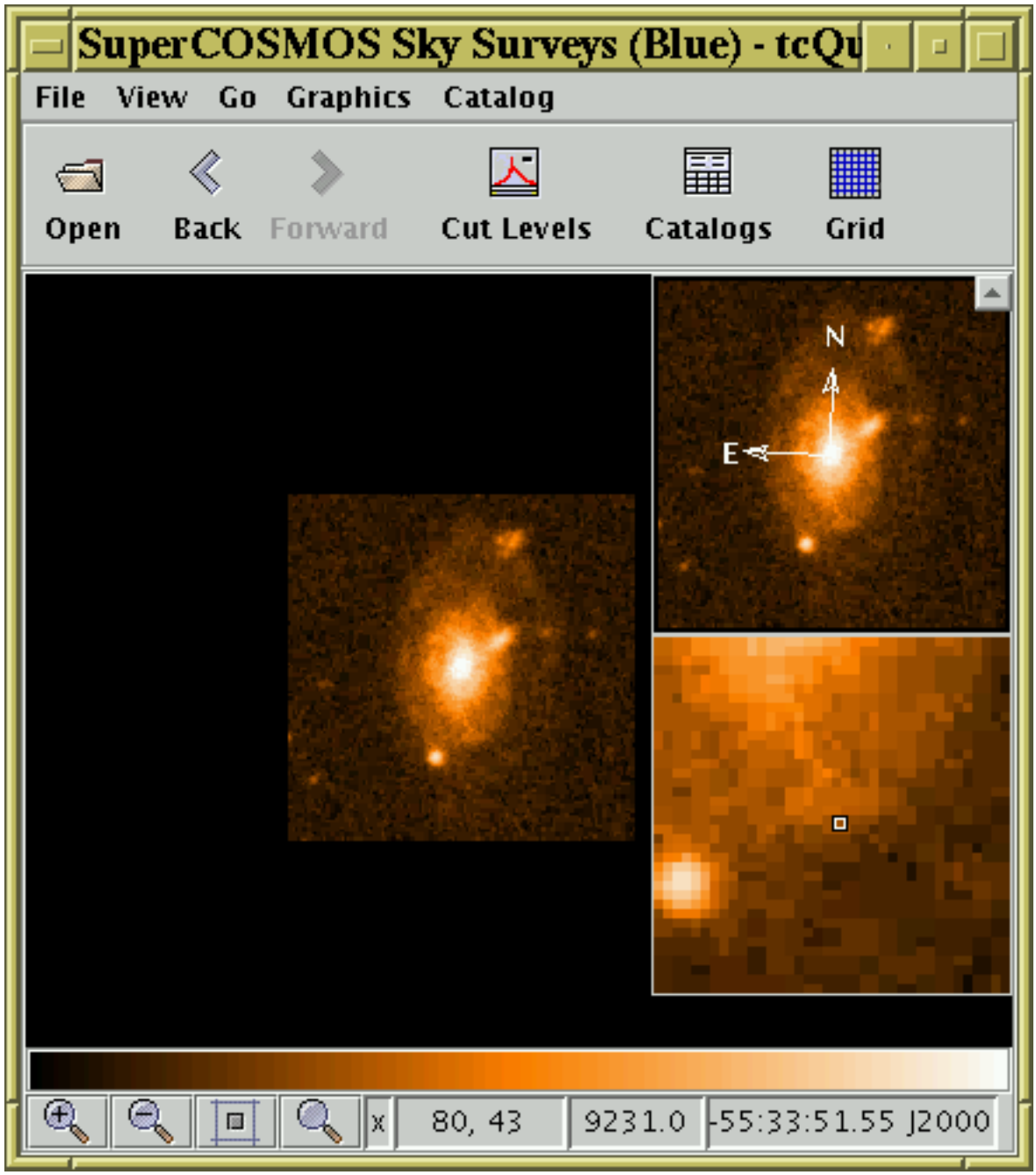
Functions which are expected to be useful for activation actions are described in Appendix B.2 and include some general-purpose ones (`displayImage` and `displaySpectrum` to display an image or spectrum in an external viewer) as well as a few which are relevant to particular survey data, for instance the `spectra2QZ()` function, which will pop up a spectrum viewer displaying all the spectra related to a given row of 2QZ survey data based on the contents of its NAME column.

As the above list shows, most of the activation actions you can define result in a viewer window of some kind popping up. Exactly what kind of viewer is used depends on how TOPCAT is set up and in some cases on your choices. More details of the viewer programs available are given in the following subsections. If these don't do what you want, you can use the **Execute Custom Code** option, perhaps in conjunction with user-defined functions or the `System exec()` functions described in Appendix B.2, to invoke your own.

A.7.4.1 Image Viewer Applications

If you choose the **Display Cutout Image** or **View URL as Image** option in the Activation Window, then activating a row will display an image in an image viewer.

The default image viewer is SoG (http://www.starlink.ac.uk/sog/), an astronomical image viewer based on JSky, which offers colourmap manipulation, image zooming, graphics overlays, and other features. For this to work JAI, otherwise known as Java Advanced Imaging (http://java.sun.com/products/java-media/jai/) must be installed. JAI is a free component available from Sun, but not a part of the Java 2 Standard Edition by default. In operation, SoG looks like this:



SoG Image Viewer

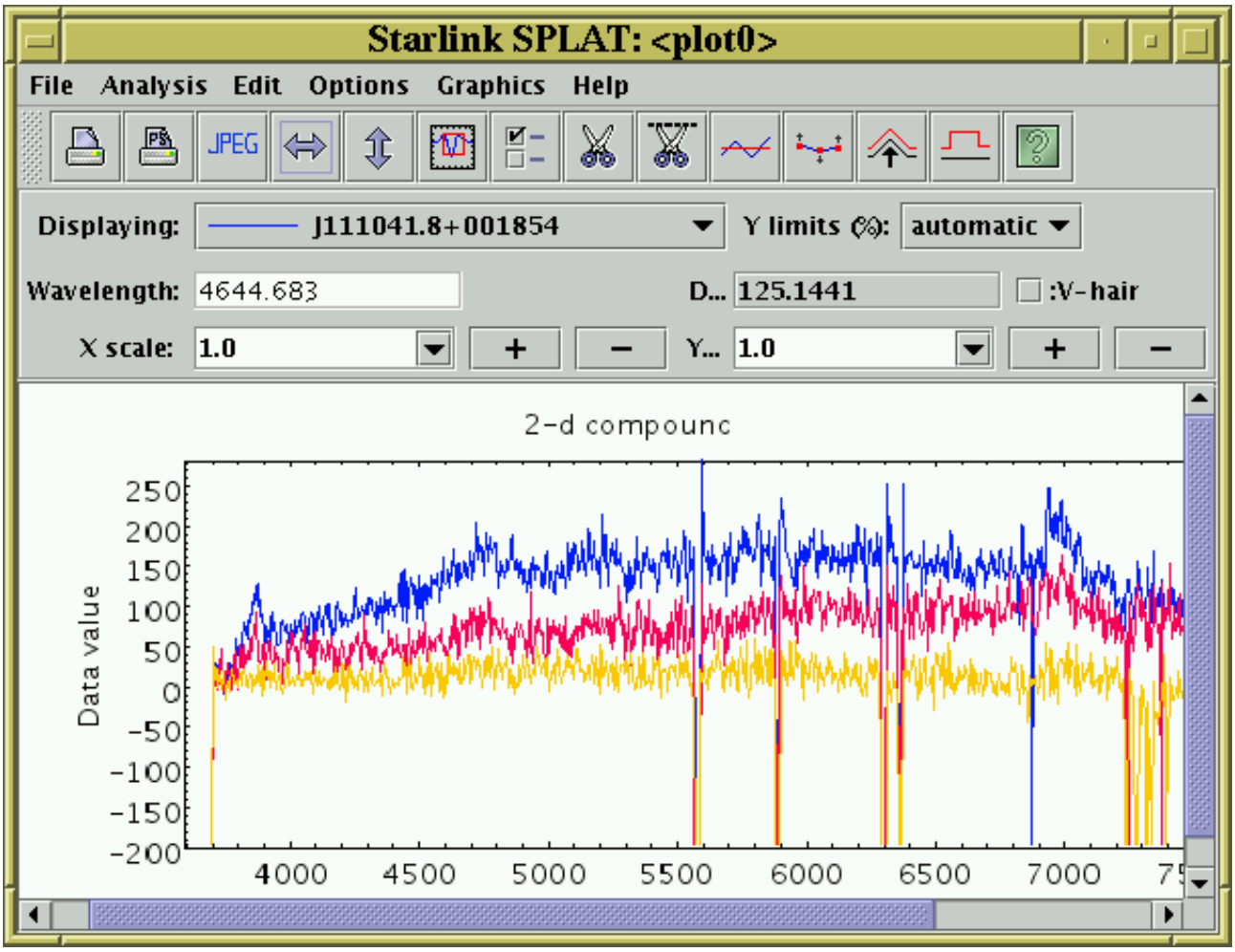
If JAI or the SoG classes themselves are absent, a fallback viewer which just displays the given image in a basic graphics window with no manipulation facilities is used. The fallback image viewer looks like this:

Fallback Image Viewer

A.7.4.2 Spectrum Viewers

If you choose the **View URL as Spectrum** option in the Activation Window, then activating a row will display a spectrum in a spectrum viewer.

