CHAPTER 1

An Introduction to R

1.1 Why R?

Often, there is much consternation among students as to why they have to learn *yet another statistical package*. Can't those darned professors make up their minds? Are they trying to drive us crazy? Are they getting kick-backs from the statistical software salesmen?

The answer, more than likely, is that the specific professor truly believes that *this* statistical package is the best (by whatever measure he or she uses) available. I am no different. I truly believe that, overall, R is the best available for the following reasons:

- It is free (both free beer and free speech).
- It is flexible.
- It is powerful.
- It matches how science *should* be done.

Let us take these four reasons in the above order and explain what I mean by each.

First, it is free. The "R Project for Statistical Computing" is a not-for-profit foundation created for the sole purpose of creating a piece of software that encourages scientific innovations in any field that uses statistics.[1] The cost of the software is zero. That means students (and professors) do not have to pay fees (license or purchase fees) to use the software. As such, as university budgets tighten, expect R to become the environment of choice. Furthermore, because you are able to load it on a USB drive, you do not have to wander around campus searching for a computer sporting R, like you do with other statistical packages.[2] It is also free in the sense that you are free to modify the code to make it better. This latter part is what allows R to become better, stronger, faster, as time passes. As of this writing, the latest R version is Version 2.11.1.

Second and third, it is flexible and powerful. The base distribution of R contains only those parts of R that are universally useful. Thus, it is small and fast to download and begin. However, there are assorted libraries to do just about any statistical analysis. And, for those methods that are not currently supported by R, you are free to create your own libraries for everyone to use. Furthermore, it is a programming (scripting) language. As such, I can program it to do the same tasks repeatedly (with slight modifications) so that robust analyses can be done.

Finally, it matches how science *should* be done. R offers a definite separation between the data and the analysis. It also offers a way of keeping track of your analysis as you do the analysis. The former allows you to keep your original dataset unmolested. The latter allows your analysis to be replicated. Both of these are important hallmarks of science.

It is for these reasons that I use the R statistical environment. It is also for these reasons that I prefer to teach using it. I am not saying it will cure world hunger, but it will help you learn the right way to do science better than some other statistical programs.

1.2 Installing R on your computer

Installing R requires two steps: downloading the most recent base package to your computer, and installing the base package from your computer. If you wish to install R to a USB drive, that is also

[1] The URL for the R Project is http://cran.r-project.org/.

[2] If you wander outside the social science buildings, it is very difficult to find a machine with SPSS installed. Where is the closest SAS machine or STATA machine to this classroom? Is it installed on the classroom computer? With R on your USB drive, you will never care.

an option.[3]

1.2.1 Step 1: Download

Use your web browser to go to `http://cran.r-project.org/`. Once there, click on the type of computer operating system (OS) you have: Linux, MacOS X, or Windows. On the next page, you will click on the following link (depending on your OS) and download it to your desktop (or someplace just as convenient):

OS	File link
Linux	your specific distribution
MacOS X	Files: (latest version)
Windows	**base**, and then the **Download** link at the top

The specifics are up to your browser and your computer operating system.

1.2.2 Step 2: Install

Once the file is on your computer, run the file (an installer) and answer the questions it asks. Usually the default selection is appropriate. The only thing you may wish to change is the destination folder. If you want to save `R` to your USB drive so you can bring `R` with you, you will have to select *that* as your destination folder.

After the installer finishes, `R` is installed on your computer.[4]

1.3 A quick, sample session

As an example, let us do a quick, sample session that checks to make sure `R` is properly installed on your computer and which does no serious statistical analysis. In this session, you will start

[3] One would want to do this if one uses several computers and cannot be certain that `R` will be installed. The steps are the same as for a normal installation, with one change: Instead of selecting `C://R` as the destination folder, you will select the `R` folder on your USB drive.

[4] Actually, if you install it to your USB drive, it is not installed on the computer, *per se*. You will have to double-click the `R/bin/Rgui.exe` file each time you wish to run `R` from the USB drive. Doing it this way will mean you should become best friends with the `setwd` function.

