Package ‘magicaxis’

May 11, 2017

Type Package

Title Pretty Scientific Plotting with Minor-Tick and Log Minor-Tick Support

Version 2.0.1

Date 2017-05-11

Author Aaron Robotham

Maintainer Aaron Robotham <aaron.robotham@uwa.edu.au>

Description Functions to make useful (and pretty) plots for scientific plotting. Additional plotting features are added for base plotting, with particular emphasis on making attractive log axis plots.

License GPL-3

Depends R (>= 2.13), MASS, plotrix, sm, mapproj, celestial

NeedsCompilation no

Repository CRAN

Date/Publication 2017-05-11 12:50:13 UTC

R topics documented:

`magicaxis-package` *Pretty Scientific Plotting with Minor-Tick and Log Minor-Tick Support*

Description

Functions to make pretty axes (major and minor) on scientific plots. Particularly effort is made on producing nice log plot outputs. The core function produces pretty axis labelling in a number of circumstances that are often used in scientific plotting. There is a higher level interface to a generic plot function that will usually produce nice plots, even without much though on the users part.

Details

Package:	magicaxis
Type:	Package
Version:	2.0.1
Date:	2017-05-11
License:	GPL-3
Depends:	MASS, plotrix, sm, mapproj, celestial

`magaxis` *Magically pretty axes*

Description

This function generates nicely arranged axes for scientific plots, including minor tick marks. It supports log settings and can unclog axes that have been logged inline by the user. When the dynamic range is 50 or less and axis is logged, axis range factors of 10 times 1, 2 and 5 are used instead of powers of 10 alone.

Usage

```
magaxis(side=1:2, majorn=5, minorn='auto', tcl=0.5, ratio=0.5, labels=TRUE,
unlog='auto', mgp=c(2,0.5,0), mtline=2, xlab=NULL, ylab=NULL, crunch=TRUE, logpretty=TRUE,
prettybase=10, powbase=10, hersh=FALSE, family='sans', frame.plot=FALSE, usepar=FALSE,
grid=FALSE, grid.col='grey', grid.lty=1, grid.lwd=1, ...)
```

Arguments

`side`	The side to be used for axis labelling in the same sense as the base axis function (1=bottom, 2=left, 3=top, 4=right). A vector of multiple entries is allowed. By default, bottom and left axes are drawn (i.e. side 1 and 2).

`majorn`	The target number of major-axis sub-divisions for pretty plotting. If length is 1 and length of side is longer than this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. Obvious reason for varying this is different pretty labelling between a and y axes.
`minorn`	The exact number of minor-axis divisions (i.e. desired minor ticks + 1) to display in plotting. Auto will produce `pretty` ticks for linear scaling, and powbase-2 minor ticks for logged (this might seem odd, but for base 10 this means ticks at 2/3/4/5/6/7/8/9, which is probably as desired). If set manually, must be greater than 1 to have a visible effect. Minor ticks are always calculated to be equally spaced in linear space, so tick spaces vary when using log plotting. If length is 1 and length of side is longer than this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. An obvious reason for varying this is different pretty labelling between x and y axes.
`tcl`	The length of major tick marks as a fraction of the height of a line of text. By default these face into the plot (in common with scientific plotting) with a value of 0.5, rather than the R default of -0.5. It is possible to force magaxis to inherit directly from par by setting usepar=TRUE (see below). See `par` for more details.
`ratio`	Ratio of minor to major tick mark lengths.
`labels`	Specifies whether major-axis ticks should be labelled for each axis. If length is 1 and length of side is longer than this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. Default is to label all axes.
`unlog`	Determines if axis labels should be unlogged. If axis is found to be logged in par('usr') then the minor ticks are automatically log spaced, however "unlog" still controls how the labelling is done: either logged form (FALSE) or exponent form (TRUE). If axis has been explicitly logged (e.g. log10(x)) then this will can produce exponential axis marking/ labelling if set to TRUE. This case will also produce log minor tick marks. If length of unlog is 1 and length of side is longer than 1 then the assigned unlog value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. Can also take the text argument 'x', 'y', 'xy' or 'yx', where these refer to which axes have been logged. If left at the default of 'Auto' then unlog is assumed to be true when the axis in question is logged, and false otherwise.
`mgp`	The margin line (in mex units) for the axis title, axis labels and axis line. This has different (i.e. prettier) defaults than R of c(2,0.5,0) rather than c(3,1,0). This pushes the numbers and labels nearer to the plot compared to the defaults. It is possible to force magaxis to inherit directly from par by setting usepar=TRUE (see below). See `par` for more details.
`mtline`	Number of lines separating axis name from axis.
`xlab`	x axis name.
`ylab`	y axis name.
`crunch`	In cases where the scientific text would be written as 1x10^8, should the 1x be removed so it reads 10^8. If length is 1 and length of side is longer then this

	value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. TRUE by default.
`logpretty`	Should the major-ticks only be located at powers of 10. This changes cases where ticks are placed at 1, 3.1, 10, 31, 100 etc to 1, 10, 100. If length is 1 and length of side is longer then this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. TRUE by default.
`prettybase`	The unit of repitition desired. By default it is 10, implying a pretty plot is one with marks at 10, 20, 30 etc. If you are plotting degrees then it might be prettier to display 90, 180, 270 etc. In which case prettybase should be set to 90. If log=TRUE then the reference location of 10 is changed, so in the previous example the labels generated would be at 9, 90, 900 etc rather than the deafult of 1, 10, 100 etc. If length is 1 and length of side is longer then this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side.
`powbase`	Set the base to use for logarithmic axes. Default is to use 10.
`hersh`	To determines whether all plot text should be passed using Hershey vector fonts. This applies to the axis labels (which are handled automatically) and the axis names. In the case of axis names the user must be careful to use the correct plot utils escape characters: http://www.gnu.org/software/plotutils/manual/en/html_node/Text-String-Format.html. magaxis will return back to the current plotting family after the function has executed.
`family`	Specifies the plotting family to be used. Allowed options are 'sans' and 'serif'. Depending on whether hersh is TRUE or FALSE these otions are either applied to the Hershey vector fonts (hersh=TRUE) or the default R Helvetica font (hersh=FALSE). magaxis will return back to the current plotting family after the function has executed.
`frame.plot`	Logical indicating whether a box should be drawn around the plot.
`usepar`	Logical indicating whether tcl and mgp should be forced to inherit the global par values. This might be preferred when you want to define global plot settings at the start of a script.
`grid`	Logical indicating whether a background grid should be drawn onto the plotting area. This will only be done for side=1 (i.e. vertical grid lines) and side=2 (i.e. horizontal grid lines). If length is 1 and length of side is longer then this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. FALSE by default.
`grid.col`	The colour of the grid to be drawn.
`grid.lty`	The line type of the grid to be drawn.
`grid.lwd`	The line width of the grid to be drawn.
`...`	Other arguments to be passed to base `axis` function.

Details

This function tries hard to make nice plots for scientific papers.

Value

No output. Run for the side effect of producing nice plotting axes.

Author(s)

Aaron Robotham

See Also

magplot,maglab,magerr,magmap,magrun

Examples

```
x=10^{1:9}
y=1:9
plot(log10(x),y,axes=FALSE)
magaxis(unlog='x')

plot(log10(x),y,axes=FALSE)
magaxis(side=c(1,3),unlog=c(TRUE,FALSE))

plot(x,y,axes=FALSE,log='x')
magaxis()
```

magbar *Pretty colour bar*

Description

This function is a high level interface to the plotrix 'color.legend' function. It makes reasonable assumptions on the plottin window to place the colour and allows the user to specify log spacing for the colour gradient and labels, as well as add a title.

Usage

```
magbar(position = "topright", range = c(0, 1), orient = "v", log = FALSE,
col = hsv(h = seq(2/3, 0, len = 100)), scale = c(1/4, 1/20), inset = 1/40,
labN = 5, title = "", titleshift = 0, centrealign = "rb", clip='', cex=1, ...)
```

Arguments

`position`	Relative position of the colour bar. This argument is used like the 'legend' function. Specify one of 'bottom', 'bottomleft', 'left', 'topleft', 'top', 'topright', 'right', 'bottomright' and 'centre'.
`range`	The text label limits used to label the colour bar.
`orient`	Orientation. Allowed options are 'v' for vertical and 'h' for horizontal.

`log`	Should the colour spacing and labelling be log spaced.
`col`	Colour palette to use for the colouring of the bar.
`scale`	The relative (to the plot window) length and width of the colour bar.
`inset`	Relative (to the plot window) inset of the colour bar.
`labN`	The number of text labels to draw on the colour bar.
`title`	Optional title (or axis label for the labels) to add to the colour bar.
`titleshift`	Extra shift to apply to the 'title' position.
`centrealign`	Option to control the labeling position used when the position='centre'.
`clip`	Setting clip='bg' will set values outside the 'range' values to be blank on the magbar (i.e. you can see through to the background).
`cex`	Character expansion factor for the labels.
`...`	Other arguments to pass to the `color.legend` function.