The default spectrum viewer is SPLAT (http://www.starlink.ac.uk/splat/), 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. In this case it will not be possible to select the **Display Named Spectrum** item in the Activation Window. An example of SPLAT display of multiple spectra is shown below.



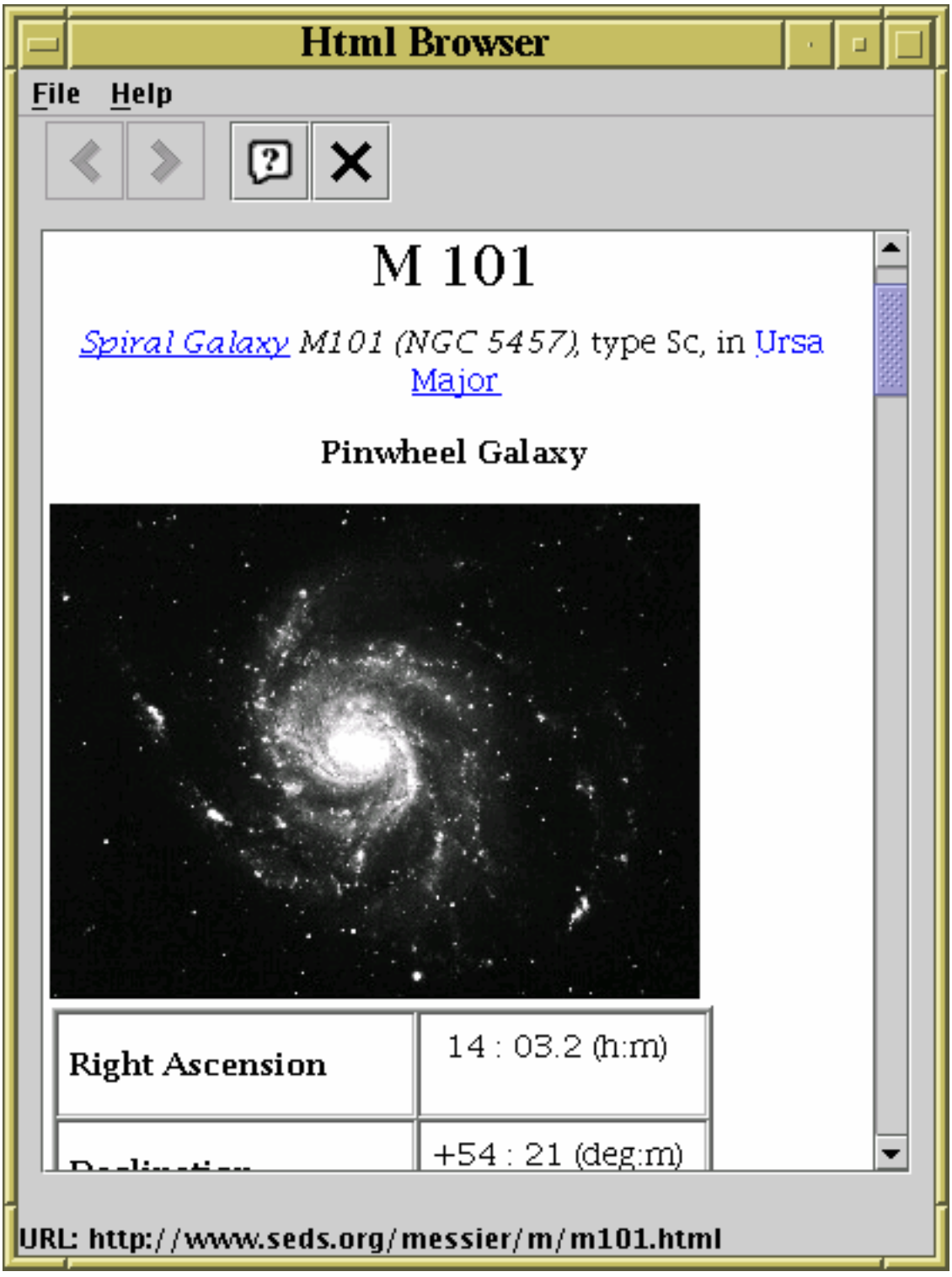
SPLAT Spectrum Viewer

Full documentation for SPLAT is available on-line within the program, or in SUN/243.

A.7.4.3 Web Browsers

If you choose the **View URL as Web Page** option in the **Activation Window**, 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 default **basic browser** option uses a simple browser which can view HTML or plain text pages and has forward and back buttons which work as you'd expect. In many cases this is fine for viewing HTML pages, and it is available regardless of the system that you are running TOPCAT on. It looks like this:



Basic HTML browser

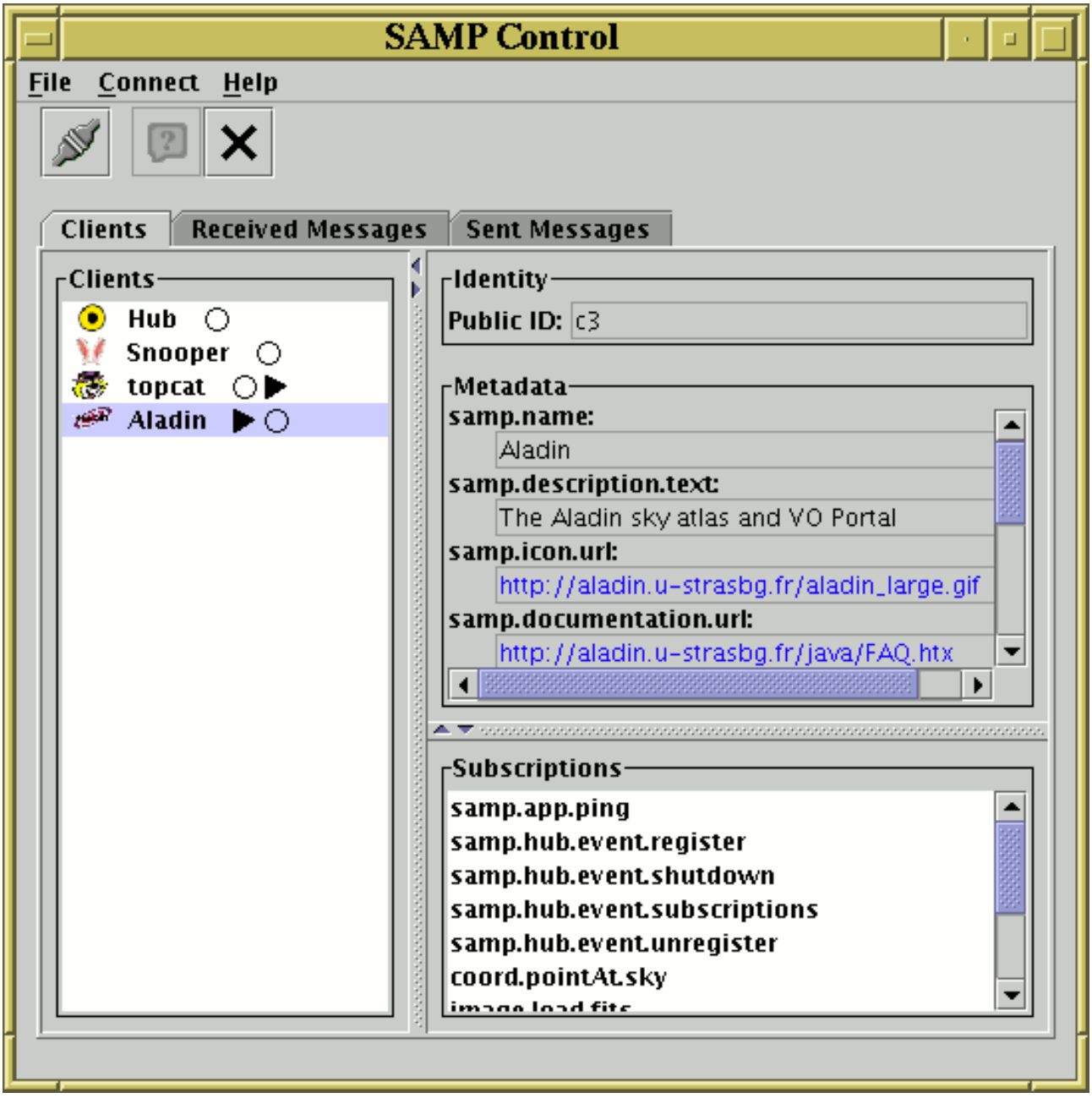
In some circumstances, it's possible to use your normal web browser for web page display instead. The list of browsers currently includes Firefox, Mozilla and Netscape as well as the basic one. Selecting these will generally only work if (1) the browser you select is installed and on your path, (2) you're on some Unix-like operating system, (3) the browser is already running when the action is invoked. In this case, the selected URL should be displayed in an existing browser window rather than opening a new one. Doing it this way has the advantage that your browser can probably display many types of document (perhaps using plugins) as well as HTML.

