NAME

rdbplt - plot columns in an RDB table

DESCRIPTION

rdbplt reads an RDB format database file and plots two columns against each other. The data may be transformed logarithmically. Independent positive and negative going error bars may also be plotted. Points may be connected, and may also have markers drawn. For more detailed usage instructions, see the later sections.

PARAMETERS

rdbplt uses an IRAF-compatible parameter interface. A template parameter file is in */proj/axaf/simul/lib/uparm/rdbplt.par*.

input

The name of the RDB file to plot. If the filename is the string `stdin`, it reads from the standard input stream.

xcol, **ycol**

The data plotted along the **X** and **Y** axes. It may be a column name, column index designator, or record index designator.

xerr, **yerr**

Either the string `none`, indicating no error bars for that axis are to be plotted, or a comma separated list of column specifications. Spaces may be used to increase readability. See *Specifying the Data* for more info.

axisxfrm *string*

This parameter specifies the transformations to be applied to the data. It is one of: `linlin`, `linlog`, `loglin`, `loglog`.

axisflip *string*

A string of flags (`0` or `1`) indicating whether the axis limits should be flipped (`1`), i.e. whether the axes have increasing coordinate values to the right (for **X**) or up (for **Y**), or to the left (for **X**) or down (for **Y**).

The flags are ordered as **X** and **Y**. For example, `01`.

autoscale *string*

A string of flags (`0` or `1`) indicating which limits are to be autoscaled (`1`). The flags are ordered as **xmin**, **xmax**, **ymin**, and **ymax**. For example, `0101`.

connect `yes`|no|*line style(s)*

If `no`, the input data are not connected by a line. If `yes`, the input data are connected by a line. It may be a comma separated list of line specifications, the string `no` (indicating no lines are to be drawn) or the string `yes` (indcating lines are to be drawn and their styles automatically selected). Spaces may be used to increase readability.

marker *list of markers*/*column name*

A comma separated list of marker specifications, or `none`, indicating that no markers should be drawn. See *Markers and Lines*. If there are more data sets than markers, new marker types are generated automatically. Spaces may be used to increase readability.

color *list of colors*

This is a comma separated list of colors for the data sets. Colors may be one of `black`, `blue`, `bluecyan`, `bluemagenta`, `chartreuse`, `cyan`, `darkgray`, `green`, `greencyan`, `lightgray`, `magenta`, `orange`, `red`, `redmagenta`, `white`, or `yellow`.

If the list is prefixed with `cycle:`, colors in the set will be cycled if the number of datasets exceeds the number of specified colors. If the list is composed only of `cycle`, then a default list of colors will be cycled through (this is a subset of the available colors selected to avoid colors which are too similar).

break *column name*

The column containing data set identifiers, or the string `none` to indicate a single data set.

xmin, **xmax**, **ymin**, **ymax**

Explicitly specified plot limits, used if the appropriate flag in **autoscale** is turned off. The limits are specified in the *transformed data space*.

xlabel, **ylabel**

The labels for the plot axes. If not specified, these will be the names of the columns which are plotted.

title *string*

The plot title.

subtitle *string*

The plot subtitle.

height *float*

The height of the plot frame, in inches. If zero, the plot will use the largest possible height.

width *float*

The width of the plot frame, in inches. If zero, the plot will use the largest possible width.

just *boolean*

If true, the scale of the axes will be the same (where scale means plot units/inch of paper)

legend *boolean*

If true, plot a legend. See **legend_opts** and **legend_text**.

legend_opts *parameter list*

legend_opts takes a list of parameters and flags which change how legends are drawn. See *Plot Legend* for more info.

legend_text *string*

A list of texts to be drawn in the legend, one element per dataset. If there are more datasets than texts, the last one is repeated. Elements in the list are separated by the characters "\n". If the string %s appears in the text, it will be replaced by the contents of the break column for the dataset.

char_height *float*

The height of the characters, in dimensionless units, where 1 is the default.

line_width *integer*

The default line width. This affects everything (including characters).

limit_alg *enumerated string*

The algorithm used to specify padding between the data and the frame. One of `none`, `pgplot`, or `percent`.

limit_val

If **limit_alg** is `percent`, this specifies the fration of the plot window to be used as padding.

device *string*

The PGPLOT device to which to output the plot.

help *boolean*

Print out a simple help message and exit.

version *boolean*

Print out **rdbplt**'s version and exit.

debug *list*

A list of debug flags. None are presently defined.

Quick Tutorial

Specifying the Data

The file to be read in is specified via the **input** parameter. If it is the string `stdin`, the data are read from the standard input stream.

