Package ‘magicaxis’

February 22, 2022

Type Package

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

Version 2.2.14

Date 2022-02-18

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

Suggests imager, fst

Imports grDevices, graphics, stats, celestial (>= 1.4.1), MASS, plotrix, sm, mapproj, RANN

NeedsCompilation no

Repository CRAN

Date/Publication 2022-02-22 03:30:02 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:	2.2.14
Date:	2022-02-18
License:	GPL-3
Depends:	R (>= 2.13)
Suggests:	imager, fst
Imports:	grDevices, graphics, stats, celestial (>= 1.4.1), MASS, plotrix, sm, mapproj, RANN

`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, axis.lwd = 1, ticks.lwd = axis.lwd, axis.col = 'black', do.tick = TRUE,
...)
```

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. If length 2 then specifies x and y axis separation respectively (else these are the same).
`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.
`axis.lwd`	The line width of the axis to be drawn. This is passed to '`lwd`' argument in `axis`.
`ticks.lwd`	The line width of the ticks to be drawn. This is passed to '`ticks.lwd`' argument in `axis`.
`axis.col`	Colour argument to pass directly to '`col`' in axis. It is a bit clunky to have to specify this, but the option 'col' clashes too much with line and point colours.
`do.tick`	Logical; should ticks be drawn? Passed to `axis` argument '`tick`'.
`...`	Other arguments to be passed to base `axis` and `mtext` functions as relevant. Options for axis are as described in `axis`, but note '`cex.lab`', '`col.lab`' and '`font.lab`' are parsed as '`cex`', '`col`' and '`font`' into `mtext` as required.

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 = hcl.colors(21), 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')
```

---

magbin *2D Binning Routines*

Description

Allows for 2D binning (counts) and summary statistics on 2D bins (medians etc).

Usage

```
magbin(x, y, z = NULL, xlim = NULL, ylim = NULL, zlim = NULL, step = NULL,
log = '', unlog = log, clustering = 10, dustlim = 0.1, shape = "hex",
plot = TRUE, colramp = hcl.colors(21), colstretch = "lin", sizestretch = "lin",
colref = "count", sizeref = "none", funstat = function(x) median(x, na.rm=TRUE),
direction = 'h', offset = 0, jitterseed = 666, projden = FALSE, ...)
```

Arguments

x	Numeric vector or matrix/data.frame; x values to 2D bin. If x is a two (or more) column matrix or data.frame and y/z is missing as an argument, then the first column is used for x and the second/third column for y/z.
y	Numeric vector; the y coordinates of points in the plot, optional if 'x' is an appropriate structure.
z	Numeric vector; the z coordinates of points in the plot (optional), optional if 'x' is an appropriate structure.
xlim	Numeric vector; the x limits to use for the data. Default of NULL calculates the range based on the provided 'x' data vector. 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.

`ylim`	Numeric vector; the y limits to use for the data. Default of NULL calculates the range based on the provided '`y`' data vector. 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`.
`zlim`	Numeric vector; the z limits to use for the data. Default of NULL calculates the range based on the provided '`y`' data vector. 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`.
`step`	Numeric vector; grid steps in x and y directions. IF NULL then this is c(diff(xlim)/50, diff(ylim)/50). If length 1, then this value is repeated.
`log`	Character scalar; log axis arguments to be passed to used. E.g. use 'x', 'y', 'xy' or 'yx' as appropriate. Default " assumes no logging of any axes. For convenience you can specify the 'z' axis too, which somewhat replaces the '`colstretch`' argument. Note that in all cases the '`x`'/'`y`'/'`z`' data is explictly logged, which means the plotting window does not know it is in logged space (via the par()$xlag and par()$ylog structures). This means is you want to add points etc to the plot you will need to apply log10 yourself, so a point with coordinates [10^2,10^-3] should be plotted at [2,-3].
`unlog`	Character scalar; determines if x/y axis labels should be unlogged (z is ignored here). By default inherits '`log`', since that is usually what you would want.
`clustering`	Numeric scalar; excess counts in densist bin relative to Uniform data. This is to optimise the binning, so can probably be ignored.
`dustlim`	Numeric scalar; if between 0 and 1 then the 2D bin count quantile to switch to showing the individual points (which visually look like 'dust'), if larger than 1 then the exact counts threshold. If this is NA or 0 then all cells are shown.
`shape`	Character scalar; type of binning, either hex/hexagon; sq/square; tri/triangle or trihex. 'trihex' is a triangle tessellation that is also arranged to have hexagonal packing (so 6 triangles can form a hexagon).
`plot`	Logical; create a plot? If FALSE then just the binning output list is created.
`colramp`	Vector; a colour scaling to use. Must be a vector and not a function.
`colstretch`	Character scalar; colour stretch, either linear (lin, default) or logarithmic (log, good for large dynamic ranges).
`sizestretch`	Character scalar; size stretch, either linear (lin, default) or logarithmic (log, good for large dynamic ranges).
`colref`	Character scalar; colour reference for call, either it should be based on the counts (count, default) or the z-axis statistic (zstat)?
`sizeref`	Character scalar; size reference for call, either it should be ignored (none, so all are the same size and closely packed), based on the counts (count) or the z-axis statistic (zstat)?
`funstat`	Function; function to use to compute a statistic over the '`z`' axis. The default is `median`, but other good options might be `mean`, `sd`, `mad`. Note, to change default arguments you might need to send through a new function, e.g. '`funstat`' = function(x) median(x, na.rm=TRUE) and similar, but if you are happy with the defaults then you can use the simpler '`funstat`' = mean etc.

`direction`	Character scalar; should there be a shape side aligned horizontally ('h', the default) or vertically ('v')? This is only relevant for hexagon and triangle bin shapes, and has the effect of leading the eye differently with some scatter structures.
`offset`	Numeric/character scalar; only relevant for '`shape`'='sq' or '`shape`'='tri'. Either a numeric value specifying the offset (relative to '`step`') to apply to alternating rows ('`direction`'='h') or columns ('`direction`'='v'); or 'jitter' which means the rows or columns are randomly jittered (only used for '`shape`'='sq' bins. This option is useful for visually breaking up strong patterns in certain types of data.
`jitterseed`	Integer scalar; the random seed to use for jittering (means you can recreate your plots exactly if the seed is the same). This argument is only used for '`shape`'='sq' bins.
`projden`	Logical; do you want projected density PDFs to be displayed above and to the side of the standard `plot.magbin` plot? If so you also need to pass the same '`xdata`' and '`ydata`' that you originally sent to `magbin`, since this is not stored in the object output from `magbin`.
`...`	Dots to be passed to `magplot`, `magmap` and `magbar`. Relevant arguments are matched, so look in those functions for optional arguments to pass.

Details

Mostly run for the side effect of making a nice plot, but the output bin statistics might also be useful.

Re performance, `magbin` works pretty well on a modern computer for up to ~1e6 points, taking only a few seconds to run usually. Beyond this you might need to carefully tune the performance with '`clutering`' otherwise it might run very slower and/or you run out of memory.