A.7.5 SAMP Window

The SAMP Window displays the status of SAMP messaging and allows some control over its configuration. SAMP is a messaging protocol which allows TOPCAT to exchange information with other desktop applications - see Section 8 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**:

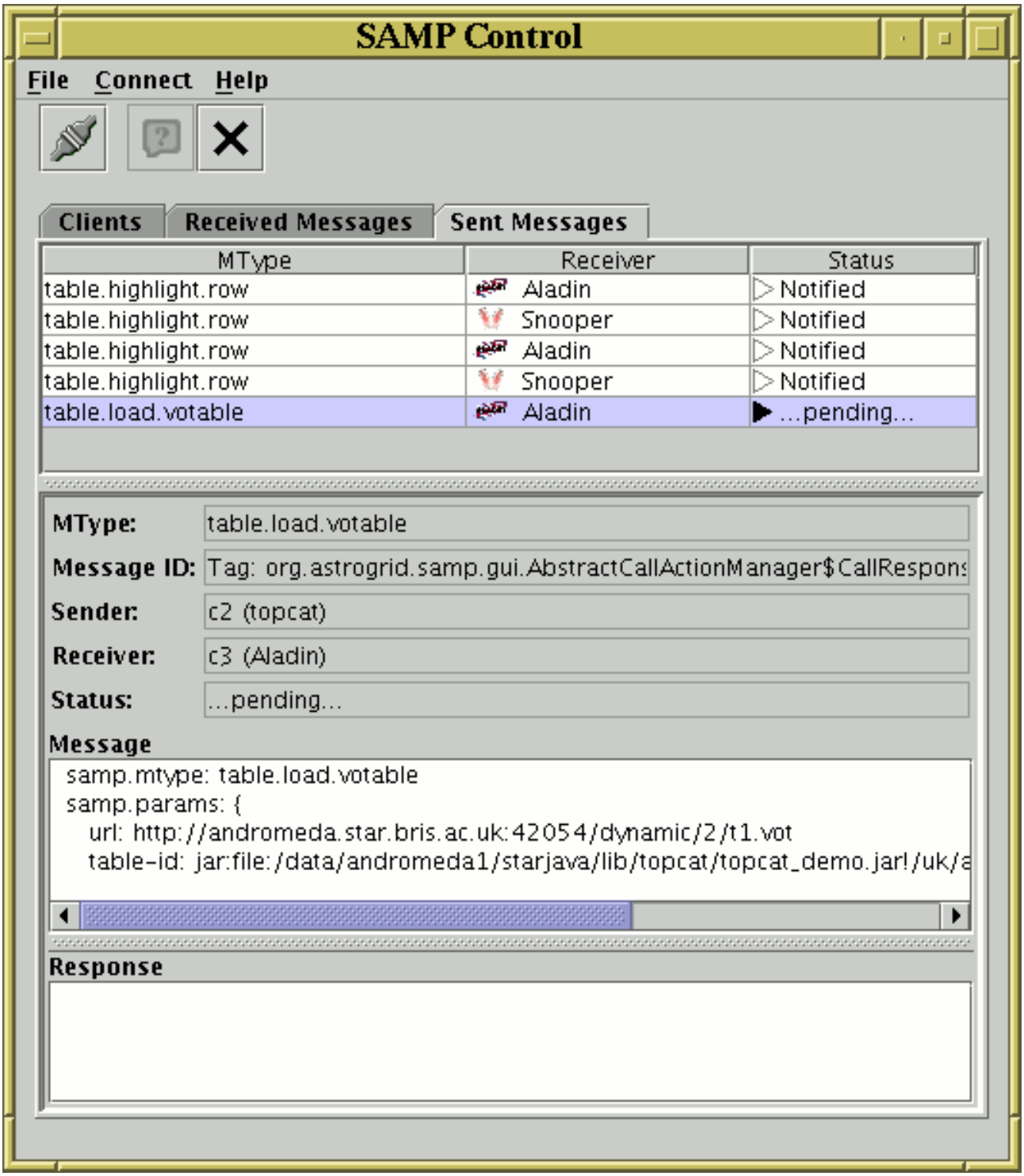


SAMP Window Clients tab

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:



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:

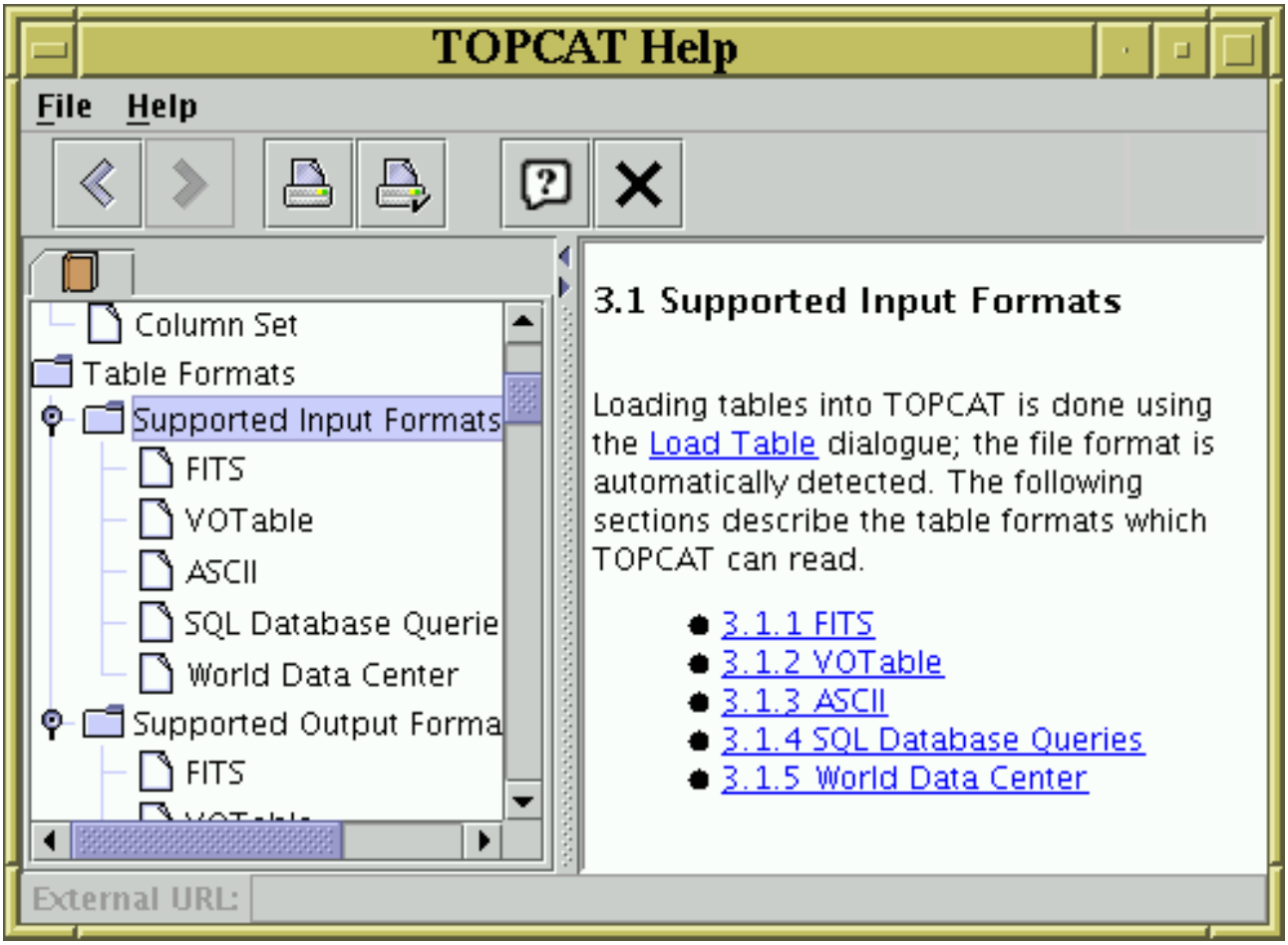
Connect/Disconnect

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.7.6 Help Window



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 left hand side gives a hierarchical view of the available help topics, and the panel on the right hand side displays the help text itself. The bar in between the two 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).