The data to plot are selected with the **xcol** and **ycol** parameters. They take a specification which has one of the following forms:

- The name of the column in the RDB file containing the data. These are case sensitive!
- The unary based index of a column. These are written as the index preceded by an underscore, e.g.

  ```
  xcol=_3
  ```

- The string `_NR`, indicating that the record numbers are to be used for the data values.

  ```
  xcol=_NR
  ```

  Record numbers start at 1

To plot error bars, specify the columns containing the errors via the **xerr** and **yerr** parameters. Error bars are correctly transformed if the logarithm of the data is performed.

Data Transformation and Plot Scaling

The transformations applied to the data before being plotted are specified with the **axisxfrm** parameter. It can take the following values:

`linlin`

No transformation is applied.

`linlog`

The base 10 logarithm of the **Y** values is plotted.

`loglin`

The base 10 logarithm of the **X** values is plotted.

`loglog`

The base 10 logarithm of both axes is plotted.

Normally **rdbplt** will autoscale the axes. To force axis limits, use the **autoscale** parameter in conjunction with the **xmin**, **xmax**, **ymin**, **ymax** parameters. **autoscale** should be set to a string of `1`'s and `0`'s, indicating whether an axis is to be autoscaled (`1`) or whether the appropriate limit parameter is to be used (`0`). The digits are ordered **xmin**, **xmax**, **ymin**, **ymax**. For instance, to autoscale on **xmax** and **ymax**, and fix the rest,

```
autoscale=0101 xmin=0 ymin=-33
```

Note that axis limits are specified in the *transformed data space*.

If it makes more sense for the axis coordinates to increase in the opposite sense from normal (i.e. right to left, or up to down), use **axisflip**. For example,

```
axisflip=01
```

flips the sense of the **Y** axis.

Plotting Multiple Data Sets

A single RDB table may contain multiple data sets to be overplotted. The X and Y values for the different data sets are *not* in separate columns, but are concatenated in a single pair of X and Y columns, with a third column indicating the set to which the data belong (this is known in RDB parlance as a break column).

Here's how it works. Each data set has a unique identifier in the break column. **rdbplt** decides that a new data set has begun when the value in the break column changes. This means that all data points for a given data set should be contiguous in the file. For example, here is a valid multiple data set RDB table:

```
x    y   break
N    N   S
0    1   set1
1    2   set1
9    9   set2
10   10  set2
```

The break column is specified via the `break` parameter. If set to `none`, a single data set is plotted. Otherwise, it should be set to the name of a column in the data file containing the data set identifier. The contents of the break column are compared as strings, regardless of the actual rdb column type.

Creating such RDB tables from multiple RDB tables is fairly straighforward, using the **rdbcat** and/or **sorttbl** RDB commands.

Markers and Lines

One would of course like to see the data, and this is accomplished by laying down either markers or lines, or both.

Markers are specified by the **marker** parameter. It can be simply set to `yes`, or `no` (or `none`), in which case one gets markers or one doesn't. In general, it takes a comma separated list of one or more marker specifications (in case there's more than one data set). Spaces may be used to increase readability.

If the markers should be read from a column in the data file, and the column contains marker codes, set

```
marker=column_name
```

If instead the column contains strings which should be plotted, set

```
marker=%column_name
```

The marker type may also be explicitly set:

```
marker=code
```

If there are more data sets than marker specifications, **rdbplt** will automatically choose a new marker to use (by incrementing the marker code for the last specified marker). There are thousands of them ranging from interesting map symbols to letters and digits, so it'll be pretty uncommon to have them repeat.

Marker codes range from -8 to several thousand. Negative markers are filled polygons, positive are symbols, letters, etc. The PGPLOT manual describes them all. `-1` is especially useful, as it's a very small dot.

To omit markers for a particular data set, set its spec to `none`:

```
marker=1,2,3,none,4,5
```

You can specify optional attributes for the markers (to override the global attributes) by appending them to the marker spec (with a `/` character between them). Attributes are of the form

```
height/linewidth/color
```

e.g.,

```
marker=column_name/height/linewidth/color
```

height

The height of the markers, specified in character height units (see *Miscellaneous* for a definition). It defaults to the value of the **char_height** parameter.

linewidth

The thickness of the line used to draw the markers. It defaults to the value of the **line_width** parameter. A thickness of `1` gives the thinnest lines.

color

The color of the marker. The available colors are listed under the **color** entry in *PARAMTERS*.

