Introduction to computational programming
Appendix 2
Plotting in *R*

David M. Rosenberg
University of Chicago
Committee on Neurobiology

Version control information:
Last changed date: 2009-10-05 15:42:55 -0500 (Mon, 05 Oct 2009)
Last changes revision:80
Version: Revision 80
Last changed by: David M. Rosenberg

October 12, 2009

Overview

Part I

Tutorial

1 Plotting in *R*

One of the greatest strengths of *R* is it's ability to produce "presentation quality" plots and graphs. Unfortunately, this strength comes at a cost: the number of options controlling plotting output can be overwhelming at first. Here we demonstrate how to use *R*'s *high-level* plotting functions to produce and save simple plots. Several tables of options and parameters are provided; refer to them as frequently as needed.

The most basic (and most frequently used) *R* command for plotting is `plot()`, which takes as arguments numeric vectors `x` and `y`. Using the default settings, `plot()` produces a scatterplot of "circles" at each point `x[i], y[i]`.

This output may not be the prettiest you've ever seen, but it works! Later, we will discuss plot parameters which allow you to customize (beautify) your plots.

Along with the `plot()` command there are a family of commands for producing individual "parts" of a plot. All four of these plotting commands are summarized in Table 1 (page 2) and Figure 1 (page 3).

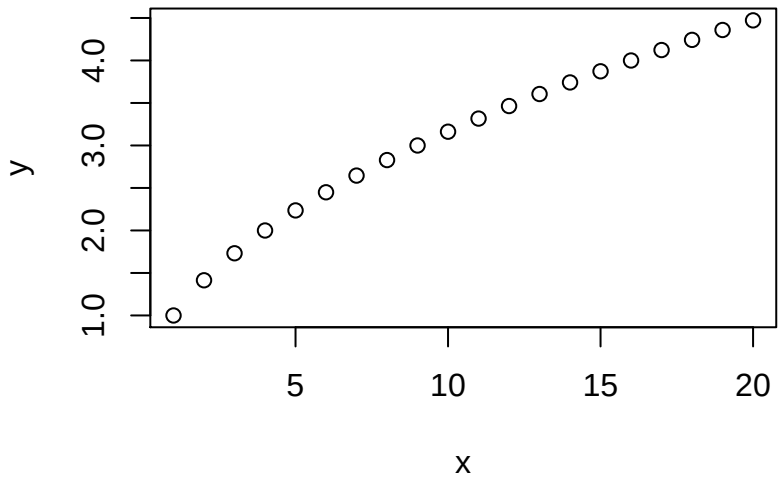
```
> x <- 1:20
> x

 [1]  1  2  3  4  5  6  7  8  9 10
     11 12 13 14 15 16 17 18 19 20

> y <- (1:20) ^ (0.5)
> y

 [1] 1.000000 1.414214 1.732051
     2.000000 2.236068 2.449490
     2.645751 2.828427
 [9] 3.000000 3.162278 3.316625
     3.464102 3.605551 3.741657
     3.872983 4.000000
[17] 4.123106 4.242641 4.358899
    4.472136

> plot(x, y)
```



Command	Required arguments	Example	Clears display[1]
`plot()`	`x` and `y` (numeric vectors)	`plot(x,y)`	Yes
`lines()`	`x` and `y` (numeric vectors)	`lines(x,y)`	No
`segments()`	`x0`, `y0`, `x1` and `y1` (all numeric vectors)	`segments(x0, y0, x1, y1)`	No
`arrows()`	`x0`, `y0`, `x1` and `y1` (all numeric vectors)	`arrows(x0, y0, x1, y1)`	No
`points()`	`x` and `y` (numeric vectors)	`points(x,y)`	No

Table 1 – Basic plotting commands in *R* as described on page 1. Example output from these commands is in Figure 1 (page 3). Note that `points()` was not shown, since its default output is identical to that shown for `plot()`.

[1]: This command clears the current plot and starts a new one.

