The **R&SS** Introduction to RStudio

Jon Starkweather, PhD

August 21, 2017

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

http://www.unt.edu

http://it.unt.edu/research

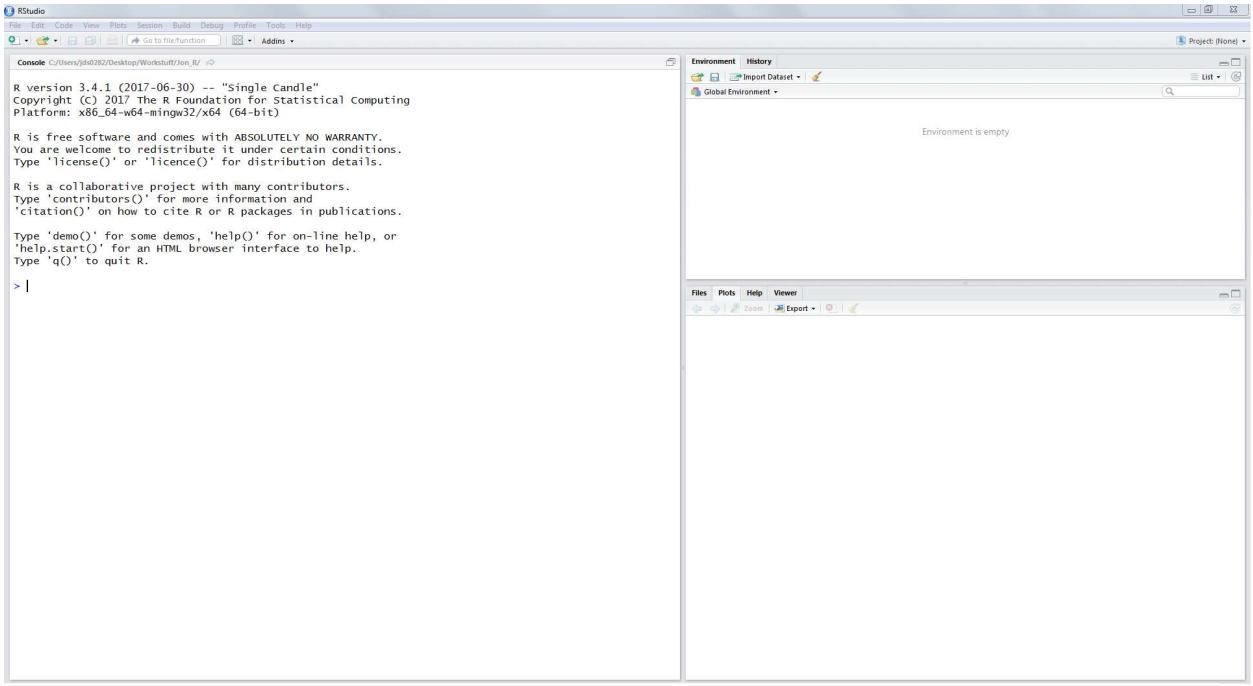
R&SS hosts a number of “Short Courses”.
A list of them is available at:
http://it.unt.edu/researchshortcourses

The material contained in this manual can also be found at:
http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/

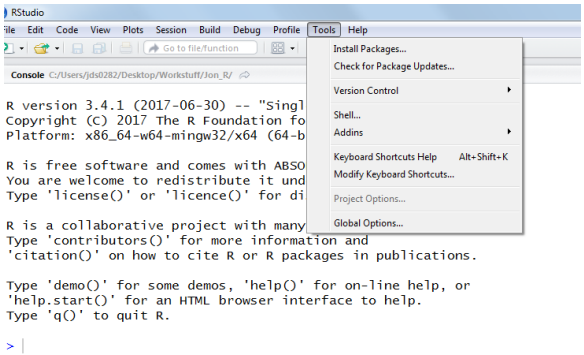
Introduction to RStudio.