R, open a new script window, type in an R script, execute the script, then save the script to your current working directory.

Before you start this session, please create a directory for your project, a place where your analysis will take place. In reality, you will have a different directory for each project. This is appropriate, as it keeps your projects, and their analyses, separate.

1.3.1 Step 1: Start R

If R is installed on your machine, find the icon and double-click on it. If it is on your USB drive, navigate to `USB:R-2.11.1/bin/` and double-click `Rgui.exe`.[5] Your screen should look something like Figure 1.1.

The R window has one subwindow right now — the 'R Console' window. The Console window is where all the analysis really gets done. All commands you type must eventually find their way to the Console window before R will actually execute them. However, the Console window should *not* be where you do your analysis. It is bad science to do your analysis in the Console window, since the commands you type there are lost once ENTER is hit, which means *replication* of your findings will be virtually impossible. It is proper to type your commands in a separate script window and send them to the Console window to be executed. Actually, 'proper' is not entirely correct here, 'the only acceptable manner' is much better.

While there are several third-party text editors which allow you to type your analysis and send it to the Console window, R provides a text editor that is more than sufficient.[6] The primary advantage to the built-in text editor is that it is easy to send lines of code in the Script window to the Console window to be executed—just type Ctrl+r.

1.3.2 Step 2: Start a new script

If you are using a third-party text editor, follow the directions provided by that vendor (which may be just copy-paste). If you are using the R text editor, open a new script: "File | New script...". You now have a second subwindow in the R window. This new window is titled

[5] If you installed a different version of R, then the name of the top folder will reflect that version.

[6] Throughout this book, I will tell you what I do and what I use. I use the text editor that comes with R. It does everything I need it to do. If it comes up short for you, then investigate some of the third-party options.