Details

This function creates pretty default colour bars by assessing the current plot window. It is a higher level implementation of the plotrix 'color.legend' function.

Value

Called for the side effect of plotting a colour bar.

Author(s)

Aaron Robotham

See Also

`magplot`,`magaxis`,`maglab`,`magmap`,`magrun`

Examples

```
magplot(sin)
magbar('top')
magbar('right',title='Just looking',titleshift=0.5)
magbar('topleft',orient='h',title='Hello!')
magbar('bottom',range=c(0.3,30),orient='h',log=TRUE,title='Log test col')
magbar('bottomleft',range=c(0.3,30),orient='v',log=TRUE,title='Log test bg',clip='bg')
```

magclip *Magical sigma clipping*

Description

This function does intelligent autoamtic sigma-clipping of data. This is optionally used by magplot and maghist.

Usage

```
magclip(x, sigma='auto', clipiters=5, sigmasel=1, estimate='both')
```

Arguments

`x`	Numeric; the values to be clipped. This can reasonably be a vector, a matrix or a dataframe.
`sigma`	The level of sigma clipping to be done. If set to default of 'auto' it will dynamically choose a sigma level to cut at based on the length of x (or the clipped version once iterations have started), i.e.: sigma=qnorm(1-2/length(x)). This have the effect of removing unlikely values based on the chance of them occurring, i.e. there is roughly a 50% chance of a 3.5 / 4.6 sigma Normal fluctuation occurring when you have 10,000 / 10,000,000 values, hence choosing this value dynamically is usually the best option.
`clipiters`	The maximum number of sigma clipping iterations to attempt. It will break out sooner than this if the iterations have converged. The default of 5 is usually plenty (up to the contamination being towards the 50% level).
`sigmasel`	The quantile to use when trying to estimate the true standard-deviation of the Normal distribution. if contamination is low then the default of 1 is about optimal in terms of S/N, but you might need to make the value lower when contamination is very high.
`estimate`	Character; determines which side/s of the distribution are used to estimate Normal properties. The default is to use both sides (both) giving better S/N, but if you know that your contamination only comes from positive flux sources (e.g., astronomical data when trying to select sky pixels) then you should only use the lower side to determine Normal statistics (lo). Similarly if the contamination is on the low side then you should use the higher side to determine Normal statistics (hi).

Details

If you know more sepcific details about your data then you should probably carry out a thorough likelihood analysis, but the ad-hoc clipping done in `magclip` works pretty well in practice.

Value

A list containing three items:

`x`	Numeric vector; the cliped 'x' values.
`clip`	Locial; logic of which values were clipped with the same type and shape attributes as the input 'x' (i.e. if the original 'x' was a matrix then 'clip' would also be a matrix that matches element to element).
`range`	The data range of clipped 'x' values returned.

Author(s)

Aaron Robotham

See Also

`maghist`, `magplot`

Examples

```
#A highly contaminated Normal distribution:
temp=c(rnorm(1e3),runif(500,-10,10))
magplot(density(temp))
lines(seq(-5,5,len=1e3),dnorm(seq(-5,5,len=1e3)),col='red')

magplot(density(magclip(temp)$x))
lines(seq(-5,5,len=1e3),dnorm(seq(-5,5,len=1e3)),col='red')

#Now we put the contamination on the high side:

temp=c(rnorm(1e3),runif(500,0,10))
magplot(density(magclip(temp)$x))
lines(seq(-5,5,len=1e3),dnorm(seq(-5,5,len=1e3)),col='red')

magplot(density(magclip(temp, estimate='lo')$x))
lines(seq(-5,5,len=1e3),dnorm(seq(-5,5,len=1e3)),col='red')
```

magcon *2D quantile images and contours*

Description

This function generates pretty images and contours that reflect the 2D quantile levels of the data. This means the user can immediately assess the 2D regime that contains an arbitrary percentage of the data. This function was designed particularly with the output of MCMC posteriors in mind, where visualising the location of the 68% and 95% 2D quantiles for covariant parameters is a necessary part of the post MCMC analysis.

Usage

```
magcon(x, y, h, doim = TRUE, docon = TRUE, dobar = TRUE, n = 100, add = FALSE,
xlab='', ylab='', imcol = rev(rainbow(1000, start = 0, end = 2/3)),
conlevels = c(0.5, pnorm(1) - pnorm(-1), 0.95), barposition = "topright",
barorient = "v",bartitle = "Contained %", bartitleshift=0,xlim=NULL,ylim=NULL,
weights=NA,...)
```

Arguments

`x`	x values to contour. If x is a two (or more) column matrix or data.frame and y is missing as an argument, then the first column is used for x and the second column for y.
`y`	y values to contour.
`h`	Smoothing parameter to pass to kde2d. Can take 1 or 2 arguments for x and optionally y smoothing.
`doim`	Should an image be generated.
`docon`	Should contours be overlain.
`dobar`	Should a magbar colour bar be added describing the image levels (doim must also be true for this to appear).
`n`	The n to send to kde2d to determine the resolution of the smoothing.
`add`	Should the output of this function be added to the current plot. If FALSE then a new plot is generated.
`xlab`	Label for x-axis, only used if add=FALSE.
`ylab`	Label for y-axis, only used if add=FALSE.
`imcol`	The colour palette to use for the image (this is also sent to magbar).
`conlevels`	Specific quantile contours to add. Default is for 50%, 68% and 95% contours, i.e. these contours contain that perecentage of the data.
`barposition`	The position to use for magbar. See magbar help for more details.
`barorient`	The orientation to use for magbar. See magbar help for more details.
`bartitle`	Title to use for magbar.
`bartitleshift`	Control of how far the magbar title is shifted away from its default position.
`xlim`	The x limits to use for the data. Default of NULL caculates the range based on the provided x data vector. Data will be clipped between the extremes given. If xlim[1]>xlim[2] plotted axes will be flipped compared to default.
`ylim`	The y limits to use for the data. Default of NULL caculates the range based on the provided y data vector. Data will be clipped between the extremes given. If ylim[1]>ylim[2] plotted axes will be flipped compared to default.
`weights`	A vector of weights to pass onto sm.density (that does the 2D density estimate). This must be the same length as the x and y vectors if specified.
`...`	Other arguments to pass to the `contour` function, e.g. lty=c(2,1,3).

Details

This function is particularly designed to assess the output for MCMC posteriors since it highlights the confidence regimes quite clearly. More generally it can show the quantile distributions for any 2D data.

Value

Called for the side effect of generating images and contours representing quantile in 2D data.

Author(s)

Aaron Robotham

See Also

magplot,magaxis,maglab,magmap,magrun,magbar

Examples

```
temp=cbind(rnorm(1e3),rnorm(1e3))
magcon(temp[,1],temp[,2])
```

`magerr` *Error bar plotting*

Description

A function to dd x and y error bars to plots. Low and high error bars can be generated.

Usage

```
magerr(x, y, xlo, ylo, xhi = xlo, yhi = ylo, corxy, length = 0.02,
col = 'black', fill=FALSE, poly=FALSE, ...)
```

Arguments

`x`	x location of data.
`y`	y location of data.
`xlo`	Error on the low side for x values. This can be positive or negative- the absolute vaue is used.
`ylo`	Error on the low side for y values. This can be positive or negative- the absolute vaue is used.
`xhi`	Error on the high side for x values. This can be positive or negative- the absolute vaue is used. By default this will inherit the xlo value.
`yhi`	Error on the high side for y values. This can be positive or negative- the absolute vaue is used. By default this will inherit the ylo value.

`corxy`	If this paramter exists then error ellipses will be drawn instead of error bars. It takes the value of the sigma_x sigma_y correlation, i.e. corxy=covxy/(xlo*ylo).
`length`	Length of error bar ends.
`col`	Either the colour of the error bars or the outline colour of the error ellipses.
`fill`	Logical; if TRUE then the error ellipses will be filled, if FALSE then only the border will be drawn.
`poly`	Logical; is FALSE then error bars or ellipses will be drawn, if TRUE then approximate error polygon will be shown instead.
`...`	Further arguments to be passed to the `arrows` / `draw.ellipse` / `polygon` functions used to draw the error bars / error ellipses (‘`corxy`’ not missing) / error polygon (‘`poly`’=TRUE).

Details

Note that with ‘`poly`’=TRUE the x values are used igoring any error terms, and the point value y errors are used to define the limits of the polygon, with straight lines joining the points. The ‘`col`’ option is used to fill the polygon with a colour (so the default black is probably not a great choice). The `polygon` function takes the argument ‘`border`’ (parsed by dots from the `magerr` function) to colour the outer lines, so for a more subtle error polygon you might want to use ‘`col`’=lightgrey, ‘`border`’=NA, where NA means no outer border lines are drawn.

Value

Called for the side effect of plotting error bars.