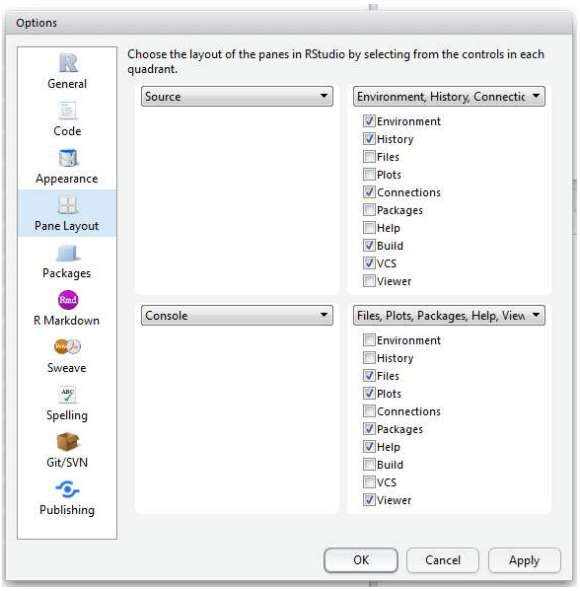
Below you will find a brief overview of the settings and setup of RStudio with my own preferences and reasons for them regarding setup. To explore all the functions, features, and customization of RStudio I would recommend starting with their website. RStudio can be downloaded from the RStudio webstie (`https://www.rstudio.com/`). Once it has been downloaded and installed (note: you will need administrator privileges when using a Windows machine) you will no longer need to open R to use it; instead, you will simply open RStudio and use R through RStudio.

When you first open RStudio it should look something like what is below, assuming the default installation.

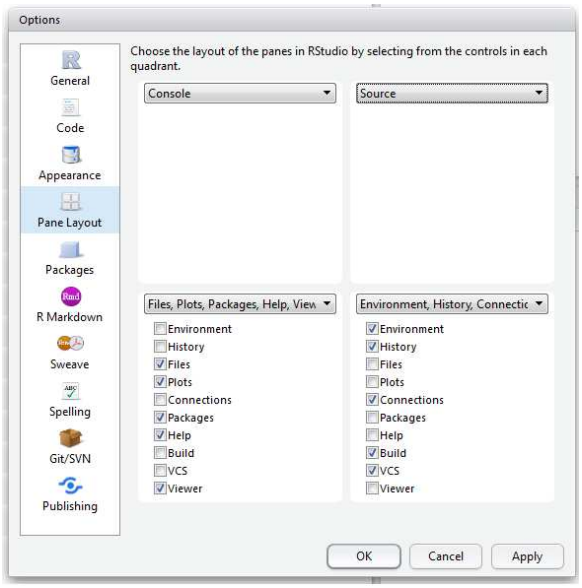


Briefly, the pane on the left in the image above is the R console, where input or script is processed and output is printed. The pane on the upper right displays the environment (i.e. any objects created or imported into the R console session) and the history (i.e. a listing of the objects which are or were in the environment). Objects in the environment can also be listed in the console using the 'list' function: `ls()`. The pane on the lower right displays: files (of the working directory), plots (i.e. graphs, plots, etc.), help (i.e. any help files called from the console), and the viewer which can be used with certain packages to display local web content. The layout of the panes can be changed by clicking on the "Global Options" in the "Tools" menu and then clicking on "Pane Layout".

