Figures in **R**

Evan Parker-Stephen

November 2, 2006

The `plot` Function

For almost any graphics task, one will want to use the `plot` function in **R**. To see the total sum of arguments that one can call using `plot`, type `args(plot.default)`, which returns the following:

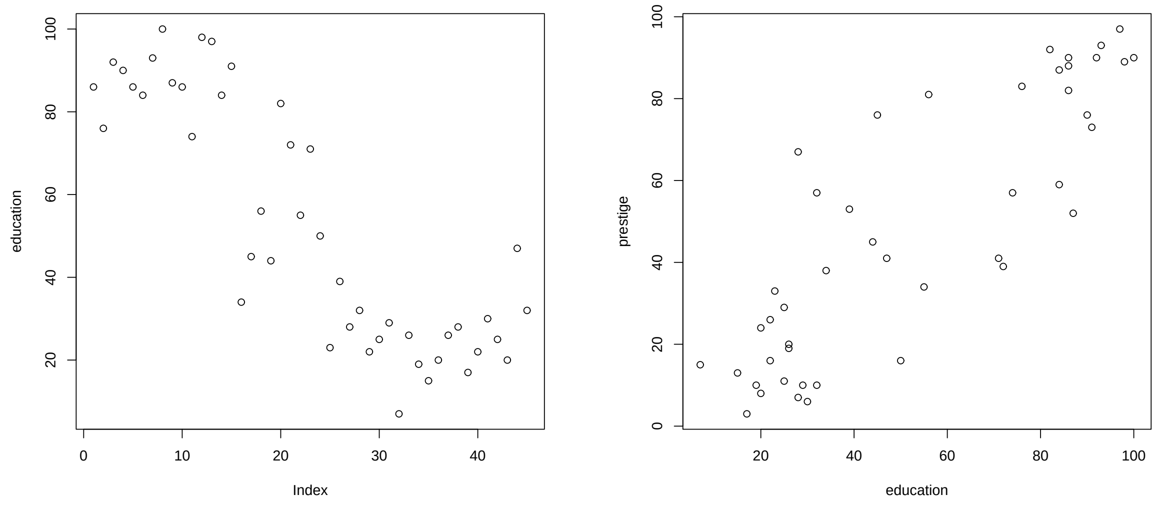
```
function (x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
    log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
    ann = par("ann"), axes = TRUE, frame.plot = axes, panel.first = NULL,
    panel.last = NULL, asp = NA, ...)
```

Obviously there is a lot going on underneath the generic `plot` function. But for the purpose of getting started with figure creation in **R** we want to ask what is essential. The answer is straightforward: one value `x` must be specified. Everything else has either a default value or is not essential. To start experimenting with `plot`, let's load data from John Fox's `car` package (car = "Companion to Applied Regression"). We're going to use Duncan's data on occupational prestige, appropriately titled "Prestige". The `summary` function gives some basic descriptive statistics of Duncan's prestige data.

```
library(car)
data(Duncan)
attach(Duncan)
summary(Duncan)
```

```
   type        income         education        prestige
 bc  :21   Min.   : 7.00   Min.   :  7.00   Min.   : 3.00
 prof:18   1st Qu.:21.00   1st Qu.: 26.00   1st Qu.:16.00
 wc  : 6   Median :42.00   Median : 45.00   Median :41.00
           Mean   :41.87   Mean   : 52.56   Mean   :47.69
           3rd Qu.:64.00   3rd Qu.: 84.00   3rd Qu.:81.00
           Max.   :81.00   Max.   :100.00   Max.   :97.00
```

As a first foray into creating figures, we can plot the variables separately with the command `plot(varname)`.[1] For example, if we want to look at the distribution of the data points for `education`, we simply type `plot(education)` and Figure 1(a) is returned in the **R** graphics interface. Note that this figure plots education against a default index. Of course, we are more often interested in bivariate relationships. We can explore these easily by incorporating an x and a y in the call to plot: `plot(education, prestige)`. This produces Figure 1(b).



(a) Education (b) Education and Occupational Prestige

Figure 1: Education and Education by Occupational Prestige

Figure Construction

R graphics are wonderful in that one begins with a blank slate. There are a good number of options that can be combined in a number of ways to create figures that suit your specific needs. These include scatterplots, line graphs, bar graphs, histograms, box plots, and more. It is perhaps most useful for getting started to work through the basic functions/options that bring considerable flexibility to creating figures in **R**.

The Coordinate System: In the above scatterplot, we were not worried about establishing the coordinate system because the data effectively did this for us. But often, you will want to establish the dimensions of the figure before plotting anything—especially if you are building up from the blank canvas. The most important point

[1]There are several alternatives to plot. Two that you might find especially useful are the histogram function (`hist(varname)`) and the boxplot function `boxplot(varname)`).

here is that your `x` and `y` must be of the same length. This is perhaps obvious, but missing data can create difficulties that will lead **R** to balk. I like to establish the coordinate system outright by creating vectors like the following:

```
xaxis <- c(1:12)
econ.inds <- c(2, 3, 3.5, 2, 3, 2.5, 3, 2.5, 3, 3.5, 4, 4)
econ.reps <- econ.inds + 2
econ.dems <- econ.inds - 1
```

We can now take these four vectors as data—that is, we can think of these four objects as variables. Let's treat these data as four separate series, where `xaxis` represents some generic index for time and the other three objects represent average perceptions on some hypothetical scale for Independents, Republicans, and Democrats respectively.

