Basic Graph Creation and Manipulation in R.

As published in Benchmarks RSS Matters, June 2014

http://web3.unt.edu/benchmarks/issues/2014/06/rss-matters

Jon Starkweather, PhD

Jon Starkweather, PhD
jonathan.starkweather@unt.edu
Consultant
Research and **S**tatistical **S**upport

http://www.unt.edu

http://www.unt.edu/rss

RSS hosts a number of "Short Courses".
A list of them is available at:
http://www.unt.edu/rss/Instructional.htm

Those interested in learning more about R, or how to use it, can find information here:
http://www.unt.edu/rss/class/Jon/R_SC

Basic Graph Creation and Manipulation in R.

The purpose of this article is to provide some key information for creating and manipulating graphs in R. Only the basics will be covered here because there have been entire books published regarding the graphical capabilities of R with and without additional packages (e.g. Andrews, 2012; Deepayan, 2008). Furthermore, those who have already mastered the basics covered in this article are encouraged to explore the CRAN Task View for graphics (Lewin-Koh, 2014). Some examples are provided below for what might be considered necessary skills for anyone working with data. The focus of the examples below is oriented toward initial data analysis (i.e. graphs which display basic descriptive and relational properties of one or two variables).

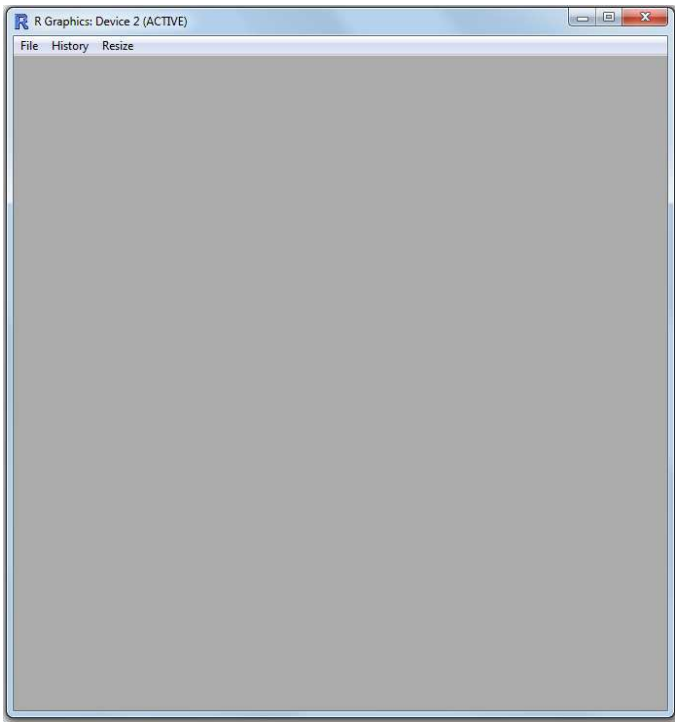
First, we import some (fictional) example data. The data can be imported directly from the RSS URL as provided in the script below (i.e. simply copy the script provided and paste into the R console to follow along):

```
df.1 <-
  read.table("http://www.unt.edu/rss/class/Jon/Benchmarks/BasicPlotData.txt",
  header = TRUE, sep = ",", na.strings = "NA", dec = ".", strip.white = TRUE)
summary(df.1)
nrow(df.1)
ncol(df.1)
names(df.1)
```

Before we actually create any graphs, it might be sensible to take a look at the graphics window and its menus first. Actually, although the habit here is to refer to it as a 'graphics window' it is really a graphics 'device' (i.e. dev). To create a blank graphics window, or graphics device, we use the simple function below:

```
dev.new()
```

which produces the empty graphics window below.



Notice in the above image, there are three menu items: File, History, and Resize. The File tab allows you to Save the image (in a variety of popular formats), Copy the image (in either of two formats), Print the image, or Close the device (i.e. window). Of course, you can also copy, save, or print the contents of the graphics window by right clicking on it with your mouse and selecting one of those operations. The History tab allows us to turn on (or off) the recorder; which will keep a record of subsequent graphs produced in this window and allow us to page up or page down to scroll through previous and next graphs produced. We can also save or clear the history from this tab. The Resize tab simply allows us to resize the graphics window based on some basic commands; of course you can also use a mouse click-and-drag from one corner to resize the graphics device. For now, we will close the graphics window using script (rather than "File", "close Device").

```
graphics.off()
```

It is often desirable to 'attach' a data frame if one is going to be repeatedly calling specific variables of it. We do so here in order to simplify the indexing of the data frame we imported.