Value

List of class 'magbin' containing:

`bins`	Bin x / y / count / and zstat info
`dust`	Dust x / y / z info
`groups`	Links input '`x`' and '`y`' data to the nearest grid cell by row number of '`bins`'
`xlim`	x limits
`ylim`	y limits
`step`	'`step`' size
`dustlim`	'`dustlim`'
`shape`	'`shape`'
`direction`	'`direction`'

See Also

`plot.magbin`, `maghist`

Examples

```
set.seed(666)
xydata = cbind(rnorm(1e4), rnorm(1e4))
magbin(xydata, shape='hexagon') #default
magbin(xydata, shape='square')
magbin(xydata, shape='triangle')
magbin(xydata, shape='trihex')
magbin(xydata, shape='hexagon', direction='v')
magbin(xydata, shape='triangle', direction='v')
magbin(xydata, shape='trihex', direction='v')
magbin(xydata, shape='hexagon', step=c(0.2,0.4)) #different aspect ratio hexagons

magbin(xydata, z=xydata[,1]^2-xydata[,2]^2, colref='zstat', sizeref='count')

magbin(xydata, z=xydata[,1]^2-xydata[,2]^2, colref='zstat', sizeref='count',
funstat=mad)
magbin(xydata, z=xydata[,1]^2-xydata[,2]^2, colref='zstat', sizeref='count',
funstat=function(x){quantile(x,0.9)})

xydata = cbind(10^rnorm(1e4), 10^rnorm(1e4))
magbin(xydata, log='xy')
magbin(xydata, z=xydata[,1]*xydata[,2], colref='zstat', sizeref='count',
log='xyz')
magbin(xydata, log='xy', unlog='xy', xlim=3, ylim=3)
```

---

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', extra = TRUE)
```

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). The number of actual iterations is returns as '`clipiters`'.
`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).
`extra`	Logical; if TRUE then '`clip`' and '`range`' are computed and returns, else these are set to NA to reduce computation and memory.

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.
`clipiters`	The number of iterations made, which might be less than the input '`clipiters`' since it might converge faster.

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')

#Setting estimate to 'lo' in this case should work better:

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, ngrid = 100, add = FALSE,
xlab = '', ylab = '', imcol = c(NA,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 = NULL,...)
```

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).

`ngrid`	The ngrid to send to kde2d / sm.density 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 calculates 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 calculates 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])
```

magcurve *Draw Function Plots*

Description

Draws a curve corresponding to a function over the interval `[from,to]` using `magplot`. `curve` can plot also an expression in the variable `xname`, default 'x'. This is almost a direct port of `curve`, with use of `magplot` rather than `plot`.

Usage

```
magcurve(expr, from = NULL, to = NULL, n = 101, add = FALSE,
      type = "l", xname = "x", xlab = xname, ylab = NULL,
      log = NULL, xlim = NULL, ...)
```

Arguments

`expr`	The name of a function, or a call or an expression written as a function of `x` which will evaluate to an object of the same length as `x`.
`x`	a 'vectorizing' numeric R function.
`from, to`	the range over which the function will be plotted.
`n`	integer; the number of x values at which to evaluate.
`add`	logical; if `TRUE` add to an already existing plot; if `NA` start a new plot taking the defaults for the limits and log-scaling of the x-axis from the previous plot. Taken as `FALSE` (with a warning if a different value is supplied) if no graphics device is open.
`xlim`	`NULL` or a numeric vector of length 2; if non-`NULL` it provides the defaults for `c(from,to)` and, unless `add = TRUE`, selects the x-limits of the plot – see `plot.window`.
`type`	magplot type: see `plot.default`.
`xname`	character string giving the name to be used for the x axis.
`xlab, ylab, log, ...`	labels and graphical parameters can also be specified as arguments. See 'Details' for the interpretation of the default for `log`. For the `"function"` method of `magplot`, `...` can include any of the other arguments of `magcurve`, except `expr`.

Details

The function or expression `expr` (for `magcurve`) or function `x` (for `magplot`) is evaluated at `n` points equally spaced over the range `[from,to]`. The points determined in this way are then plotted.

If either `from` or `to` is `NULL`, it defaults to the corresponding element of `xlim` if that is not `NULL`.

What happens when neither `from/to` nor `xlim` specifies both x-limits is a complex story. For `magplot(<function>)` and for `magcurve(add = FALSE)` the defaults are $(0, 1)$. For `magcurve(add`

= NA) and magcurve(add = TRUE) the defaults are taken from the x-limits used for the previous plot. (This differs from versions of R prior to 2.14.0.)

The value of log is used both to specify the plot axes (unless add = TRUE) and how 'equally spaced' is interpreted: if the x component indicates log-scaling, the points at which the expression or function is plotted are equally spaced on log scale.

The default value of log is taken from the current plot when add = TRUE, whereas if add = NA the x component is taken from the existing plot (if any) and the y component defaults to linear. For add = FALSE the default is ""

This used to be a quick hack which now seems to serve a useful purpose, but can give bad results for functions which are not smooth.

For expensive-to-compute expressions, you should use smarter tools.

The way magcurve handles expr has caused confusion. It first looks to see if expr is a name (also known as a symbol), in which case it is taken to be the name of a function, and expr is replaced by a call to expr with a single argument with name given by xname. Otherwise it checks that expr is either a call or an expression, and that it contains a reference to the variable given by xname (using all.vars): anything else is an error. Then expr is evaluated in an environment which supplies a vector of name given by xname of length n, and should evaluate to an object of length n. Note that this means that magcurve(x,...) is taken as a request to plot a function named x (and it is used as such in the function method for magplot).

Value

A list with components x and y of the points that were drawn is returned invisibly.

Warning

For historical reasons, add is allowed as an argument to the "function" method of plot, but its behaviour may surprise you. It is recommended to use add only with magcurve.

See Also

curve, magplot

Examples

```
magcurve(sin, -2*pi, 2*pi, xname = "t")
magcurve(tan, xname = "t", add = NA,
      main = "magcurve(tan)  --> same x-scale as previous plot")

op <- par(mfrow = c(2, 2))
magcurve(x^3 - 3*x, -2, 2, ylab='y')
magcurve(x^2 - 2, add = TRUE, col = "violet")

## simple and advanced versions, quite similar:
magcurve(cos, xlim = c(-pi, 3*pi), n = 1001, col = "blue", add = TRUE)

chippy <- function(x) sin(cos(x)*exp(-x/2))
magcurve(chippy, -8, 7, n = 2001)
```

```
for(ll in c("", "x", "y", "xy"))
   magcurve(log(1+x), 1, 100, log = ll,
         sub = paste("log= '", ll, "'", sep = ""))
par(op)
```

---

magcutout *Image Cutout Utilities*

Description

Functions to subset both raw images and images with associated WCS systems.

Usage

```
magcutout(image, loc = dim(image)/2, box = c(100, 100), shiftloc = FALSE, paddim = TRUE,
  padval = NA, plot = FALSE, ...)

magcutoutWCS(image, header, loc, box = c(100, 100), shiftloc = FALSE, paddim = TRUE,
 padval = NA, plot = FALSE, CRVAL1 = 0, CRVAL2 = 0, CRPIX1 = 0, CRPIX2 = 0, CD1_1 = 1,
  CD1_2 = 0, CD2_1 = 0, CD2_2 = 1, coord.type = "deg", sep = ":",
  loc.type=c('coord','coord'), approx.map = FALSE, ...)

