

More Advanced Graphics in R

Martyn Plummer

University of Warwick, UK

SPE 2023, Tartu

Outline

Overview of graphics systems

Device handling

Base graphics

Grid graphics

Graphics Systems in R

R has several different graphics systems:

- ▶ Base graphics (the `graphics` package)
- ▶ Lattice graphics (the `lattice` package)
- ▶ Grid graphics (the `grid` package)
- ▶ Grammar of graphics (the `ggplot2` package)

Why so many? Which one to use?

Base Graphics

- ▶ The oldest graphics system in R.
- ▶ Based on S graphics (Becker, Chambers and Wilks, *The New S Language*, 1988)
- ▶ Implemented in the base package `graphics`
 - ▶ Loaded automatically so always available
- ▶ Ink on paper model; once something is drawn “the ink is dry” and it cannot be erased or modified.

Grid Graphics

- ▶ A complete rewrite of the graphics system of R, independent of base graphics.
- ▶ Programming with graphics:
 - ▶ Grid graphics commands create graphical objects (Grobs)
 - ▶ Printing a Grob displays it on a graphics device
 - ▶ Functions can act on grobs to modify or combine them
- ▶ Implemented in the base package `grid`, and extended by CRAN packages `gridExtra`, `gridDebug`, ...
- ▶ Described by the package author Paul Murrell in the book *R Graphics (2nd edition)*, 2011.

Grammar of Graphics

- ▶ Originally described by Leland Wilkinson in the book *The Grammar of Graphics*, 1999 and implemented in the statistical software nViZn (part of SPSS)
- ▶ Statistical graphics, like natural languages, can be broken down into components that must be combined according to certain rules.
- ▶ Provides a *pattern language* for graphics:
 - ▶ geometries, statistics, scales, coordinate systems, aesthetics, themes, ...
- ▶ Implemented in R in the CRAN package `ggplot2`
- ▶ Described more fully by the `ggplot2` package author Hadley Wickham in the book *ggplot2: Elegant Graphics for Data Analysis*, 2009.

Putting It All Together

- ▶ Base graphics are the default, and are used almost exclusively in this course
- ▶ `grid` provides alternate low-level graphics functions.
 - ▶ Experts only
- ▶ `ggplot2` is an alternate, high-level graphics package built on `grid`.
- ▶ All graphics packages take time to learn...

Graphics Devices

Graphics devices are used by all graphics systems.

- ▶ Plotting commands will draw on the current *graphics device*
- ▶ The *default* graphics device is a window on your screen:

 In RStudio `RStudioGD()`

 On Windows `windows()`

 On Unix/Linux `x11()`

 On Mac OS X `quartz()`

It normally opens up automatically when you need it.

- ▶ You can have several graphics devices open at the same time (but only one is current)

Graphics Device in RStudio

RStudio has its own graphics device `RStudioGD` built into the graphical user interface

- ▶ You can see the contents in a temporary, larger window by clicking the zoom button.
- ▶ You can write the contents directly to a file with the export menu
- ▶ Sometimes the small size of the `RStudioGD` device causes problems. Open up a new device calling `RStudioGD()`. This will appear in its own window, free from the GUI.

Writing Graphs to Files

There are also non-interactive graphics devices that write to a file instead of the screen.

`pdf` produces Portable Document Format files

`win.metafile` produces Windows metafiles that can be included in Microsoft Office documents (windows only)

`postscript` produces postscript files

`png`, `bmp`, `jpeg` all produce bitmap graphics files

- ▶ Turn off a graphics device with `dev.off()`. Particularly important for non-interactive devices.
- ▶ Plots may look different in different devices

Types of Plotting Functions

- ▶ High level
 - ▶ Create a new page of plots with reasonable default appearance.
- ▶ Low level
 - ▶ Draw elements of a plot on an existing page:
 - ▶ Draw title, subtitle, axes, legend ...
 - ▶ Add points, lines, text, math expressions ...
- ▶ Interactive
 - ▶ Querying mouse position (`locator`), highlighting points (`identify`)

Base x-y Plots

- ▶ The `plot` function with one or two numeric arguments
- ▶ Scatterplot or line plot (or both) depending on `type` argument: "l" for lines, "p" for points (the default), "b" for both, plus quite a few more
- ▶ Also: formula interface, `plot(y~x)`, with arguments similar to the modeling functions like `lm`

Adding to Plots in Base

- ▶ `title()` add a title above the plot
- ▶ `points()`, `lines()` adds points and (poly-)lines
- ▶ `text()` text strings at given coordinates
- ▶ `abline()` line given by coefficients (a and b) or by fitted linear model
- ▶ `axis()` adds an axis to one edge of the plot region.
Allows some options not otherwise available.