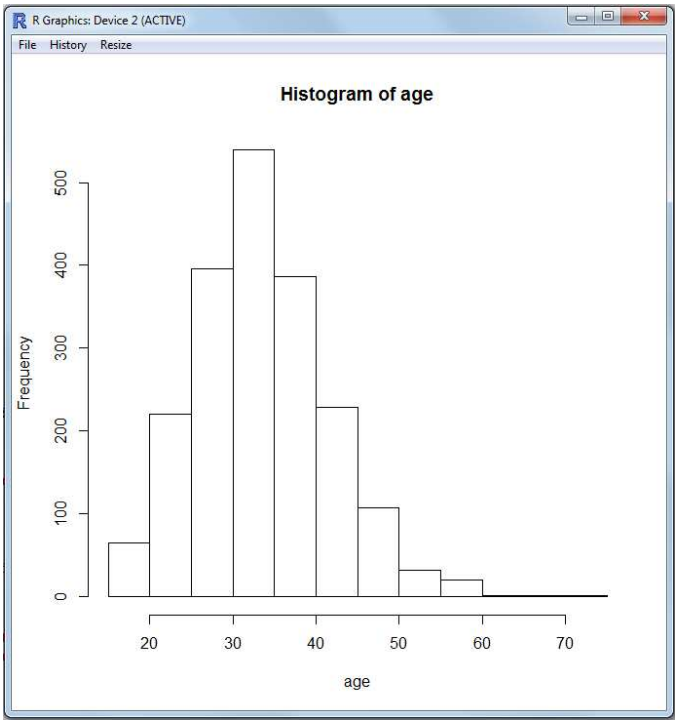
```
attach(df.1)
```

1 Defaults

One of the most commonly used graphic displays is the simple histogram; which is used to display a distribution of values (i.e. continuous or nearly continuous variable). Below we create a simple histogram of the 'age' variable (of the 'df.1' data frame); supplying only the variable name (because we attached the data frame) and omitting all other arguments (which provides a default histogram):

```
hist(age)
```

which produces the following simple histogram.

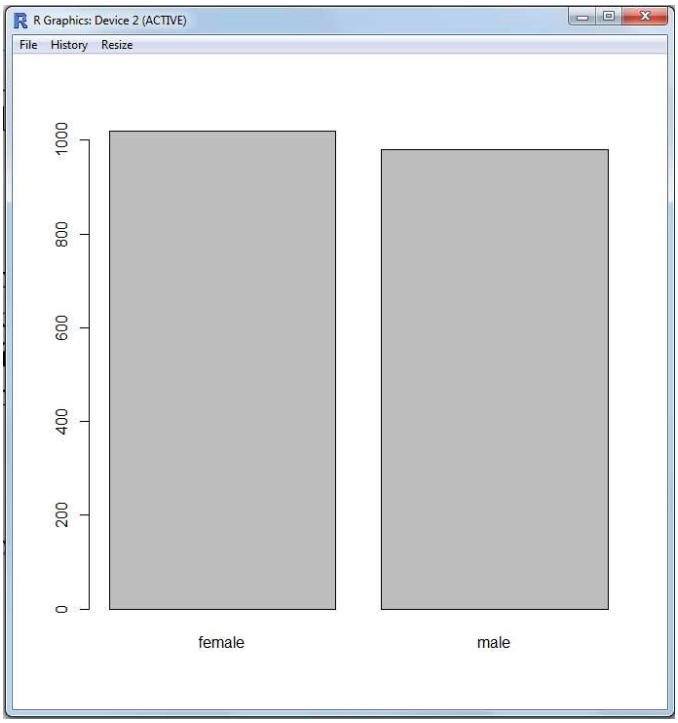


The default graph produced by the 'hist' function provides a most basic histogram based on default options for the function's many arguments. The histogram above provides the necessary information; it is very plain, some might even consider it boring.

By far the most commonly used graphical function is the simple 'plot' function. The 'plot' function can be used to display single variables (e.g. categorical variables' frequency counts) or multiple variables' relationships (e.g. scatterplots & scatterplot matrices). To use the 'plot' function in its most primitive form, we will supply it with the gender variable of our data frame:

```
plot(gender)
```