magWCSradec2xy(RA, Dec, header, CRVAL1 = 0, CRVAL2 = 0, CRPIX1 = 0, CRPIX2 = 0, CD1_1 = 1,
  CD1_2 = 0, CD2_1 = 0, CD2_2 = 1, CTYPE1 = 'RA--TAN', CTYPE2 = 'DEC--TAN',
  loc.diff = c(0, 0), coord.type = "deg", sep = ":")

magWCSxy2radec(x, y, header, CRVAL1 = 0, CRVAL2 = 0, CRPIX1 = 0, CRPIX2 = 0, CD1_1 = 1,
  CD1_2 = 0, CD2_1 = 0, CD2_2 = 1, CTYPE1 = 'RA--TAN', CTYPE2 = 'DEC--TAN',
  loc.diff = c(0, 0))
```

Arguments

image	Numeric matrix; required, the image we want to decorate. If 'image' is a list as created by readFITS, read.fits of magcutoutWCS then the image part of these lists is parsed to 'image' and the correct header part is parsed to 'header'. If 'image' is a path to an fst format file then the cutout can be done on disk directly.
header	Full FITS header in table or vector format. Legal table format headers are provided by the read.fitshdr function or the 'hdr' list output of read.fits in the astro package; the 'hdr' output of readFITS in the FITSio package or the 'header' output of magcutoutWCS. If a header is provided then key words will be taken from here as a priority. Missing header keywords are printed out and other header option arguments are used in these cases.
loc	Numeric vector; the [x,y] (magcutout) or [x,y] / [RA,Dec] (magcutoutWCS) location where we want to cutout the image. For magcutoutWCS the unit type can be specified with the 'loc.type' option. Either it is WCS in degrees [RA,Dec] (coord) or pixel [x,y] of the 'image' (image).

`box`	Numeric vector; the dimensions of the box to cut out from ‘`image`’ centred on ‘`loc`’. For `magcutout` the ‘`box`’ unit is always pixels. For `magcutoutWCS` the unit type can be specified with the ‘`loc.type`’ option. Either it is pixels or asec (see ‘`loc.type`’).
`RA`	Vector or matrix; target right ascension in degrees. If matrix then the first column will be used as ‘`RA`’ and the second column as ‘`Dec`’.
`Dec`	Vector; target declination in degrees. Ignored if ‘`RA`’ is a matrix.
`x`	Vector or matrix; target x-pixel. If Matrix then the first column will be used as the x-axis and the second column as y-axis. Note this is the R convention of [x,y] (see Notes) not FITS.
`y`	Vector; target y-pixel. Ignored if x is a matrix. Note this is the R convention of [x,y] not FITS (see Notes).
`loc.diff`	The pixel offset to apply. Only relevant if the image being plotted is a cutout from within a FITS legal image.
`shiftloc`	Logical; should the cutout centre shift from ‘`loc`’ away from the ‘`image`’ edge if the desired ‘`box`’ size extends beyond the edge of the ‘`image`’?
`paddim`	Logical; should the cutout dimensions be padded with with value of ‘`padval`’ for data outside the ‘`image`’ boundary (TRUE)? If FALSE the dimensions will truncate when the edge of the input ‘`image`’ has been reached.
`padval`	Numeric scalar; the value to use for padding if ‘`paddim`’=TRUE.
`plot`	Logical; should a `magimage` (`magcutout`) or `magimageWCS` (`magcutoutWCS`) plot of the output be generated?
`CRVAL1`	FITS header CRVAL1 for the ‘`CTYPE1`’ projection system. This is the RA in degrees at the location of ‘`CRPIX1`’.
`CRVAL2`	FITS header CRVAL2 for the ‘`CTYPE2`’ projection system. This is the Dec in degrees at the location of ‘`CRPIX2`’.
`CRPIX1`	FITS header CRPIX1 for the ‘`CTYPE1`’ projection system. This is the x pixel value at the location of ‘`CRVAL1`’.
`CRPIX2`	FITS header CRPIX2 for the ‘`CTYPE2`’ projection system. This is the y pixel value at the location of ‘`CRVAL2`’.
`CD1_1`	FITS header CD1_1 for the ‘`CTYPE1`’ projection system. Change in ‘`CTYPE1`’ in degrees along x-Axis.
`CD1_2`	FITS header CD1_2 for the ‘`CTYPE1`’ projection system. Change in ‘`CTYPE1`’ in degrees along y-Axis.
`CD2_1`	FITS header CD2_1 for the ‘`CTYPE2`’ projection system. Change in ‘`CTYPE2`’ in degrees along x-Axis.
`CD2_2`	FITS header CD2_2 for the ‘`CTYPE2`’ projection system. Change in ‘`CTYPE2`’ in degrees along y-Axis.
`CTYPE1`	The RA projection system type. Either ’RA–TAN’ for Tan Gnomonic (default), or ’RA–SIN’ for Sine Orthographic. ’RA–NCP’ is approximated by Sine Orthographic with a warning. Over-ridden by the FITS header.
`CTYPE2`	The DEC projection system type. Either ’DEC–TAN’ for Tan Gnomonic (default), or ’DEC–SIN’ for Sine Orthographic. ’DEC–NCP’ is approximated by Sine Orthographic with a warning. Over-ridden by the FITS header.

`coord.type`	The units of ‘`loc`’ for `magcutoutWCS`. Allowed options are ’deg’ for degress and ’sex’ for sexigesimal (i.e. HMS for RA and DMS for Deg).
`sep`	When ‘`coord.type`’=’sex’, ‘`sep`’ defines the type of separator used for the HMS and DMS strings (i.e. H:M:S and D:M:S would be sep=’:’, which is the default). See `hms2deg` and `dms2deg` for more details.
`loc.type`	Character vector; specifies what type of location is being provided. The first element specifies the coordinate type for ‘`loc`’ and the second element is the coordinate type for ‘`box`’. Either it is WCS in degrees [RA,Dec] / asec (’coord’) or pixel [x,y] of the ‘`image`’ (’image’). If only one element is provided then the same coordinate type is used for both ‘`loc`’ and ‘`box`’.
`approx.map`	Logical; should an approximate coordinate mapping scheme be computed? This should be left as the default FALSE if saving the cut down object, and only TRUE if you are experimenting with the image cutouts within the same R session.
`...`	Dots are parsed to either `magimage` (`magcutout`) or `magimageWCS` (`magcutoutWCS`).

Details

These functions are on a level trivial, since it is easy to subset matrices and therefore images within R. However these functions track important properties of the subset region that makes it easy to track its location with respect to the original image. Also, they allow direct plotting of the resultant cutout with the most appropriate image functions. In many cases these functions will be used purely for their plotting side effects.

The ‘`shiftloc`’ and ‘`paddim`’ control the behaviour of the function in the non-trivial case when the desired box size extendeds beyond the edge of the image. If ‘`shiftloc`’ is FALSE (the default behaviour), the cutout is guaranteed to be centred on the pixel specified by ‘`loc`’. Then, if ‘`paddim`’ is FALSE, the cutout extends only as far as possible until it reaches the edge of the image; otherwise if ‘`paddim`’ is TRUE the cutout image is padded with NAs in regions outside the supplied ‘`image`’ (the default behaviour). If ‘`shiftloc`’ is TRUE, the centre of the cutout will be shifted. In this case, if ‘`paddim`’ is FALSE, the cutout will extend at most half of the supplied ‘`box`’ size from the given ‘`loc`’; otherwise if ‘`paddim`’ is TRUE the cutout will be expanded until it reaches the desired ‘`box`’ size or spans the entire image.

Note that if ‘`shiftloc`’ is TRUE and ‘`paddim`’ is FALSE, the cutout can be larger than ‘`box`’; otherwise, the cutout is guaranteed to be no larger than the specified ‘`box`’ size.

Value

A list containing:

`image`	Numeric matrix; the cutout region of interst centred around ‘`loc`’.
`loc`	The new ‘`loc`’ vector that tranforms the input ‘`loc`’ x and y location to the new ‘`cutim`’ coordinates. This is in ProFit coordinates, so these values can be used when, e.g., constructing a ProFit modellist structure.
`loc.orig`	The original location is provided by the input ‘`loc`’.
`loc.diff`	The x and y offsets of the cutout compared to the original image, where ‘`loc`’ + ‘`loc.diff`’ = ‘`loc.orig`’ exactly.

`xsel`	Integer vector; the extracted x pixels from the original '`image`' that form '`cutim`'.
`ysel`	Integer vector; the extracted y pixels from the original '`image`' that form '`cutim`'.
`loc.WCS`	Only output from `magcutoutWCS`. Numeric vector of the central RA and Dec of the cutout region in degrees.
`scale.WCS`	Only output from `magcutoutWCS`. Numeric scalar value of the pixel scale in asec/pixel.
`usr.WCS`	Only output from `magcutoutWCS`. Numeric matrix of the RA/Dec extremes of the image in order BL/TL/BR/TR.
`approx.map`	Only output from `magcutoutWCS`. A helper function that will usually approximately map RA and Dec numeric vector inputs to a matrix with columns x and y. This is less accurate than `magWCSradec2xy`, so use that function is the projection is too extreme. Should work for most N-S aligned data pretty well though, and it saves having to correctly track the appropriate header.
`header`	Only output from `magcutoutWCS`. The updated header with the correct WCS for the cutout region. Basically this means CRPIX1.new=CRPIX1.org-loc.diff[1] and CRPIX2.new=CRPIX2.org-loc.diff[2].

Note

By R convention the bottom-left part of the bottom-left pixel when plotting the image matrix is c(0,0) and the top-right part of the bottom-left pixel is c(1,1), i.e. the mid-point of pixels are half integer values in x and y. This differs to the FITS convention of pixel mid points being integer values. As such the R [x,y] = FITS [x-0.5,y-0.5]. This rarely matters too much in practice, but for accurate overlays you will want to get it right (see Examples).

It is ambiguous what the desired outcome is in some cutting scenarios, e.g. what should be returned if a 3x3 cutout is requested at the "centre" of a 8x8 image? For this reason, and to avoid unexpected results due to numerical precision, you should only cut out even pixel dimensions if integer pixel coordinates are provided, and odd pixel dimensions if half-integer pixel coordinates are provided. Regardless, the '`loc`' and '`loc.orig`' outputs will always help you locate the absolute coordinates of your desired cut out centre in both the cut out and the original image coordinate system.

Author(s)

Aaron Robotham & Dan Taranu

See Also

`magimageWCS`

Examples