Author(s)

Aaron Robotham

See Also

`magplot`, `magaxis`, `maglab`, `magmap`, `magrun`, `arrows`, `draw.ellipse`, `polygon`

Examples

```
# Basic x and y errors added to plot
temp=cbind(x=runif(10),y=runif(10),xerr=runif(10,0.05,0.2),yerr=runif(10,0.1,0.3),
corxy=runif(10,-1,1))
magplot(temp[,1:2])
magerr(x=temp[,1],y=temp[,2],xlo=temp[,3],ylo=temp[,4])
# Example of errors on plots wityh log axes
magplot(temp[,1:2],log='xy')
magerr(x=temp[,1],y=temp[,2],xlo=temp[,3],ylo=temp[,4])

#Example of error ellipses

magplot(temp[,1:2])
magerr(x=temp[,1],y=temp[,2],xlo=temp[,3],ylo=temp[,4])
magerr(x=temp[,1],y=temp[,2],xlo=temp[,3],ylo=temp[,4],corxy=temp[,5])
```

maghist *Magically pretty histograms*

Description

A fairly simple function that produces pretty histograms. The main difference to base `hist` is that it allows for easy truncation of the data provided via '`xlim`'.

Usage

```
maghist(x, breaks = "Sturges", freq = TRUE, include.lowest = TRUE, right = TRUE,
density = NULL, angle = 45, col = NULL, border = NULL, xlim = NULL, ylim = NULL,
plot = TRUE, verbose=TRUE, add=FALSE, log='', ...)
```

Arguments

`x`	A vector of values for which the histogram is desired.
`breaks`	One of: • A vector giving the breakpoints between histogram cells, • A function to compute the vector of breakpoints, • A single number giving the number of cells for the histogram, • A character string naming an algorithm to compute the number of cells, • A function to compute the number of cells. In the last three cases the number is a suggestion only; the breakpoints will be set to `pretty` values. If breaks is a function, the x vector is supplied to it as the only argument.
`freq`	Logical; if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). Defaults to TRUE if and only if breaks are equidistant (and probability is not specified).
`include.lowest`	Logical; if TRUE, an x[i] equal to the breaks value will be included in the first (or last, for right = FALSE) bar. This will be ignored (with a warning) unless breaks is a vector.
`right`	Logical; if TRUE, the histogram cells are right-closed (left open) intervals.
`density`	The density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. Non-positive values of density also inhibit the drawing of shading lines.
`angle`	The slope of shading lines, given as an angle in degrees (counter-clockwise).
`col`	A colour to be used to fill the bars. The default of NULL yields unfilled bars.
`border`	The color of the border around the bars. The default is to use the standard foreground color.

`xlim`	Vector; range of ‘`x`’ values to use for both counting and plotting. The default NULL will span the range of histogram breaks. If length equals 1 then the argument is taken to mean the sigma range to select for plotting and the clipping is done by `magclip`. If this is set to ’auto’ then the limits will be estimated from the data dynamically. See examples.
`ylim`	Vector; range of y limits to show in the histogram plot.
`plot`	Logical; draw the histogram (otherwise it just returns the count data).
`verbose`	Logical; if TRUE and ‘`xlim`’ is used then the followign is printed out: summary of the data selected, standard-deviation the 1/2-sigma implied quantiles, and number and fraction of displayed data. Note all numbers are computed for the logged values of the ‘`x`’ input if ‘`log`’= x | xy | yx.
`add`	Logical, if TRUE the histogram will be added to the current plot. Be careful to match ‘`log`’ properties if adding, else the comparison will be of little use and hard to interpret.
`log`	Log axis arguments to be passed to hist and plot. E.g. use ’x’, ’y’, ’xy’ or ’yx’ as appropriate. Default ” assumes no logging of any axes. If the x axis is logged then the histogram will be calculated in log-space.
`...`	Arguments to be parsed to `magplot`.

Details

To better replicate the base `hist` plot you might consider setting ‘`frame.plot`’=FALSE, which will be parsed to `magplot` and turn off the outer box. The default behaviour might change in the future.

Value

An object of class "histogram", basically the same output as produced by `hist`. Note where axes are logged, the corresponding hist list values will not be logged when returned. This is to make it easy to take a histogram object and plot it with different log scalings on the axes (see Examples). For the x axis this means the "breaks" and the "mids" items, and for the y axis this means the "counts" and the "density" items.

Appended to the end of the usual `hist` output are the summary of the sample (list element "summary") and the standard-deviation / 1 and 2-sigma quantile range (list element "ranges").

Author(s)

Aaron Robotham

See Also

`hist`

Examples

```
maghist(rnorm(1e4))
maghist(rnorm(1e4), xlim=c(-2,4))

#Notice the x-limits are close to -3/3, since  if we ask for xlim=3 (a 3-sigma range)
```

```
maghist(rnorm(1e4), xlim=3, verbose = FALSE)

#The 'auto' option allows magclip to dynamically estimate a clip value (which is similar
#in this case, but need not be in general).

maghist(rnorm(1e4), xlim='auto', verbose = FALSE)

#Test of log histograms:

testdata=10^(runif(1e3,0,4))
maghist(testdata)
maghist(testdata,log='x')
maghist(testdata,log='y')
maghist(testdata,log='xy')

maghist(testdata,freq=FALSE)
maghist(testdata,freq=FALSE,log='x')
maghist(testdata,freq=FALSE,log='y')
maghist(testdata,freq=FALSE,log='xy')

#Test of plotting histogram objects:

testhist=maghist(testdata,log='xy')
maghist(testhist)
maghist(testhist,log='x')
magplot(testhist,log='y')
magplot(testhist,log='xy')

#Nice to see a grid with large ranges:

maghist(rnorm(1e6), grid=TRUE)
maghist(rnorm(1e6), log='y', grid=TRUE)
```

`magimage` *Magically pretty images*

Description

magimage is a level replacement for base image with hooks into magaxis for the tick marks and magmap for the image scaling. The default behavious is a bit different to base (e.g. x/y scales are automatically the number of pixels in the image matrix). magimageRGB is similar, but is for the creation colour images where the user can provide R G B input matrix chanels (or similar).

Usage

```
magimage(x, y, z, zlim, xlim, ylim, col = grey((0:1000)/1000), add = FALSE,
useRaster = TRUE, asp = 1, magmap=TRUE, lo = 0.4, hi = 0.995, flip = FALSE,
range = c(0, 1), type = "quan", stretch = "asinh", stretchscale = 'auto', bad = NA,
clip = "", axes = TRUE, frame.plot = TRUE, sparse='auto', ...)
```

```
magimageRGB(x, y, R, G, B, zlim, xlim, ylim, add = FALSE, useRaster = TRUE, asp = 1,
magmap = TRUE, lo = 0.4, hi = 0.995, flip = FALSE, range = c(0, 1), type = "quan",
stretch = "asinh", stretchscale = "auto", bad = range[1], clip = "", axes = TRUE,
frame.plot = TRUE, sparse='auto', ...)
```

Arguments

`x, y`	Locations of grid lines at which the values in z are measured. These must be finite and non-missing (order may be reversed). By default, equally spaced values from 0 to dim(z)[1] are used. If x is a list, its components x$x and x$y are used for x and y, respectively. If the list has component z/R/G/B this is used for z/R/G/B.
`z`	A numeric or logical matrix containing the values to be plotted (NAs are allowed). Note that x can be used instead of z for convenience.
`R`	A numeric or logical matrix containing the red colour values to be plotted (NAs are allowed). Note that a 3D array x can be used instead of R for convenience, where R=x[,,1].
`G`	A numeric or logical matrix containing the green colour values to be plotted (NAs are allowed). Note that a 3D array x can be used instead of G for convenience, where G=x[,,2].
`B`	A numeric or logical matrix containing the blue colour values to be plotted (NAs are allowed). Note that a 3D array x can be used instead of B for convenience, where B=x[,,3].
`zlim`	The z limit with respect to the output of magmap$map. If '`magmap`'=FALSE (default) '`zlim`' should be with respect to the provided z matrix (like base `image`). If '`magmap`'=TRUE '`zlim`' should be with respect to the '`range`' output of `magmap`. By default the `magmap` function scales between 0 and 1, so to only show the brighter pixels '`zlim`' could be set to c(0.5,1).
`xlim, ylim`	Ranges for the plotted x and y values, defaulting to the ranges of x and y.
`col`	A list of colours for the `magmap` re-mapping of z to be parsed into (e.g. `rainbow`, `heat.colors`, `topo.colors`, `terrain.colors` or similar).
`add`	If true add the new image to the current plot.
`useRaster`	If TRUE a bitmap raster is used to plot the image instead of polygons. The grid must be regular in that case, otherwise an error is raised. Raster is much faster, so use when pixels are equal sized.
`asp`	The y/x aspect ratio
`magmap`	
`lo`	The low limit to clip the z data at (what this means varies depending on the 'type' option). This can be a single value (used for R, G and B) or a vector of length 3 (used for R, G and B respectively). See `magmap` for more information.
`hi`	The high limit to clip the z data at (what this means varies depending on the 'type' option). This can be a single value (used for R, G and B) or a vector of length 3 (used for R, G and B respectively). See `magmap` for more information.