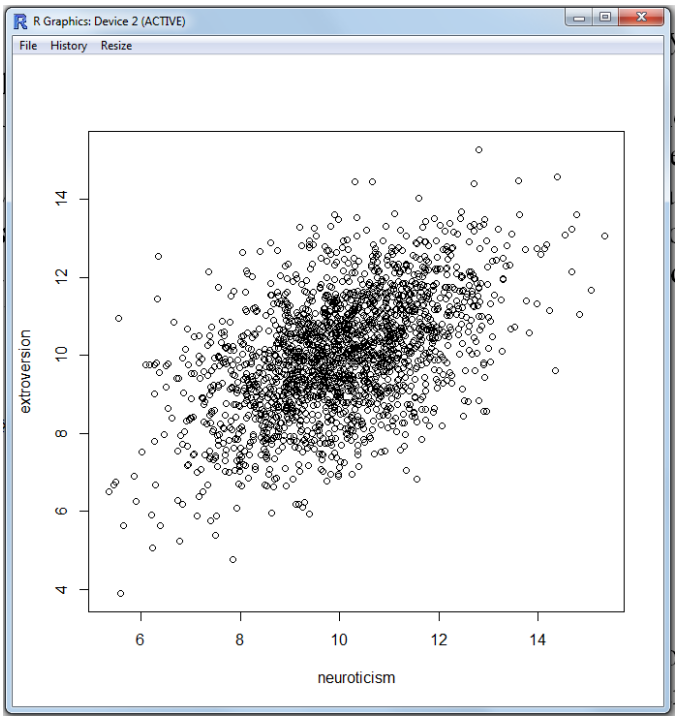
which produces the following graph (i.e. an example of a bar chart or bar graph).



Notice in the above, we supplied only the variable, without specifying any optional parameters to other arguments of the 'plot' function. Notice there is no main title to the graph, as there was with the histogram previously; nor is there an x-axis line. Furthermore, the x-axis labels (female, male) are taken directly from the variable supplied; as was the case for the x-axis label (age) in the histogram previously. Both default graphs are produced using gray scale; which makes printing them easy and some publications require manuscripts to contain only gray scale (i.e. no colors). Therefore, the default graphs can be quite handy, even if they tend to be a little boring. Below is an example of the 'plot' function producing a simple scatterplot by supplying only the two variables:

```
plot(neuroticism, extroversion)
```

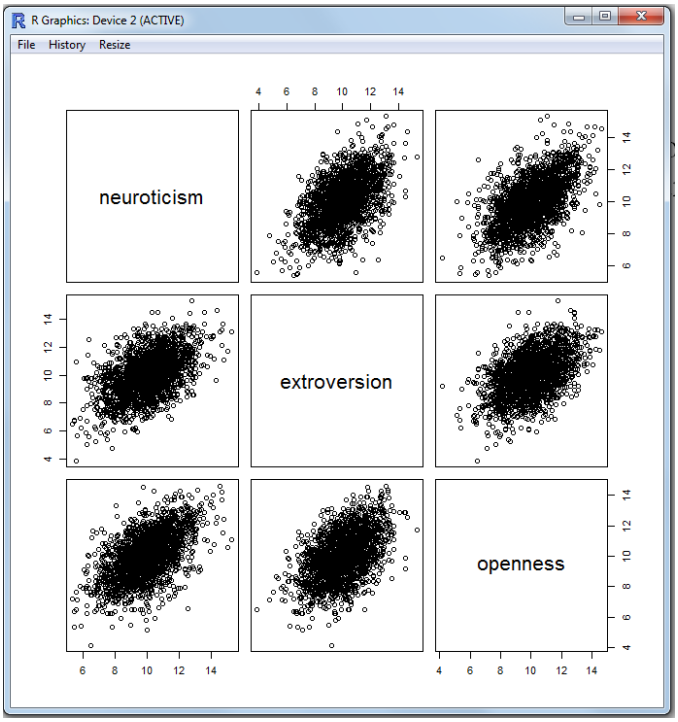
which produces the following graph (i.e. scatterplot).



The ‘plot’ function does not supply a main title to the graph when supplied with two (or more) variables. Also notice, the x-axis and y-axis labels (neuroticism, extroversion) are supplied exactly as they are given from the data frame. And again, the graph is displayed in gray scale by default. To create a scatterplot matrix you can simply supply the ‘plot’ function with the columns of a data frame (or matrix); such as:

```
plot(df.1[,22:24])
```

which produces the following graph.



Keep in mind, depending on the number of cases, more than a few variables in a single scatterplot matrix can defeat the purpose of displaying the data in this way (i.e. more than 4 or 5 variables in a scatterplot matrix often makes each cell of the matrix too small to interpret). Again, notice there is no main title, x-axis label, y-axis label, or colors in the default scatterplot matrix.