All attributes are optional (leave the field empty), and trailing empty attributes may be removed:

```
marker=col/3//blue
marker=col/3
```

Connecting lines are controlled by the **connect** parameter. It can be simply set to `yes`, or `no`, in which case one gets lines or one doesn't. In this case, if there is more than one data set, **rdbplt** will automatically cycle through the available line styles. There are unfortunately only 5 styles, so it can get confusing.

For more control over the lines' appearance, **connect** may also be given a comma separated list of line specifications (one per data set), which have the following forms:

```
style/width/color
```

Only `style` is required; the rest are optional, and follow the same rules as the marker attribute specs as to leaving out attributes. Spaces may be used to increase readability.

style

A line style may be one of the following types: `dash`, `dashdot1`, `dashdot2`, `dot`, `full`, `none`. A style of `none` indicates that no line should be drawn for that particular data set.

width

This is the width of the line. It defaults to the value of the `line_width` parameter. A thickness of `1` gives the thinnest lines.

color

The color of the marker. The available colors are listed under the **color** entry in *PARAMTERS*.

Just as with markers, if fewer line specifications are present then there are data sets, **rdbplt** will cycle through line styles.

Colors for markers and lines are specified with the **color** parameter, which takes a comma separated list of colors, one per data set. If there are more data sets than there are color specifications, the default color (whatever that means) will be used for the unspecified ones. This parameter is useful for easily getting the same color for both lines and markers in a data set.

If the list is prefixed with `cycle:`, colors in the set will be cycled if the number of datasets exceeds the number of specified colors. If the list is composed only of `cycle`, then a default list of colors will be cycled through (this is a subset of the available colors selected to avoid colors which are too similar).

The available colors are listed under the **color** entry in *PARAMTERS*.

Plot Frame appearance

Because PGPLOT attempts to fill the plot window, and the window may not be square, plotting data where both axes have the same units may lead to squashed or stretched looking plots. To ensure that both axes have the same ratio of plot units to output pixels (or inches), set the **justify** parameter to **yes**. This will plot square regions as square.

Nicely determined padding between the plotted data and the plot frame makes for a well balanced plot. The padding between the data and the labeled axes is determined by the **limit_alg** parameter, with these possible values:

none

The determined or specified data min and max limits will be used. This may cause data values to appear on top of the plot frame.

pgplot

PGPLOT's algorithm is used, which uses the determined or specified data min and max and finds "nice" rounded numbers near them.

percent

The padding is a fraction of the plot window; the **limit_val** parameter specifies the fraction.

The size of the frame (in inches) may be specified with the **height** and **width** parameters. A value of zero indicates the frame should be the maximum possible width in that direction allowed by the plotting device.

A title, subtitle, and X and Y axis labels are specified via the **title**, **subtitle**, **xlabel**, and **ylabel** parameters, respectively. PGPLOT allows one to specify different fonts (including Greek letters), symbols, and super- and sub- scripts.

Plot Legend

It's nice to know which points went with which data set. **rdbplt** can construct a legend fairly automatically. To have it do so, set the **legend** parameter to `yes`.

The text associated with each data set is taken from the **legend_text** parameter. This is a set of strings (optionally one per data set) to be written next to the marker and line in the legend. Strings are separated by the two characters `\n`. If the characters `%s` are found within a string, they are replaced by the contents of the break column for that parameter set. If there are fewer strings than data sets, the last is repeated.

This sounds a bit complicated, but it's not. If you want to make very spiffy legends, and know how many data sets are there ahead of time, specify the legend strings completely:

```
legend_text='My First Data Set\nMy Second Data Set'
```

Or, say the break column contains the independent parameter which varies between data sets, and you want to insert that into the legend:

```
legend_text='%s [ppm / day]'
```

Or, say the break column has everything in there that you want:

```
legend_text=%s
```

The position of the legend and how it looks is controlled by the **legend_opts** parameter, which takes a space or semicolon separated list of keyword-value pairs or booleans. Pairs have the form

```
keyword=value
```

while booleans appear as simply

```
keyword
```

For example,

```
legend_opts='x=0.3 ll'
```

Note the use of the quotes to get the spaces past the shell.

x

The **x** coordinate of the legend box, as a fraction of the plot frame width. By default this refers to the left edge of the legend box. See **ul** or **ll**. Defaults to zero.

y

The **y** coordinate of the legend box, as a fraction of the plot frame height. By default this refers to the top edge of the legend box. See **ul** or **ll**. Defaults to one.

lineskip

The fraction of a character height to skip between legend entries. Defaults to 1.5.

linelen

The length of illustrative lines drawn in the legend, as a fraction of the plot frame width. Defaults to 0.05.

box

Boolean. Draw a box around the legend. Defaults to off.

boxpad

Padding between the legend box and its contents, in character heights.

char_ht

The legend character height, in dimensionless units, where 1 is the default.

ul

Boolean. Specifies that the **x** and **y** legend positions refer to the upper left of the legend.

ll

Boolean. Specifies that the **x** and **y** legend positions refer to the lower left of the legend.