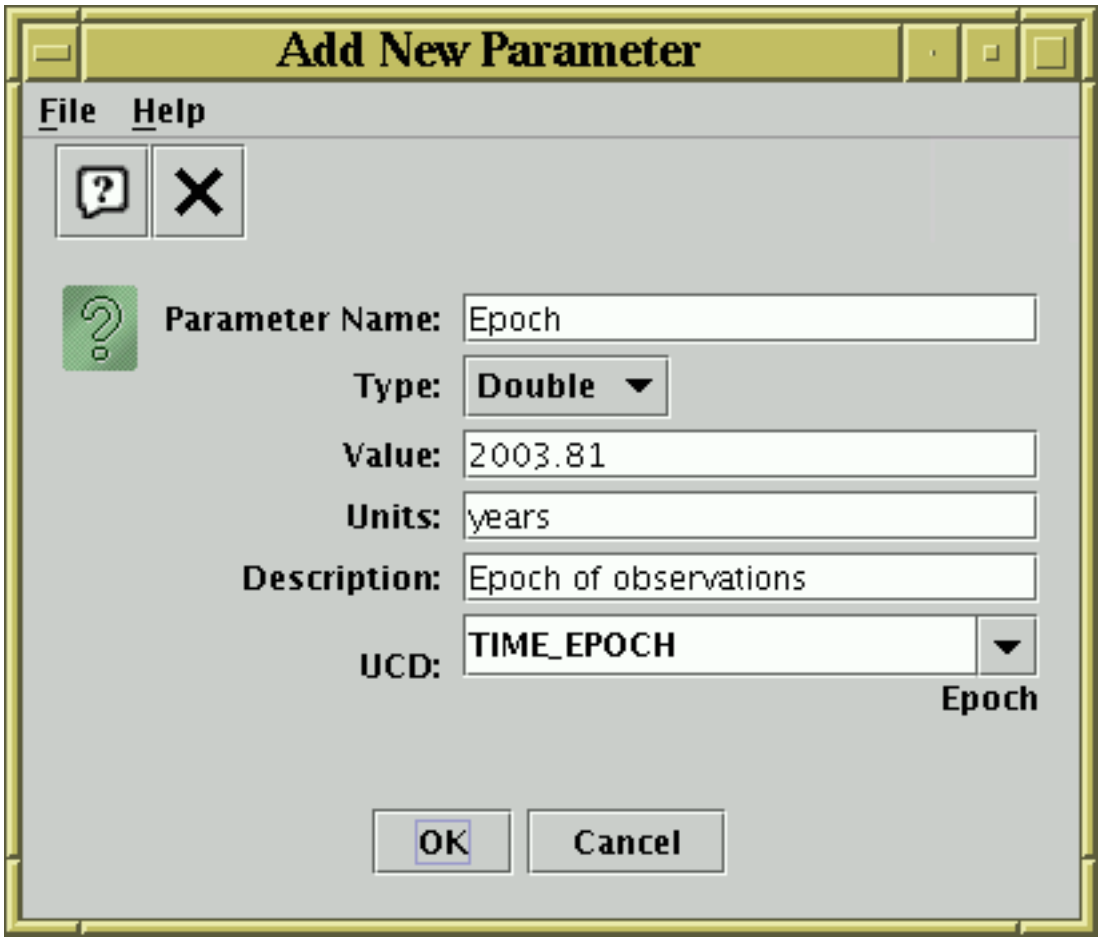
Page Setup

Pops up a dialogue to do printer setup.

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.7.7 New Parameter Window

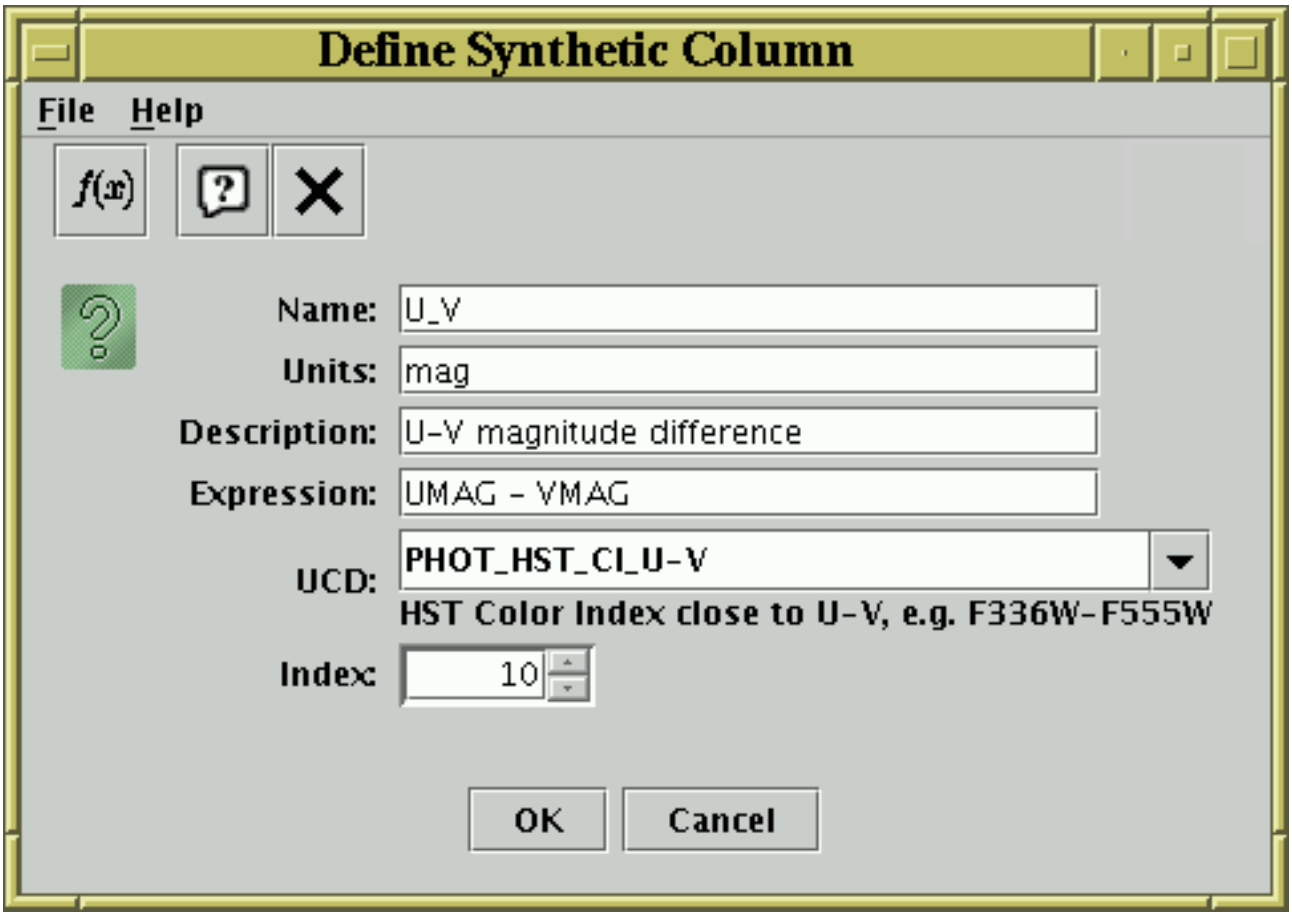


New Parameter dialogue window

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.7.8 Synthetic Column Window



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 6, 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.