It should also be noted that there are other graphics functions available in a base install of R which may occasionally be useful; such as the 'boxplot' function which produces a box and whisker plot;

```
boxplot(age ~ gender)
```

the 'coplot' function which produces a conditional plot;

```
coplot(income ~ age | gender)
```

and the 'pairs' function which produces a scatterplot or scatterplot matrix; in fact, the 'plot' command from above calls this function to produce these types of graphs;

```
pairs(df.1[,22:24])
```

For a demonstration of some of the base functionality of R (in terms of graphics), use the 'demo' function:

```
graphics.off()
demo(graphics)
```

For more complex data, the 'persp' function provides the ability to produce a 3-dimensional perspective plot.

```
graphics.off()
demo(persp)
```

2 Non-Defaults

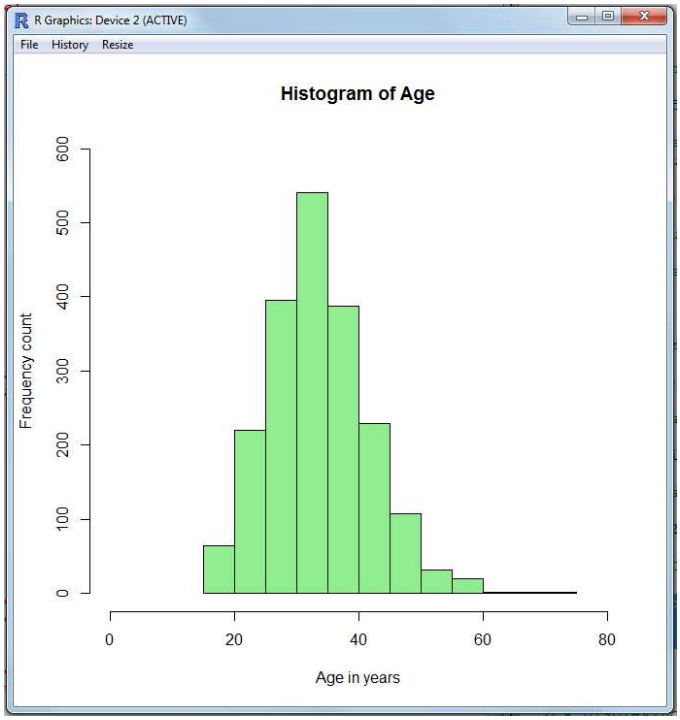
Like most things in R, however, everything displayed by the 'hist' function and the 'plot' function can be manipulated using optional arguments. Below we provide a few examples using some common non-default options for arguments of the 'hist' and 'plot' functions. To start fresh, we will again close the graphics device:

```
graphics.off()
```

Revisiting the histogram of age from above, but adding some color (col), providing a specific main title (main), x-axis label (xlab), y-axis label (ylab), x-axis limits (i.e. zero to 80), and y-axis limits (i.e. zero to 600):

```
hist(age, col = "lightgreen", main = "Histogram of Age",
     xlab = "Age in years", ylab = "Frequency count",
     xlim = c(0,80), ylim = c(0,600))
```

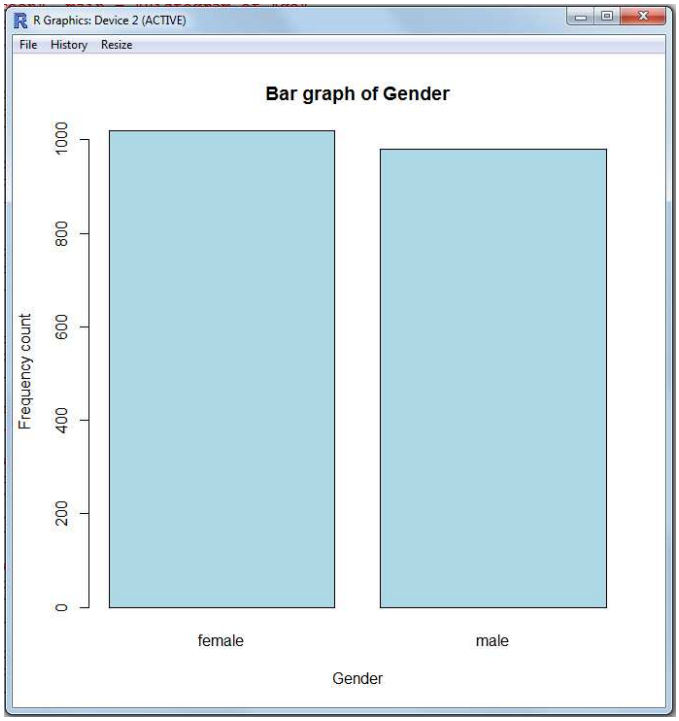
which produces the following graph.