```
## Not run:
temp=matrix(1:121,11)

#The central value is at:

temp[6,6]
```

```
print(magcutout(temp, dim(temp)/2, box=c(3,3))$image)

#Given we cutout around the centre of the central pixel [5.5,5.5], the new centre
#relative to the cutout image output should be at [1.5,1.5]:

print(magcutout(temp, dim(temp)/2, box=c(3,3))$loc.orig)
print(magcutout(temp, dim(temp)/2, box=c(3,3))$loc)

# A simple WCS cutout example:

image=readFITS(system.file("extdata", 'VIKING/mystery_VIKING_Z.fits', package="ProFound"))

par(mar=c(3.1,3.1,1.1,1.1))
magcutout(image$imDat, loc=c(50.5,50.5), plot=TRUE)
magcutoutWCS(image, loc=c(50.5,50.5), loc.type = 'image', plot=TRUE)

#We can cut out using the coordinates instead:
print({tempcoord=magWCSxy2radec(50.5,50.5,header=image$hdr)})
magcutoutWCS(image, loc=tempcoord, loc.type=c('coord','image'), plot=TRUE)

#You can select coordinates too:

magcutoutWCS(image, loc=c(352.29167, -31.827777), box=c(30,30), plot=TRUE)$loc.WCS
magcutoutWCS(image, loc=c("23:29:10", "-31:49:40"), box=c(30,30) , coord.type = 'sex',
plot=TRUE)$loc.WCS

#We can add RA Dec specific decorations easily:

cutim=magcutoutWCS(image, loc=c(352.2918, -31.82652), box=c(30,30), plot=TRUE,
approx.map=TRUE)

#Approx overlays:

points(cutim$approx.map(c(352.2918, 352.2897), c(-31.82652, -31.8252)), pch=3, col='blue')

#Exact overlays:

points(magWCSradec2xy(c(352.2918, 352.2897), c(-31.82652, -31.8252), header=cutim$header),
col='red')

#Given we correctly modify the header, we can actually use the cut down image directly:

magimageWCS(cutim)

# Now test the various cutout size options for a large cutout near the image boundary

loc = c(300,340)
box = c(200,200)
loc.type = c("image","image")

magimage(image$imDat)
points(loc[1], loc[2], col='red')
```

```
# Setting shiftloc=FALSE and paddim=TRUE pads the image with NAs (default):

cutim=magcutout(image$imDat, loc=loc, box=box, plot=TRUE, shiftloc=FALSE, paddim=TRUE)
points(cutim$loc[1], cutim$loc[2], col='red')

cutim=magcutoutWCS(image, loc=loc, box=box, coord.type="image", loc.type=loc.type,
plot=TRUE, shiftloc=FALSE, paddim=TRUE)
points(cutim$loc[1], cutim$loc[2], col='red')

# The cutout is exactly the request size, but the centre is shifted:

cutim=magcutout(image$imDat, loc=loc, box=box, plot=TRUE, shiftloc=TRUE, paddim=TRUE)
points(cutim$loc[1], cutim$loc[2], col='red')

cutim=magcutoutWCS(image, loc=loc, box=box, coord.type="image", loc.type=loc.type,
plot=TRUE, shiftloc=TRUE, paddim=TRUE)
points(cutim$loc[1], cutim$loc[2], col='red')

# Setting paddim=FALSE returns the largest possible cutout within the image bounds,
# without shifting the centre:

cutim=magcutout(image$imDat, loc=loc, box=box, plot=TRUE, shiftloc=FALSE, paddim=FALSE)
points(cutim$loc[1], cutim$loc[2], col='red')

cutim=magcutoutWCS(image, loc=loc, box=box, coord.type="image", loc.type=loc.type,
  plot=TRUE, shiftloc=FALSE, paddim=FALSE)
points(cutim$loc[1], cutim$loc[2], col='red')

# Setting paddim=FALSE and shiftloc=TRUE returns a larger cutout, but with at most
# box/2 padding on either side:

cutim=magcutout(image$imDat, loc=loc, box=box, plot=TRUE, shiftloc=TRUE, paddim=FALSE)
points(cutim$loc[1], cutim$loc[2], col='red')

cutim=magcutoutWCS(image, loc=loc, box=box, coord.type="image", loc.type=loc.type,
  plot=TRUE, shiftloc=TRUE, paddim=FALSE)
points(cutim$loc[1], cutim$loc[2], col='red')

# Setting shiftloc=FALSE and requesting a box size larger than the image returns a cutout
# with the requested box size:

box = c(400,400)
cutim=magcutoutWCS(image, loc=loc, box=box, coord.type="image", loc.type=loc.type,
plot=TRUE, shiftloc=FALSE, paddim=TRUE)
points(cutim$loc[1], cutim$loc[2], col='red')

## End(Not run)
```

`magerr` *Error bar plotting*

Description

A function to add 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`	Numeric vector; x location of data.
`y`	Numeric vector; y location of data.
`xlo`	Numeric vector; error on the low side for x values. This can be positive or negative- the absolute vaue is used. If length 1, then will be repeated length('x') times.
`ylo`	Numeric vector; error on the low side for y values. This can be positive or negative- the absolute vaue is used. If length 1, then will be repeated length('x') times.
`xhi`	Numeric vector; 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. If length 1, then will be repeated length('x') times.
`yhi`	Numeric vector; 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. If length 1, then will be repeated length('x') times.
`corxy`	Numeric vector; if this parameter 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). If length 1, then will be repeated length('x') times.
`length`	Numeric vector; length of error bar ends. If length 1, then will be repeated length('x') times.
`col`	Either the colour of the error bars or the outline colour of the error ellipses. If length 1, then will be repeated length('x') times.
`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' and useful logging options.

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 = '', unlog = log, scale = 1,
cumsum = FALSE, p.test = NULL, ...)
```

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.