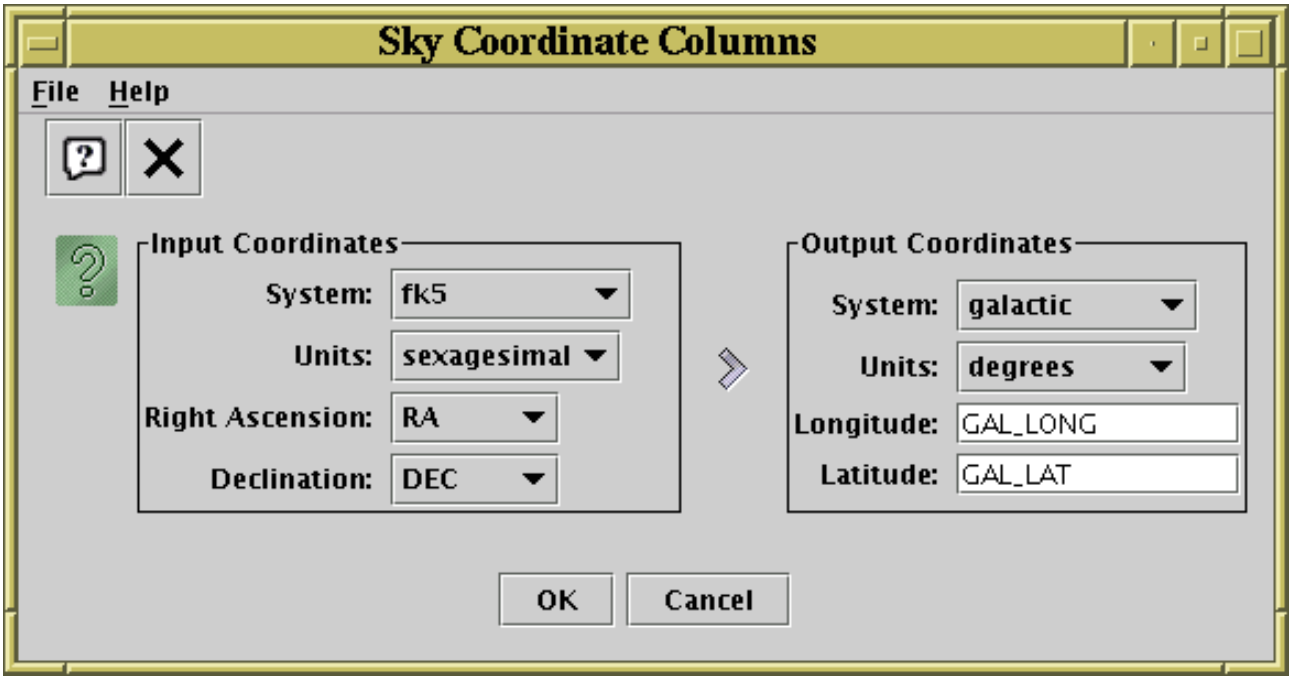
Of these, the **Expression** is the only one which must be filled in.

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.7.9 Sky Coordinates Window



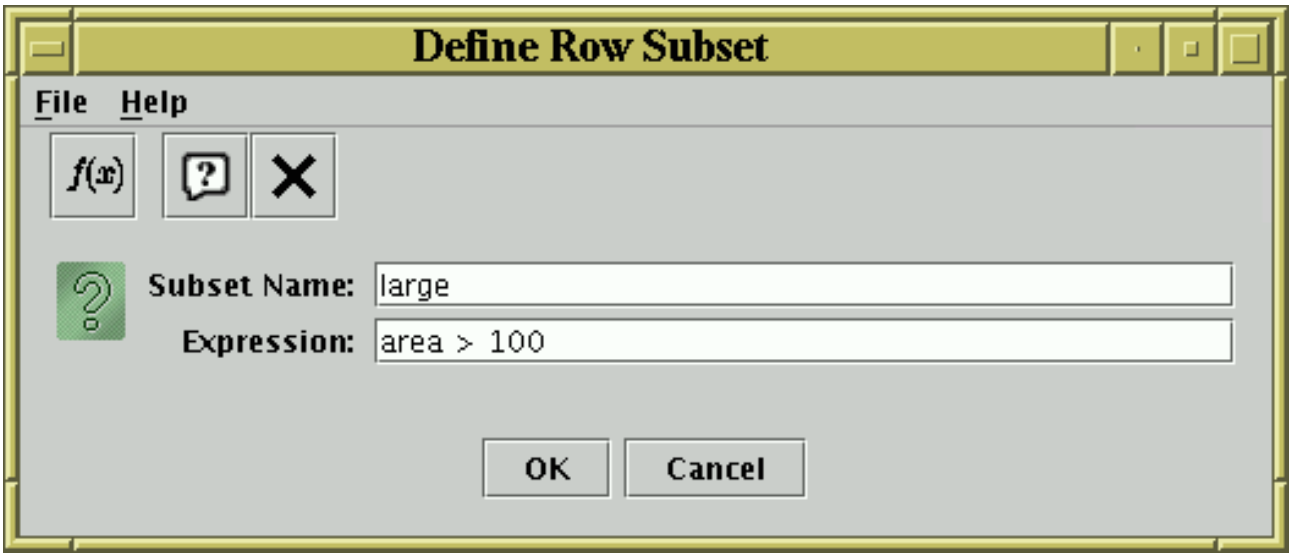
Sky Coordinates Window

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 new columns will be appended at the right of the table.

A.7.10 Algebraic Subset Window



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:

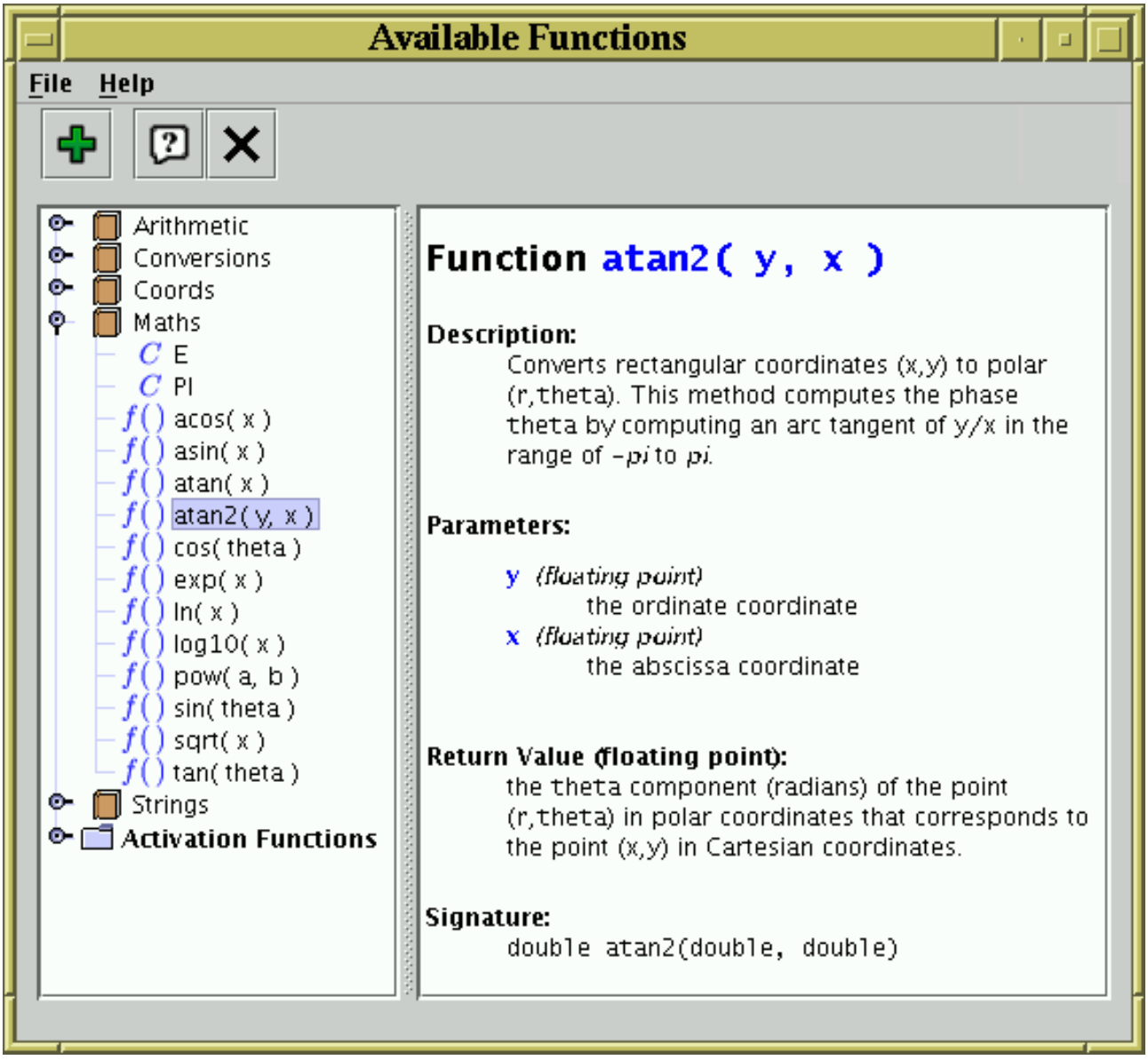
- Different from other columns even apart from upper/lower case distinctions
- In the form of a java identifier (starts with a letter, continues with alphanumerics including underscore)
- Not too long