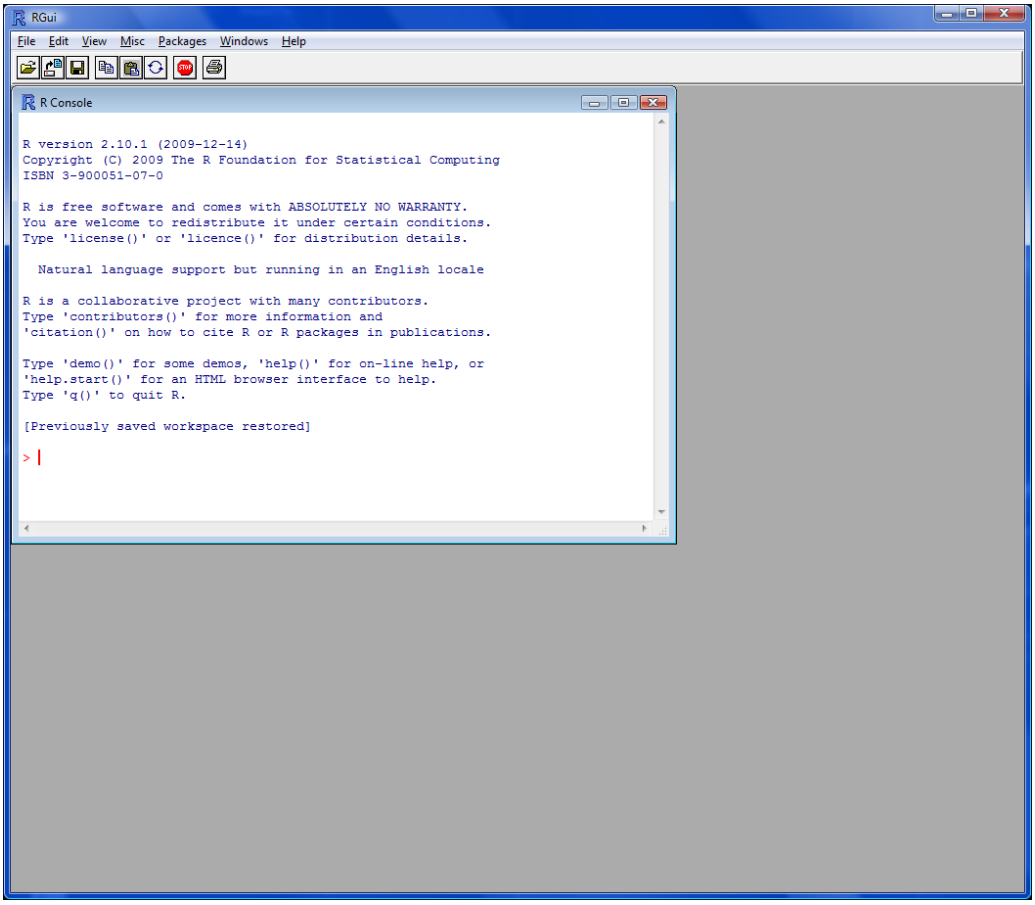

Figure 1.1: *The opening screen for* `R` *in Windows.*

"`Untitled - R Editor`". Tile the two windows so you can see both at the same time: "`Windows | Tile Vertically`". At this point, your `R` window should look similar to Figure 1.2

1.3.3 Step 3: Type in the script

The script you will be typing in to the `Script` window is a simple script that does the following three things: creates a univariate dataset, analyzes the dataset, and plots the data. The first goal is accomplished with a single line. The second by as many lines as levels of analysis you wish to perform. The last by two lines—one for each of the two manners of graphing univariate data.

The code is as follows (make sure you type it in correctly):