`unlog`	Determines if axis labels should be unlogged. E.g. use 'x', 'y', 'xy' or 'yx' as appropriate. See `magaxis`.
`scale`	Numeric scalar; an additional scaling parameter to apply to the frequnecy counts. This is useful if you want to get the y axis in to certain units, e.g. the counts represent objects in 10 square degrees of sky, so to get the y-axis into units of N/sq.deg you would set '`scale`'=1/10.
`cumsum`	Logical; if FALSE (default) then '`counts`' and '`density`' are totals for the current bin (usual sense of a histogram), if TRUE then '`counts`' and '`density`' are cumulative totals of all counts up to and including the current bin (always from lowest '`x`' upwards).
`p.test`	Function; optionally the user can provide a standard (or custom) p-test to be computed on the display (i.e. trimmed) data. This might be e.g. a Normality test with the `shapiro.test` function etc. The results are printed to screen below the other standard summary information if '`verbose`'=TRUE.
`...`	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, locut = 0.4, hicut = 0.995, flip = FALSE,
range = c(0, 1), type = "quan", stretch = "asinh", stretchscale = 'auto', bad = NA,
clip = "", axes = TRUE, frame.plot = TRUE, sparse = 'auto', qdiff = FALSE,
rem_med = FALSE, ...)
magimageRGB(x, y, R, G, B, saturation = 1, zlim, xlim, ylim, add = FALSE,
useRaster = TRUE, asp = 1, magmap = TRUE, locut = 0.4, hicut = 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 '`x`' can be a 3D array or list (containing elements R/G/B) 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 '`x`' can be a 3D array or list (containing elements R/G/B) 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 '`x`' can be a 3D array or list (containing elements R/G/B) instead of B for convenience, where B=x[,,3].
`saturation`	The visual saturation of the colours, a bit like the dial on a TV. 1 is native, 0 is black and white, 2 is very intense nigh-on trippy.
`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`	If TRUE then `magmap` scaling is applied to the z dimension (default), if FALSE then it is not.
`locut`	The low limit to clip the z data at (what this means varies depending on the 'type' option). For `magimageRGB` 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.
`hicut`	The high limit to clip the z data at (what this means varies depending on the 'type' option). For `magimageRGB` 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. For `magimageRGB` 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.
`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).
`qdiff`	Logical; if the z axis being plotted, this will set all the various parameters that make for a visually useful difference plot, where -ve difference values become blue, 0 difference values are yellow, and +ve difference values are red.
`rem_med`	Logical; should the `median` of the '`image`' be subtracted or not? This is often useful in conjunction with '`qdiff`' = TRUE.
`...`	Arguments to be parsed to `image` and `magaxis` as relevant (this is checked for internally by argument name). See `image` and `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

`magimageWCS`, `image`, `magcutout`, `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))

## Not run:
#Some astro data:
image=readFITS(system.file("extdata", 'VIKING/mystery_VIKING_Z.fits', package="ProFound"))

#A monotone image:
magimage(image$imDat)
#A faked colour image (this won't look great):
magimageRGB(R=image$imDat^3, G=image$imDat^1.5, B=image$imDat^2)

## End(Not run)
```

magimageWCS *Tan Gnomonic WCS Image Decoration*

Description

These functions add various WCS information to R images. It is particularly designed to work in unison with 'magimage' that is used extensively in the wider ProFit package.

Usage

```
magimageWCS(image, header, n, grid.col = "grey", grid.lty = 2, grid.lwd = 0.5,
lab.col = "green", coord.type = "sex", margin = TRUE, loc.diff = c(0, 0),
xlab = "Right Ascension", ylab = "Declination", mgp = c(2, 0.5, 0), mtline = 2,
position = "topright", com.col = "green", com.length = 0.05, coord.axis = 'auto',
pretty = 'auto', CRVAL1 = 0, CRVAL2 = 0, CRPIX1 = 0, CRPIX2 = 0, CD1_1 = 1, CD1_2 = 0,
CD2_1 = 0, CD2_2 = 1, CTYPE1 = 'RA--TAN', CTYPE2 = 'DEC--TAN', ...)

magimageWCSRGB(R, G, B, header_out, Rheader, Gheader, Bheader, dowarp='auto',
direction = "auto", boundary = "dirichlet", interpolation = "cubic", n, grid.col='grey',
grid.lty=2, grid.lwd=0.5, lab.col='green', coord.type='sex', margin=TRUE, loc.diff=c(0,0),
xlab='Right Ascension', ylab='Declination', mgp=c(2,0.5,0), mtline=2, position='topright',
com.col="green", com.length=0.05, coord.axis='auto', pretty='auto', CRVAL1=0, CRVAL2=0,
CRPIX1=0, CRPIX2=0, CD1_1=1, CD1_2=0, CD2_1=0, CD2_2=1, CTYPE1 = 'RA--TAN',
CTYPE2 = 'DEC--TAN', ...)

magimageWCSGrid(header, n, grid.col = "grey", grid.lty = 1, grid.lwd = 1,
coord.type = "sex", loc.diff = c(0, 0), pretty= 'auto', CRVAL1 = 0, CRVAL2 = 0,
CRPIX1 = 0, CRPIX2 = 0, CD1_1 = 1, CD1_2 = 0, CD2_1 = 0, CD2_2 = 1, CTYPE1 = 'RA--TAN',
CTYPE2 = 'DEC--TAN', ...)

magimageWCSLabels(header, n, lab.col = "green", coord.type = "sex", margin = TRUE,
loc.diff = c(0, 0), xlab = "Right Ascension", ylab = "Declination", mgp = c(2, 0.5, 0),
mtline = 2, coord.axis='auto', pretty= 'auto', CRVAL1 = 0, CRVAL2 = 0, CRPIX1 = 0,
CRPIX2 = 0, CD1_1 = 1, CD1_2 = 0, CD2_1 = 0, CD2_2 = 1, CTYPE1 = 'RA--TAN',
CTYPE2 = 'DEC--TAN', ...)