Revisiting our bar graph from earlier, but supplying some color, a main title, and axis labels:

```
plot(gender, col = "lightblue", main = "Bar graph of Gender",
     xlab = "Gender", ylab = "Frequency count")
```

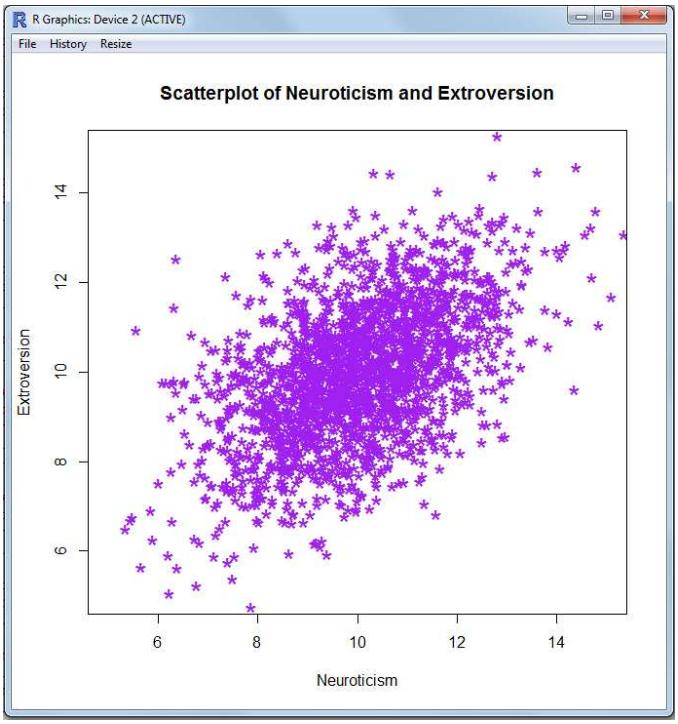
which produces the following graph.



And finally, we revisit our scatterplot with some color, main and axis titles, as well as specific limits for both axes. Also notice we used the 'pch' argument to specify a particular character for the points in the scatterplot. We also used the 'cex' argument to specify the size of those characters; where smaller numbers produce smaller points or characters in the scatterplot.

```
plot(neuroticism, extroversion, col = "purple",
     main = "Scatterplot of Neuroticism and Extroversion",
     xlab = "Neuroticism", ylab = "Extroversion",
     xlim = c(5,15), ylim = c(5,15), pch = "*", cex = 2)
```

which produces the following graph.



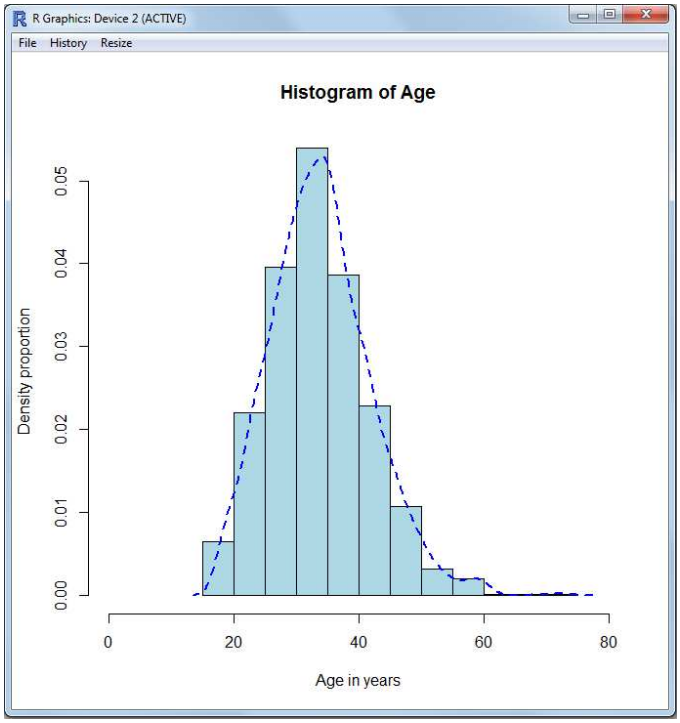
3 Supplemental Graphics Functions

There are other useful graphics device functions which can be used in conjunction with what we have done above. Some of the popular operations involve adding a line of some type to histograms, such as what is done below with the 'lines' function:

```
hist(age, col = "lightblue", main = "Histogram of Age",
     xlab = "Age in years", ylab = "Density proportion",
     xlim = c(0,80), prob = TRUE)
lines(density(age), col = "blue", lty = 2, lwd = 2)
```

which produces the following graph. Notice in the script directly above, we used 'prob = TRUE' to indicate that the histogram should reflect probability densities rather than frequency counts and our y-axis label (ylab) has been changed accordingly. The 'lines' function can be used in a variety of ways to add a line to a graphic; here we specify a density line using line type dashed (lty = 2) with a line width