`flip`	Should the z scaling be flipped. This allows numbers from 0 to 10 to be mapped from 1 to 0 (so ordered back to front with respect to the input). See `magmap` for more information.
`range`	The numerical range of the output z mapping which should be a vector of length two specifying c(low,high). See `magmap` for more information.
`type`	The type of z mapping attempted. Options are 'quan' (default), 'num', 'sig' and 'rank'. See `magmap` for more information.
`stretch`	stretch='lin' gives linear mapping. stretch='log' gives logarithmic mapping. stretch='atan' gives atan mapping. stretch='asinh' gives asinh mapping. stretch='sqrt' gives sqrt mapping. See `magmap` for more information.
`stretchscale`	A number to multiply the z data by before applying the stretch. This only has a user impact for stretch='atan' and stretch='asinh' since it controls what parts of the data is in the linear or logarithmic regime of the stretch procedure. If set to 'auto' (the default) it uses 1/median(abs(data)) to find a useful scale.See `magmap` for more information.
`bad`	Sets the value that NA, NaN and infinite input z data should be set to in the final map output. This should be thought of in the context of the range argument, i.e. if bad=range[1] then bad values will be the low range value and if bad=range[2] bad values will be the high range value. See `magmap` for more information. For `magimageRGB` bad is set to range[1] by default since this removes RGB conversion errors that would be experiences with NA values (i.e. negative values when '`stretch`'='log').
`clip`	By default clipped z values inherit the nearest lo/hi value (depending on which side they are clipped). Setting clip='NA' will set values outside the 'lo' and 'hi' values to be NA (currently this is the only other clip option). See `magmap` for more information.
`axes`	Specify if any axes be drawn on the image. If FALSE then only the pixels (with appropriate `magmap` scaling) are shown.
`frame.plot`	Specify if a box be drawn around the image frame. Only happens if '`add`'=TRUE and '`axes`'=TRUE.
`sparse`	Determines whether the image pixels are sparse sampled to speed up plotting. If set to 2 it will only determine every 2nd pixel, and if 3 every 3rd etc. The default 'auto' means it will scale to produce a maximum number of 1,000 pixels on any side (on most monitors this is a fairly useful maximum, and ensures quick displaying of even very large images).
`...`	Arguments to be parsed to `magaxis`. See `magaxis` for details.

Details

See `image`, `magmap` and `magaxis` for more details.

Value

Outputs the final image list containing x,y and z (magimage) or R/G/B (magimageRGB). Generally run for the side effect of producing rapid and well-scaled image plots.

Author(s)

Aaron Robotham

See Also

`image`, `magmap`, `magaxis`

Examples

```
#Basic
magimage(matrix(1:9,3))

#Mid pixel versus pixel edge:
magimage(3:0,1:3,matrix(1:9,3))

#Standard scaling is not very useful in this instance:
magimage(matrix(10^(1:9),3))
#Linear scaling is not very useful in this instance, though it does now map from [0,1]:
magimage(matrix(10^(1:9),3),magmap=TRUE,zlim=c(0,0.5))
#Log scaling with magmap makes it much clearer:
magimage(matrix(10^(1:9),3),magmap=TRUE,stretch='log')
#And it's easy just to show the lowest half now:
magimage(matrix(10^(1:9),3),magmap=TRUE,stretch='log',zlim=c(0,0.5))
```

`maglab` *Pretty scientific labelling*

Description

Utilises pretty for the major-tick locations, but makes prettier decisions if log axes are being used. Translates the default text into nicely formatted expressions- this is particularly successful when axes are logged and exponents are used since formats like 1e5 should not be used in scientific academic journals.

Usage

```
maglab(lims, n, log=FALSE, exptext=TRUE, crunch=TRUE, logpretty=TRUE,
usemultloc=FALSE, multloc=c(1,2,5), prettybase=10, powbase=10, hersh=FALSE, trim=FALSE)
```

Arguments

`lims`	Limits over which pretty major-tick locations will be calculated.
`n`	The target number of major-axis sub-divisions. Will not necessarily be achieved.
`log`	Should the limits be evenly distributed over log space. Usually what you want if an axis has been logged.
`exptext`	Should log==TRUE then should the text be written in exponent form (e.g. 10^8, default when exptext==TRUE) or logged (e.g. 8 in this case).

`crunch`	In cases where the scientific text would be written as 1x10^8, should the 1x be removed so it reads 10^8. TRUE by default.
`logpretty`	Should the major-ticks only be located at powers of 10, or when dynamic range is small (less than 50) powers of 10 times 1, 2 and 5. This changes cases where ticks are placed at 1, 3.1, 10, 31, 100 etc to 1, 10, 100.
`usemultloc`	For log=TRUE, if usemultloc=FALSE then label locations are only at powers of 10, if usemultloc=TRUE then they are at multiples of powers of 10 as defined by multloc.
`multloc`	If usemultloc is TRUE then multloc provides the multiples of powers of 10 for the location of labels. Default will give them at 0.1, 0.2, 0.5, 1 etc.
`prettybase`	The unit of repitition desired. By default it is 10, implying a pretty plot is one with marks at 10, 20, 30 etc. If you are plotting degrees then it might be prettier to display 90, 180, 270 etc. In which case prettybase should be set to 90. If log=TRUE then the reference location of 10 is changed, so in the previous example the labels generated would be at 9, 90, 900 etc rather than the deafult of 1, 10, 100 etc.
`powbase`	Set the base to use for logarithmic axes. Default is to use 10.
`hersh`	Determines whether the text format output by maglab should be Hershey vector font compatable text (TRUE), or normal plotmath style expressions (FALSE).
`trim`	If trim is TRUE the outputs are not allowed to exceed the stated limits, if FALSE then the whole range of pretty values calculated are used. This will usually extend beyond the limits to ensure the plots look pretty, but if maglab is being used for something other than plotting axis labels then trimmed values might be useful.

Details

This function is a mid level routine for producing nice ticks and text, with particularly effort on improving the outcome of logged axis cases. The end user will probably not require axis to it except in unusual circumstances. I note that my method of translating the default representation of the exponents is not very elegant, so any suggestions for improvement are welcome!

Value

`tickat`	Location of proposed major-tick marks.
`labat`	Location of proposed label locations (not necessarily the same as major-tick locations).
`exp`	Expressions to be used at label locations.

Author(s)

Aaron Robotham

See Also

`magplot`,`magaxis`,`magerr`,`magmap`,`magrun`

Examples

```
x=10^{1:9}
y=1:9
plot(log10(x),y,axes=FALSE)
ticks=maglab(range(x),log=TRUE)
print(ticks)
axis(1,at=log10(ticks$labat),labels=ticks$exp)

# Same outcome a different way:

plot(x,y,axes=FALSE,log='x')
ticks=maglab(range(x),log=TRUE)
print(ticks)
axis(1,at=ticks$labat,labels=ticks$exp)

# For small dynamic range

x=seq(1,40,len=9)
y=1:9
plot(x,y,axes=FALSE,log='x')
ticks=maglab(range(x),log=TRUE,usemultloc=TRUE)
axis(1,at=ticks$labat,labels=ticks$exp,tick=FALSE)
axis(1,at=ticks$tickat,labels=FALSE)

# Different base prettiness

x=0:270
y=sin(x*pi/180)
plot(x,y,axes=FALSE,type='l')
ticks=maglab(range(x))
axis(1,at=ticks$labat,labels=ticks$exp)
# Not very pretty for degree plotting
ticks=maglab(range(x),prettybase=45)
axis(3,at=ticks$labat,labels=ticks$exp)
# Much nicer!
```

`magmap` *Value remapper*

Description

This function allows the use to remap a vector of values onto a different system. For instance you might have values stretching from -10 to 100 which you want mapped from 0 to 2/3 so you can then sue the output as an input for point colour or size. It allows clipping of values, rejection of bad values, and log stretching.

Usage

```
magmap(data, lo = 0, hi = 1, flip = FALSE, range = c(0, 2/3), type = "quan",
stretch = 'lin', stretchscale=1, bad = NA, clip='')
```

Arguments

`data`	A vector of values. This can contain bad values (NA, NaN, infinite), but these will be ignored during mapping and set to the value of input parameter 'bad'.
`lo`	The low limit to clip the data at (what this means varies depending on the 'type' option). This should be a single value.
`hi`	The high limit to clip the data at (what this means varies depending on the 'type' option). This should be a single value.
`flip`	Should the scaling be flipped. This allows numbers from 0 to 10 to be mapped from 1 to 0 (so ordered back to front with respect to the input).
`range`	The numerical range of the output mapping which should be a vector of length two specifying c(low,high).
`type`	The type of mapping attempted. Options are 'quan' (default), 'num', 'sig' and 'rank'.
`stretch`	'`stretch`'='lin' gives linear mapping. '`stretch`'='log' gives logarithmic mapping. '`stretch`'='atan' gives atan mapping. '`stretch`'='asinh' gives asinh mapping. '`stretch`'='sqrt' gives sqrt mapping. '`stretch`'='cdf' gives CDF mapping onto the cumulative range 0-1
`stretchscale`	A number to multiply the data by before applying the stretch. This only has a user impact for stretch='atan' and stretch='asinh' since it controls what parts of the data is in the linear or logarithmic regime of the stretch procedure. If set to 'auto' it uses 1/median(abs(data)) to find a useful scale.
`bad`	Sets the value that NA, NaN and infinite input data should be set to in the final map output. This should be thought of in the context of the range argument, i.e. if bad=range[1] then bad values will be the low range value and if bad=range[2] bad values will be the high range value.
`clip`	By default clipped values inherit the nearest lo/hi value (depending on which side they are clipped). Setting clip='NA' will set values outside the 'lo' and 'hi' values to be NA (currently this is the only other clip option).

