1 Getting Started

CHAPTER OUTLINE

LEARNING OBJECTIVES

- Explain the difference between R, RStudio, and R Markdown.
- Distinguish between functions and objects.
- Download, install, and navigate both R and RStudio.
- Identify the main features of R Markdown.
- Create a Word or HTML file from R Markdown.

Overview

Learning how to use a high-powered microscope in biology class is a useful analogy to our task in this chapter. Learning how to manipulate and examine data in RStudio is analogous to positioning biological material on a slide, correctly placing it underneath the lens, and bringing it into focus. This analogy can be taken one step further. In biology class, you also learn how to take lab notes and present your findings. RStudio makes it easy to record your analysis and to present your results with professional-looking documents.

The goal of this chapter is to introduce you to R, RStudio, and R Markdown. We will be using RStudio to execute commands in R. RStudio makes many tasks in R easier since it provides an intuitive interface between you and R. Within RStudio, you will use R Markdown to take notes and generate professional-looking documents.

Mastering statistical software is an important part of analyzing data. Good analysis follows from scientists' skillful use of their lab equipment; scientists skilled in the lab are more likely to make interesting discoveries. The same holds true for you.

R, RStudio, and R Markdown

R is a statistical programming language based on **S**, a programming language developed at Bell Labs in the 1960s. R is an open-source language, administered by the R Core Development Team and the R Foundation. The program is free. Thousands of users contribute by writing their own R code and making it available in **packages**. Thousands of packages exist, all designed to help with specific aspects of data analysis. Packages are add-on features that you install to help make your analysis easier and more effective. Users submit packages to the R online community to test the code. Once testing is complete, the code is made available to all (https://www.r-project.org).

The interface for R, what the user actually sees when running it, is rather austere and based primarily on typing instructions to a prompt. In 2010, J. J. Allaire developed RStudio, an integrated development environment (IDE) that makes using R much easier. While the work you'll be doing in this book will technically be in R, you will be executing the commands, taking notes, and producing reports using RStudio.

Within RStudio, there is a note-taking and document-generating feature called **R Markdown**. R Markdown is a lightweight markup language for working with R based on Markdown, a more general lightweight markup language. Why bother with R Markdown? Why not simply cut and paste graphics and statistical output into a Word file? While you can simply cut and paste graphs and statistical output into Word, I recommend using R Markdown. R Markdown serves two important purposes. First, it makes combining data graphics with text extremely easy, generating documents in HTML, Word, or $\LaTeX$.[1] Second, R Markdown encourages the careful practice of taking notes so that you can easily replicate your results. Communicating results, both to yourself and to a wider audience, is an important but often overlooked part of data analysis. Not here.

This book is designed to be read with your computer at your side. Once you download R and RStudio onto your computer (which we will describe in more detail next), you will be using RStudio to do all of the analysis. You do not need to open R, just RStudio. Almost every table and figure in this book was created with RStudio. There are exceptions, in which case I began with RStudio and edited the figure with Adobe Illustrator. To follow along with this book, you will only need RStudio.

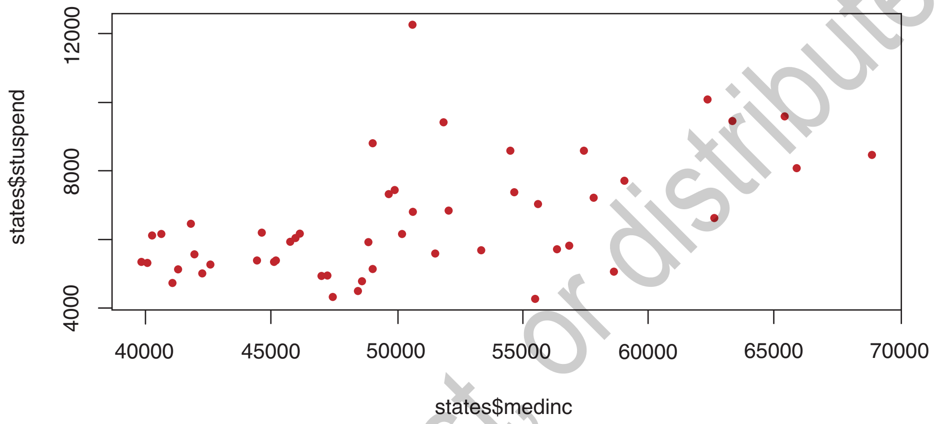
This book is designed so that the examples can be easily replicated. The instructions you type into RStudio (the R code) for each table and figure are presented before the actual table and figure appear in the text so that you can take the code, implement it, and reproduce the same tables and figures. The code will always appear in what will be referred to as a "code chunk" just above the table or figure produced. Just to give an example, the code in Code Chunk 1-1 produces a figure called a scatter plot. I'll go into more detail on scatter plots in this and subsequent chapters. For now, it's important to establish the direct connection between the code chunk and the figure that follows. In Figure 1-1, I plot the relationship between state spending on education per student and the state's median income.