magimageWCSCompass(header, position = "topright", com.col = "green", com.length = 0.05,
loc.diff = c(0, 0), CRVAL1 = 0, CRVAL2 = 0, CRPIX1 = 0, CRPIX2 = 0, CD1_1 = 1, CD1_2 = 0,
CD2_1 = 0, CD2_2 = 1, CTYPE1 = 'RA--TAN', CTYPE2 = 'DEC--TAN', ...)
```

Arguments

image	Numeric matrix; required, the image we want to decorate. If 'image' is a list as created by readFITS, read.fits of magcutoutWCS then the image part of the list is parsed to 'image' and the correct header part is parsed to 'header'.
header	Full FITS header in table or vector format. Legal table format headers are provided by the read.fitshdr function or the 'hdr' list output of read.fits in

	the astro package; the 'hdr' output of readFITS in the FITSio package or the 'header' output of magcutoutWCS. If a header is provided then key words will be taken from here as a priority. Missing header keywords are printed out and other header option arguments are used in these cases.
R	Numeric matrix; containing the red colour values to be plotted (NAs are allowed). If 'R' is a list as created by readFITS, read.fits of magcutoutWCS then the image part of the list is parsed to 'R' and the correct header part is parsed to 'Rheader'.
G	Numeric matrix; containing the green colour values to be plotted (NAs are allowed). If 'G' is a list as created by readFITS, read.fits of magcutoutWCS then the image part of the list is parsed to 'G' and the correct header part is parsed to 'Gheader'.
B	Numeric matrix; containing the blue colour values to be plotted (NAs are allowed). If 'B' is a list as created by readFITS, read.fits of magcutoutWCS then the image part of the list is parsed to 'B' and the correct header part is parsed to 'Bheader'.
header_out	Full FITS header in table or vector format. This is the target WCS projection that the RGB image will be mapped onto. Legal table format headers are provided by the read.fitshdr function or the 'hdr' list output of read.fits in the astro package; the 'hdr' output of readFITS in the FITSio package or the 'header' output of magcutoutWCS. If a header is provided then key words will be taken from here as a priority. Missing header keywords are printed out and other header option arguments are used in these cases.
Rheader	Full 'R' FITS header in table or vector format. It is usually easier to include the header with the 'R' list input as described above.
Gheader	Full 'G' FITS header in table or vector format. It is usually easier to include the header with the 'G' list input as described above.
Bheader	Full 'B' FITS header in table or vector format. It is usually easier to include the header with the 'B' list input as described above.
dowarp	Image warping flag, either TRUE, FALSE or 'auto' (default). If TRUE then the images will be warped onto the 'header_out' WCS scheme, if FALSE they will not be (which will break if the WCSs are not truly the same and the iamges are pixel matched). 'auto' tries to detect if the headers are all the same, and if they are then it will set 'dowarp'=FALSE, otherwise it will set to TRUE. This can be fooled by small (and unimportant) differences in the headers, e.g. they have different SWarp dates etc, since it just checks if the text differs anywhere.
direction	Only used if re-mapping via magwarp. "auto" (default), "forward" or "backward", see imwarp. Since it is usally better to go from the higher resolution image and map this onto the lower resolution grid, "auto" selects the better direction given the pixel scales recovered from the header information.
boundary	Only used if re-mapping via magwarp. Boundary conditions: "dirichlet", "neumann", "periodic" (default "dirichlet"), see imwarp
interpolation	Only used if re-mapping via magwarp. "nearest", "linear", "cubic" (default "linear"), see imwarp
n	The target number of major-axis sub-divisions. Will not necessarily be achieved.

`grid.col`	The colour of the overlaid grid lines.
`grid.lty`	The line type of the overlaid grid lines.
`grid.lwd`	The line width of the overlaid grid lines.
`lab.col`	The colour of the labels when ‘`margin`’=FALSE.
`coord.type`	Should the labels be drawn using degrees (deg) or colon delimited sexigesimal (sex).
`margin`	Should the labels be drawn in the outer margin region (default).
`loc.diff`	The pixel offset to apply. Only relevant if the image being plotted is a cutout from within a FITS legal image.
`xlab`	x axis name. If left as default either H:M:S or D:M:S (‘`coord.type`’=’sex’) or deg (‘`coord.type`’=’deg’) will be appended.
`ylab`	y axis name. If left as default either H:M:S or D:M:S (‘`coord.type`’=’sex’) or deg (‘`coord.type`’=’deg’) will be appended.
`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. For ‘`margin`’=FALSE ‘`mgp`’ = -‘`mgp`’ - 3, which has the effect of shifting the tick labels nicely inside the margin.
`mtline`	Number of lines separating axis name from axis. For ‘`margin`’=FALSE ‘`mtline`’ = -‘`mtline`’, which has the effect of shifting the axis labels nicely inside the margin.
`coord.axis`	Integer vector; specifies whether the RA and Dec axes should be 1 or 2 (i.e. x or y axis). The default ’auto’ tries to guess based on the header information (and this usually works okay). Otherwise ‘`coord.axis`’=c(1,2) would be sensible for a N/S vertical aligend frame and ‘`coord.axis`’=c(2,1) would be sensible for a E/W vertically aligned frame. When the orientation is between the two then it is not always obvious which will work better.
`pretty`	If ’auto’ then it will try to compute the prettiest grid and label scaling *usually this is pretty good). Otherwise set to 1 to be degrees major ticks, 60 for minutes (DMS or HMS) and 3600 for seconds (DMS or HMS).
`position`	Relative position of the compass bar. This argument is used like the ’legend’ function. Specify one of ’bottom’, ’bottomleft’, ’left’, ’topleft’, ’top’, ’topright’, ’right’, ’bottomright’ and ’centre’.
`com.col`	Colour of the compass.
`com.length`	Length of the edges of the arrow head.
`CRVAL1`	FITS header CRVAL1 for the ‘`CTYPE1`’ projection system. This is the RA in degrees at the location of ‘`CRPIX1`’.
`CRVAL2`	FITS header CRVAL2 for the ‘`CTYPE2`’ projection system. This is the Dec in degrees at the location of ‘`CRPIX2`’.
`CRPIX1`	FITS header CRPIX1 for the ‘`CTYPE1`’ projection system. This is the x pixel value at the location of ‘`CRVAL1`’.
`CRPIX2`	FITS header CRPIX2 for the ‘`CTYPE2`’ projection system. This is the y pixel value at the location of ‘`CRVAL2`’.

`CD1_1`	FITS header CD1_1 for the '`CTYPE1`' projection system. Change in '`CTYPE1`' in degrees along x-Axis.
`CD1_2`	FITS header CD1_2 for the '`CTYPE1`' projection system. Change in '`CTYPE1`' in degrees along y-Axis.
`CD2_1`	FITS header CD2_1 for the '`CTYPE2`' projection system. Change in '`CTYPE2`' in degrees along x-Axis.
`CD2_2`	FITS header CD2_2 for the '`CTYPE2`' projection system. Change in '`CTYPE2`' in degrees along y-Axis.
`CTYPE1`	The RA projection system type. Either 'RA–TAN' for Tan Gnomonic (default), or 'RA–SIN' for Sine Orthographic. 'RA–NCP' is approximated by Sine Orthographic with a warning. Over-ridden by the FITS header.
`CTYPE2`	The DEC projection system type. Either 'DEC–TAN' for Tan Gnomonic (default), or 'DEC–SIN' for Sine Orthographic. 'DEC–NCP' is approximated by Sine Orthographic with a warning. Over-ridden by the FITS header.
`...`	These are parsed to `magimage`, (`magimageWCS`), `magimageRGB`, (`magimageWCSRGB`), `lines` (`magimageWCSGrid`), `axis` (`magimageWCSLabels`) or `arrows` (`magimageWCSCompass`).

Details

Most people will be content to use the higher level `magimageWCS` function, which calls (in order) `magimage`, `magimageWCSGrid`, `magimageWCSLabels` and `magimageWCSCompass`.

`magimageWCSRGB` can be used in a few different ways, but the recommended route is to supply combined image and header list objects to the '`R`', '`G`' and '`B`' arguments. If '`header_out`' is not supplied then by default it will project the three images onto the first available WCS header it finds searching '`R`', '`G`' and '`B`' in order. If you are happy to map the colour image into the '`R`' WCS then you do not need to supply '`header_out`' at all. Note if image remapping is required (the images are different sizes or the headers differ) then the `imager` library will need to be installed in order to use `magwarp`.

Value

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

Note

For convenience users can use the header outputs produced by both the `readFITS` and `astro` package.

By R convention the bottom-left part of the bottom-left pixel when plotting the image matrix is c(0,0) and the top-right part of the bottom-left pixel is c(1,1), i.e. the mid-point of pixels are half integer values in x and y. This differs to the FITS convention of pixel mid points being integer values. As such the R [x,y] = FITS [x-0.5,y-0.5]. This rarely matters too much in practice, but for accurate overlays you will want to get it right.

Author(s)

Aaron Robotham

See Also

magimage, magcutoutWCS, radec2xy, xy2radec

Examples

```
## Not run:
image=readFITS(system.file("extdata", 'VIKING/mystery_VIKING_Z.fits', package="ProFound"))

#Convenient image plotting for lists containing headers:

magimageWCS(image$imDat, header=image$hdr)
magimageWCS(image)

#First using the outer margins for tick labels:

par(mar=c(3.1,3.1,1.1,1.1))
magimageWCS(image)
magimageWCS(image, coord.type='deg')