Details

'`type`'='quan' means the '`lo`' and '`hi`' options are interpreted as the quantile limits to clip the data at (so lo=0.05 and hi 0.95 would clip the data at the 5% and 95% quantile limits and scale values between these). '`type`'='num' interprets '`lo`' and '`hi`' as the exact values to clip the data at and scale between. '`type`'='sig' treats '`lo`' and '`hi`' as the sigma offsets in a Normal distribution, with the probabilities at these positions used to clip and scale that data (so '`lo`'=-1 and '`hi`'=1 is interpretted as +/- 1 sigma, so the data is clipped and scaled at the 16% and 84% levels, i.e. the 1 sigma range). '`type`'='rank' means the data mapping is done by rank value only, with '`lo`' and '`hi`' specifying the quantile limits used to clip and scale the ranks. In all cases lo and hi clipped values are set to the relevant extreme values of 'range'.

If range is between 1 and 100 and stretch='lin' the midpoint in the mapping will be 50.5. If stretch='log' the midpoint becomes 10. This enhances the local dynamic range of the mapping for data that has a logarithmic distribution.

Value

`map`	The remapped data. This is the same length and order as the input data.
`datalim`	The a vector of the low and high limits actually applied to the data. Unless type='num' this will probably be different to the lo and hi arguments provided.
`maplim`	The output range (same is the requested input range, but included for book-keeping).
`loclip`	The fraction of objects clipped from the input data at the low end.
`hiclip`	The fraction of objects clipped from the input data at the high end.

Author(s)

Aaron Robotham

See Also

`magimage`, `magbar`

Examples

```
set.seed(650)
temp=cbind(runif(100),runif(100))
temp=cbind(temp,sqrt(temp[,1]^2+temp[,2]^2))
magplot(temp)
magplot(temp[,1:2],col=hsv(h=magmap(temp[,3])$map))

# A different mapping type:
magplot(temp[,1:2],col=hsv(h=magmap(temp[,3],type='rank')$map))

# Flipped:
magplot(temp[,1:2],col=hsv(h=magmap(temp[,3],flip=TRUE,type='rank')$map))

# Example of linear/log/atan/asinh mapping:
temp=cbind(temp,10^temp[,3])
magplot(temp[,1:2],col=hsv(h=magmap(temp[,4])$map))
magplot(temp[,1:2],col=hsv(h=magmap(temp[,4],stretch='log')$map))
magplot(temp[,1:2],col=hsv(h=magmap(temp[,4],stretch='atan')$map))
magplot(temp[,1:2],col=hsv(h=magmap(temp[,4],stretch='asinh')$map))

#atan and asinh can be useful when data spans negative to positive:
temp=cbind(temp,temp[,4]-10)
magplot(temp[,1:2],col=hsv(h=magmap(temp[,5],stretch='atan')$map))
magplot(temp[,1:2],col=hsv(h=magmap(temp[,5],stretch='asinh')$map))
#effect of stretchscale
magplot(temp[,1:2],col=hsv(h=magmap(temp[,5],stretch='atan',stretchscale=0.5)$map))
magplot(temp[,1:2],col=hsv(h=magmap(temp[,5],stretch='atan',stretchscale=2)$map))
magplot(temp[,1:2],col=hsv(h=magmap(temp[,5],stretch='asinh',stretchscale=0.5)$map))
magplot(temp[,1:2],col=hsv(h=magmap(temp[,5],stretch='asinh',stretchscale=2)$map))

#Using multiple mappings for plots:
magplot(temp[,1:2],col=hsv(h=magmap(temp[,4],stretch='log')$map),
cex=magmap(temp[,3],lo=0.5,hi=1,range=c(1,6),type='num')$map)

#Different combinations of mapping options:
magmap(c(-1,0.1,1,NA,0.3,3),lo=0,hi=2.5,type='num',stretch='lin',bad=0.5)$map
magmap(c(-1,0.1,1,NA,0.3,3),lo=0.1,hi=0.9,type='quan',stretch='log',bad=0.8)$map
magmap(c(-1,0.1,1,NA,0.3,3),lo=-1,hi=1,type='sig',stretch='asinh',bad=0,stretchscale=2)$map
magmap(c(-1,0.1,1,NA,0.3,3),type='rank',stretch='atan',bad=NA,stretchscale=2)$map

#Example showing using asinh to generate a different axis mapping:
datastretch=cbind(runif(1e3),10^runif(1e3,0,4)-10^runif(1e3,0,4))
#This isn't a very helpful view of the data
magplot(datastretch[,1:2])
#This only shows the positive half of the data:
magplot(datastretch[,1:2],log='y')
#We can do a better job by remapping using the asinh option in magmap:
datastretch=cbind(datastretch,magmap(datastretch[,2],lo=-1e4,hi=1e4,range=c(0,1),
type='num',stretch='asinh')$map)
asinhticks=magmap(c(-10^(4:0),0,10^(0:4)),lo=-1e4,hi=1e4,range=c(0,1),type='num',
stretch='asinh')$map
magplot(datastretch[,1],datastretch[,3],side=1)
axis(2,asinhticks,labels=c(-10^(4:0),0,10^(0:4)))
abline(h=magmap(0,lo=-1e4,hi=1e4,range=c(0,1),type='num',stretch='asinh')$map)
```

---

magplot *Magically pretty plots*

Description

Makes scientific plots based on magaxis axes. Particularly designed for log plotting. Utilises base plot for the most part, but the axis drawing is replaced by a call to the magaxis fuction.

Usage

```
magplot(x, y, log = "", main = "", side = 1:2, majorn = 5, minorn = 'auto', tcl = 0.5,
ratio = 0.5, labels = TRUE, unlog = "auto", mgp = c(2,0.5,0), mtline = 2, xlab = '',
ylab = '', crunch = TRUE, logpretty = TRUE, prettybase = 10, powbase=10, hersh = FALSE,
family = "sans", frame.plot = TRUE, usepar=FALSE, grid=FALSE, grid.col='grey',
grid.lty=1, grid.lwd=1, axes=TRUE, xlim=NULL, ylim=NULL, ...)
```

Arguments

x	The x coordinates of points in the plot. Alternatively, a single plotting structure, function or any R object with a plot method can be provided.

`y`	The y coordinates of points in the plot, optional if x is an appropriate structure.
`log`	Log axis arguments to be passed to plot. E.g. use 'x', 'y', 'xy' or 'yx' as appropriate. Default " assumes no logging of any axes.
`main`	Title for the plot. Default is no title.
`side`	The side to be used for axis labelling in the same sense as the base axis function (1=bottom, 2=left, 3=top, 4=right). A vector of multiple entries is allowed. By default, bottom and left axes are drawn (i.e. side 1 and 2). If '`side`'=FALSE (or '`axes`'=FALSE) then no sides or labels will be drawn.
`majorn`	The target number of major-axis sub-divisions for pretty plotting. If length is 1 and length of side is longer than this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. Obvious reason for varying this is different pretty labelling between a and y axes.
`minorn`	The exact number of minor-axis divisions (i.e. desired minor ticks + 1) to display in plotting. Auto will produce `pretty` ticks for linear scaling, and powbase-2 minor ticks for logged (this might seem odd, but for base 10 this means ticks at 2/3/4/5/6/7/8/9, which is probably as desired). If set manually, must be greater than 1 to have a visible effect. Minor ticks are always calculated to be equally spaced in linear space, so tick spaces vary when using log plotting. If length is 1 and length of side is longer than this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. An obvious reason for varying this is different pretty labelling between x and y axes.
`tcl`	The length of major tick marks as a fraction of the height of a line of text. By default these face into the plot (in common with scientific plotting) with a value of 0.5, rather than the R default of -0.5. It is possible to force magaxis to inherit directly from par by setting usepar=TRUE (see below). See `par` for more details.
`ratio`	Ratio of minor to major tick mark lengths.
`labels`	Specifies whether major-axis ticks should be labelled for each axis. If length is 1 and length of side is longer than this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. Default is to label all axes.
`unlog`	Determines if axis labels should be unlogged. If axis is found to be logged in par('usr') then the minor ticks are automatically log spaced, however "unlog" still controls how the labelling is done: either logged form (FALSE) or exponent form (TRUE). If axis has been explicitly logged (e.g. log10(x)) then this will can produce exponential axis marking/ labelling if set to TRUE. This case will also produce log minor tick marks. If length of unlog is 1 and length of side is longer than 1 then the assigned unlog value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. Can also take the text argument 'x', 'y', 'xy' or 'yx', where these refer to which axes have been logged. If left at the default of 'auto' then unlog is assumed to be true when the axis in question is logged, and false otherwise.
`mgp`	The margin line (in mex units) for the axis title, axis labels and axis line. This has different (i.e. prettier) defaults than R of c(2,0.5,0) rather than c(3,1,0). This pushes the numbers and labels nearer to the plot compared to the defaults. It is