[1] PDF files can only be produced if the appropriate $\LaTeX$ files are loaded onto your machine. $\LaTeX$ accomplishes the same thing as R Markdown but is a much more complete and therefore complicated markup language. If you do have the proper $\LaTeX$ files on your computer, when you generate a PDF file RStudio converts the R Markdown markup language to $\LaTeX$. While files generated this way are of exceptional quality since they use the formatting language based on TeX, it can open up a whole new world of syntax that requires mastery. To produce PDF documents, a suite of files must be downloaded. The required software can be found at https://www.latex-project.org/get/#tex-distributions and is available for Mac, Windows, and Linux platforms.

Code Chunk 1-1

```
plot(states$medinc, states$stuspend,
     main = "Figure 1-1: Spending is Higher in Wealthier States",
     col = "#bf0000", pch = 20, font.main = 1)
```

FIGURE 1-1 Spending Is Higher in Wealthier States



Art and Practice of Data Visualization

COPY RELEVANT BITS OF CODE

Experienced practitioners rely on several pieces of code that they know produce specific graphs and statistical analysis. Having a file where that code can be found, copied, and edited is highly recommended. RStudio makes that easy. Rarely are entire analyses created by typing code from scratch (there's simply too much to remember and therefore there are too many possibilities for mistakes).

KNOWLEDGE CHECK: Explain the difference between R, RStudio, and R Markdown.

1. Which of the following statements accurately describes R?
   a. A statistical programming language based on S
   b. An integrated development environment (IDE)
   c. A lightweight markup language
   d. A statistical programming language based on RStudio

2. Which of the following statements accurately describes RStudio?
    a. A statistical programming language based on S
    b. An integrated development environment for R
    c. A lightweight markup language
    d. A statistical programming language based on R
3. Which of the following statements accurately describes R Markdown?
    a. A statistical programming language based on RStudio
    b. A lightweight markup language
    c. A feature in RStudio that helps replicate results
    d. A feature in RStudio that helps presents results

Objects and Functions

While this is not a book dedicated solely to learning R, there are a couple of concepts worth mentioning before we embark on our journey. For comprehensive treatments of the R language, I find Kurt Gaubatz's *A Survivor's Guide to R* the most helpful (Gaubatz, 2015). Instead of a full-blown chapter (or two or three) on R, I want to briefly discuss two features of R that I think will help make sense of the instructions you'll be typing or copying into RStudio.

Let's start with functions. In R, the words function and command are interchangeable. **Functions** tell R to take specific actions. They are like the functions you learned in high school math or like the verbs you use every day: *open* the door, *eat* your broccoli, or worse, *finish* your homework. In these examples, the commands or functions are *open*, *eat*, or *finish*. Also note that each function is paired with an object: *door*, *broccoli*, *homework*. In R, functions are denoted by a word directly followed by a set of parentheses that contain an object. R code for the examples above would look like *open(door)*, *eat(broccoli)*, or *finish(homework)*. Since R really doesn't physically open doors, eat broccoli, or finish homework (although one could argue it does open doors and it at least helps finish homework), let's use a more realistic example from data we will be using in the book. Here is an actual example from R:

```
hist (states$hsdiploma)
```

Here *hist* is the function and *states$hsdiploma* is the object. In this simple example, we're telling R to create a histogram (a simple picture of the data we'll use in our first session). I will provide all of the data we use in this book, which includes information on the 50 U.S. states, information about respondents from a survey, and information collected from countries throughout the world. In this example, the data we will use is the percentage of each U.S. state's adult population that finished high school. In the data, it is called *states$hsdiploma*. Whenever you see a word before parentheses, you'll know that word is a function. There are lots of functions in R, and you can define your own.

