APPENDIX C

Getting Started with R

What is R?

R is a powerful computer language that merges the convenience of statistical packages with the power of coding. It is open source as well as cross-platform compatible. This means that there is zero cost to download R, and it can be run on Windows, macOS, or Linux. In this appendix, we will introduce you to some fundamental features of R and provide instructions on how to obtain solutions for many of the exercises in the text.

What is RStudio?

RStudio is a program that makes R easier to use. On its own, R acts like a programming language and, as such, comes with a minimal user interface. As standalone software, R shows a single prompt for you to enter commands; this is called the Console. While everything we will ever need from R can be done by combining Console commands with other programs, things can quickly get messy. To make coding in R easier, we use an integrated development environment (IDE). IDEs are programs that combine in one place many common features needed in programming and give them a graphical user interface.[1] In this text, we use an open source version of an IDE called RStudio, which is very popular among students, professionals, and researchers who use R.

Installation

Installation of both R and RStudio is straightforward and requires no special modifications to your system. However, it should be noted that RStudio does not come with R; *therefore, both pieces of software need to be installed separately.* Also, all R code in the text and its accompanying output are based on **R version 3.5.3 on Microsoft Windows**. Even though there will be newer versions of R when you prepare to download it, we suggest that you download **R version 3.5.3**. Newer versions of R may not be compatible with certain R packages, especially those packages that we use in later chapters of this text. All versions of R for Windows can be found at the following website: https://cran.r-project.org/bin/windows/base/old/. We discuss R packages at the end of this Appendix.

Installing R for Windows

A. Navigate to https://cran.r-project.org/bin/windows/base/old/.
B. Select *R 3.5.3 (March 2019)*.
C. Select *Download R 3.5.3 for Windows*.
D. Locate the downloaded file and double-click.
E. Select *Yes* when asked about verifying the software publisher, and then select the language that you prefer. We select *English*.
F. Follow the instructions in the R Setup window.

For Mac and Linux, follow the instructions at https://cran.r-project.org/bin/macosx/ and https://cran.r-project.org/bin/linux/, respectively, and choose the appropriate link for your operating system and version.

[1]More formally, this is called a "graphical user interface," or a GUI. In practice, this means that charts, graphs, and buttons can be seen and used.

Installing RStudio

A. Navigate to https://www.rstudio.com/products/rstudio/.

B. Select *RStudio Desktop,* then select *Download RStudio Desktop.*

C. Scroll down to the *Installers for Supported Platforms* section, select the link that corresponds to your operating system, and then select *Open* or *Run.*

D. Select *Yes* when asked about verifying the software publisher.

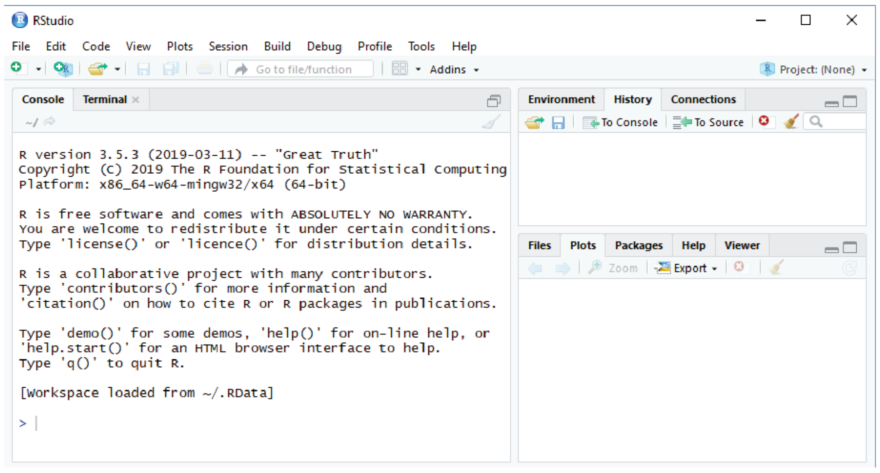
E. Follow the instructions in the RStudio Setup window.