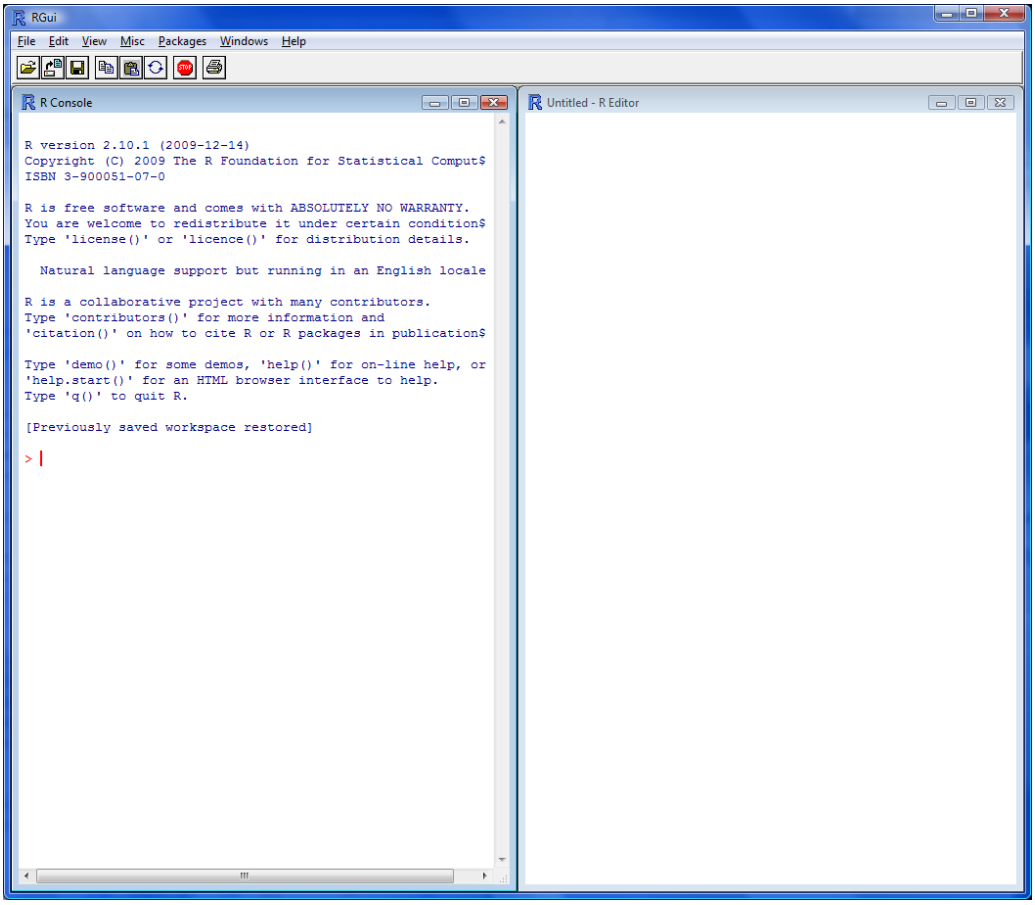

Figure 1.2: *The R Window after tiling the two subwindows (`Console`, left, and `Script`, right).*

```
# Sample script

set.seed(370)
x <- runif(500, min=10, max=20)
x
length(x)

mean(x)
median(x)
mode(x)

var(x)
sd(x)
IQR(x)

min(x)
```

```
17 max(x)
18 quantile(x)
19
20 mean(x, trim=0.05)
21 quantile(x, 1:100/100)
22
23 boxplot(x)
24 hist(x)
```

What does this script do? The first line is a comment. The comment character is the hash symbol, `#`. Anything on the line following the `#` is ignored by `R`. Commenting your script is a very good idea; it makes the script more understandable by everyone. The second line sets the random number seed, which guarantees your dataset will be the same as mine. When we get to simulation, you will have need of this function, and we will discuss it in greater detail then.

The next line creates a variable named '`x`' and puts 500 uniform random numbers between 10 and 20 in the variable `x`.[7] In the language of probability,

$$x \overset{\text{iid}}{\sim} \mathcal{U}(10, 20) \tag{1.1}$$

The '`r`' in `runif` indicates its random aspect. The '`unif`' in `runif` indicates the uniform distribution. Other options include `rnorm` (normal distribution), `rt` (Student's t-distribution), and `rexp` (exponential distribution). Each of the other random number generator distributions have different options, see the `R` Help for specifics.[8]

The next line displays the contents of `x` — the entire dataset. The next line displays how many elements are in `x` — the sample size. In statistical notation, the sample size is represented by the variable `n`. As such, you will see, in my programs, a line such as:

```
n <- length(x)
```

The next lines find the following information about the data stored in `x`: mean ($\bar{x}$), median ($\tilde{x}$), mode, variance (s^2), standard deviation (s), interquartile range, minimum value, maximum value, the five-number summary (quartiles 0 through 4), the 5%-trimmed mean, and all 100 percentiles.[9]

[7] The assignment operator in `R` is not the equals sign, `=`, it is the left-arrow `<-`, a less-than sign and a hyphen.

[8] This is as good a place as any to introduce the `R` Help and `R` Search functions. If you know the actual command or statement, but you forget its specifics, type a question mark followed by the command in quotation marks(for example, `? "rnorm"`). However, if you do not know the actual command, but you know a word close to it or what it does, type two questions marks followed by the word or words in quotation marks (for example, `?? "random number"`).

[9] Where quartiles divide the dataset into 4 equal quarters (hence 'quartile'), percentiles divide the dataset into 100 equal parts. 'Equal' in this sense is number of elements. Note, however, that equal is not truly equal unless the number of elements in a dataset has certain properties; it means 'approximately equals.'