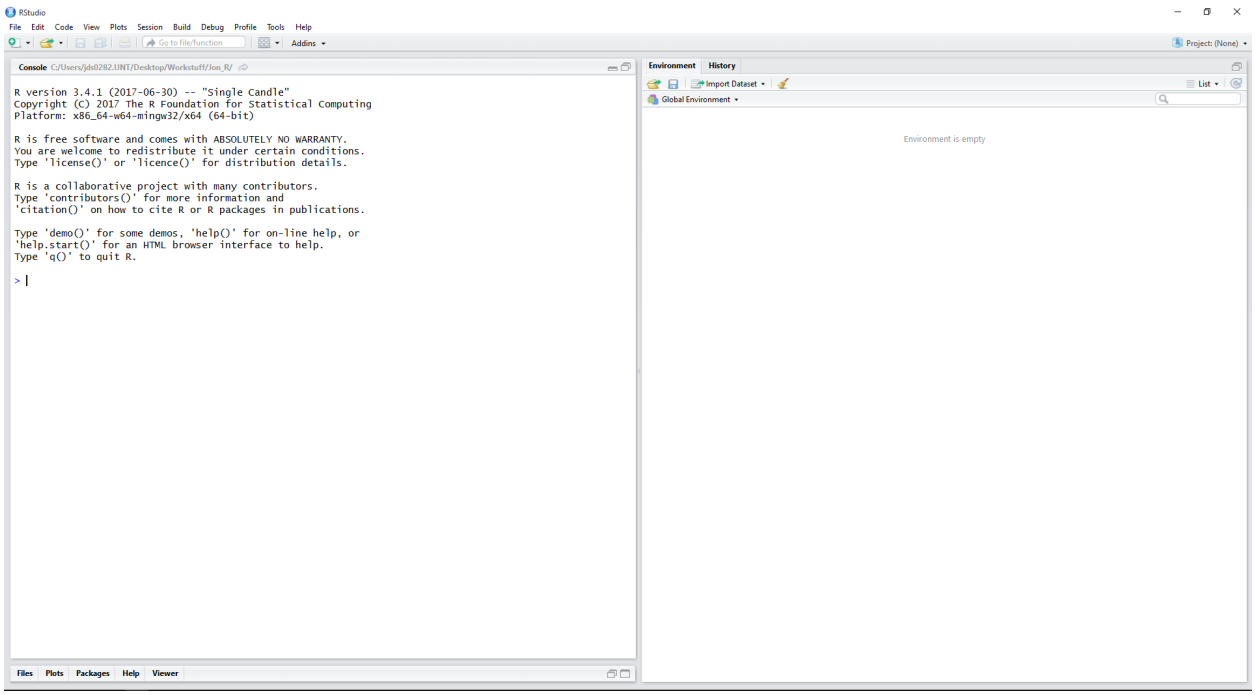




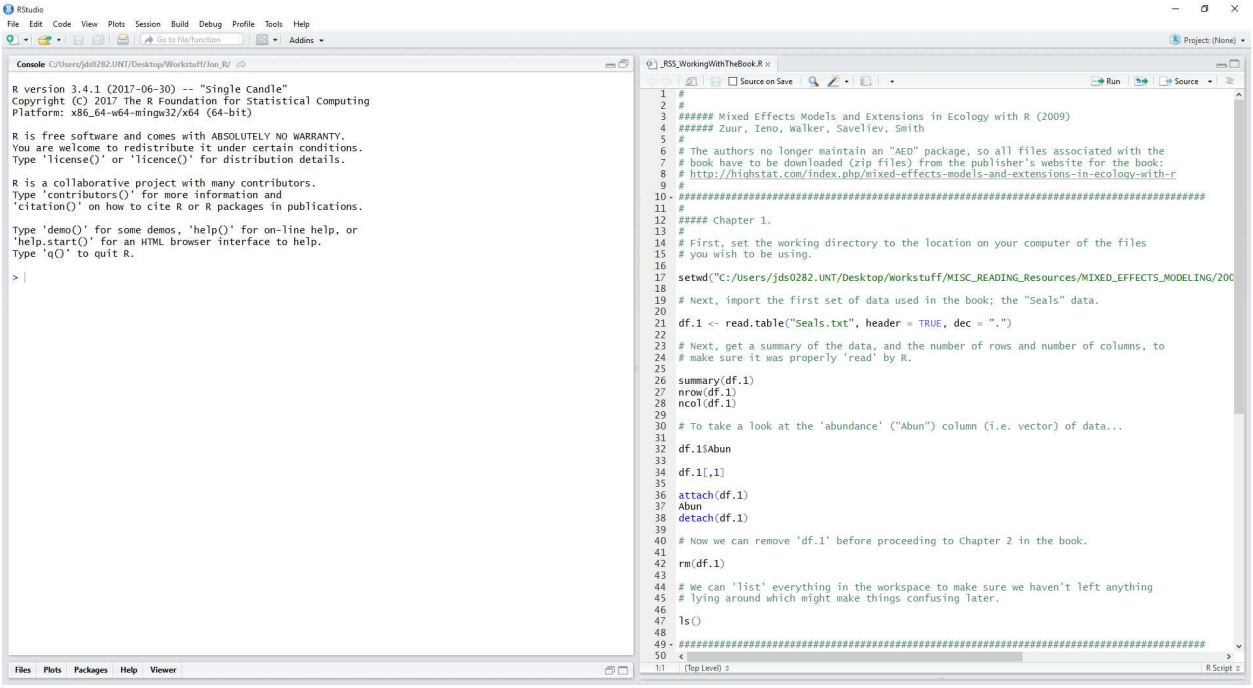
Obviously, personal preference should dictate how the panes are arranged and which tabs of each pane. My own preferences are to have the console on the left with the other side dominated by the script or “Source” with all other panes minimized.