The Interface

Installation should now be complete. You can close all windows and then double-click on the RStudio icon.

The RStudio interface consists of several panes. By default, three panes are visible. We will refer to these by the names of the default tab shown in each: Console, Environment, and Help. We will also briefly discuss the Source pane, which is hidden until you open it. Figure C.1 shows what you should see when you open RStudio for the first time.

FIGURE C.1 The Console, Environment, and Help Panes


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
~/
Environment History Connections
To Console
To Source
Files Plots Packages Help Viewer
Zoom
Export

```
R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> |
```

Source: R Studio

- **Console pane:** The Console pane is the primary way that you interact with R. It is here that you input commands (at the > prompt) and then view most of your output.
- **Environment pane:** Two relevant tabs in the Environment pane are: Environment and History. A common feature between them is the broom icon, which clears the content of each tab. The Environment tab shows the data, objects, and variables in the current R session. The History tab provides a list of all console commands issued in the session.
- **Help pane (or Files pane):** The help section has five tabs. We discuss two of these here: Help and Plots.
  - The Help tab is where you can view R documentation (help files). For example, to learn about the **print** function, select the Help tab and then enter `print` next to the magnifying glass icon. (You can also view R documentation by entering a question mark followed immediately by the topic of interest in the Console pane; so, for this example, you would enter `?print` after the prompt.)
  - The Plots tab is where you can see all graphs and charts. Any graph or chart can be cleared with the broom icon.
- **The Source pane:** The Source pane is hidden by default in R. This is where you can write your own scripts. As you will see, most of what we do in this text can be accomplished by importing a data set and then using a single command in the Console. Nonetheless, here is an example of how you would write a simple script:

  A. From the menu, select **File > New File > R Script**

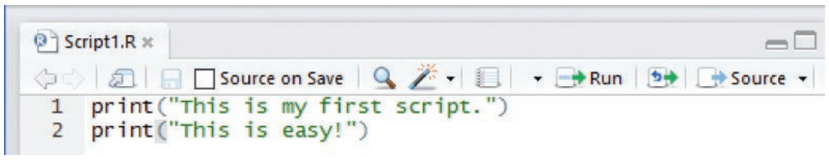
  B. In the new window, enter the following:

```
print("This is my first script.")
print("This is easy!")
```

Save the script with **File > Save As.** Name your script `Script1`. Figure C.2 shows what you should see in the Source pane.

Important: Due to different fonts and type settings, copying and pasting formulas and functions from this text directly into Excel may cause errors. When such errors occur, you may need to replace special characters such as quotation marks and parentheses or delete extra spaces in the functions.

FIGURE C.2 The Source Pane After Writing First Script

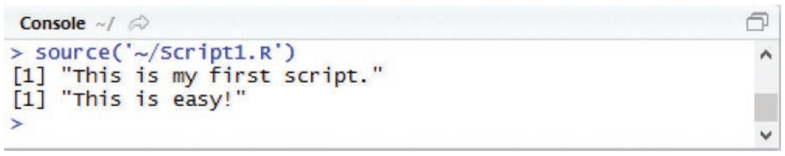


```
print("This is my first script.")
print("This is easy!")
```

Source: R Studio

C. Again refer to Figure C.2. Click the Source button from the menu on the Source pane; this tells R to read and execute the script. Figure C.3 shows what you should see in the Console pane after executing your first script.

FIGURE C.3 The Console Pane After Executing First Script



```
> source('~/Script1.R')
[1] "This is my first script."
[1] "This is easy!"
>
```

Source: R Studio

R executes complete statements in the order that they appear. Unique to RStudio, there is also a way to run specific sections of scripts. This is done by highlighting the desired section of the script in RStudio and selecting the Run button from the menu on the Source pane.

Entering Data and Using Functions

Throughout this text, our goal is to provide the simplest way to obtain the relevant output. Seasoned users of R might argue that there are "better" approaches than the ones we suggest, but we feel that they may distract from learning the important concepts.