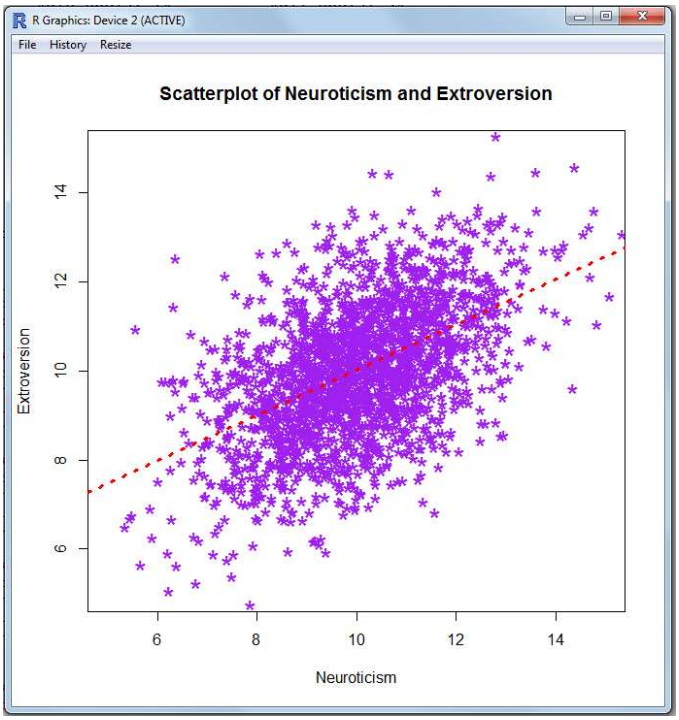
of moderately thick (lwd = 2). The defaults for both line type and line width are 1; which correspond to solid line and thin line respectively.



The ‘abline’ function is an alternative to the ‘lines’ function for placing a line over an existing graph. As an example, we apply a line which represents a linear model (lm) to our previous scatterplot.

```
plot(neuroticism, extroversion, col = "purple",
     main = "Scatterplot of Neuroticism and Extroversion",
     xlab = "Neuroticism", ylab = "Extroversion",
     xlim = c(5,15), ylim = c(5,15), pch = "*", cex = 2)
abline(lm(extroversion ~ neuroticism), col = "red",
     lty = 3, lwd = 3)
```

which produces the following graph. Notice here we chose to use line type dotted (lty = 3) and a line width of thick (lwd = 3).



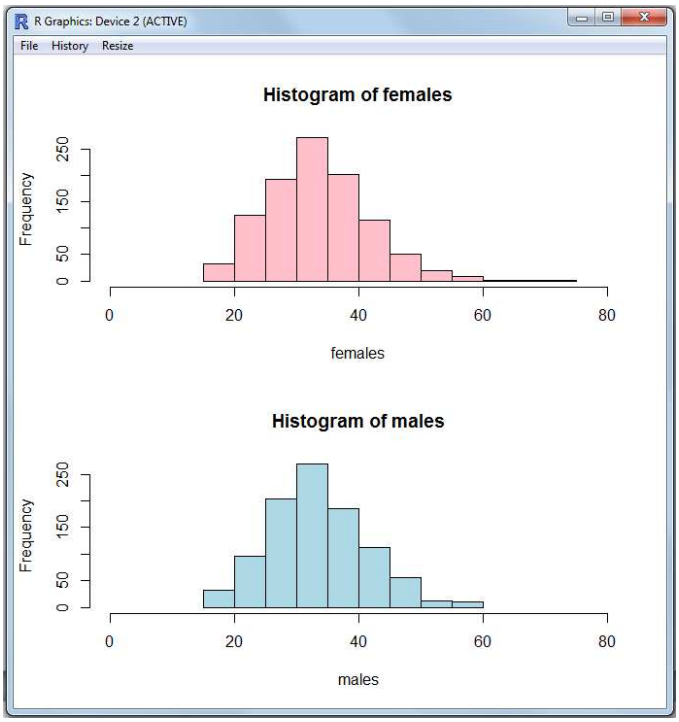
4 Graphics Parameters

The ‘par’ function is used to specify graphics parameters; most commonly applied to the base ‘plot’ function and resulting graphics devices (i.e. graphics windows). One of the most useful applications of the ‘par’ function is the ability to place more than one graph in a graphics device. For example, you may want to place one graph below another:

```
gen <- which(df.1[,4] == "female")
females <- df.1[gen,5]
males <- df.1[-gen,5]

par(mfrow = c(2,1))
hist(females, xlim = c(0,80), col = "pink")
hist(males, xlim = c(0,80), col = "lightblue")
```