The other basic concept is the **object**. In the everyday examples above, the objects are *door*, *broccoli*, and *homework*. While functions might be considered verbs, objects would be nouns. In the R example above, *states$hsdiploma* is the object on which the function *hist* operates. R is what we call an object-oriented language. That means R is designed to work with objects. In the example above, the object *states$hsdiploma* is a list of numbers. An object can be a list of numbers, several lists of numbers (what we'll call a data set), or a variety of other things. You can define your own objects. R code that defines objects can be recognized by the combination of the *lesser than* sign and the dash ("<-"), which essentially means "is equal to." Consider the following example:

```
myobject <- hist (states$hsdiploma)
```

In this example, I am defining an object named *myobject* as a histogram of the measure named *states$hsdiploma*, the percentage of each state's adult population with a high school diploma. The structure of *states$hsdiploma* is simply a list of numbers, one number for each state. To help solidify the concept of functions and objects, let's define a function and an object with some numbers we'll simply type into the computer. Let's start by defining an object:

```
newobject <- c(1,1,1,1,3,4,5,5,5,5,6,6)
```

The code above tells R to create an object called *newobject* and to define it as a list of 12 numbers. Here I decided to make a list of 12 numbers ranging from 1 to 6. The list of numbers has the same structure as *states$hsdiploma*, only we're using numbers we entered ourselves. The list called *states$hsdiploma* contains 50 numbers (one number for each state). The *c* in this instance tells R to combine the numbers into a list. Once we have *newobject* defined as a list of 12 numbers, we can use a function to make a histogram of that list, which is now an object called *myobject*. The command is written as follows:

```
hist(newobject)
```

This barely scratches the surface as an introduction to the object-oriented programming language R. The primary reason we're not dwelling too long on the basics of R is simple: the best way to learn data analysis using R is *to do* data analysis using R. Second, current practice in coding, regardless of the language (R, C^{++}, JavaScript, Python, etc.), involves identifying useful chunks of code (often from a variety of different sites on the Internet), copying them, and using them in your work. Being able to recognize a function and an object will carry you a long way. Rest assured, this book contains all of the chunks of code you'll need to do sophisticated, effective, and insightful work.

There is one more note before we proceed. Most of the code you'll see in R is a function (or command) with a set of parentheses. As I noted, in those parentheses is an object of some sort. Just like our example of a door, broccoli, and homework, the object can be many different things. You'll notice as we proceed that in the parentheses there are often additional instructions we give, always separated by commas. Those extra commands allow us to be more specific. In our example of *open(door)*, we could have *open(door, slowly)* or *open(door, just a crack)*. These commands can get fairly long, but they give you that much more control over how the data are manipulated, providing even more insight.

KNOWLEDGE CHECK: Distinguish between functions and objects.

4. Given the analogy between verbs and functions in R, which of the following makes sense?
   a. *homework()*
   b. *finish()*
   c. *combine()*
   d. *numbers()*
5. Given the analogy between nouns and objects in R, which of the following makes sense?
   a. *finish(homework)*
   b. *numbers(combine)*
   c. *homework(finish)*
   d. *combine(numbers)*

Getting Started in RStudio

To follow the material in this book, you need to download four separate things: R, RStudio, and two files from SAGE. Once you've loaded the correct version of R onto your computer (https://www.r-project.org), you can download and use RStudio (https://www.rstudio.com). Once R and RStudio have been installed on your computer, two files are available from the SAGE server that are required: *Getting_Started_with_R.Rmd* and *Art_and_Practice.RData*. Both files can be found at **edge.sagepub.com/brownstats1e**. The first file contains material that will help you get started with RStudio and R Markdown. The second file contains the data you'll be using and a few functions I've created.