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 6, 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** ($f(x)$) 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.7.11 Available Functions Window



Available Functions Window

This window displays all the functions (Java methods) which are available for use when writing algebraic expressions (Section 6). 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** ($f(x)$) 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 7). 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.

Function

A function that you can use in an expression.

Constant

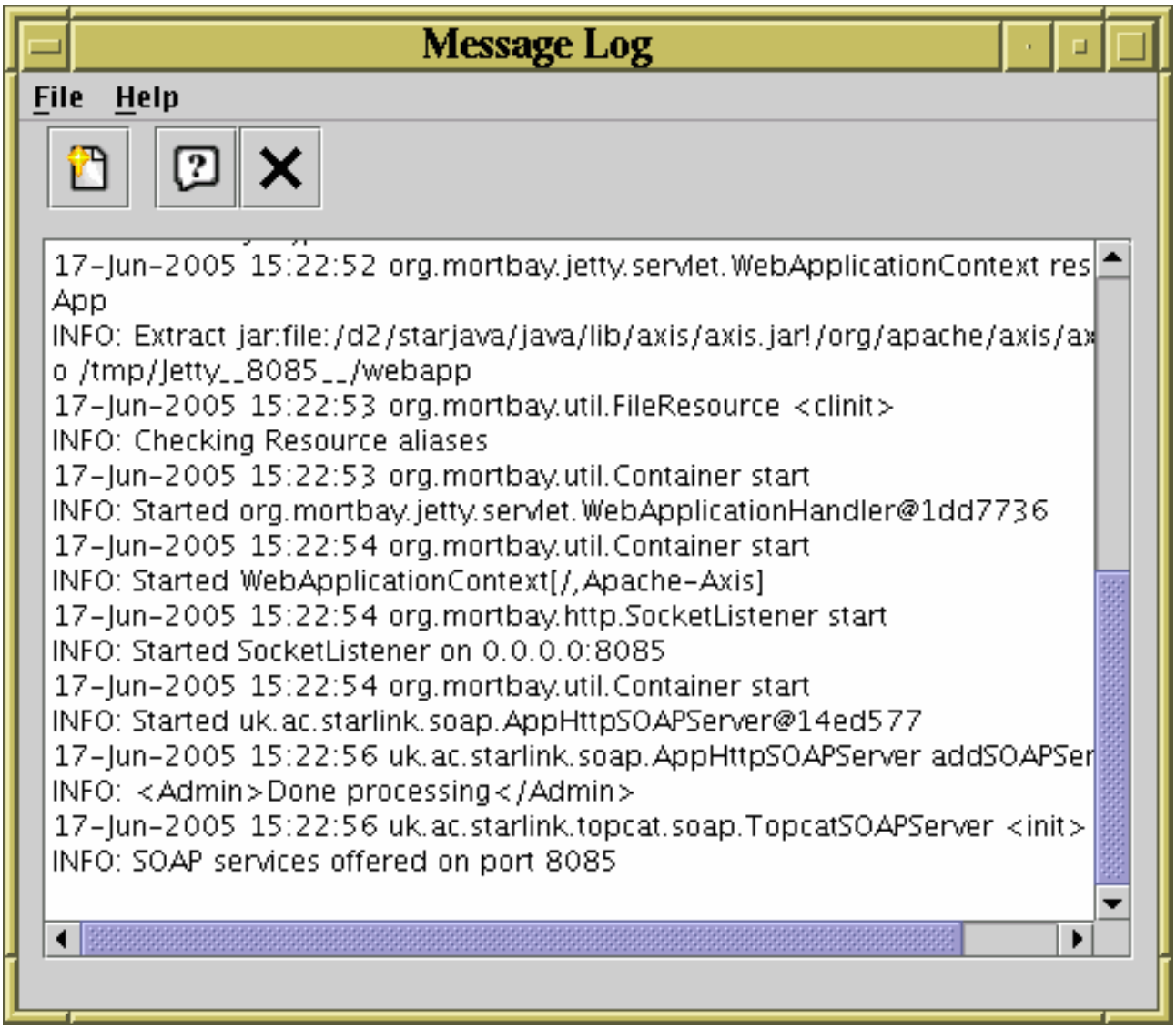
A constant value which you can refer to by name in an expression (as long as it doesn't clash with a column name).

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 `atan2(X_POS,Y_POS)` as the expression for a new table column which gives the angle from the X axis of a point whose position is given by columns with the names X_POS and Y_POS. Examples of a number of these functions are given in Section 6.8.

Using the Add button ( ) you can specify the name of a class to add 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 9.2.1. 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 6.9. Classes added in this way will be visible in the tree, but may not have proper documentation (clicking on them may not reveal a description in the right hand panel).

A.7.12 Log Window



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. Depending on whether the `-verbose` flag has been specified, some or all of these messages may have been written to console as well (if there is a console - this depends on how you have invoked TOPCAT). Under some circumstances, messages way back in the list may not be displayed.

To clear the display of all the existing messages you can use the **Clear Log** button ( ).

The messages displayed here are those written through Java's logging system - in general they are intended for debugging purposes and not for users to read, but if something unexpected is happening, or if you are filing a bug report, it may provide some clues about what's going on. 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 6). 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 7).

Note that although all the available functions are listed here with short descriptions, their full explanation, including parameter descriptions and examples, is only available from the Available Functions Window (Appendix A.7.11), 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. More complete documentation of them is available from within TOPCAT in the Available Functions Window.

B.1.1 Coords

Functions for angle transformations and manipulations. In particular, methods for translating between radians and HH:MM:SS.S or DDD:MM:SS.S type sexagesimal representations are provided.

DEGREE
The size of one degree in radians.

HOUR
The size of one hour of right ascension in radians.

ARC_MINUTE
The size of one arcminute in radians.

ARC_SECOND
The size of one arcsecond in radians.

radiansToDms(rad)
Converts an angle in radians to a formatted degrees:minutes:seconds string. No fractional part of the seconds field is given.

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.

radiansToHms(rad)
Converts an angle in radians to a formatted hours:minutes:seconds string. No fractional part of the seconds field is given.

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.

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.

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.

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.

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).

skyDistance(ra1, dec1, ra2, dec2)
Calculates the separation (distance around a great circle) of two points on the sky.

skyDistanceDegrees(ra1, dec1, ra2, dec2)
Calculates the separation (distance around a great circle) of two points on the sky in degrees.

hoursToRadians(hours)
Converts hours to radians.

degreesToRadians(deg)
Converts degrees to radians.

radiansToDegrees(rad)
Converts radians to degrees.

raFK4toFK5(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.

decFK4toFK5(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.

raFK5toFK4(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.

decFK5toFK4(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.

raFK4toFK5(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.

decFK4toFK5(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.

raFK5toFK4(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.

decFK5toFK4(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.

B.1.2 Arithmetic

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

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.

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.

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.

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.

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.

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.

max(a, b)
Returns the greater of two integer values. If the arguments have the same value, the result is that same value.

max(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.

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.

min(a, b)
Returns the smaller of two integer values. If the arguments have the same value, the result is that same value.

min(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.

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.