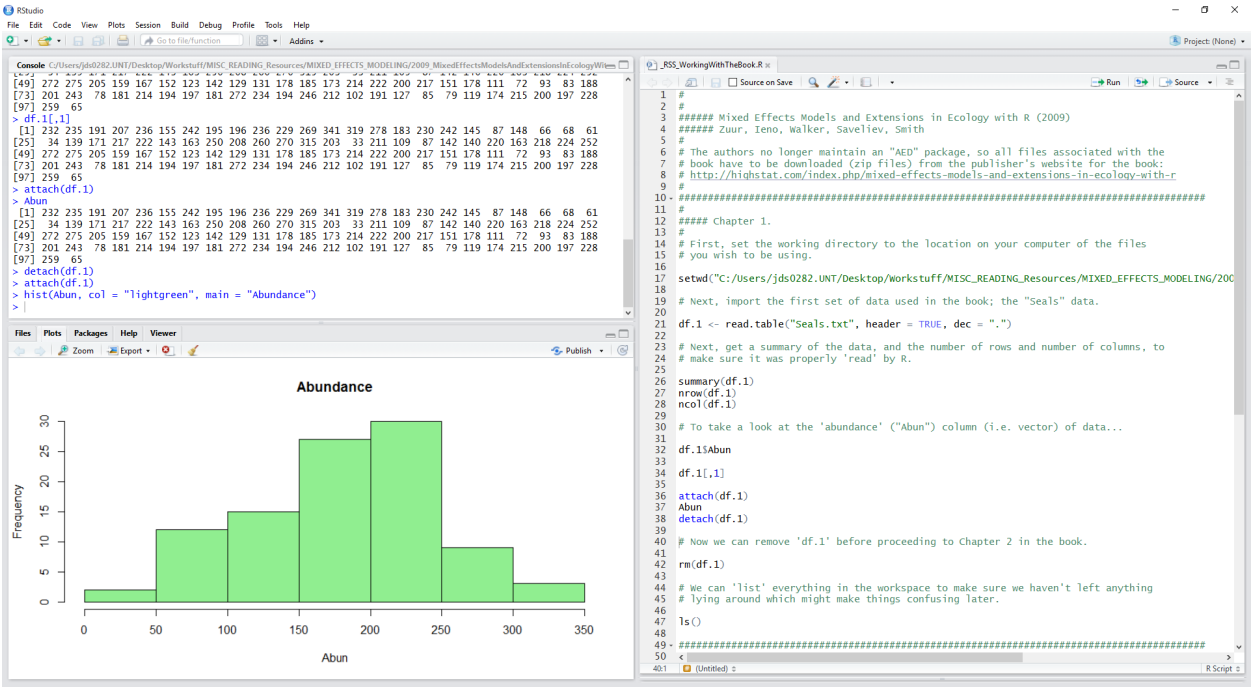
Here's what it looks like with that pane layout.



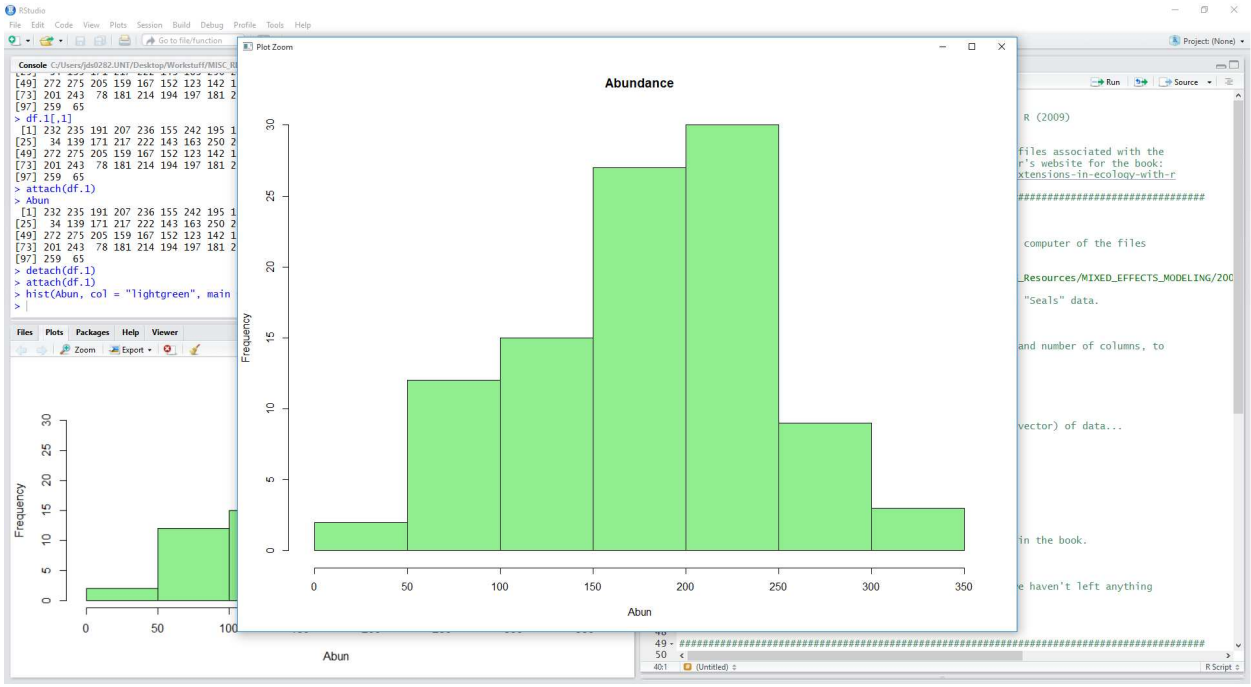
Here it is with a script (i.e. "Source") file opened.