	possible to force magaxis to inherit directly from par by setting usepar=TRUE (see below). See `par` for more details.
`mtline`	Number of lines separating axis name from axis.
`xlab`	x axis name.
`ylab`	y axis name.
`crunch`	In cases where the scientific text would be written as 1x10^8, should the 1x be removed so it reads 10^8. If length is 1 and length of side is longer then this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. TRUE by default.
`logpretty`	Should the major-ticks only be located at powers of 10. This changes cases where ticks are placed at 1, 3.1, 10, 31, 100 etc to 1, 10, 100. If length is 1 and length of side is longer then this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side. TRUE by default.
`prettybase`	The unit of repitition desired. By default it is 10, implying a pretty plot is one with marks at 10, 20, 30 etc. If you are plotting degrees then it might be prettier to display 90, 180, 270 etc. In which case prettybase should be set to 90. If log=TRUE then the reference location of 10 is changed, so in the previous example the labels generated would be at 9, 90, 900 etc rather than the deafult of 1, 10, 100 etc. If length is 1 and length of side is longer then this value is used for all axes. If length of arguments is longer than 1 then these should tally with the relevant axes in side.
`powbase`	Set the base to use for logarithmic axes. Default is to use 10.
`hersh`	To determines whether all plot text should be passed using Hershey vector fonts. This applies to the axis labels (which are handled automatically) and the axis names. In the case of axis names the user must be careful to use the correct plot utils escape characters: http://www.gnu.org/software/plotutils/manual/en/html_node/Text-String-Format.html. magaxis will return back to the current plotting family after the function has executed.
`family`	Specifies the plotting family to be used. Allowed options are 'sans' and 'serif'. Depending on whether hersh is TRUE or FALSE these otions are either applied to the Hershey vector fonts (hersh=TRUE) or the default R Helvetica font (hersh=FALSE). magaxis will return back to the current plotting family after the function has executed.
`frame.plot`	Logical indicating whether a box should be drawn around the plot.
`usepar`	Logical indicating whether tcl and mgp should be forced to inherit the global par values. This might be preferred when you want to define global plot settings at the start of a script.
`grid`	Logical indicating whether a background grid should be drawn onto the plotting area. If true this will generate vertical and horiztonal grid lines. For more control (i.e. to only draw horizontal or verical lines) see `link{magaxis}`.
`grid.col`	The colour of the grid to be drawn.
`grid.lty`	The line type of the grid to be drawn.
`grid.lwd`	The line width of the grid to be drawn.

`axes`	If ‘`axes`’=FALSE (or ‘`side`’=FALSE) then no sides or labels will be drawn.
`xlim`	Vector; range of data to display. Default of NULL shows the full range. If length equals 1 then the argument is taken to mean the sigma range to select for plotting and the clipping is done by `magclip`. If this is set to ’auto’ then the limits will be estimated from the data dynamically.
`ylim`	Vector; range of data to display. Default of NULL shows the full range. If length equals 1 then the argument is taken to mean the sigma range to select for plotting and the clipping is done by `magclip`. If this is set to ’auto’ then the limits will be estimated from the data dynamically.
`...`	Further arguments to be passed to base `plot` and also to `magaxis` -> `axis`.

Details

This is a simple function that just turns off most of the plotting output of base plot, and replaces where possible those present in magaxis.

If ‘`x`’ is a data.frame with more than 2 columns then the utility base `plot` data.frame plotting function is used to create a full plotting grid. This ignores `magaxis` settings entirely.

Setting ‘`xlim`’ and ‘`ylim`’

Value

No output. Run for the side effect of producing nice plotting axes.

Author(s)

Aaron Robotham

See Also

`magaxis`,`maglab`,`magerr`,`magmap`,`magrun`

Examples

```
x=10^{1:9}
y=1:9
magplot(log10(x),y,unlog='x')
magplot(x,y,log='x')

#Not ideal to have two decades between major labels:

magplot(x,y,log='x',majorn=c(10,5))
magplot(x,y,log='xy',majorn=c(10,5,5,5),side=1:4)

#Sometimes it is helpful to focus on where most of the data actually is.
#Using a single value for xlim and ylim sigma clips the data to that range.
#Here a value of 2 means we only show the inner 2-sigma (2% to 98%) range.
#The 'auto' option allows magclip to dynamically estimate a clip value.

temp=cbind(rt(1e3,1.5),rt(1e3,1.5))
```

```
magplot(temp)
magplot(temp, xlim=2, ylim=2)
magplot(temp, xlim='auto', ylim='auto')

#Some astronomy related examples (and how to display the solar symbol):

temp=cbind(runif(10,8,12),runif(10,0,5))

magplot(temp[,1:2], xlab=expression(M['\u0298']), ylab=expression(M['\u0298']/Yr), unlog='xy')
```

---

magproj *Magic longitude / latitude projection function*

Description

High level methods for producing pretty plot of projected data. Particularly useful in astronomy or geography, where many datasets are in longitude (right ascension) / latitude (declination) format. magproj is the highest level function, creating a projected image grid with labels and data. magprojgrid and magprojlabels are functions to simply overplot a grid and add labels respectively.

Usage

```
magproj(long, lat, type = "b", plottext, longlim = c(-180, 180), latlim = c(-90, 90),
projection = "aitoff", parameters = NULL, centre = c(0, 0), add = FALSE, fliplong=FALSE,
nlat=6, nlong=6, prettybase=30, labels = TRUE, grid=TRUE, grid.col = "grey", grid.lty = 2,
auto = FALSE, upres = 100, box = TRUE, labloc = c(90, -45), labeltype = "deg",
crunch=FALSE, ...)
magprojgrid(nlat=6, nlong=6, prettybase=30, box=TRUE, ...)
magprojlabels(nlat=6, nlong=6, prettybase=30, labloc = c(90, -45), labeltype='deg',
crunch=FALSE, ...)
```

Arguments

`long`	Vector of longitude values to use. If this is a matrix or data.frame with two columns and ‘`lat`’ is missing then column 1 is taken to be longitude values and column 2 is taken to be latitude values. long should have 2 elements only when ‘`type`’="b".
`lat`	Vector of latitude values to use. If the input for ‘`long`’ is a matrix or data.frame with two columns and lat is missing then column 1 is taken to be longitude values and column 2 is taken to be latitude values. lat should have 2 elements only when type="b".
`type`	The display type, either points (p), lines (l), polygon (pl), text (t), or box (b, the default). Points simply projects longitude and latitude positions into particle positions. Lines will join the positions together into a line, using `approxfun` to interpolate between positions at resolution ‘`upres`’. Polygon will join the positions together into a polygon, using `approxfun` to interpolate between positions

	at resolution 'upres'. Text will display the text provided in plottext at the positions. Box will draw a polygon box, where the limits are given as a two element vector for 'long' and a two element vector for 'lat'.
plottext	A vector of text to display at the provided longitude and latitude positions. Only used if 'type='t''.
longlim	The longitude limits to use in the plot. Vector of length 2.
latlim	The latitude limits to use in the plot. Vector of length 2.
projection	Map projection to use. This function directly uses mapproject, and all of the inputs allowed for the 'projection' argument in that function are also allowed here.
parameters	Map parameters to use. For details see the 'parameters' argument in mapproject.
centre	For most popular projections this argument specifies the longitude and latitude that is centred in the plot. Strictly 'orientation' in mapproject is set to c(90+centre[2], centre[1], 0).
add	Should a fresh plot be drawn ('add=FALSE'), or should the new data be added to the current plot ('add=TRUE').
fliplong	Should the the longitude axis be flipped so that low values are on the right hand side (normal for celestial sphere plots in astronomy).
nlong	The target number of gridlines in the longitude direction. Uses pretty, so the result may not be what is requested.
nlat	The target number of gridlines in the latitude direction. Uses pretty, so the result may not be what is requested.
prettybase	The unit of repitition desired for the grid lines and labels. See 'prettybase' in maglab. By default it is 30, implying a pretty plot is one with marks at 30, 60, 90 etc (i.e. attractive for large scale plots covering large longitude and latitude limits).
labels	Should text coordinate labels be added to the plot.
grid	Should a background grid be drawn.
grid.col	The colour of the background grid.
grid.lty	The line type for the background grid.
auto	If 'auto=FALSE' the plot is set up using all options specified. If 'auto=TRUE' then 'longlim', 'latlim', 'centre' and 'labloc' is estimated from the data. This mostly behaves sensibly, but do not be too surprised if the automatic plot is not ideal, and some manual tweaking is required.
upres	The resolution at which to do internal interpolation when drawing lines and boxes.
box	Should a black outline be drawn following the 'longlim' and 'latlim' limits.
labloc	The longitude and latitude at which labels should be drawn.
labeltype	Should the labels be drawn using degrees (deg) or column delimited sexigesimal (sex).
crunch	If set to FALSE the full output of deg2hms and deg2dms is printed. If set to TRUE a simplified output is used, where only the hours and degrees parts are extracted and appended with a 'h' and a degree symbol respectively.