Like Excel and other statistical packages, R has many built-in formulas or functions. In the text, we denote all function names in **boldface.** Within each function, R also provides various options, such as labeling the axes of a graph, inserting colors in a chart, and so on. We will not use every option within a function; rather, we use those that we feel are most useful and least cumbersome. We denote all option names in *italics.*

Most of the time we will be importing data files, as we explain in the next section. However, suppose we want to use R to perform a simple calculation. Suppose we want to calculate the mean given the following data: −4, 0, 6, 1, −3, −4. In order to input these values into R, we use the **c** function, which combines the values to form a list; or, perhaps more mathematically precise, the **c** function combines the values to form a vector. We label this data as Example_1 and use the expression "<-" which is equivalent to the equal sign. We enter:

```
Example_1 <- c(−4, 0, 6, 1, −3, −4)
```

You should see Example_1 listed in the Environment pane. You can view the data in the Console pane by entering Example_1 after the prompt. Additionally, you can use the **View** function and the data will appear in the Source pane. (Note that R is case sensitive.) We enter:

```
View(Example_1)
```

Another common function is the **mean** function which we discuss in more detail in Chapter 3. In order to calculate the mean of the data, we enter:

```
> mean(Example_1)
```

And R returns: −0.6666667.

FILE
Admission

Importing Data and Using Functions

All the data for *Business Analytics—Communicating with Numbers* have been stored in Excel spreadsheets. We will assume that you have stored all the relevant spreadsheets in a Data folder. When we import a spreadsheet into R, it is referred to as a data frame. A data frame is a table, or two-dimensional array-like structure, in which each column contains measurements on one variable and each row contains one observation, record, or case. A data frame is used for storing data tables.

We illustrate the mechanics of importing an Excel file using the ***Admission*** data from Chapter 2. The data set contains the student record number (Student), the college decision on acceptance (Decision = Admit or Deny), the student's SAT score, whether the student is female or male (Female = yes or no), and the student's high school GPA (HSGPA). Table C.1 shows a portion of the ***Admission*** data.

TABLE C.1 Portion of the Admission Data

Student	Decision	SAT	Female	HSGPA
1	Deny	873	No	2.57
2	Deny	861	Yes	2.65
⋮	⋮	⋮	⋮	⋮
1230	Admit	1410	No	4.28

In order to import this data file into R, we select **File > Import Dataset > From Excel,** as shown in Figure C.4.[2] (The first time you import data, R might prompt you to add updates. Simply follow the steps to add the relevant updates.)