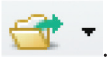
Art and Practice of Data Visualization

KEEP IT TOGETHER

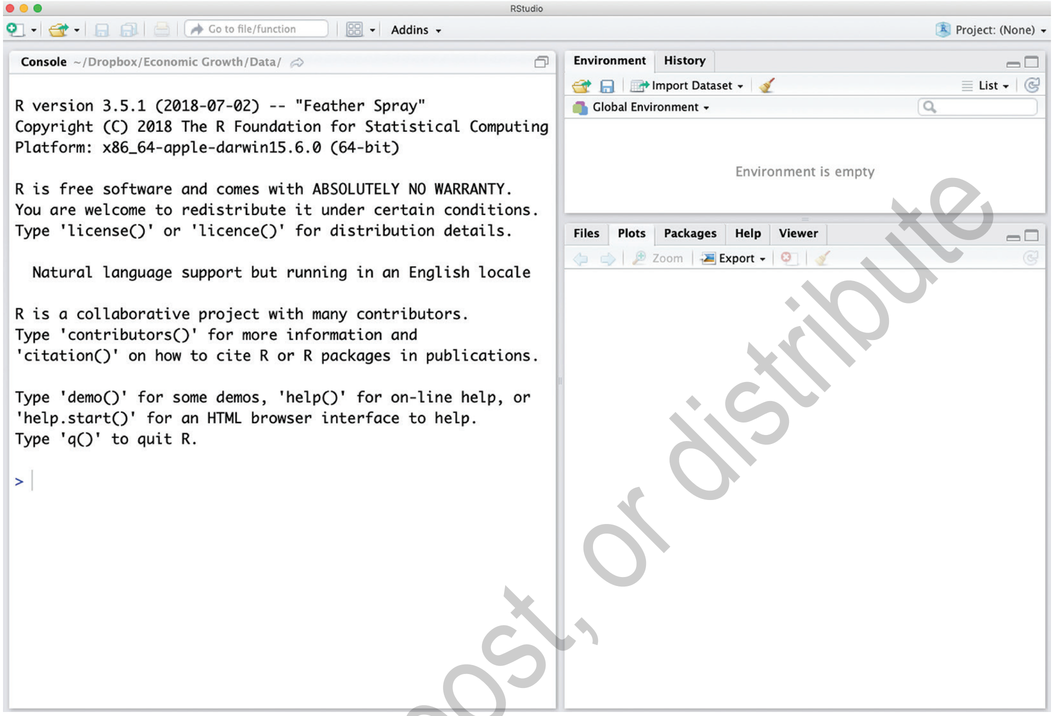
It's best to create a file folder somewhere on your computer to keep your files together in one place. In terms of best practices, if you keep this folder somewhere in the Cloud (using Dropbox, iCloud, or any equivalent service), your work will always be available from any computer and won't be lost if something happens to your computer.

- Step 1. Open the RStudio application. It should look like Figure 1-2.

There are a number of ways to work with R. In this book, we will use R Markdown so we can create professional-looking documents in Word, HTML, or $\LaTeX$.

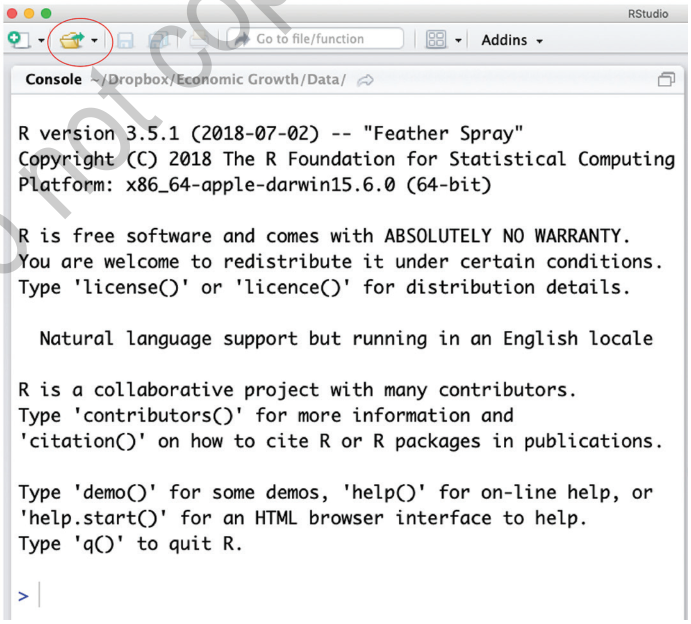
- Step 2. Navigate to the upper-left pane and select the yellow folder button . This step is illustrated in Figure 1-2a.