`...`	For magproj, Extra options that are either passed to `points` (‘`type='p'`’), `lines` (‘`type='l'`’), `polygon` (‘`type='pl'`’), `text` (‘`type='t'`’), or `polygon` (‘`type='b'`’). For magprojgrid dots are pased to `lines` for drawing the grid lines. For magprojlabels dots are passed to `text` for adding text labels.

Value

No output. Run for the side effect of producing nice projected plots.

Author(s)

Aaron Robotham

See Also

`magplot`, `magaxis`, `maglab`, `magmap`, `magrun`, `magbar`, `magprojextra`

Examples

```
# GAMA fields:
par(mar=c(0.1,0.1,0.1,0.1))
magproj(c(129,141), c(-2,3), type='b', projection='aitoff', centre=c(180,0),
fliplong=TRUE, labloc=c(90,-45), col='red', labeltype = 'sex', crunch=TRUE)
magproj(c(211.5,223.5), c(-2,3), col='red', add=TRUE)
magproj(c(30.2,38.8), c(-10.25,-3.72), col='red', add=TRUE)
magproj(c(30.2,38.8), -6, type='l', add=TRUE, col='grey')
magproj(c(339,351), c(-35,-30), col='red', add=TRUE)

mageclipticl(width=10,col=hsv(1/12,alpha=0.3),border=NA)
mageclipticl(width=0,col='orange')
magMWplane(width=20,col=hsv(v=0,alpha=0.1),border=NA)
magMWplane(width=0,col='darkgrey')
magMW(pch=16, cex=2, col='darkgrey')
magsun(c(7,26), pch=16, cex=2, col='orange2') #An important date!

magproj(c(174,186), c(-3,2), col='red', add=TRUE)

#Plus SDSS:
magproj(c(110,260), c(-4,70), border='blue', add=TRUE)

magproj(c(35,135,180,217.5,345), c(-3.72,3,2,3,-30)+10, type='t',
plottext=c('G02','G09','G12','G15','G23'), add=TRUE)

legend('topleft', legend=c('GAMA Regions','SDSS Main Survey'), col=c('red','blue'),
pch=c(15,NA), lty=c(NA,1), bty='n')
legend('topright', legend=c('Ecliptic','MW Plane'), col=c(hsv(c(1/12,0), v=c(1,0),
alpha=0.5)), pch=c(15,15), lty=c(1,1), bty='n')
legend('bottomleft', legend=c('Sun', 'MW Centre'), col=c('orange2','darkgrey'), pch=16,
bty='n')
```

`magprojextra` *Attractive great circles and thick bands on magproj plots*

Description

High level functions to add great circles and thick bands on projections plots. In astronomy these are popular for indicating regions of exclusion surrounding the ecliptic or the Milky-Way plane. Also simple functions to add either the MW bluge to the current projection (magMW) or the sun on a given date (magsun).

Usage

```
magring(crosseq = 0, peaklat = 0, offset = 0, res = 1000, ...)
magband(crosseq = 0, peaklat = 0, width = 10, res = 1000, ...)
mageclıptic(width=10, ...)
magMWplane(width=10, ...)
magsun(Ydate='get', anti=FALSE, ...)
magMW(...)
```

Arguments

`crosseq`	The longitude below the latitude peak of the great circle (or centre of the band) where the centre crosses the equator. See examples to see how this is used in practice.
`peaklat`	The positive maximum latitude obtained by the great circle (or centre of the band). See examples to see how this is used in practice.
`offset`	Whether the ring drawn if systematically offset from the great cirlce defined by ‘`crosseq`’ and ‘`peaklat`’. Leave at 0 to draw a great circle.
`width`	How wide whould the band be in degrees. For `mageclıptic` and `magMWplane`, if this is zero it will draw a line instead.
`res`	Number of elements making up each side of the band (default should be fine for most plots).
`Ydate`	The date for the location of the Sun on the spherical grid. Vector in c(M,D) format. If set to 'get' then the function will return the Sun's location for today.
`anti`	Should the anti-sun position be computed (i.e. the RA and Dec of the position diametrically opposed to the Sun).
`...`	Arguments passed on to `lines` (`magring`), `polygon` (`magband`), `points` (`magMW` and `magsun`).

Value

No output. Run for the side effect of producing nice projected plots.

Author(s)

Aaron Robotham

See Also

magplot, magaxis, maglab, magmap, magrun, magbar, magproj

Examples

```
# GAMA fields:
par(mar=c(0.1,0.1,0.1,0.1))
magproj(c(129,141), c(-2,3), type='b', projection='aitoff', centre=c(180,0),
fliplong=TRUE, labloc=c(90,-45), col='red', labeltype = 'sex', crunch=TRUE)
magproj(c(211.5,223.5), c(-2,3), col='red', add=TRUE)
magproj(c(30.2,38.8), c(-10.25,-3.72), col='red', add=TRUE)
magproj(c(30.2,38.8), -6, type='l', add=TRUE, col='grey')
magproj(c(339,351), c(-35,-30), col='red', add=TRUE)

magecliptic(width=10,col=hsv(1/12,alpha=0.3),border=NA)
magecliptic(width=0,col='orange')
# Note this a shortcut for: magring(0,23.4,col='orange')
magMWplane(width=20,col=hsv(v=0,alpha=0.1),border=NA)
magMWplane(width=0,col='darkgrey')
# Note this a shortcut for: magring(76.75,62.6,col='darkgrey')
magMW(pch=16, cex=2, col='darkgrey')
magsun(c(7,26), pch=16, cex=2, col='orange2') #An important date!

magproj(c(174,186), c(-3,2), col='red', add=TRUE)

#Plus SDSS:
magproj(c(110,260), c(-4,70), border='blue', add=TRUE)

magproj(c(35,135,180,217.5,345), c(-3.72,3,2,3,-30)+10, type='t',
plottext=c('G02','G09','G12','G15','G23'), add=TRUE)

legend('topleft', legend=c('GAMA Regions','SDSS Main Survey'), col=c('red','blue'),
pch=c(15,NA), lty=c(NA,1), bty='n')
legend('topright', legend=c('Ecliptic','MW Plane'), col=c(hsv(c(1/12,0), v=c(1,0),
alpha=0.5)), pch=c(15,15), lty=c(1,1), bty='n')
legend('bottomleft', legend=c('Sun', 'MW Centre'), col=c('orange2','darkgrey'), pch=16,
bty='n')
```

---

magrun *Running averages*

Description

Computes running averages (medians / means / modes), user defined quantiles and standard deviations for x and y scatter data.

Usage

```
magrun(x, y, bins = 10, type='median', ranges = pnorm(c(-1, 1)), binaxis = "x",
equalN = TRUE, xcut, ycut, log='', Nscale=FALSE, diff=FALSE)
```

Arguments

`x`	Data x coordinates. This can be a 1D vector (in which case y is required) or a 2D matrix or data frame, where the first two columns will be treated as x and y.
`y`	Data y coordinates, optional if x is an appropriate structure.
`bins`	If a single integer value, how many bins the data should be split into. If a vector is provoided then these values are treated as the explicit bin limits to use.
`type`	The type of running average to determine. Options are 'median' (the default), 'mean', 'mode' and 'mode2d'. 'median' calculates the median for binned x and y values. 'mean' calculates the mean for binned x and y values. 'mode' uses the default R 'density' function, and finds the mode of the resulting smoothed 1D distributions for binned x and y values. 'mode2d' uses the MASS package 'kde2d' function, and finds the mode of the resulting smoothed 2d distribution for binned x and y values. 'cen' just calucates the geometric centre of the bin in x and y directions and is useful for using in conjuction with another 'type' option for plotting purposes. 'mean', 'mode' and 'mode2d' should be used with some thought if 'log' is used, since the central values will be determined for the logged data, which may or may not be desired.
`ranges`	The quantile ranges desired, can set to NULL if quantiles are not desired. The default adds 1-sigma equivilant quantile ranges.
`binaxis`	Which axis to bin across. Must be set to 'x' or 'y'.
`equalN`	Should the data be split into bins with equal numbers of objects (default, TRUE), or into regular spaces from min to max (FALSE). Only relevant if 'bins' paramter is set to a single integer value and 'magrun' is determining the explicit bin limits automatically.
`xcut`	A two element vector containing optional lower and upper x limits to apply to the data.
`ycut`	A two element vector containing optional lower and upper y limits to apply to the data.
`log`	Specify axes that should be logged. Allowed arguments are 'x'
`Nscale`	Sets whether the quantile ranges and standard deviations calculated are reduced with respect to the median by the square-root of the number of contributing data within each bin. The result of setting Nscale to TRUE is to scale the data like you are calculating the error-in-the-mean, rather than the scatter. For describing the 'significance' of trends in scatter data this is often what you want to show.
`diff`	Should the output quantiles and standard deviations be expressed as differences from the chosen type of running avergage (TRUE) or the actual values (default, FALSE). The advantage of the former is plotting the results as errorbars using magerr, which expects differences (so error like values). If set to TRUE then the output of 'xsd' and 'ysd' is a 1D vector rather than a data.frame with x/y-sd and x/y+sd columns. See the examples below for usage guidance.