Strategy for Customization of Base Graphics

- ▶ Start with default plots
- ▶ Modify parameters (using `par()` settings or plotting arguments)
- ▶ Add more graphics elements. Notice that there are graphics parameters that turn things *off*, e.g. `plot(x, y, xaxt="n")` so that you can add completely customized axes with the `axis` function.
- ▶ Put all your plotting commands in a script or inside a function so you can start again

Demo 1

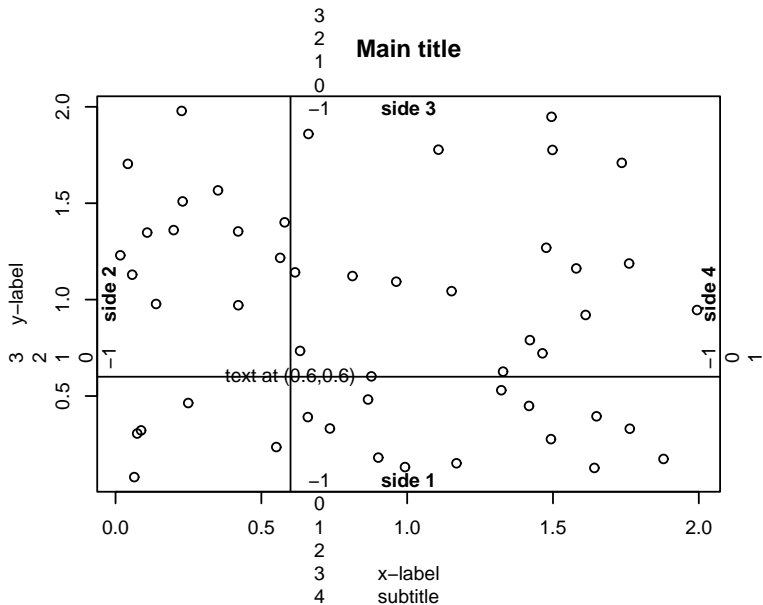
```
library(ISwR)
par(mfrow=c(2,2))
matplot(intake)
matplot(t(intake))
matplot(t(intake),type="b")
matplot(t(intake),type="b",pch=1:11,col="black",
        lty="solid", xaxt="n")
axis(1,at=1:2,labels=names(intake))
```


Margins

- ▶ R sometimes seems to leave too much empty space around plots (especially in multi-frame layouts).
- ▶ There is a good reason for it: You might want to put something there (titles, axes).
- ▶ This is controlled by the `mar` parameter. By default, it is `c(5, 4, 4, 2) + 0.1`
 - ▶ The units are *lines of text*, so depend on the setting of `pointsize` and `cex`
 - ▶ The sides are indexed in clockwise order, starting at the bottom (1=bottom, 2=left, 3=top, 4=right)
- ▶ The `mtext` function is designed to write in the margins of the plot
- ▶ There is also an *outer margin* settable via the `oma` parameter. Useful for adding overall titles etc. to multiframe plots

Demo 2

```
x <- runif(50,0,2)
y <- runif(50,0,2)
plot(x, y, main="Main title", sub="subtitle",
      xlab="x-label", ylab="y-label")
text(0.6,0.6,"text at (0.6,0.6)")
abline(h=.6,v=.6)
for (side in 1:4)
  mtext(-1:4,side=side,at=.7,line=-1:4)
mtext(paste("side",1:4), side=1:4, line=-1,font=2)
```



A Few Words on Grid Graphics

- ▶ Experts only, but ...
- ▶ Recall that `ggplot2` uses `grid`
- ▶ The key concepts you need are *grobs* and *viewports*

Grobs: Graphical Objects

- ▶ Grobs are created by plotting functions in `grid`, and `ggplot2`
- ▶ Grobs are only displayed when they are printed
- ▶ Grobs can be modified or combined before being displayed
- ▶ The `ggplot2` package uses the `+` operator to combine grobs representing different elements of the plot

Viewports

- ▶ The plotting region is divided into viewports
- ▶ Grobs are displayed inside a viewport
 - ▶ Viewports can be different sizes (inches, centimetres, lines of text, or relative units)
 - ▶ Each viewport may have its own coordinate systems