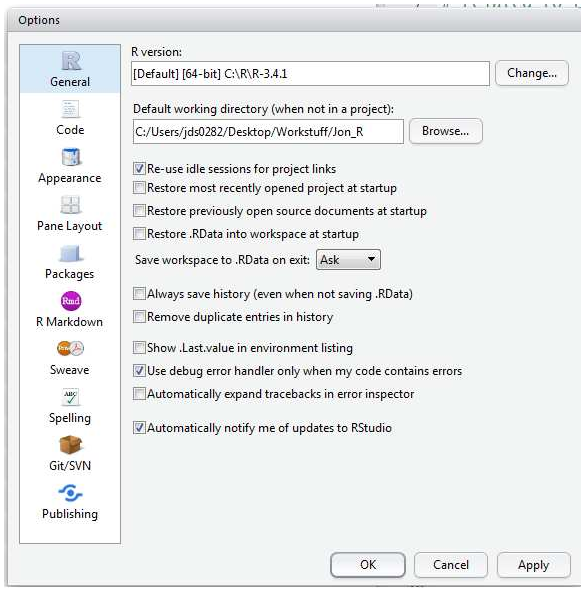
The layout panes will automatically change when one creates a graphical object (e.g. a histogram); such that the plots pane will expand to half the height of the left side and the console pane will collapse to half size.



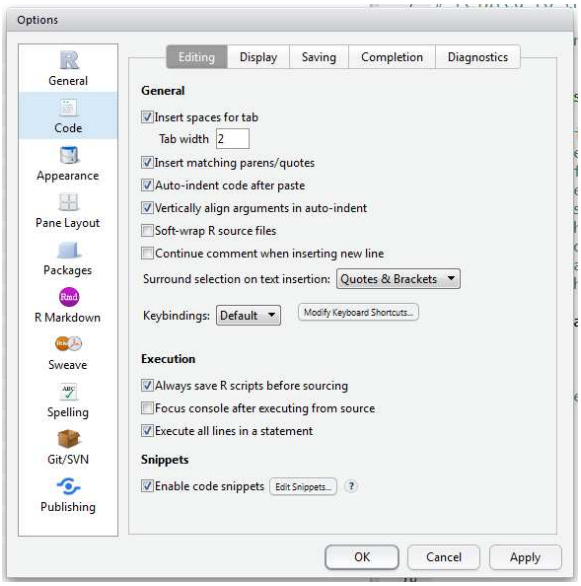
Keep in mind, you can click on the “Zoom” button of the plot pane to open the graph or plot in a new window (i.e. outside of RStudio) to get a better view of it. Then, minimize the plot pane to re-expand the console pane and continue working.



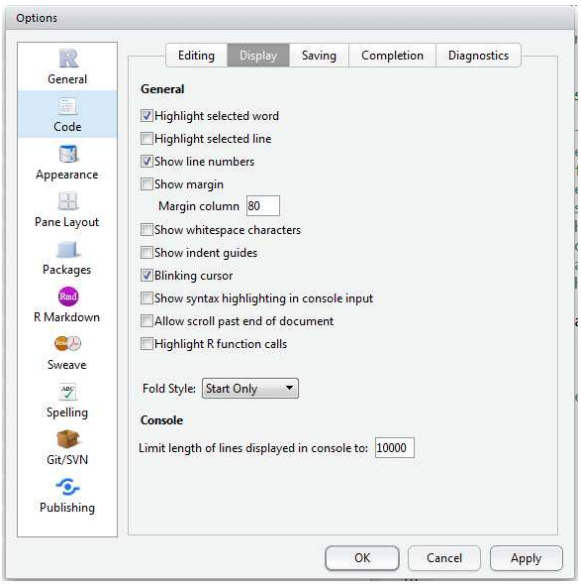
There are, over course, other options one may want to experiment with changing. Let's begin with the General options. I typically set the default working directory to a location which is more general than specific. Most of my work computers have a file for all my work stuff and within which is a general R oriented folder (e.g. what is displayed below). The desktop might be a good general location too. As a habit I always set the working directory at the top of each script file to the location of all the files related to a particular project. One must be careful when specifying paths as R only recognizes forward slashes / and not back slashes \which are commonly used by Windows operating systems.