■ **FIGURE 1-2** View When RStudio Is Opened



```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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.

  Natural language support but running in an English locale

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.

>
```

■ **FIGURE 1-2A** Open the R Markdown File



```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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.

  Natural language support but running in an English locale

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.

>
```

It will open a browsing window where you can navigate to the R Markdown file *Getting_Started_with_R.Rmd*. When you've opened the file, you should see a window pane open up in the upper-left corner of RStudio. Figure 1-3 illustrates what you should see.

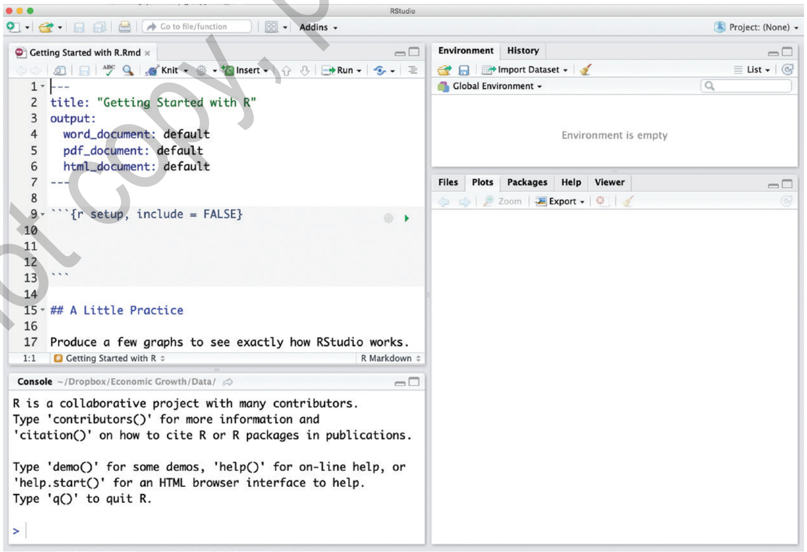
Notice there's a gray area within the R Markdown file. The gray area is called a **code chunk**, which is where we place our instructions (the actual R code). Since this is the first code chunk that appears in the window, I've entitled it *setup* and it should include instructions to load the data and the address of where they reside on your computer.

- Step 3. Navigate to the upper-right pane and click on the yellow folder 📂▾, which will allow you to browse for the data file (Figure 1-4a).[2] Find *Art_and_Practice.RData* on your computer and then select it. If you complete this step correctly, you should see three data sets in the upper-right pane: *nes*, *states*, and *world* (Figure 1-4a).

Note that when you select the file, the load command with the actual address of that file appears in the lower-left pane (the console pane). It will look similar to what is depicted in Figure 1-4b but will vary depending on where you placed the file on your computer: load ("~/Art_and_Practice.RData"). You can copy that command starting with the "l" in *load*.

Once you've copied that command, paste it on any line in the code chunk (Figure 1-4c). In Figure 1-4c, I drew a red ellipse that indicates where you can paste the code. That tells R Markdown where your data are so that you can **knit**[3] this file into a Word, HTML, or LaTeX document.