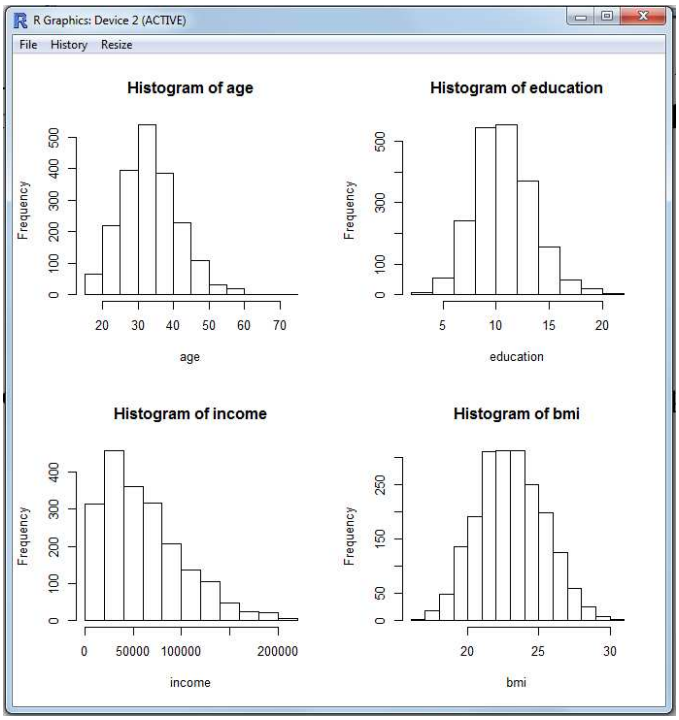
Notice in the ‘par’ function we are specifying rows and columns of a single graphics device display. The example here specifies two rows and one column (within the graphics device). The resulting graphics device displays a histogram of female participants’ age over a histogram of male participants’ age.



A second example uses the ‘par’ function to specify 2 rows and 2 columns — keep in mind you can put any graph in each row / column (i.e. cell); this example simply uses histograms.

```
par(mfrow = c(2,2))
hist(age)
hist(education)
hist(income)
hist(bmi)
```

The above script produces the matrix of four graphs in the single graphics device displayed below. Notice how the order of each histogram function corresponds to each cell of the graphics device window (i.e. age in the upper left, education in the upper right, income in the lower left, and BMI in the lower right).



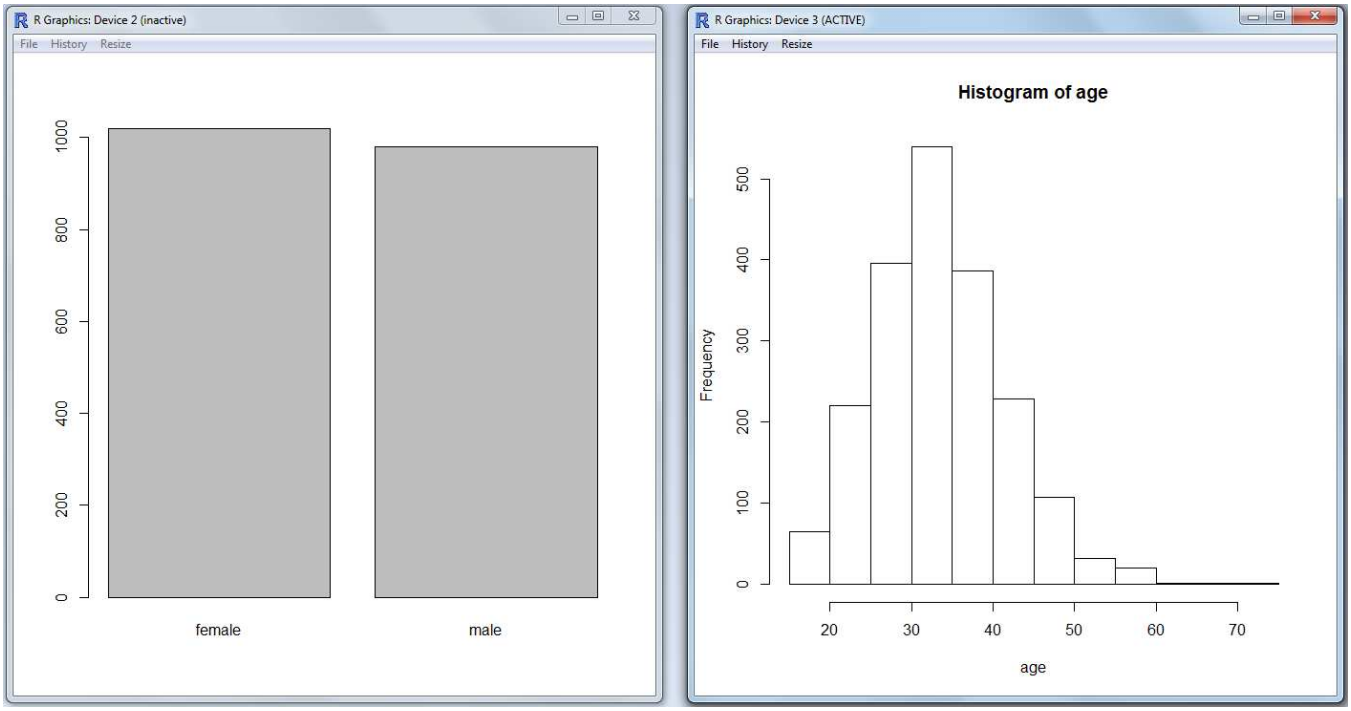
There are two ways to reset the rows and columns of your graphics device. First, simply close the graphics device (as was shown earlier) or second, simply re-specify the rows and columns as 1 each (which is the default when a graphics device is originally opened):

```
par(mfrow = c(1,1))
```