Plot Types: We now want to plot these series, but the `plot` function allows for different types of plots. The different types that one can include within the generic `plot` function include:

`type=‘‘p"` This is the default and it plots the x and y coordinates as *points.*

`type=‘‘l’’` This plots the x and y coordinates as *lines.*

`type=‘‘n’’` This plots the x and y coordinates as *nothing* (it sets up the coordinate space only).

`type=‘‘o’’` This plots the x and y coordinates as *points and lines* overlaid (i.e., it "overplots").

`type=‘‘h’’` This plots the x and y coordinates as *histogram-like vertical lines.*

`type=‘‘s’’` This plots the x and y coordinates as *stair-step like lines.*

Axes: It is possible to turn off the axes, to adjust the coordinate space by using the `xlim` and `ylim` options, and to create your own labels for the axes.

`axes=` Allows you to control whether the axes appear in the figure or not; I often like to turn them off by selecting `axes=F`. Then I create my own labels as follows:

- `axis(side=1, at=c(2, 4, 6, 8, 10, 12), labels=c("Feb", "Apr", "June", "Aug", "Oct", "Dec"))`

`xlim=, ylim=` For example, if we wanted to expand the space from the **R** default, we could enter:

- `plot(xaxis, econ.inds, type="o", xlim=c(-5, 17), ylim=c(-5, 15))`

`xlab="", ylab=""` Creates labels for the x- and y-axis (if you use this option you will want to be sure that you select `axes=F`).

Style: There are a number of options to adjust the style in the figure, including changes in the line type, line weight, color, point style, and more. Some that I commonly use include:

`lty=` Selects the type of line (solid, dashed, short-long dash, etc.)

`lwd=` Selects the line width (fat or skinny lines)

`pch=` Selects the plotting symbol, can either be a numbered symbol (`pch=1`) or a letter (`pch="R"`)

`col=` Selects the color of the lines/points in the figure (see "Color in R" for the many options)

`par` The `par` function brings added functionality to plotting in **R**. What is perhaps most important is that the `par` allows you to plot multiple (x, y)'s in a single graphic. This is accomplished by selecting `par(new=T)` following each call to `plot`.

Add-on Functions

There are also a number of add-on functions that one can use once the basic coordinate system has been created using `plot`. Some of these are listed below, and their implementation appears in the following sample code.

`arrows(x1, y1, x2, y2)` Create arrows within the plot (useful for labeling particular data points, series, etc

`text(x1, x2, "text")` Create text within the plot (modify size of text using the character expansion option `cex`

`lines(x, y)` Create a plot that connects lines

`points(x, y)` Create a plot of points

`polygon()` Create a polygon of any shape (rectangles, triangles, etc.

`legend(x, y, at = c("", "",), labels=c("", "")))` Create a legend to identify the components in the figure

Example Code: Time Series

```
postscript("parallel.eps", horizontal=FALSE, width=7, height=7,
            onefile=FALSE, paper="special", family="ComputerModern")
    plot(xaxis, econ.inds, type="o", lwd=2, pch=5, ylim=c(0, 6.5),
        axes=F, xlab="", ylab="")
    par(new=T)
    plot(xaxis, econ.dems, type="o", lwd=2, pch=1, col="blue", ylim=c(0, 6.5),
         axes=F, xlab="", ylab="")
    par(new=T)
    plot(xaxis, econ.reps, type="o", lwd=2, pch=2, col="red", ylim=c(0, 6.5),
        axes=F, xlab="Month", ylab="Average Perception (Hypothetical)")
    axis(side=1, at=c(2, 4, 6, 8, 10, 12),
        labels=c("Feb", "Apr", "June", "Aug", "Oct", "Dec"))
    box()
    text(2.25, 6.25, "Republicans", cex=1.2)
    text(6, 3.7, "Independents", cex=1.2)
    text(10.5, 1.2, "Democrats", cex=1.2)
    arrows(2, 6.05, 3, 5.65, length=.10, lwd=2)
    arrows(5.75, 3.5, 6.75, 3.1, length=.10, lwd=2)
    arrows(10.25, 1.35, 9.25, 1.75, length=.10, lwd=2)
    dev.off()
```