■ **FIGURE 1-3** R Markdown File Appears



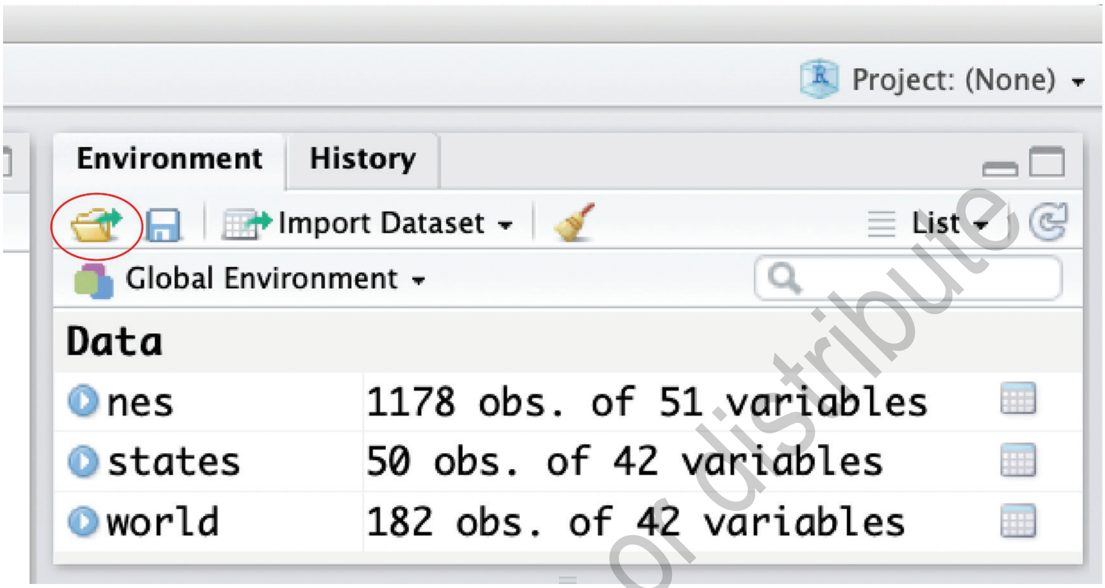
~~~
---
title: "Getting Started with R"
output:
  word_document: default
  pdf_document: default
  html_document: default
---

```{r setup, include = FALSE}



```

## A Little Practice

Produce a few graphs to see exactly how RStudio works.
~~~

```
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.

>
```

[2] The data file *Art_and_Practice.RData* includes data but also some functions. By data, I mean the actual numbers you'll need to make graphs, perform statistical tests, and start exploring important subjects (e.g., gun control laws in the United States, democracy across the world). Defining data and data sets more fully is saved for Chapter 2.

[3] "Knitting" is the term we use when converting the R Markdown file to a document.

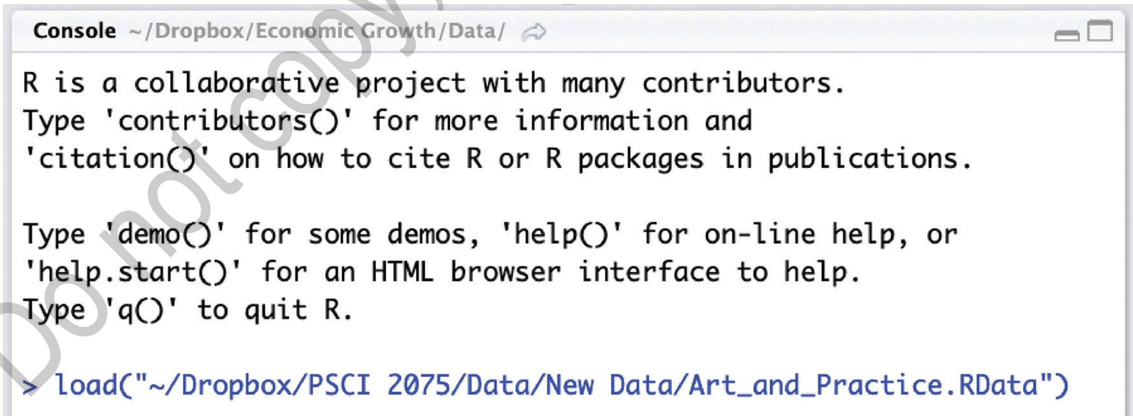
■ **FIGURE 1-4A** Opening the Data File



Data

nes	1178 obs. of 51 variables
states	50 obs. of 42 variables
world	182 obs. of 42 variables