B.1.3 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 (http://www.astro.utoronto.ca/~patton/astro/mags.html (http://www.astro.utoronto.ca/~patton/astro/mags.html), citing Frei and Gunn 1995).
- Constants `VEGA_AB_*`, for Vega <-> AB magnitude conversions (Blanton et al., Astronomical Journal *127*, 2562-2578 (2005), eqs.(5)).

JOHNSON_AB_V
Approximate offset between Johnson and AB magnitudes in V band. $V_J \sim= V_{AB} +$`JOHNSON_AB_V`.

JOHNSON_AB_B
Approximate offset between Johnson and AB magnitudes in B band. $B_J \sim= B_{AB} +$`JOHNSON_AB_B`.

JOHNSON_AB_Bj
Approximate offset between Johnson and AB magnitudes in Bj band. $Bj_J \sim= Bj_{AB} +$`JOHNSON_AB_Bj`.

JOHNSON_AB_R
Approximate offset between Johnson and AB magnitudes in R band. $R_J \sim= R_{AB} +$`JOHNSON_AB_R`.

JOHNSON_AB_I
Approximate offset between Johnson and AB magnitudes in I band. $I_J \sim= I_{AB} +$`JOHNSON_AB_I`.

JOHNSON_AB_g
Approximate offset between Johnson and AB magnitudes in g band. $g_J \sim= g_{AB} +$`JOHNSON_AB_g`.

JOHNSON_AB_r
Approximate offset between Johnson and AB magnitudes in r band. $r_J \sim= r_{AB} +$`JOHNSON_AB_r`.

JOHNSON_AB_i
Approximate offset between Johnson and AB magnitudes in i band. $i_J \sim= i_{AB} +$`JOHNSON_AB_i`.

JOHNSON_AB_Rc
Approximate offset between Johnson and AB magnitudes in Rc band. $Rc_J \sim= Rc_{AB} +$`JOHNSON_AB_Rc`.

JOHNSON_AB_Ic
Approximate offset between Johnson and AB magnitudes in Ic band. $Ic_J \sim= Ic_{AB} +$`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} \sim = J_{AB} + $`VEGA_AB_J`.

VEGA_AB_H
Approximate offset between Vega (as in 2MASS) and AB magnitudes in H band. $H_{Vega} \sim = H_{AB} + $`VEGA_AB_H`.

VEGA_AB_K
Approximate offset between Vega (as in 2MASS) and AB magnitudes in K band. $K_{Vega} \sim = K_{AB} + $`VEGA_AB_K`.

abToJansky(magAB)
Converts AB magnitude to flux in Jansky.

$F/Jy = 10^{(23-(AB+48.6)/2.5)}$

janskyToAb(fluxJansky)
Converts flux in Jansky to AB magnitude.

$AB = 2.5*(23-\log_{10}(F/Jy))-48.6$

luminosityToFlux(lumin, dist)
Converts luminosity to flux given a luminosity distance.

$F = lumin/(4 \times Pi \times dist^2)$

fluxToLuminosity(flux, dist)
Converts flux to luminosity given a luminosity distance.

$lumin = (4 \times Pi \times dist^2)\ F$

B.1.4 Arrays

Functions which operate on array-valued cells. You can only use these functions on values which are already 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.

sum(array)
Returns the sum of all the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

mean(array)
Returns the mean of all the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

minimum(array)
Returns the smallest of the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

maximum(array)
Returns the largest of the non-blank elements in the array. If `array` is not a numeric array, `null` is returned.

size(array)
Returns the number of elements in the array. If `array` is not an array, zero is returned.

B.1.5 Tilings

Pixel tiling functions for the celestial sphere.

htmIndex(level, ra, dec)
Gives the HTM (Hierachical Triangular Mesh) pixel index for a given sky position.

healpixNestIndex(k, ra, dec)
Gives the pixel index for a given sky position in the HEALPix NEST scheme.

healpixRingIndex(k, ra, dec)
Gives the pixel index for a given sky position in the HEALPix RING scheme.

healpixK(pixelsize)
Gives the HEALPix resolution parameter suitable for a given pixel size. This `k` value is the logarithm to base 2 of the Nside parameter.

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.

htmLevel(pixelsize)
Gives the HTM `level` parameter suitable for a given pixel size.

htmResolution(level)
Gives the approximate resolution in degrees for a given HTM depth level.

B.1.6 Strings

String manipulation and query functions.

concat(s1, s2)
Concatenates two strings. In most cases the same effect can be achieved by writing `s1+s2`, but blank values can sometimes appear as the string "`null`" if you do it like that.

concat(s1, s2, s3)
Concatenates three strings. In most cases the same effect can be achieved by writing `s1+s2+s3`, but blank values can sometimes appear as the string "`null`" if you do it like that.

concat(s1, s2, s3, s4)
Concatenates four strings. In most cases the same effect can be achieved by writing `s1+s2+s3+s4`, but blank values can sometimes appear as the string "`null`" if you do it like that.

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.

equalsIgnoreCase(s1, s2)
Determines whether two strings are equal apart from possible upper/lower case distinctions.

startsWith(whole, start)
Determines whether a string starts with a certain substring.

endsWith(whole, end)
Determines whether a string ends with a certain substring.

contains(whole, sub)
Determines whether a string contains a given substring.

length(str)
Returns the length of a string in characters.

matches(str, regex)
Tests whether a string matches a given regular expression.

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.

replaceFirst(str, regex, replacement)
Replaces the first occurrence of a regular expression in a string with a different substring value.

replaceAll(str, regex, replacement)
Replaces all occurrences of a regular expression in a string with a different substring value.

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.

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`.

toUpperCase(str)
Returns an uppercased version of a string.

toLowerCase(str)
Returns an uppercased version of a string.

trim(str)
Trims whitespace from both ends of a string.

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.

B.1.7 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.

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.

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://java.sun.com/j2se/1.4.2/docs/api/java/text/DecimalFormat.html) class.

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.

B.1.8 Maths

Standard mathematical and trigonometric functions.

E
Euler's number *e*, the base of natural logarithms.

PI
Pi, the ratio of the circumference of a circle to its diameter.

RANDOM
Evaluates to a random number in the range 0<=x<1. This is different for each cell of the table. The quality of the randomness may not be particularly good.

sin(theta)
Sine of an angle.

cos(theta)
Cosine of an angle.

tan(theta)
Tangent of an angle.

asin(x)
Arc sine of an angle. The result is in the range of -*pi*/2 through *pi*/2.

acos(x)
Arc cosine of an angle. The result is in the range of 0.0 through *pi*.

atan(x)
Arc tangent of an angle. The result is in the range of -*pi*/2 through *pi*/2.

exp(x)
Euler's number *e* raised to a power.

log10(x)
Logarithm to base 10.

ln(x)
Natural logarithm.

sqrt(x)
Square root. The result is correctly rounded and positive.

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*.

pow(a, b)
Exponentiation. The result is the value of the first argument raised to the power of the second argument.

sinh(x)
Hyperbolic sine.

cosh(x)
Hyperbolic cosine.

tanh(x)

Hyperbolic tangent.

asinh(x)
Inverse hyperbolic sine.

acosh(x)
Inverse hyperbolic cosine.

atanh(x)
Inverse hyperbolic tangent.

B.1.9 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.

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
- 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`", and "`2112-05-25 16:45Z`".

dateToMjd(year, month, day, hour, min, sec)
Converts a calendar date and time to Modified Julian Date.

dateToMjd(year, month, day)
Converts a calendar date to Modified Julian Date.

decYearToMjd(decYear)
Converts a Decimal Year to a Modified Julian Date.

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`.

mjdToDate(mjd)
Converts a Modified Julian Date value to an ISO 8601-format date string. The output format is `yyyy-mm-dd`.

mjdToTime(mjd)
Converts a Modified Julian Date value to an ISO 8601-format time-only string. The output format is `hh:mm:ss`.

mjdToDecYear(mjd)
Converts a Modified Julian Date to Decimal Year.

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://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html) class. The default output corresponds to the string "`yyyy-MM-dd'T'HH:mm:ss`"

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.

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.

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.

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.

unixMillisToMjd(unixMillis)
Converts from milliseconds since the Unix epoch (1970-01-01T00:00:00) to a modified Julian date value

mjdToUnixMillis(mjd)
Converts from modified Julian date to milliseconds since the Unix epoch (1970-01-01T00:00:00).

B.1.10 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).

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.

MpcToM(distMpc)
Converts from MegaParsecs to metres.

mToMpc(distM)
Converts from metres to MegaParsecs.

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.