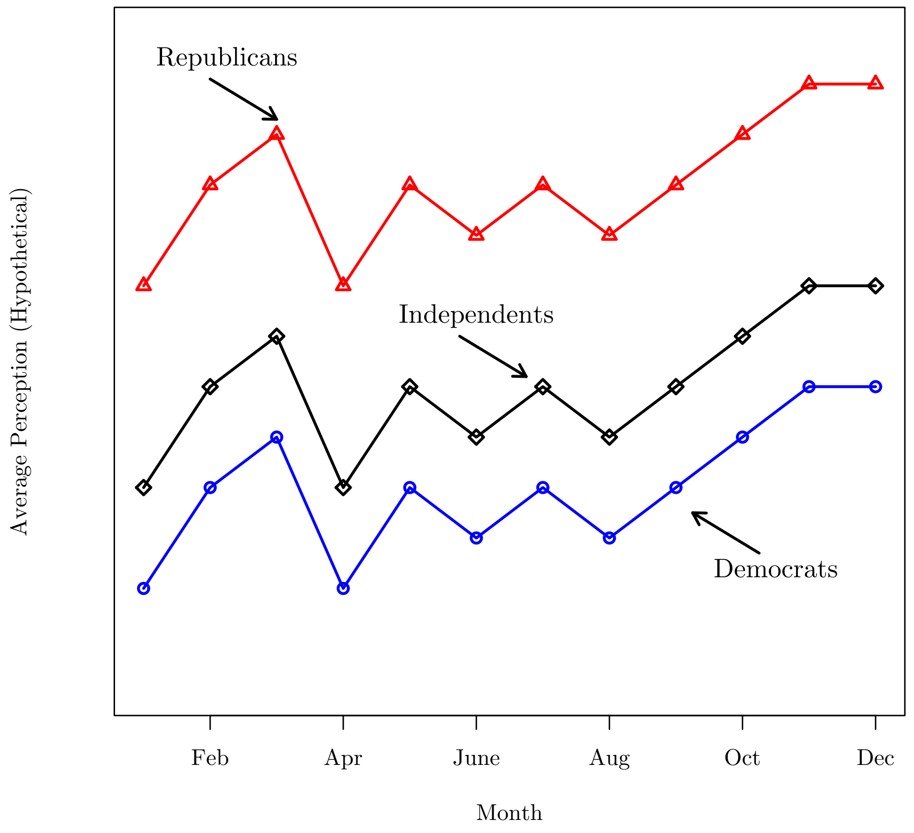


Figure 2: Hypothetical Example of Objective Information Updating

Example Code: Bar Graph

```
postscript("folpa.eps", horizontal=FALSE, width=7, height=7,
onefile=FALSE, paper="special", family="ComputerModern")
    plot(c(0, 4), c(0, 0.35), type='n', xlab="How Interested in Politics?",
        ylab="Proportion",axes=F)
    axis(2)
    axis(side = 1, at = c(.5, 1.5, 2.5, 3.5),
        labels = c("Not at All", "Hardly", "Quite", "Very"))
    polygon(c(0.25, 0.25, .75, .75), c(0, .33, .33, 0), col="gray76")
    polygon(c(1.25, 1.25, 1.75, 1.75), c(0, .20, .20, 0), col="gray56")
    polygon(c(2.25, 2.25, 2.75, 2.75), c(0, .32, .32, 0), col="gray36")
    polygon(c(3.25, 3.25, 3.75, 3.75), c(0, .16, .16, 0), col="gray16")
    abline(h=0, col="gray70")
    title("Distribution of Political Interest")
    text(.5, .35, ".33", cex=1.2)
    text(1.5, .22, ".20", cex=1.2)
    text(2.5, .34, ".32", cex=1.2)
    text(3.5, .18, ".16", cex=1.2)
    dev.off()
```

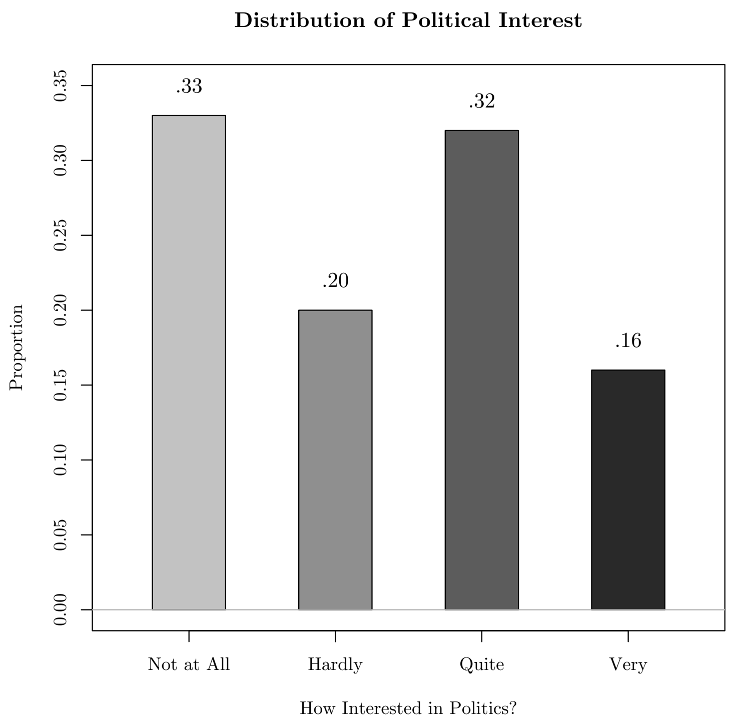


Figure 3: Attentiveness to Politics in the Albanian Mass Public