#Now removing the margins and putting labels inside the image:

par(mar=c(0,0,0,0))
magimageWCS(image, margin=FALSE)
magimageWCS(image, margin=FALSE, coord.type='deg')

#We can make a WCS colour image of mismatched images:

VISTA_K=readFITS(system.file("extdata", 'VISTA_K.fits', package="magicaxis"))
VST_r=readFITS(system.file("extdata", 'VST_r.fits', package="magicaxis"))
GALEX_NUV=readFITS(system.file("extdata", 'GALEX_NUV.fits', package="magicaxis"))

magimageWCSRGB(VISTA_K, VST_r, GALEX_NUV)
magimageWCSRGB(VISTA_K, VST_r, GALEX_NUV, saturation=0.5)

#To make direct magimageRGB plots of the outputs you must turn off magmap scaling:

temp=magimageWCSRGB(VISTA_K, VST_r, GALEX_NUV)
magimageRGB(R=temp$R, G=temp$G, B=temp$B, magmap=FALSE)

#We can map onto various WCS schemes easily too:

magimageWCSRGB(VISTA_K, VST_r, GALEX_NUV, VISTA_K$hdr)
magimageWCSRGB(VISTA_K, VST_r, GALEX_NUV, VST_r$hdr)
magimageWCSRGB(VISTA_K, VST_r, GALEX_NUV, GALEX_NUV$hdr)

## End(Not run)
```

---

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, locut = 0, hicut = 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'.
`locut`	The low limit to clip the data at (what this means varies depending on the 'type' option). This should be a single value.
`hicut`	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, z = NULL, 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 = TRUE,
  grid.col = 'grey90', grid.lty = 1, grid.lwd = 1, xlim = NULL, ylim = NULL, lwd = 1,
  axis.lwd = 1, ticks.lwd = axis.lwd, axis.col = 'black', zcol = hcl.colors(21),
  zstretch = 'lin', dobar = TRUE,  ...)
```

Arguments

`x`	The x coordinates of points/lines 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/lines in the plot, optional if x is an appropriate structure.
`z`	The z coordinates for colour scaling of points in the plot. This will be passed through `magmap`, with dots passed as relevant.
`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 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. If length 2 then specifies x and y axis separation respectively (else these are the same).
`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.
`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.
`lwd`	The width of plot lines to be drawn. This has different behaviour depending on the plot type.
`axis.lwd`	The line width of the axis to be drawn. This is passed to '`lwd`' argument in `axis`.
`ticks.lwd`	The line width of the ticks to be drawn. This is passed to '`ticks.lwd`' argument in `axis`.
`axis.col`	Colour argument to pass directly to '`col`' in axis. It is a bit clunky to have to specify this, but the option 'col' clashes too much with line and point colours.
`zcol`	Vector; a colour palette to use for '`z`' mapped colours. Must be a vector and not a function.. Only relevant if data has been passed to '`z`'
`zstretch`	Character scalar; '`z`' colour stretch, either linear (lin, default) or logarithmic (log, good for large dynamic ranges).
`dobar`	Logical; should a colour bar be added to the plot?
`...`	Further arguments to be passed to: base `plot`; `magaxis` -> `axis`; `magmap` and `magbar` if '`z`' scaling is being used.

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')

#With z scaling

z=sqrt(9:1)
magplot(x, y, z, log='x', position='topleft')
```

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 colon 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)

magecliptic(width=10,col=hsv(1/12,alpha=0.3),border=NA)
magecliptic(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, ...)
magecliptic(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 `magecliptic` 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', 'y' and 'xy'
`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 (AKA corner) 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 = 1000, thin = 1, samptype = 'end', grid = FALSE, do.tick = FALSE,
refvals = NULL, lab = NULL, ...)
```

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 plot 1000 samples.
`thin`	Specify the thinning of chain samples, default (1) processes the whole chain.
`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), randomly selected ('ran', should only be used if you are confident the posterior chains supplied are true stationary samples) or evenly selected ('thin', which behaves much like thinning except we specify the target number of outputs, not the fraction of samples kept).
`grid`	Should a background grid be added to the sub-panels? See `magaxis` for details.
`do.tick`	Logical; should ticks be drawn on each sub-panel? Passed to `axis` argument '`tick`'.
`refvals`	Numeric vector; this gives reference values to overdraw. If provided it must be the same length as dim(chains)[2], i.e. this would usually be the number of parameters being compared in the triangle plot.
`lab`	Character vector; optional over-ride for column names when plotting the grid of scatter plots.
`...`	Extra arguments are passed to '`\link{magcon}`' for plotting.

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=MASS::mvrnorm(n=1000, mu=1:3, Sigma=Sigma)
magtri(chains,tick=TRUE)
```

---

magwarp *Remap Image WCS via Warping*

Description

Remaps an input Tan Gnomonic or Sine Orthographic projection system to a different target WCS.

Usage