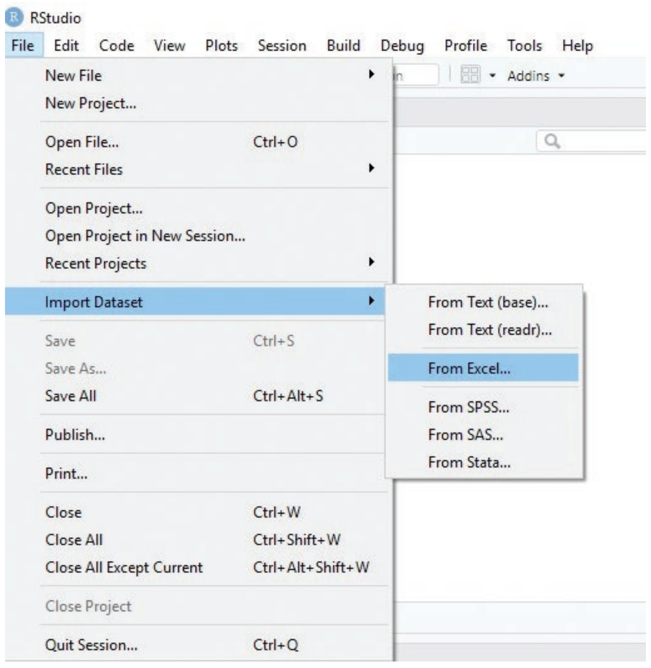


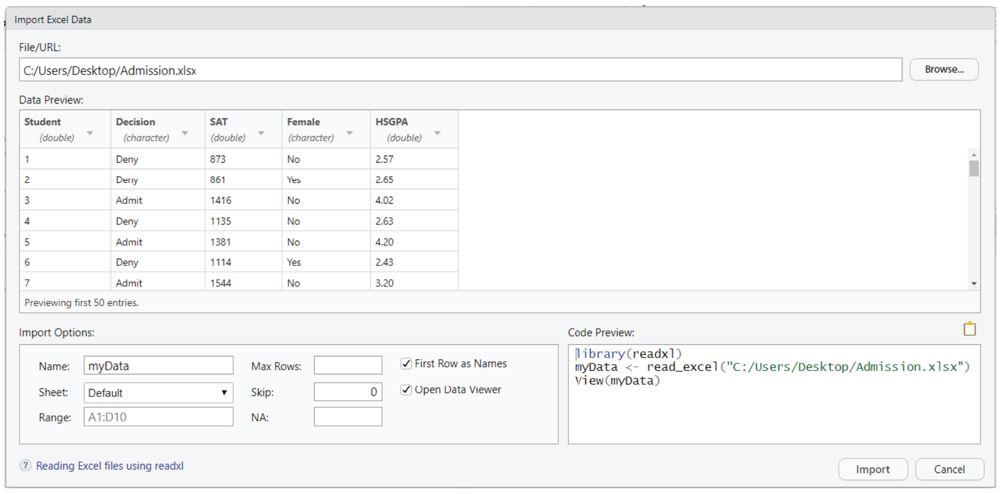
FIGURE C.4 Importing the Admission Data into R

Source: R Studio

[2]Note that you can also import a comma- or tab-delimited text file by selecting **File > Import Dataset > From Text (base)** or **File > Import Dataset >From Text (readr),** respectively.

See Figure C.5. We select the Browse button and then navigate to the ***Admission*** data in the Data folder. Once we select the ***Admission*** data, we should see the data in the *Data Preview* dialog box. In the R instructions in this text, we label all data files as myData for simplicity and consistency. Because of this, in the *Import Options* dialog box, replace Admission with myData. Once you select the Import button (see the bottom of Figure C.5), you have successfully imported the data. You can verify this in a number of ways. For instance, you should now see myData in the Environment pane under Data, or you can enter `View(myData)` in the Console pane and a portion of the data will appear in the Source pane.

FIGURE C.5 Viewing the Admission Data Prior to Importing Source: R Studio



File/URL: C:/Users/Desktop/Admission.xlsx

Data Preview:

Student (double)	Decision (character)	SAT (double)	Female (character)	HSGPA (double)
1	Deny	873	No	2.57
2	Deny	861	Yes	2.65
3	Admit	1416	No	4.02
4	Deny	1135	No	2.63
5	Admit	1381	No	4.20
6	Deny	1114	Yes	2.43
7	Admit	1544	No	3.20

Previewing first 50 entries.

Import Options:

Name: myData
Max Rows:
✓ First Row as Names
Sheet: Default
Skip: 0
✓ Open Data Viewer
Range: A1:D10
NA:

Code Preview:

```
library(readxl)
myData <- read_excel("C:/Users/Desktop/Admission.xlsx")
View(myData)
```

Reading Excel files using readxl

Note: For an Excel file with multiple worksheets, select the appropriate worksheet from the *Sheet* drop-down option.

Suppose we want to calculate the mean SAT score in myData. In order to select a variable from a data frame, we attach the expression $VariableName to the name of the data frame. Here, we enter:

```
> mean(myData$SAT)
```

And R returns: 1197.348.

If the variable name in the data frame is more than one word or numeric (such as year), then it is necessary to enclose the variable name with single quotation marks. For instance, if the variable name was SAT score instead of SAT, then we would have entered mean(myData$'SAT score').