Output Devices

The device (or file) to which to write the plot is specified with the **device** parameter.

X window display devices are:

`/xserve`

`/xdisp`

(an alternate device which doesn't require as many colors)

To create a PostScript file that can be printed, use a device name of *filename/??* where *filename* is the name of the file to be created, and */??* is one of the following:

`/ps`

landscape PostScript

`/vps`

portrait PostScript

`/cps`

landscape color PostScript

`/vcps`

portrait color PostScript

For the entire list of supported devices, specify ?

```
device='?'
```

Note the quotes to pass the question mark by the shell.

Miscellaneous

The default character height for most things is specified via the **char_height** parameter. A height of `1` is approximately 1/40 of the plot window's height

The default line width is specified by the **line_width** parameter. A width of `1` is very thin.

Advanced Usage

Markers

It is possible to specify a global marker spec, and then override it for individual data sets. The global spec should be separated from the data set ones by a colon.

```
marker=global_def:ds1,ds2

marker=col/1/3/blue://red,none,/3/yellow
```

Note that the global default may be `none` if the majority of data sets should not have markers associated with them. Remember that spaces may be used to increase readability.

With a global spec, one need not specify any data set specific attributes; the marker type will be incremented automatically as usual.

Connecting lines

It is possible to specify a global connect spec, and then override it for individual data sets. The global spec should be separated from the data set ones by a colon.

```
connect=global_def:ds1,ds2

connect=full/1/3/blue://red,none,/3/yellow
```

Note that the global default may be `none` if the majority of data sets should not be connected. Remember that spaces may be used to increase readability.

With a global spec, one need not specify any data set specific attributes; the line style will be changed automatically as usual.

Error Bars

The **xerr** and **yerr** parameters control drawing of error bars Spaces are ignored in error bar specifications, so use them as required to make things readable. Error bar specs come in two forms: as a column name with optional attributes, or as attributes. Each axis has in general two specs for the error bars, one for the positive going side, the other for the negative going side. If only one spec is provided, it'll be used for both. When plotting multiple data sets, one can give a global default spec and then override that for individual data sets.

The column name spec (hereafter **col_spec**) looks like

```
column/X/line_style/line_width/color
```

where **column** is the column name, **X** is the cross bar length (in units of the character height), **line_width** is the thickness of the line used to plot the bar and **color** is the error bar color. The attributes are optional, and may be empty to specify only a particular attribute:

```
xerr='column / / full / / blue'
```

rdbplt does a reasonable job for setting defaults if an attribute isn't set.

The pure attributes spec hereafter (**attr_spec**) looks like a **col_spec**, but the column name (and the first `/` character) aren't required:

```
X/line_style/line_width/color
```

There are several ways to provide the specs for a single error bar:

- If the same error is to be used for negative and positive going error bars (i.e, x - error, x + error), specify a single spec.
- for independent error bars, specify two specs, with the negative going errors first. For **col_specs**, separate them with commas, for **attr_specs**, separate them with `%` characters:

  ```
  neg_col_spec, pos_col_spec
  neg_attr_spec % pos_attr_spec
  ```

  (note that spaces are allowed)

  This results in `y - err_neg` and `y + err_pos`.
- for one-sided error bars, set the column name for the other side to `none`:

  ```
  none,pos_col_spec
  none % pos_attr_spec
  ```

Finally, here's how to combine the specs for one or more data sets.

- If there's a single data set, or you're going to use the same specs for all of your data sets,

  ```
  xerr=col_spec
  ```
- To provide a default set of specs and then override them for multiple data sets,

  ```
  xerr=col_spec:attr_spec,attr_spec
  ```

  If the default spec is to be used for a dataset, provide an empty **attr_spec**:

  ```
  xerr='col_spec : , attr_spec'
  ```

  To not plot error bars for a data set, give it an **attr_spec** of `none`:

  ```
  xerr='col_spec : none, attr_spec'
  ```

Hints

- If you unexpectedly find lots of strange symbols plotted, you've probably plotting multiple data sets and have forgotten to sort the RDB table on the input column.

COPYRIGHT & LICENSE

AUTHOR

D. Jerius <djerius@cfa.harvard.edu>