In order to modify the plotting style, additional parameters can be specified as arguments to the plotting function using `parameter=value` syntax. Table 2 (page 6) summarizes the plotting parameters and possible values that you are most likely to use. Two more typical examples of the usage and output of the `plot()` command are shown in Code Listing 1 (page 4) and Code Listing 2 (page 5).

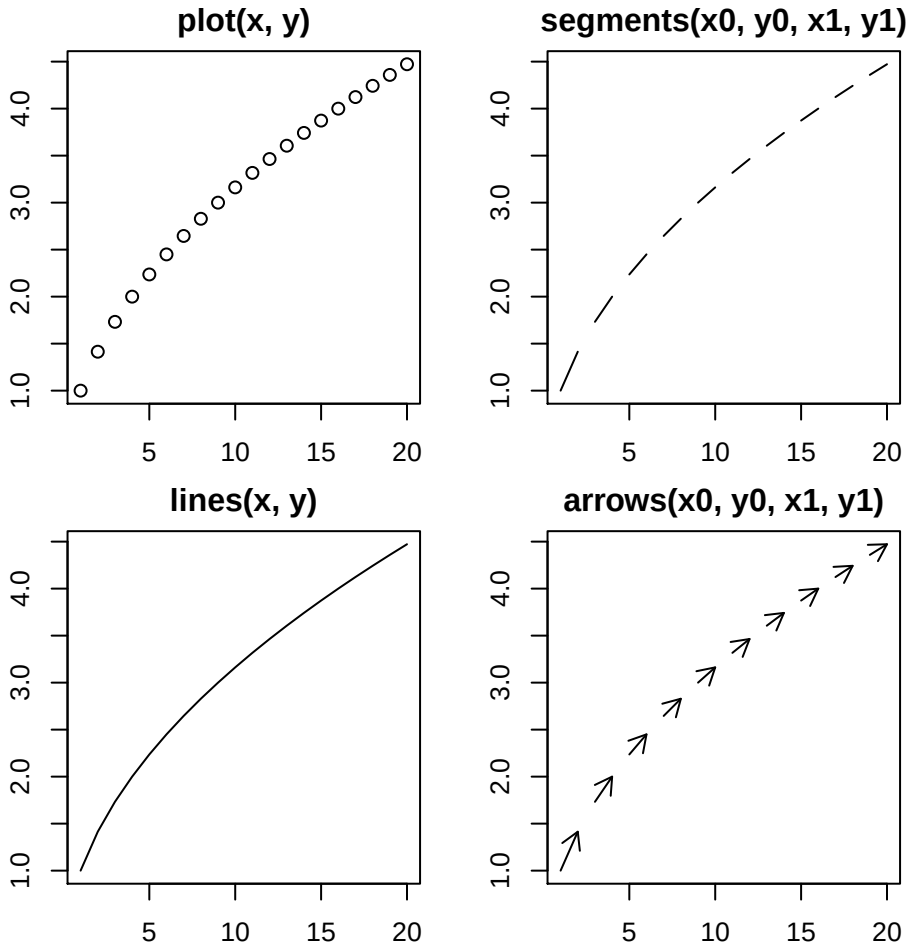
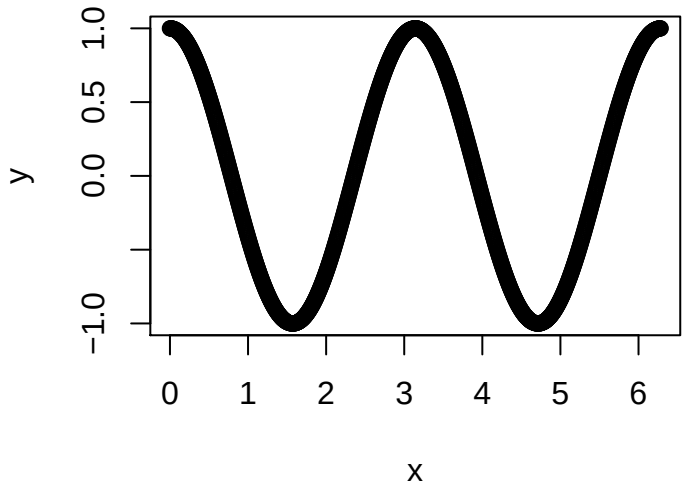