```
magwarp(image_in, header_out = NULL, header_in = NULL, dim_out, direction = "auto",
boundary = "dirichlet", interpolation = "cubic", doscale = TRUE, CRVAL1_in = 0,
CRVAL2_in = 0, CRPIX1_in = 0, CRPIX2_in = 0, CD1_1_in = 1, CD1_2_in = 0, CD2_1_in = 0,
CD2_2_in = 1, CRVAL1_out = 0, CRVAL2_out = 0, CRPIX1_out = 0, CRPIX2_out = 0,
CD1_1_out = 1, CD1_2_out = 0, CD2_1_out = 0, CD2_2_out = 1, plot = FALSE, ...)
```

Arguments

`image_in`	Numeric matrix; required, the image we want to remap. If '`image_in`' is a list as created by `readFITS`, `read.fits` of `magcutoutWCS` then the image part of the list is parsed to '`image_in`' and the correct header part is parsed to '`header_in`'.
`header_out`	Full FITS header in table or vector format. This is the target WCS projection that '`image_in`' will be mapped onto. Legal table format headers are provided by the `read.fitshdr` function or the '`hdr`' list output of `read.fits` in the `astro` package; the '`hdr`' output of `readFITS` in the `FITSio` package or the '`header`' output of `magcutoutWCS`. If '`header_out`' is provided then key words will be taken from here as a priority. Missing header keywords are printed out and other header option arguments are used in these cases.
`header_in`	Full FITS header in table or vector format. This should be the header WCS that matches '`image_in`'. Legal table format headers are provided by the `read.fitshdr` function or the '`hdr`' list output of `read.fits` in the astro package; the '`hdr`' output of `readFITS` in the `FITSio` package or the '`header`' output of `magcutoutWCS`. If '`header_in`' is provided then key words will be taken from here as a priority. Missing header keywords are printed out and other header option arguments are used in these cases.
`dim_out`	Integer vector; this defines the desired dimensions of the output image. If this is not provided then the output image is made to be the same size as the NAXIS1 and NAXIS2 arguments taken from '`header_out`' (which is usually what you will want TBH).
`direction`	"auto" (default), "forward" or "backward", see `imwarp`. Since it is usally better to go from the higher resolution image and map this onto the lower resolution grid, "auto" selects the better direction given the pixel scales recovered from the header information.
`boundary`	boundary conditions: "dirichlet", "neumann", "periodic" (default "dirichlet"), see `imwarp`
`interpolation`	"nearest", "linear", "cubic" (default "linear"), see `imwarp`
`doscale`	Logical; if TRUE (default) then the image is scaled by the relative change in the pixel scale. This is usually what you want when converting images between different WCS (since we really want to make sure we conserve overall flux). If FALSE then nothing is done to rescale the image. This is useful when the input '`image_in`' is something like a segmentation map, where the actual values matter.
`CRVAL1_in`	FITS header CRVAL1 for the Tan Gnomonic projection system of the input WCS. This is the RA in degrees at the location of '`CRPIX1`'.
`CRVAL2_in`	FITS header CRVAL2 for the Tan Gnomonic projection system of the input WCS. This is the Dec in degrees at the location of '`CRPIX2`'.
`CRPIX1_in`	FITS header CRPIX1 for the Tan Gnomonic projection system of the input WCS. This is the x pixel value at the location of '`CRVAL1`'.
`CRPIX2_in`	FITS header CRPIX2 for the Tan Gnomonic projection system of the input WCS. This is the y pixel value at the location of '`CRVAL2`'.
`CD1_1_in`	FITS header CD1_1 for the Tan Gnomonic projection system of the input WCS. Change in RA-Tan in degrees along x-Axis.

`CD1_2_in`	FITS header CD1_2 for the Tan Gnomonic projection system of the input WCS. Change in RA-Tan in degrees along y-Axis.
`CD2_1_in`	FITS header CD2_1 for the Tan Gnomonic projection system of the input WCS. Change in Dec-Tan in degrees along x-Axis.
`CD2_2_in`	FITS header CD2_2 for the Tan Gnomonic projection system of the input WCS. Change in Dec-Tan in degrees along y-Axis.
`CRVAL1_out`	FITS header CRVAL1 for the Tan Gnomonic projection system of the output WCS. This is the RA in degrees at the location of '`CRPIX1`'.
`CRVAL2_out`	FITS header CRVAL2 for the Tan Gnomonic projection system of the output WCS. This is the Dec in degrees at the location of '`CRPIX2`'.
`CRPIX1_out`	FITS header CRPIX1 for the Tan Gnomonic projection system of the output WCS. This is the x pixel value at the location of '`CRVAL1`'.
`CRPIX2_out`	FITS header CRPIX2 for the Tan Gnomonic projection system of the output WCS. This is the y pixel value at the location of '`CRVAL2`'.
`CD1_1_out`	FITS header CD1_1 for the Tan Gnomonic projection system of the output WCS. Change in RA-Tan in degrees along x-Axis.
`CD1_2_out`	FITS header CD1_2 for the Tan Gnomonic projection system of the output WCS. Change in RA-Tan in degrees along y-Axis.
`CD2_1_out`	FITS header CD2_1 for the Tan Gnomonic projection system of the output WCS. Change in Dec-Tan in degrees along x-Axis.
`CD2_2_out`	FITS header CD2_2 for the Tan Gnomonic projection system of the output WCS. Change in Dec-Tan in degrees along y-Axis.
`plot`	Logical; should a `magimageWCS` plot of the output be generated?
`...`	Dots are parsed to either `magimageWCS` (only relevant if '`plot`'=TRUE).

Details

The function allows for arbitrary WCS remapping, as long as the input and output WCS both use the Tan Gnomonic projection system (which is by far the most common with modern survey imaging data). The process internally does the following:

- xy2radec; maps the input image to RA and Dec per pixel using the input header
- radec2xy; maps the pixel RA and Dec coordinates onto the desired output pixel grid using the output header

There are a few different ways to consider doing pixel remapping (or warping). The main question is usually whether to operate on a forward or backwards manner (see `imwarp`). Backwards mapping (the default) finds the best interpolation of every output pixel in the input image, whereas forward mapping finds the best interpolation of every input pixel on the output image. Backwards mapping usually provides the fewest artefacts (hence it is the default), but in general if the input image is higher resolution then you might prefer forward mapping and vica-versa.

The actual warping is done using the `imwarp` function in the `imager` package, so this will need to be installed prior to use (it is available on CRAN).

Value

A list containing:

`image`	Numeric matrix; the remapped image using the target WCS.
`header`	The target '`header_out`'.

Note

This function uses a bi-cubic interpolation scheme by default. It should approximately conserve the flux in '`image_in`', but this is not guaranteed. The 'linear' interpolation scheme may be closer to conserving flux in images with sharp features. The conservation is usually good to about 0.1% (i.e. 0.01 mag). If you require better conservation than this then alternative schemes (e.g. SWarp) will be required.

Author(s)

Aaron Robotham

See Also

`magimageWCSRGB`

Examples

```
## Not run:
VISTA_K=readFITS(system.file("extdata", 'VISTA_K.fits', package="magicaxis"))
VST_r=readFITS(system.file("extdata", 'VST_r.fits', package="magicaxis"))
GALEX_NUV=readFITS(system.file("extdata", 'GALEX_NUV.fits', package="magicaxis"))

magwarp(VST_r, GALEX_NUV$hdr, plot=TRUE)
magwarp(VISTA_K, GALEX_NUV$hdr, plot=TRUE)

magwarp(GALEX_NUV, VST_r$hdr, plot=TRUE)
magwarp(VISTA_K, VST_r$hdr, plot=TRUE)

magwarp(GALEX_NUV, VISTA_K$hdr, plot=TRUE)
magwarp(VST_r, VISTA_K$hdr, plot=TRUE)

#Check we can warp forwards and backwards correctly:

magimageWCS(GALEX_NUV)
magwarp(GALEX_NUV, GALEX_NUV$hdr, plot=TRUE)
magwarp(magwarp(GALEX_NUV, VST_r$hdr, interpolation='nearest'),
        GALEX_NUV$hdr, interpolation='nearest', plot=TRUE)

## End(Not run)
```

plot.magbin *Plot 2D Histogram or Bin Data*

Description

Plot 2D histogram counts or a third (z) axis summary statistic in 2D cells of different shapes.

Usage

```
## S3 method for class 'magbin'
plot(x, colramp = hcl.colors(21), colstretch = "lin",
  sizestretch = "lin", colref = "count", sizeref = "none", add = FALSE,
  dobar = TRUE, title = colref, colnorm = FALSE, projden = FALSE, xdata = NULL,
  ydata = NULL, pch.dust = '.', cex.dust = 1, ...)
```

Arguments

`x`	Object of class 'magbin'.
`colramp`	Vector; a colour palette to use. Must be a vector and not a function.
`colstretch`	Character scalar; colour stretch, either linear (lin, default) or logarithmic (log, good for large dynamic ranges).
`sizestretch`	Character scalar; size stretch, either linear (lin, default) or logarithmic (log, good for large dynamic ranges).
`colref`	Character scalar; colour reference for call, either it should be based on the counts (count) or the z-axis statistic (zstat)?
`sizeref`	Character scalar; colour reference for call, either it should be ignored (none, so all are the same size and closely packed), based on the counts (count) or the z-axis statistic (zstat)?
`add`	Logical; should bins be added to the current plot? If FALSE then a new plot will be made.
`dobar`	Logical; should a colour bar be added to the plot?
`title`	Character scalar; title to use for the '`\link{magbar}`' label.
`colnorm`	Logical; should the colour bar be normalised so the maximum value equals 1?
`projden`	Logical; do you want projected density PDFs to be displayed above and to the side of the standard `plot.magbin` plot? If so you also need to pass the same '`xdata`' and '`ydata`' that you originally sent to `magbin`, since this is not stored in the object output from `magbin`.
`xdata`	Numeric vector; the original x data sent to `magbin`. Only relevant if '`projden`'=TRUE.
`ydata`	Numeric vector; the original y data sent to `magbin`. Only relevant if '`projden`'=TRUE.
`pch.dust`	Scalar; pch symbol to use for the dust points.
`cex.dust`	Scalar; cex size to use for the dust points.
`...`	Dots to be passed to `magplot`, `magmap` and `magmap`. Relevant arguments are matched, so look in those functions for optional arguments to pass.

Value

Run for the side effect of making a nice plot.

Author(s)

Aaron Robotham

See Also

`magbin`

Examples

```
temp = magbin(rnorm(1e4), rnorm(1e4), plot=FALSE)
plot(temp, xlab='x', ylab='y')
```

Index