Keep in mind, you can have multiple graphics devices open at the same time simply using the 'dev.new' function:

```
plot(gender)
dev.new()
hist(age)
```

which produces two graphs, each in its own device (i.e. window). Notice each device is numbered, starting with 2. This allows you to reference each device (active versus inactive) individually if so desired.



Keep in mind we have only used one argument (of more than 70) of the 'par' function. If you would like to see all the available graphics device parameter ('par') arguments, please take a look at the 'help' file; which can be accessed with the following:

```
help(par)
```

Of course, each of the functions used in this article has an associated 'help' file which can be accessed from the R console and each function is included in the base install of R (no additional packages were used to produce any of the above graphs). Lastly, you will surely have noticed the limited number and hue of colors used in this article. At the time of writing (June 5, 2014), there were 657 colors available for use in R. To see the entire list of available colors, use the 'colors' function:

```
colors()
```

5 Conclusions

Clearly there are many, many other ways to graphically display data using R; as the CRAN Task View for graphics (Lewin-Koh, 2014) shows. There are two especially popular packages for graphical display of multivariate data; the 'lattice' package (Deepayan, 2014) and the 'latticist' package (Andrews, 2014) which provides a Graphical User Interface (GUI) for the functions of 'lattice'. There are two other packages worth looking into. The 'car' package (Fox & Weisberg, 2014) provides very good 'scatterplot' and 'scatterplotMatrix' functions which fit linear model (regression) lines, smoothed (e.g. lowess) lines, and boxplots for each axis by default. Also, the 'scatterplot3d' package (Ligges & Mchler, 2003), as one might expect from the name, provides a function for producing 3-dimensional scatterplots which look very good. Please visit Module 12 of the Research and Statistical Support *Do-It-Yourself Introduction*

to R[1] course page for examples of everything mentioned in this article (including the more complex data displays mentioned in this paragraph).

Until next time, remember what George Carlin said: *"Intelligence tests are biased toward the literate."*

6 References & Resources

Andrews, F. (2014). Package latticist. Documentation available at CRAN:
http://cran.r-project.org/web/packages/latticist/index.html

Andrews, F. (2012). latticist: A Graphical User Interface for Exploratory Visualisation R package version 0.9-44.
http://latticist.googlecode.com/

Baron, J. (2014). R Reference Card. Available at CRAN Contributed Documents:
http://cran.r-project.org/doc/contrib/refcard.pdf

Carlin, G. (1937 - 2008). http://www.just-one-liners.com/ppl/george-carlin

Deepayan, S. (2014). Package lattice. Documentation available at CRAN:
http://cran.r-project.org/web/packages/lattice/index.html

Deepayan, S. (2008). Lattice: Multivariate Data Visualization with R. New York: Springer Science+Business Media, LLC.

Fox, J., & Weisberg, S. (2014). Package car. Documentation available at CRAN:
http://cran.r-project.org/web/packages/car/index.html

Fox, J., & Weisberg, S. (2011). An R Companion to Applied Regression, Second Edition. Thousand Oaks CA: Sage.
http://socserv.socsci.mcmaster.ca/jfox/Books/Companion

Lewin-Koh, N. (2014). CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization.
http://cran.r-project.org/web/views/Graphics.html

Ligges, U. & Mchler, M. (2003). Scatterplot3d - an R Package for Visualizing Multivariate Data. Journal of Statistical Software 8(11), 1-20. Documentation available at CRAN:
http://cran.r-project.org/web/packages/scatterplot3d/index.html

Short, T. (2004). R Reference Card. Available at CRAN Contributed Documents:
http://cran.r-project.org/doc/contrib/Short-refcard.pdf

[1] http://www.unt.edu/rss/class/Jon/R_SC/

This article was last updated on June 9, 2014.

This document was created using LaTeX