Plotting commands

Adapted from by Tom Short, EPRI Solutions, Inc., tshort@eprisolutions.com

High-level plotting commands (these create new plots)

`plot(x)` plot of the values of x (on the *y-axis)* ordered on the *x-axis*

`plot(x, y)` bivariate plot of x (on the *x-axis)* and y (on the *y-axis)*

`hist(x)` histogram of the frequencies of x

`barplot(x)` histogram of the values of x; use horiz=FALSE for horizontal bars

`dotchart(x)` if x is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)

`pie(x)` circular pie-chart NEVER USE THIS

`boxplot(x)` "box-and-whiskers" plot

`interaction.plot (f1, f2, y)` if f1 and f2 are factors, plots the means of y (on the *y-axis)* with respect to the values of f1 (on the *x-axis)* and of f2 (different curves); the option fun allows to choose the summary statistic of y (by default fun=mean)

`matplot(x,y)` bivariate plot of the first column of x *vs.* the first one of y, the second one of x *vs.* the second one of y, etc.

`pairs(x)` if x is a matrix or a data frame, draws all possible bivariate plots between the columns of x

`qqnorm(x)` quantiles of x with respect to the values expected under a normal law

`qqplot(x, y)` quantiles of y with respect to the quantiles of x

`contour(x, y, z)` contour plot (data are interpolated to draw the curves), x and y must be vectors and z must be a matrix so that dim(z)=c(length(x), length(y)) (x and y may be omitted)

`filled.contour(x, y, z)` id. but the areas between the contours are coloured, and a legend of the colours is drawn as well

`image(x, y, z)` id. but with colours (actual data are plotted) **`persp(x, y, z)`** id. but in perspective (actual data are plotted) **`stars(x)`** if x is a matrix or a data frame, draws a graph with segments or a star where each row of x is represented by a star and the columns are the lengths of the segments

`symbols(x, y, ...)` draws, at the coordinates given by x and y, symbols (circles, squares, rectangles, stars, thermometres or "boxplots") which sizes, colours . . . are specified by supplementary arguments

`termplot(mod.obj)` plot of the (partial) effects of a regression model (mod.obj)

Arguments common to many high-level plotting functions:

`add=FALSE` if TRUE superposes the plot on the previous one (if it exists)

`axes=TRUE` if FALSE does not draw the axes and the box

`type="p"` specifies the type of plot, "p": points, "l": lines, "b": points connected by lines, "o": id. but the lines are over the points, "h": vertical lines, "s": steps, the data are represented by the top of the vertical lines, "S": id. but the data are represented by the bottom of the vertical lines

`xlim=, ylim=` specifies the lower and upper limits of the axes, for example with xlim=c(1, 10) or xlim= range(x)

`xlab=, ylab=` annotates the axes, must be variables of mode character

`main=` main title, must be a variable of mode character

`sub=` sub-title (written in a smaller font)

Low-level plotting commands (these ADD to an existing plot)

`points(x, y)` adds points (the option type= can be used)

`lines(x, y)` id. but with lines

`text(x, y, labels, ...)` adds text given by labels at coordinates (x,y); a typical use is: plot(x, y, type="n"); text(x, y, names)

`mtext(text, side=3, line=0, ...)` adds text given by text in the margin specified by side (see axis() below); line specifies the line from the plotting area

`segments(x0, y0, x1, y1)` draws lines from points (x0, y0) to points (x1,y1)

`arrows(x0, y0, x1, y1, angle= 30, code=2)` id. with arrows at points (x0,y0) if code=2, at points (x1,y1) if code=1, or both if code=3; angle controls the angle from the shaft of the arrow to the edge of the arrow head

`abline(a,b)` draws a line of slope b and intercept a

`abline(h=y)` draws a horizontal line at ordinate y

`abline(v=x)` draws a vertical line at abcissa x

`abline(lm.obj)` draws the regression line given by lm.obj

`rect(x1, y1, x2, y2)` draws a rectangle which left, right, bottom, and top limits are x1, x2, y1, and y2, respectively

`polygon(x, y)` draws a polygon linking the points with coordinates given by x and y