Another function that we discuss in Chapter 3 is the **summary** function. This function provides various summary measures for all variables in a data frame. We could enter `summary(myData)` and R would return summary measures for all the variables in the myData data frame, including the categorical variables. Suppose we would like summary measures only on SAT and HSGPA. In this case, we attach square brackets to the name of the data frame, and within the brackets we indicate the columns that should be included in the calculations using the **c** function. In order to obtain summary measures for SAT (third variable) and HSGPA (fifth variable), we enter:

```
> summary(myData[,c(3,5)])
```

Notice in the above command that we enter a comma directly after the left square bracket. This implies that we are including all 1,230 observations (all 1,230 rows) in the calculations. If for some reason we only wanted to include the first 100 observations in the calculations, we would have entered `summary(myData[1:100,c(3,5)])`.

Finally, suppose we want to delete the myData data frame. We use the **rm** function and enter:

```
> rm(myData)
```

You will find that myData no longer appears under *Data* in the Environment pane.

A Note on Line Breaks

The commands that we have outlined here have been relatively short. There are some instances, however, when the commands get long and become difficult to read. To mitigate this, we can break up a command into parts. R will prompt you to finish the command with plus + symbols in lines following the first line. For example, in Chapter 3 we discuss a scatterplot. Suppose we want to construct a scatterplot of SAT against HSGPA for the first 20 observations using R's **plot** function. In addition to constructing the scatterplot, we use the *ylab* and *xlab* options to add titles on the *y*-axis and the *x*-axis. Two entries for constructing a scatterplot are shown below. Entry 1 uses a single line in R (even though two lines are shown on the page). Entry 2 uses three lines. Both entries result in the same scatterplot, as shown in Figure C.6.

Entry 1:

```
> plot(myData$SAT[1:20] ~ myData$HSGPA[1:20],
  ylab="SAT Score", xlab="High School GPA")
```

Entry 2:

```
> plot(myData$SAT[1:20] ~ myData$HSGPA[1:20],
  + ylab="SAT Score",
  + xlab="High School GPA")
```

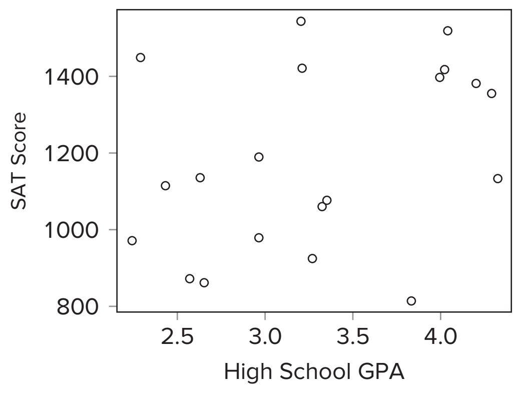


FIGURE C.6 Scatterplot of SAT against HSGPA

Packages

Part of what makes R so powerful is its large collection of packages, or collections of objects not included in the base version. Packages greatly expand what can be done with R by adding custom functions and data structures. As mentioned earlier in this Appendix, we use these packages in conjunction with **R version 3.5.3.** Compatibility issues may arise if you use another version of R.

To use a package, you must install it and then load it. We use the *caret* package, which stands for <u>**C**</u>lassification <u>**a**</u>nd <u>**Re**</u>gression <u>**T**</u>raining, to demonstrate how this is done:

```
> install.packages("caret")
> library(caret)
```

The **install.packages** function connects to the official R servers (CRAN), downloads the specified package(s) and those it depends on, and installs it. This must be done with each package only once on each computer used. The **library** function loads the installed package(s). Suppose you want to install multiple packages, such as *caret, gains,* and *pROC.* Instead of installing each package separately, you can use the following command:

```
> install.packages(c("caret","gains","pROC"))
```

However, you still need to use the **library** function separately for each package as follows:

```
> library(caret)
> library(gains)
> library(pROC)
```

Sometimes, R may prompt you to install additional packages. If this is the case, follow the steps outlined here to download and install these additional packages.

Note that each package only needs to be loaded once per R session. Once the package is downloaded and loaded, documentation for commands it contains can be viewed in R using the help feature discussed earlier. Documentation files for an entire package can be viewed online. All information associated with available packages can be found at https://cran.r-project.org/web/packages/.