Figure 1 – Basic plotting command output

1.0.1 Saving plots

The *R* syntax for saving / printing plots is a little different than one might expect. Rather than provide facilities for the printing / post-hoc manipulation of plots *R* instead provides a pair of functions for "saving" plots to a file. The two commands you should use for this are `dev.copy2pdf(file=...)` and `dev.copy2eps(file=...)`. Both take a single argument (`file`), designating the filename to save the plot to. The first command, `dev.copy2pdf()` produces a *pdf* output file, which is likely a format you have encountered and used before. The second command `dev.copy2eps()` produces an *encapsulated postscript file* suitable for use in LaTeX documents and other postscript formats. If in doubt, you should use `dev.copy2pdf()`.

```
> x <- seq(from=0, to=2*pi, len
    =1000)
> y <- cos(2*x)
> ## just provide data; sensible
    labelling
> plot(x,y)
> dev.new()
```

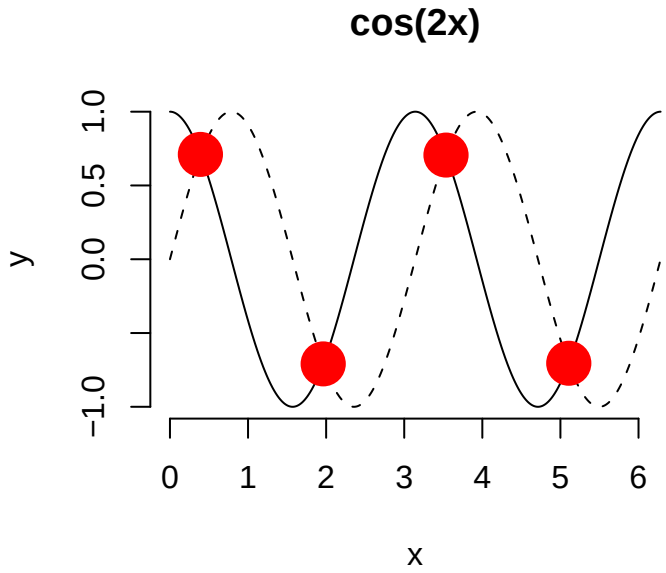


Code Listing 1 – Typical simple example plot

1.0.2 Building complex plots

- complex plots
  1. building part at a time
  2. be careful of clearing
  3. axes, etc

```
> ## Expand on previous plot ...
> plot(x,y, main='cos(2x)', type='l'
    ,
+      lty=1, bty='n')
> y2 <- sin(2*x)
> lines(x,y2, main='sin(2x)', type='
    l',
+       lty=2)
> same <- which( abs(y - y2) < 0.01)
> points(x[same], y[same], pch=19,
+        col='red', cex=3)
```



Code Listing 2 – Typical layered example plot

Parameter	Effect	Values	Default	Notes
`type`	*what* is plotted	'p', 'l', 'n'	'p'	Type is only valid as an argument to `plot`. It determines what kind of plot is generated. The default, `'p'` gives "points" (scatterplot); `'l'` gives "lines" (like `lines()`); `'n'` gives "nothing"
`xlab`, `ylab`	axis labels	`character()`	*	Defaults are the names of the first to arguments to `plot()`.
`main`	Plot title	`character()`	None	
`col`	color	Color name	`'black'`	RGB values can be used instead and are denoted by strings of hexadecimal triples of the form `'#RRGGBB'`. See `colors()` for a list of named colors.
`xlim`, `ylim`	x, y range	`numeric` pairs	*	Default values are determined from the data
`lty`	line type	`numeric`	`1`	`1` is a solid line, ≥ 2 are dashed lines
`pch`	plot "symbol"	see `example(points)`	`1`	`'.'` is most common
`log`	axes log scale?	`'x'`, `'y'`, `'xy'` or `''`	`''`	

Table 2 – Common plotting parameters