Details

This function will be default calculate the running median along the x axis for y values, it is intended to be used to trace the spread in scattered data.

Value

`x`	The chosen averages (default median) of the x bins.
`y`	The chosen averages (default median) of the y bins.
`xquan`	Matrix containing the extra user defined x quantile ranges (columns are in the same order as the requested quantiles). If Nscale is set to TRUE then this is also divided by sqrt the contributing objects in each bin.
`yquan`	Matrix containing the extra user defined y quantile ranges (columns are in the same order as the requested quantiles). If Nscale is set to TRUE then this is also divided by sqrt the contributing objects in each bin.
`xsd`	The standard deviations in the x bins. This is a two column data.frame if 'diff' is set to FALSE, giving the x-sd and x+sd values, or a single vector if 'diff' is set to TRUE. If Nscale is set to TRUE then this is also divided by sqrt the contributing objects in each bin.
`ysd`	The standard deviations in the y bins. This is a two column data.frame if 'diff' is set to FALSE, giving the y-sd and y+sd values, or a single vector if 'diff' is set to TRUE. If Nscale is set to TRUE then this is also divided by sqrt the contributing objects in each bin.
`bincen`	The bin centres used in the chosen binning direction.
`binlim`	The bin limits used in the chosen binning direction.
`Nbins`	The number of items contributing to each running bin. This effectively produces a histogram counts output for the final bin limits.

Author(s)

Aaron Robotham

See Also

`magplot`,`magaxis`,`maglab`,`magerr`,`magmap`

Examples

```
#Simple example

temp=cbind(seq(0,2,len=1e4),rnorm(1e4))
temprun=magrun(temp)
magplot(temp,col='lightgreen',pch='.')
lines(temprun,col='red')
lines(temprun$x,temprun$yquan[,1],lty=2,col='red')
lines(temprun$x,temprun$yquan[,2],lty=2,col='red')
temprun=magrun(temp,binaxis='y')
lines(temprun,col='blue')
lines(temprun$xquan[,1],temprun$y,lty=2,col='blue')
lines(temprun$xquan[,2],temprun$y,lty=2,col='blue')

#Now with a gradient- makes it clear why the axis choice matters for simple line fitting.
```

```
temp=cbind(seq(0,2,len=1e4),rnorm(1e4)+1+seq(0,2,len=1e4))
temprun=magrun(temp)
magplot(temp,col='lightgreen',pch='.')
lines(temprun,col='red')
lines(temprun$x,temprun$yquan[,1],lty=2,col='red')
lines(temprun$x,temprun$yquan[,2],lty=2,col='red')
temprun=magrun(temp,binaxis='y')
lines(temprun,col='blue')
lines(temprun$xquan[,1],temprun$y,lty=2,col='blue')
lines(temprun$xquan[,2],temprun$y,lty=2,col='blue')

#Compare the different centres.

temp=cbind(seq(0,2,len=1e4),rnorm(1e4)^2+seq(0,2,len=1e4))
temprunmedian=magrun(temp,type='median')
temprunmean=magrun(temp,type='mean')
temprunmode=magrun(temp,type='mode')
temprunmode2d=magrun(temp,type='mode2d')
magplot(temp,col='grey',pch='.',ylim=c(-2,5))
lines(temprunmedian,col='red')
lines(temprunmean,col='green')
lines(temprunmode,col='blue')
lines(temprunmode2d,col='orange')

#Choose your own bins.

temp=cbind(seq(0,2,len=1e4),rnorm(1e4)+1+seq(0,2,len=1e4))
temprun=magrun(temp,bins=c(0.1,0.5,0.7,1.2,1.3,2))
magplot(temp,col='lightgreen',pch='.')
points(temprun,col='red')

#Show the 'error in the mean' type data points. Comparing to the best fit line,
#it is clear they are much more meaningful at reflecting the error in the trend seen,
#but not the distribution (or scatter) of data around this.

temp=cbind(seq(0,2,len=1e3),rnorm(1e3)+1+seq(0,2,len=1e3))
temprun=magrun(temp,bins=5)
temprunNscale=magrun(temp,bins=5,Nscale=TRUE)
magplot(temp,col='lightgreen',pch='.')
magerr(temprun$x,temprun$y,temprun$x-temprun$xquan[,1], temprun$y-temprun$yquan[,1],
temprun$xquan[,2]-temprun$x, temprun$yquan[,2]-temprun$y, lty=2,length=0,col='blue')
magerr(temprunNscale$x,temprunNscale$y,temprunNscale$x-temprunNscale$xquan[,1],
temprunNscale$y-temprunNscale$yquan[,1],temprunNscale$xquan[,2]-temprunNscale$x,
temprunNscale$yquan[,2]-temprunNscale$y,col='red')
abline(lm(temp[,2]~temp[,1]),col='black')

#Or the above type of plot can be done more simply using the 'diff' flag.

temprun=magrun(temp,bins=5,diff=TRUE)
temprunNscale=magrun(temp,bins=5,Nscale=TRUE,diff=TRUE)
magplot(temp,col='lightgreen',pch='.')
magerr(temprun$x,temprun$y,temprun$xquan[,1], temprun$yquan[,1], temprun$xquan[,2],
temprun$yquan[,2],lty=2,length=0,col='blue')
```

```
magerr(temprunNscale$x,temprunNscale$y,temprunNscale$xquan[,1], temprunNscale$yquan[,1],
temprunNscale$xquan[,2],temprunNscale$yquan[,2],col='red')
abline(lm(temp[,2]~temp[,1]),col='black')

#Similar, but using the 'sd' output.

magplot(temp,col='lightgreen',pch='.')
magerr(temprun$x,temprun$y,temprun$xsd,temprun$ysd,lty=2,length=0,col='blue')
magerr(temprunNscale$x,temprunNscale$y,temprunNscale$xsd,temprunNscale$ysd,col='red')
abline(lm(temp[,2]~temp[,1]),col='black')
```

magtri *High level triangle plotting code for MCMC chains*

Description

A very high level (minimal options) MCMC chain triangle plot function. The default is deliberately spartan in terms of options, but the result should be a clear set of covariance plots that should give quick insight into the stationary sampling quality of a set of MCMC posterior chains.

Usage

```
magtri(chains, samples, samptype = "end", grid = FALSE, tick = FALSE)
```

Arguments

`chains`	A matrix or data.frame of the posterior chains, arranged so that the columns are the parameters and rows are the individual chain samples. The column names are inherited as the parameter names from the input to `chains`.
`samples`	Specify the number of sub-samples desired. To speed up plotting it is often a good idea not to plot all chain samples (the reduced set is plotted as the top-left points and used to generate the bottom-right contours). The default (empty) will plot all samples provided by '`chains`'.
`samptype`	Specifies whether to take all of the samples from the end of the supplied `chains` ('end', the default since samples are usually better towards the end of a set of psoterior chain samples) or randomly selected ('ran', should only be used if you are confident the posterior chains supplied are true stationary samples).
`grid`	Should a background grid be added to the sub-panels? See `magaxis` for details.
`tick`	Should tick marks be added to every sub-panel?

Details

This interface is deliberately very high level with few options. It is really designed to allow quick exploratory views of posterior samples from MCMC chains, and publication grade plots should be designed by the user. That said, in many situations the plots generated are of pleasant, clear and publishable quality.

Other types of data can be plotted using this function of course, but the default setup is tuned towards being useful for MCMC posterior chain samples.

The contour levels shown are the defaults for magcon, i.e. they contain 50% (lty=2), 68% (lty=1) and 95% (lty=3) of the posterior chains.

The red cross shows the mean for the sampled posterior chain. The red vertical dashed line traces this over the contour plots. The red dotted line shows the +/- SD range of the sampled posterior chain.

Value

Outputs a two column matrix containing the means and standard deviations fo the parameters. Generally run for the side effect of producing nice projected plots.

Author(s)

Aaron Robotham

See Also

magcon

Examples

```
Sigma=matrix(c(10,3,-5,3,12,8,-5,8,20),3,3)
chains=mvrnorm(n=1000, mu=1:3, Sigma=Sigma)
magtri(chains,tick=TRUE)
```

Index