Next, let's take a look at the Editing tab of the Code settings.



Inserting a specific number of spaces for tabs (e.g. 2 spaces, as default) is a good idea. That's because writing functions and loops, which often contain many lines and functions and loops within them, it can be otherwise difficult to *read* the code. RStudio brace-matches by default. This means that whenever the user types a parenthesis, brace, or bracket; RStudio automatically prints a corresponding version to *close* the parentheses, braces, or brackets. Most people see this as a convenient feature, I do not. I find it interrupts my typing flow so I turn that *feature* off. The only thing I change in the Display settings (tab) is the console limit length from 1000 to 10000.



I do not change any settings in the Saving, Completion, or Diagnostics tabs. Nor do I change any of the other settings (e.g. Appearance, Packages, R Markdown, etc.). However, I do set the mirror

(for downloading and updating packages) in the "Rprofile.site" file; which is located in the "etc" folder of the R installation. An example of which can be found on the R&SS Introduction to R short course website at:

`http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/Module2/JonsRpr`

That's about all I have to offer at this time. Again, I refer interested parties to consult the RStudio website to explore the functions, features, and customization of RStudio.

General R tips.

I believe there are two general perspectives when it comes to programming. One values efficient, compact, and elegant code. The other favors read-ability; can others, particularly those with less experience, decipher what was done and why? I imagine most folks with a computer engineering / computer science background tend to be the former. I am, clearly, of the latter perspective. Most of the scripts I produce, whether for the R course page or as templates / examples for specific clients...must be readable. Therefore, what follows below are my own thoughts on *good* R scripting habits or *good* R user habits.

Generally, good habits when working in R, or any programming language, include commenting copiously. This is perhaps the most important thing; using comments to describe what was done and why in the script files. The script files are the backbone of any analysis project *for me*. Why was a gender variable recoded and what were the codes used? Why were two variables eliminated from analysis? Answering these types of questions with comments in the script as the script is being created allows me, or anyone, to review the script to replicate what was done and provides an explanation for why it was done. In the current research climate one focus has been replication and reproduction of results. Documentation is the first key to allowing replication, or even reproduction, and comments in script are invaluable. Comments are also very useful for writing up results; again if habitually commenting while working through the data analysis it will make writing up a manuscript later much easier.

Returning to the settings and specific spaces for tabs, it's generally good for readability sake to indent subsequent lines which are contained inside a set of parentheses or braces, such as:

```
for (i in 1:1000){
  if(x[i] >= 0){y[i] <- y[i] + 25}
  if(x[i] <  0){y[i] <- y[i] - 25}
  z[i] <- x[i] + y[i]
  }
```

As one becomes more familiar with R it will become important to standardize the way one does certain things. For example; there are several ways to index or call a specific variable or column of a data frame. After learning all those ways, I would suggest settling on the one you find most comfortable and sticking with it in all subsequent scripts. Other general tips include trying to use recursion as much as possible. This can make the script hard to read, but is much more efficient and efficiency is a good thing to strive for when writing script. Also, saving workspace objects (.RData) can consume a lot of space; so, I tend to avoid saving the workspace of a project and

only saving the script – to save space on my machines. However, if I have conducted a resource intensive operation; say an iterative imputation technique which took 20 minutes to complete then I would certainly save the workspace and annotate the script to reflect that for later.

Lastly, it is important to note that many functions in R rely upon a random number generator to properly conduct *random* operations (e.g. `rnorm(n = 1000, mean = 10, sd = 1.5)`). If replicability and / or reproducibility are needed, then one must 'set the seed'. Setting the seed at the beginning of a script allows randomized functions to return the *exact* same results. Setting the seed is done with the following: `set.seed(12345678)`, with any numeric combination inside the parentheses. Obviously, the specific numeric combination used allows the same result to be produced (i.e. changing the seed number will produce differing results).

This document was created using LaTeX