Package ‘magicaxis’

September 29, 2014

Type Package

Title Pretty scientific plotting with minor-tick and log minor-tick support

Version 1.9.3

Date 2014-09-25

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-2

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

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-29 17:53:04

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:	1.9.3
Date:	2014-09-25
License:	GPL-2
Depends:	MASS, plotrix

`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=1:4, majorn=5, minorn=5, tcl=0.5, ratio=0.5, labels=TRUE,
unlog='Auto', tline=0.5, mtline=2, xlab=NULL, ylab=NULL, box=FALSE,
crunch=TRUE, logpretty=TRUE, prettybase=10, hersh=FALSE, family='sans',...)
```

Arguments

`side`	The side to be used for axis labelling in the same sense as the base axis function. Multiple entries are allowed. If blank all 4 axes are 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 target number of minor-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.
`tcl`	The length of major ticks. By default these face into the plot (in common with scientific plotting).
`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.
`tline`	Number of lines separating labels from axis.
`mtline`	Number of lines separating axis name from axis.
`xlab`	x axis name.
`ylab`	y axis name.
`box`	Should a box be added around the plotting region.
`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. 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.
`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.
`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.
`...`	Other arguments to be passed to base axis function.

Details

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

Value

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

Author(s)

Aaron Robotham

See Also

`magplot`,`maglab`,`magerr`,`magmap`,`magrun`

Examples

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

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

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

`magbar` *Pretty colour bar*

Description

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

Usage

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

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).

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')
```

---

magcon *2D quantile images and contours*

Description

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

Usage

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

Arguments

`x`	x values to contour.
`y`	y values to contour.
`h`	Smoothing parameter to pass to kde2d. Can take 1 or 2 arguments for x and optionally y smoothing.
`doim`	Should an image be generated.
`docon`	Should contours be overlain.
`dobar`	Should a magbar colour bar be added describing the image levels (doim must also be true for this to appear).
`n`	The n to send to kde2d to determine the resolution of the smoothing.
`add`	Should the output of this function be added to the current plot. If FALSE then a new plot is generated.

`xlab`	Label for x-axis, only used if add=FALSE.
`ylab`	Label for y-axis, only used if add=FALSE.
`imcol`	The colour palette to use for the image (this is also sent to magbar).
`conlevels`	Specific quantile contours to add. Default is for 50
`barposition`	The position to use for magbar. See magbar help for more details.
`barorient`	The orientation to use for magbar. See magbar help for more details.
`bartitle`	Title to use for magbar.
`bartitleshift`	Control of how far the magbar title is shifted away from its default position.
`xlim`	The x limits to use for the data. Default of NULL caculates the range based on the provided x data vector. Data will be clipped between the extremes given. If xlim[1]>xlim[2] plotted axes will be flipped compared to default.
`ylim`	The y limits to use for the data. Default of NULL caculates the range based on the provided y data vector. Data will be clipped between the extremes given. If ylim[1]>ylim[2] plotted axes will be flipped compared to default.
`weights`	A vector of weights to pass onto sm.density (that does the 2D density estimate). This must be the same length as the x and y vectors if specified.
`...`	Other arguments to pass to the contour function, e.g. lty=c(2,1,3).

Details

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

Value

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

Author(s)

Aaron Robotham

See Also

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

Examples

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

magerr *Error bar plotting*

Description

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

Usage

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

Arguments

`x`	x location of data.
`y`	y location of data.
`xlo`	Error on the low side for x values. This can be positive or negative- the absolute vaue is used.
`ylo`	Error on the low side for y values. This can be positive or negative- the absolute vaue is used.
`xhi`	Error on the high side for x values. This can be positive or negative- the absolute vaue is used. By default this will inherit the xlo value.
`yhi`	Error on the high side for y values. This can be positive or negative- the absolute vaue is used. By default this will inherit the ylo value.
`corxy`	If this paramter exists then error ellipses will be drawn instead of error bars. It takes the value of the sigma_x sigma_y correlation, i.e. corxy=covxy/(xlo*ylo).
`length`	Length of error bar ends.
`col`	Either the colour of the error bars or the outline colour of the error ellipses.
`fill`	If fill=TRUE then the error ellipses will be filled, if FALSE then only the border will be drawn.
`...`	Further arguments to be passed to the arrows / draw.ellipse functions used to draw the error bars / error ellipses.

Value

Called for the side effect of plotting error bars.

Author(s)

Aaron Robotham

See Also

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

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])
```

---

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, 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.
`hersh`	Determines whether the text format output by maglab should be Hershey vector font compatable text (TRUE), or normal plotmath style expressions (FALSE).
`trim`	If trim is TRUE the outputs are not allowed to exceed the stated limits, if FALSE then the whole range of pretty values calculated are used. This will usually extend beyond the limits to ensure the plots look pretty, but if maglab is being used for something other than plotting axis labels then trimmed values might be useful.

Details

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

Value

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

Author(s)

Aaron Robotham

See Also

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

Examples

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

```
axis(1,at=log10(ticks$labat),labels=ticks$exp)

# Same outcome a different way:

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

# For small dynamic range

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

# Different base prettiness

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

magmap *Value remapper*

Description

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

Usage

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

Arguments

`data`	A vector of values. This can contain bad values (NA, NaN, infinite), but these will be ignored during mapping and set to the value of input parameter 'bad'.