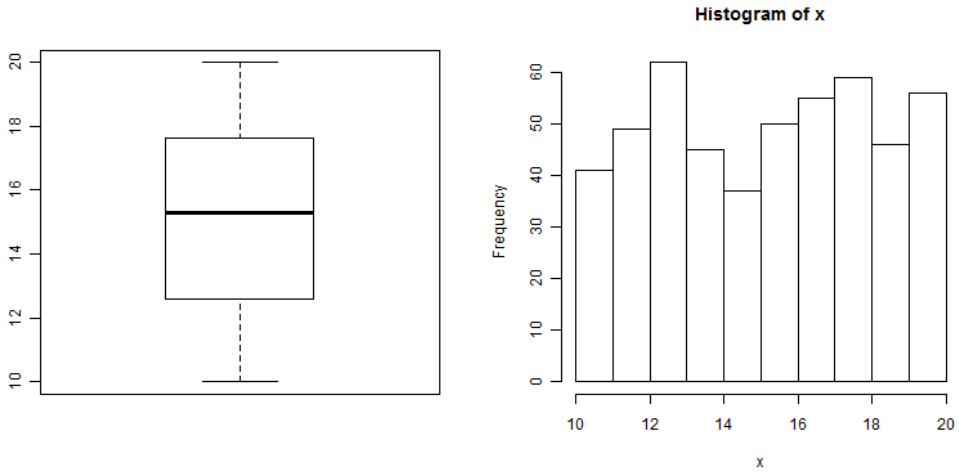

Figure 1.3: *The results of initial graphical analysis of the random uniform dataset. Left Panel: Boxplot. Right Panel: Histogram.*

The last two lines plot the data in two different ways. The boxplot displays the median (heavy bar in center) and Quartiles 1 and 3 (the ends of the central box). It also displays bars at either the minimum and maximum values or, if there are outliers, at $\tilde{x} \pm 1.5 \times IQR$, which provides a boundary for the outlier data values (dots in the boxplot). There are no outliers in this dataset, so the upper bar is the maximum value in the dataset and the lower bar is the minimum value in the dataset. Your boxplot should look like that in Figure1.3, Left Panel.

The histogram separates the data into bins of equal width (as a default) and plots the frequency of data in each bin. Your histogram should look like Figure 1.3, Right Panel. As the data comes from a uniform distribution, we expected the histogram to be flat. All bumpiness is due to the inherent randomness of sampling and the small sample size.[10]

1.3.4 Step 4: Save the script

If this were an analysis you used for your research, you would definitely want to save it. Saving the script is rather straight-forward: `File | Save`. If you do not see the `Save` option, then the

[10]Technically, the histogram is an approximation of the probability density function (pdf). The pdf is the 'formula' for the distribution. Different distributions have different pdfs and (therefore) different histograms. As it is an approximation, the histogram will not *be* the pdf; it will only approximate it. Better approximations occur with larger sample sizes.

active subwindow is not the script. Click on the script window and retry the save procedure.

1.4 Conclusion

In this chapter, you have learned why I like R and why I will use it in the classroom. You also learned how to download and install R on your computer (or USB drive); how to type in a simple, yet informative, program; and how to use help and search functions to locate information about statements, commands, or functions in R. The next chapter will deal with reading in your data in its various forms, as well as additional topics in univariate statistics.

1.5 Extensions

This section offers suggestions on things you can practice from just the information in this chapter. Completing these extension exercises requires judicious use of the help and search functions in R, as well as some trial and error—both things I use quite frequently.

1. Start a new session in R, open a new script, and create a random dataset from the Gaussian distribution. [The Gaussian distribution is also known as the normal distribution.]

2. Start a new session in R, open a new script, write a script that creates a dataset of size one million (n=1 000 000), with each element in the dataset being a random number between 1 and 6. Now, find the average value of the dataset. [Unless stated otherwise, we assume that 'random' numbers are from a uniform distribution.]

3. Extending the previous extension, simulate rolling a fair six-sided die by making all of those random numbers integers. [You will want to look up the following three functions: `floor`, `ceiling`, and `round` to determine which of the three is appropriate (and what other changes you may need to make).]

4. Extending the histogram graph done above, make the following alterations: Change the x-axis label to "Pipe Length", the histogram title to "Histogram of Pipe Lengths", and the orientation of the axis labels to horizontal.