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.

comovingDistanceL(z, H0, omegaM, omegaLambda)
Line-of-sight comoving distance.

comovingDistanceT(z, H0, omegaM, omegaLambda)
Transverse comoving distance.

angularDiameterDistance(z, H0, omegaM, omegaLambda)
Angular diameter distance.

luminosityDistance(z, H0, omegaM, omegaLambda)
Luminosity distance.

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.

comovingVolume(z, H0, omegaM, omegaLambda)
Comoving volume. This returns the all-sky total comoving volume out to a given redshift `z`.

B.1.11 Conversions

Functions for converting between strings and numeric values.

toString(value)
Turns a numeric value into a string.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

toDouble(value)
Converts the numeric argument to a double (64-bit signed integer) result.

toHex(value)
Converts the integer argument to hexadecimal form.

fromHex(hexVal)
Converts a string representing a hexadecimal number to its integer value.

B.2 Activation Functions

The following functions can be used only for defining custom Activation Actions (Section 7) - 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). More complete documentation of them is available from within TOPCAT in the Available Functions Window.

B.2.1 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.

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.

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.

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.

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.

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.

B.2.2 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.

B.2.3 Splat

Functions for display of spectra in the external viewer SPLAT.

splat(label, loc)
Displays the resource at a given location as a spectrum in a spectrum viewer program (SPLAT). `label` may be any string which identifies the window for display, so that multiple (sets of) spectra may be displayed in different windows without getting in each others' way. `loc` should be a filename pointing to a spectrum in a format that SPLAT understands (includes FITS, NDF). In some cases, a URL can be used too.

splat2(label, loc1, loc2)
Displays two spectra in the same (SPLAT) viewer. This may be useful to compare two spectra which correspond to the same table row.

splatMulti(label, locs)
Generic routine for displaying multiple spectra simultaneously in the same SPLAT plot.

B.2.4 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.

SSS_BASE_URL
Base URL for SuperCOSMOS image cutout service.

sssCutout(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.

sssCutout(ra, dec)
Displays a cutout image of default size in one of the available bands from the SuperCOSMOS Sky Surveys. Sky coverage is complete.

sssCutoutBlue(ra, dec, pixels)
Displays a cutout image of default size from one of the blue-band surveys from SuperCOSMOS. Sky coverage is complete.

sssCutoutRed(ra, dec, pixels)
Displays a cutout image of default size from one of the red-band surveys from SuperCOSMOS. Sky coverage is complete.

displayUkstB(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).

displayUkstR(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).

displayUkstI(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).

displayEsoR(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).

displayPossE(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).

B.2.5 Mgc

Specialist functions for use with data from the the Millennium Galaxy Survey.

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.

imageMgc(mgc_id)
Displays the postage stamp FITS image for an MGC object in an image viewer.

B.2.6 Output

Functions for simple logging output.

print(str)
Outputs a string value to the user log.

print(num)
Outputs a numeric value to the user log.

B.2.7 System

Executes commands on the local operating system. These are executed as if typed in from the shell, or command line.

exec(cmd, arg1)
Executes an operating system command with one argument.

exec(cmd, arg1, arg2)
Executes an operating system command with two arguments.

exec(cmd, arg1, arg2, arg3)
Executes an operating system command with three arguments.

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).

B.2.8 Image

Functions for display of images in a window. Supported image formats include GIF, JPEG, PNG and FITS, which may be compressed. The SoG program (http://www.starlink.ac.uk/sog/ (http://www.starlink.ac.uk/sog/)) will be used if it is available, otherwise a no-frills image viewer will be used instead.

displayImage(label, location)
Displays the file at the given location in an image viewer.

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/).

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.

spectra2QZ(name, nobs)
Displays all the spectra relating to a 2QZ object in an external viewer (SPLAT).

image2QZ(name)
Displays the postage stamp FITS image for a 2QZ object in an image viewer.

jpeg2QZ(name)
Displays the postage stamp JPEG image for a 2QZ object in an external viewer.

get2qzSubdir(name)
Returns the name of the subdirectory (such as "ra03_04") for a given 2QZ object name (ID).

B.2.10 Sdss

Specialist display functions for use with the Sloane Digital Sky Server.

SDSS_DR2_BASE_URL
Base URL for SkyServer JPEG retrieval service, DR2.

SDSS_BASE_URL
Base URL for SkyServer JPEG retrieval service.

sdssCutout(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.

sdssCutout(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.

sdssCutout(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.

B.2.11 Sog

Functions for display of images in external viewer SOG (http://www.starlink.ac.uk/sog/ (http://www.starlink.ac.uk/sog/)).

sog(label, loc)
Displays the file at a given location as an image in a graphical (SoG) 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 SOG understands (this includes FITS, compressed FITS, and NDFs).

B.2.12 Spectrum

Functions for general display of spectra in a window. Display is currently done using the SPLAT program, if available (http://www.starlink.ac.uk/splat/ (http://www.starlink.ac.uk/splat/)). Recognised spectrum formats include 1-dimensional FITS arrays and NDF files.

displaySpectrum(label, location)
Displays the file at the given location in a spectrum viewer.

displaySpectra(label, locations)
Displays the files at the given locations in a spectrum viewer. Each file represents a single spectrum, but they will be displayed within the same viewer window.

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-graphical counterpart to TOPCAT, providing many of the same facilities (matching, row selection, format conversion 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.). From July 2005 until June 2006, it was supported by grant PP/D002486/1 from the UK's Particle Physics and Astronomy Research Council. Maintenance and development has been funded from July 2006 until December 2007 by the European VOTech project within the UK's AstroGrid, and directly from AstroGrid funding beyond that.

Inspiration for many of TOPCAT's features has been taken from the following pre-existing tools:

- Mirage (Bell labs)
- VOPlot (VO-India)

Apart from the excellent Java 2 Standard Edition itself, the following external libraries provide important parts of TOPCAT's functionality:

- JEL (GNU) for algebraic expression evaluation
- PixTools (Fermilab EAG) for HEALPix-based celestial sphere row matching
- IVOARegistry (NVO) for IVOA registry access
- HTM (Sloan Digital Sky Survey) for HTM-based celestial sphere row matching (now deprecated within TOPCAT)
- Ptplot (Ptolemy) for axis plotting
- EPSGraphics2D (Jibble) for encapsulated postscript output
- GifEncoder (Acme) for GIF output
- IVOA FITS Package (Sloan Digital Sky Survey) for simple (non-SoG) display of FITS images
- BrowserLauncher2 for launching default web browser

The following users, testers and programmers have supplied useful comments (apologies for any missed out):

- Malcolm Currie (Starlink, RAL)
- David Giaretta (Starlink, RAL)
- Clive Page (Leicester)
- Clive Davenhall (Royal Observatory Edinburgh)
- Tim Jenness (JACH)
- Mike Read (ROE)
- Peter Draper (Starlink, Durham)
- Alasdair Allan (Starlink, Exeter)
- Roy Platon (RAL)
- Norman Gray (Starlink, Glasgow)
- Jonathan Irwin (Cambridge)
- Mark Allen (CDS)
- Francois Ochsenbein (CDS)
- Mike Watson (Leicester)
- Bob Mann (Royal Observatory Edinburgh)
- Richard McMahon (IoA)
- Anita Richards (Jodrell Bank)
- Eduardo Gonzalez-Solares (IoA)
- Gerard Lemson (MPE Garching)
- Evanthia Hatziminaoglou (ESO)
- James Price (Bristol)
- Aaron Robotham (Bristol)
- Danny Boxhoorn (Kapteyn, Groningen)
- David Mills (Bristol)
- Douglas Morgan (Bristol)
- Haggis Harris (Bath)
- Shashikiran Ganesh (PRL, India)
- Ivan Zolotukhin (SAI MSU)
- Luke Davies (Bristol)
- Sebastien Derriere (CDS)

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.4.4) 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 8.

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 9.4).

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.4.1.7) 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 9.2.3.

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.4.1.3.

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.4.1.5.

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.4.1.1). 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$` notation (Section 6.3), and both columns and parameters can be referenced by UCD using `ucd$` notation (Section 6.1).

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.5.6). 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.4.1.4) in 2d and 3d scatter plots - use the **Draw Label** (

) 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** (

) 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 ( ) 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 9.2.3.
- 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 8.

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.5.4.

Other enhancements

- Table columns and parameters can now be referenced by Utype using utype$ syntax in a similar way to using UCDs - see Section 6.1.
- 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).