`lo`	The low limit to clip the data at (what this means varies depending on the 'type' option). This should be a single value.
`hi`	The high limit to clip the data at (what this means varies depending on the 'type' option). This should be a single value.
`flip`	Should the scaling be flipped. This allows numbers from 0 to 10 to be mapped from 1 to 0 (so ordered back to front with respect to the input).
`range`	The numerical range of the output mapping which should be a vector of length two specifying c(low,high).
`type`	The type of mapping attempted. Options are 'quan' (default), 'num', 'sig' and 'rank'.
`stretch`	If stretch='lin' linear output mapping is used. If stretch='log' logarithmic mapping is used. If stretch='atan' atan mapping is used. If stretch='asinh' asinh mapping is used.
`stretchscale`	A number to multiply the data by before applying the stretch. This only has a user impact for stretch='atan' and stretch='atanh' since it controls what parts of the data is in the linear or logarithmic regime of the stretch procedure.
`bad`	Sets the value that NA, NaN and infinite input data should be set to in the final map output.
`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 'lo' and 'hi'.
`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

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))

#Combination of mapping:
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)
```

magplot *Magically pretty plots*

Description

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

Usage

```
magplot(x, y, log='', xlab=NULL, ylab=NULL, unlog='Auto', majorn=5, minorn=5,
main='', labels=TRUE, crunch=TRUE, logpretty=TRUE, prettybase=10, hersh=FALSE,
family='sans',side=1:2,frame.plot=TRUE,...)
```

Arguments

`x`	The x coordinates of points in the plot. Alternatively, a single plotting structure, function or any R object with a plot method can be provided.
`y`	The y coordinates of points in the plot, optional if x is an appropriate structure.
`log`	log axis arguments to be passed to plot. E.g. use ’x’, ’y’, ’xy’ or ’yx’ as appropriate. Default ” assumes no logging of any axes.
`xlab`	If desired, label for x-axis. Default produces no label.
`ylab`	If desired, label for y-axis. Default produces no label.
`unlog`	unlog argument to be passed to magaxis. If axis has been explicitly logged (e.g. log10(x)) then this will produce logged axis marking/ labelling if set to TRUE. If length is 1 then this value is used for all axes. Otherwise should be of length 4 (for the 4 sides: c(bottom,left,top,right)). Can also take the text argument ’x’, ’y’, ’xy’ or ’yx’, where these refer to which axes have been logged.
`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. Otherwise should be of length 4 (for the 4 sides: c(bottom,left,top,right)). Obvious reason for varying this is different pretty labelling between x and y axes.
`minorn`	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. Otherwise should be of length 4 (for the 4 sides: c(bottom,left,top,right)). Obvious reason for varying this is different pretty labelling between x and y axes.
`main`	Title for the plot. Default is no title.
`labels`	labels argument to be passed to magaxis. Specifies whether major-axis ticks should be labelled for each axis. If length is 1 then this value is used for all axes. Otherwise should be of length 4 (for the 4 sides: c(bottom,left,top,right)). Default is to label all axes.
`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.If length is 1 then this value is used for all axes. Otherwise should be of length 4 (for the 4 sides: c(bottom,left,top,right)).
`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 then this value is used for all axes. Otherwise should be of length 4 (for the 4 sides: c(bottom,left,top,right)). 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. Otherwise should be of length 4 (for the 4 sides: c(bottom,left,top,right)).
`hersh`	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.
`side`	The side to be used for axis labelling in the same sense as the base axis function. Multiple entries are allowed. If blank all 4 axes are drawn.
`frame.plot`	A logical indicating whether a box should be drawn around the plot.
`...`	Further arguments to be passed to base plot.

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.

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)

#Some astronomy related examples:

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')
```

magrun *Running averages*

Description

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

Usage

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

Arguments

`x`	Data x coordinates. This can be a 1D vector (in which case y is required) or a 2D matrix or data frame, where the first two columns will be treated as x and y.
`y`	Data y coordinates, optional if x is an appropriate structure.
`bins`	If a single integer value, how many bins the data should be split into. If a vector is provoided then these values are treated as the explicit bin limits to use.
`type`	The type of running average to determine. Options are 'median' (the default), 'mean', 'mode' and 'mode2d'. 'median' calculates the median for binned x and y values. 'mean' calculates the mean for binned x and y values. 'mode' uses the default R 'density' function, and finds the mode of the resulting smoothed 1D distributions for binned x and y values. 'mode2d' uses the MASS package 'kde2d' function, and finds the mode of the resulting smoothed 2d distribution for binned x and y values. 'cen' just calucates the geometric centre of the bin in x and y directions and is useful for using in conjuction with another 'type' option for plotting purposes. 'mean', 'mode' and 'mode2d' should be used with some thought if 'log' is used, since the central values will be determined for the logged data, which may or may not be desired.
`ranges`	The quantile ranges desired, can set to NULL if quantiles are not desired. The default adds 1-sigma equivilant quantile ranges.
`binaxis`	Which axis to bin across. Must be set to 'x' or 'y'.
`equalN`	Should the data be split into bins with equal numbers of objects (default, TRUE), or into regular spaces from min to max (FALSE). Only relevant if 'bins' paramter is set to a single integer value and 'magrun' is determining the explicit bin limits automatically.
`xcut`	A two element vector containing optional lower and upper x limits to apply to the data.
`ycut`	A two element vector containing optional lower and upper y limits to apply to the data.
`log`	Specify axes that should be logged. Allowed arguments are 'x'

`Nscale`	Sets whether the quantile ranges and standard deviations calculated are reduced with respect to the median by the square-root of the number of contributing data within each bin. The result of setting Nscale to TRUE is to scale the data like you are calculating the error-in-the-mean, rather than the scatter. For describing he '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.

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 etc
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')
```

Index