graphical functions in R

lorenz gygax

December 17, 2001

In general, help in R is either accessed by the html based help system or e. g. by writing `help(functionname)` at the R prompt (the help files always include executable examples). `demo(graphics)` shows some examples of the potential of using R for graphics. Information on R is available at `www.r-project.org`.

1 General procedure

To produce a graphic in R a device has to be opened first. This can be e. g. an empty window or a file. The commands producing the graphics are then executed. For the files, the device has to be closed to produce complete files. Several devices can be open at the same time and it is possible to switch among the devices. Plots and commands like closing are directed to the so called active device (generally the last that was in use).

1.1 Opening a device

A graphics window is opened using either of `x11()` or `X11()` (there is also: `win.graph()`, `windows()` for a windows environment). The multiple names are for compatibility with other systems.

`postscript()` directs the printing commands to a postscript file, `win.metafile()` windows meta format file and `win.print()` to the Windows print system.

The device opening functions take parameters to specify the size of the plotting area, file name, pointsize and similar. `postscript()` has some more parameters specifying e. g. whether to include several graphics in one file or put each graphic in a single file to be included in a document as encapsulated postscript files later on.

1.2 Device handling commands

`dev.cur()` returns the number and name of the current (active) device (starting to count at 2, because there is a null device with number one) and `dev.list()` returns the numbers and names of all devices. To switch between devices one can use `dev.next()`, `dev.prev()`, which return the number and name of the next/previous device in the list of devices, or `dev.set(X)`, which returns the name and number of the new active device with number X.

`bringToTop()` brings the specified screen device's window to the front of the window stack (and gives it focus).

1.3 Closing a device

The current device is closed using `dev.off()` and the Xth device by `dev.off(which= X)`. All graphics devices can be shut off using `graphics.off()`.

2 Plotting commands

In R, there are high level plotting functions, which set up a new graphic. I. e. in a graphic window a new plot is made while overwriting a pre-existing plot. Other functions add elements to the current plot.

It is possible to define standard plotting functions for specific object classes (and many have predefined plotting functions).

2.1 General plotting parameters

Using `par()`, the current settings of the graphic parameters can be checked and new defaults for the graphic parameters can be defined. There are many possible parameters to:

- choose the type of plot (`type`)
- specify the box surrounding the plot (`bty`) and the shape of the plot region (`pty`)
- specify the size of the margins (`mai`, `mar`) and the outer margins (`oma`, `omd`, `omi`) in a graphic with several plotting regions
- specify aspects of or suppress the axis (`lab`, `las`, `tck`, `tcl`, `xaxp`, `xaxs`, `xaxt`, `xlog`, `yaxp`, `yaxs`, `yaxt`, `ylog`)
- chose the symbol for plotted points (`pch`, `pin`) or the line type (`lty`) and width (`lwd`)
- specify multiple regions within one graphic (`fig`, `mfcol`, `mfrow`, `mfg`)
- specify the adjustment of text strings (adj), the size of the plotting symbols and strings in different functions (`cex`, `cex.axis`, `cex.lab`, `cex.main`, `cex.sub`, `cin`, `csi`, `csxi`) and the font (`font`, `font.axis`, `font.lab`, `font.main`, `font.sub`)
- specify colours (`bg`, `fg`, `col`, `col.axis`, `col.lab`, `col.main`, `col.sub`)
- suppress the annotations of axis and titles (`ann`)
- to prompt the user before a (new) figure is drawn (`ask`)
- and a variety of others details (see `help(par)`).

Many of the parameters that can be specified with `par()` can be directly specified in the plotting functions.

There are different methods to place several different plots on one page (into one graphic). Often it can be accomplished by setting the parameters `mfcol`, `mfrow` or `fig` with `par()`. But see also `layout()` for an approach to place figures in a matrix where not all the cells have the same size.

2.2 Creating a new plot

Some of the high level plots have a parameter that allows to overlay them on an already existing plot. The first call of a high level plot also defines the coordinate system of the current plot.

The most important plotting function is the generic `plot()`. With it scatterplots and line plots can be made or coordinate systems can be set up to use for any later user specified plotting of objects. Standard parameters include those for specifying the range of the axes (`ylim`, `xlim`), the axis labels (`ylab`, `xlab`) and a title if desirable (`main`).

Some standard forms of plots have predefined methods, such as `boxplot()`, `barplot()`, `hist()`. For small samples instead of boxplots and barplots, `stripchart()` and `dotchart()` can be used, repectively.

Three dimensional data can be plotted using `image()` (differential shading) or `contour()` (lines of equal 'elevation'), a combination of both or `persp()` (surfaces in 3D perspective; there is also a package for 3D scatterplots: `scatterplot3D`). Higher dimensional data can be plotted using `pairs()` (plotting every variable against every other variable) or using the R interface to XGobi and XGvis (GGobi for windows is under development) which allows e. g. easy spinning of high dimensional data.

Conditioning plots can be done using `coplot()` and with `matplot()` columns of one matrix can be plotted against the columns of another.

2.3 Adding elements to a plot

Points, lines (also fits from statistical models) and straight lines can be added to a plot using `points()`, `lines()` and `abline()`, respectively. One or several line segments or arrows can be plotted using `segments()` and `arrows()`, respectively. `polygons()` draws the polygons whose vertices are given in x and y.

If axes have been omitted in the high level call to a graphics function, they can be individually set up and adjusted using `axis()`. Text can be added to a graph using `text()`, `legend()` or `mtext()`. Using `expression()` it is possible to include mathematical expressions in text, titles, subtitles and x- and y-axis labels. The output will be formatted according to TEX -like rules (see also `help(plotmath)`)