`legend(x, y, legend)` adds the legend at the point (x,y) with the symbols given by legend

`title()` adds titles, sub-titles, axis labels – can be on outer edges of plotting region

`axis(side)` adds an axis at the bottom (side=1), on the left (2), at the top (3), or on the right (4); at=vect (optional) gives the abcissa (or ordinates) where tick-marks are drawn

`box()` draw a box around the current plot

`rug(x)` draws the data x on the *x-axis* as small vertical lines

`locator(n, type="n", ...)` returns the coordinates (x,y) after the user has clicked n times on the

plot with the mouse; also draws symbols (type="p") or lines (type="l") with respect to optional graphic parameters (...); by default nothing is drawn (type="n")

Graphical parameters

These can be set globally with **`par(...);`** many can be passed as parameters to plotting commands.

`adj` controls text justification (0 left-justified, 0.5 centred, 1 right-justified)

`bg` specifies the colour of the background (ex. : bg="red", bg="blue", . . . the list of the 657 available colours is displayed with colors())

`bty` controls the type of box drawn around the plot, allowed values are: "o", "l", "7", "c", "u" or "]" (the box looks like the corresponding character); if bty="n" the box is not drawn

`cex` a value controlling the size of texts and symbols with respect to the default; the following parameters have the same control for numbers on the axes, cex.axis, the axis labels, cex.lab, the title, cex.main, and the sub-title, cex.sub

`col` controls the color of symbols and lines; use color names: "red", "blue" see colors() or as "#RRGGBB"; see rgb(), hsv(), gray(), and rainbow(); as for cex there are: col.axis, col.lab, col.main, col.sub

`font` an integer which controls the style of text (1: normal, 2: italics, 3: bold, 4: bold italics); as for cex there are: font.axis, font.lab, font.main, font.sub

`las` an integer which controls the orientation of the axis labels (0: parallel to the axes, 1: horizontal, 2: perpendicular to the axes, 3: vertical)

`lty` controls the type of lines, can be an integer or string (1: "solid", 2: "dashed", 3: "dotted", 4: "dotdash", 5: "longdash", 6: "twodash", or a string of up to eight characters (between "0" and "9") which specifies alternatively the length, in points or pixels, of the drawn elements and the blanks, for example lty="44" will have the same effect as lty=2

`lwd` a numeric which controls the width of lines, default 1

`mar` a vector of 4 numeric values which control the space between the axes and the border of the graph of the form c(bottom, left, top, right), the default values are c(5.1, 4.1, 4.1, 2.1)

`mfcol` a vector of the form c(nr,nc) which partitions the graphic window as a matrix of nr lines and nc columns, the plots are then drawn in columns

`mfrow` id. but the plots are drawn by row

`pch` controls the type of symbol, either an integer between 1 and 25, or any single character

`ps` an integer which controls the size in points of texts and symbols

`pty` a character which specifies the type of the plotting region, "s": square, "m": maximal

`tck` a value which specifies the length of tick-marks on the axes as a fraction of the smallest of the width or height of the plot; if tck=1 a grid is drawn

`tcl` a value which specifies the length of tick-marks on the axes as a fraction of the height of a line of text (by default tcl=-0.5)

`xaxs, yaxs` style of axis interval calculation; default "r" for an extra space; "i" for no extra space

`xaxt` if xaxt="n" the *x-axis* is set but not drawn (useful in conjunction with axis(side=1, ...))

`yaxt` if yaxt="n" the *y-axis* is set but not drawn (useful in conjunction with axis(side=2, ...))

Graphics devices

`x11(), windows(), quartz()` open a graphics window

`postscript(file)` starts the graphics device driver for producing PostScript graphics; use horizontal = FALSE, onefile = FALSE, paper = "special" for EPS files; family= specifies the font (AvantGarde, Bookman, Courier, Helvetica, Times, or ComputerModern); width= and height= specifies the size of the region in inches (for paper="special", these specify the paper size).

`ps.options()` set and view (if called without arguments) default values for the arguments to postscript

`pdf, png, jpeg, bitmap` see ?Devices

`dev.off()` shuts down the specified (default is the current) graphics device;
see also **`dev.cur, dev.set`**