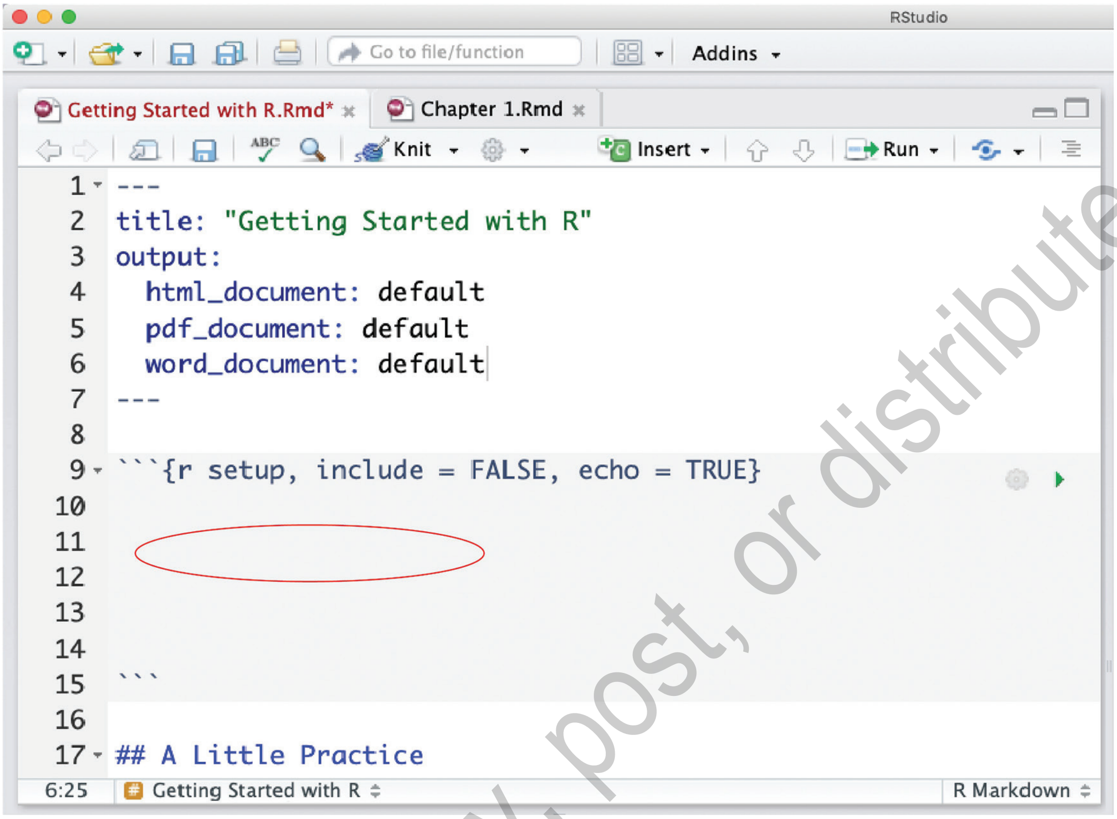
■ **FIGURE 1-4B** Copying the *load* Command



```
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.

> load("~/Dropbox/PSCI 2075/Data/New Data/Art_and_Practice.RData")
```

- Step 4. Install the packages you'll need for this book. Execute the function I defined as *installD()* by typing it at the prompt in the console window (see Figure 1-5a). Included in the data file are a few functions written for this book. The function *installD()* is one of them and it installs all of the packages you need.

FIGURE 1-4C Paste the Command Into the Code Chunk



~~~
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---

```{r setup, include = FALSE, echo = TRUE}





```

## A Little Practice
~~~

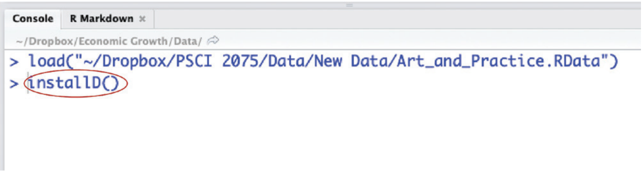
Once you've typed *installD()* and pressed enter to the prompt in the console pane (lower-left), the computer will start installing the packages. This might take 3–5 minutes or a bit longer depending on the speed of your computer. Once all of the packages are installed, the console pane will return you to the prompt.

- Step 5. Type *libraries()* at the prompt in the console pane (lower-left) and press enter to load all of the packages you just installed into RStudio so that you can use them (see Figure 1-5b). This is a specific command written for this book.

The *libraries()* command is similar to the *library()* command, which you can use to load one package at a time. You'll see some activity in the console as each package is loaded. This procedure usually lasts only a manner of seconds. Once you're back to the prompt in the console, you're ready to go!

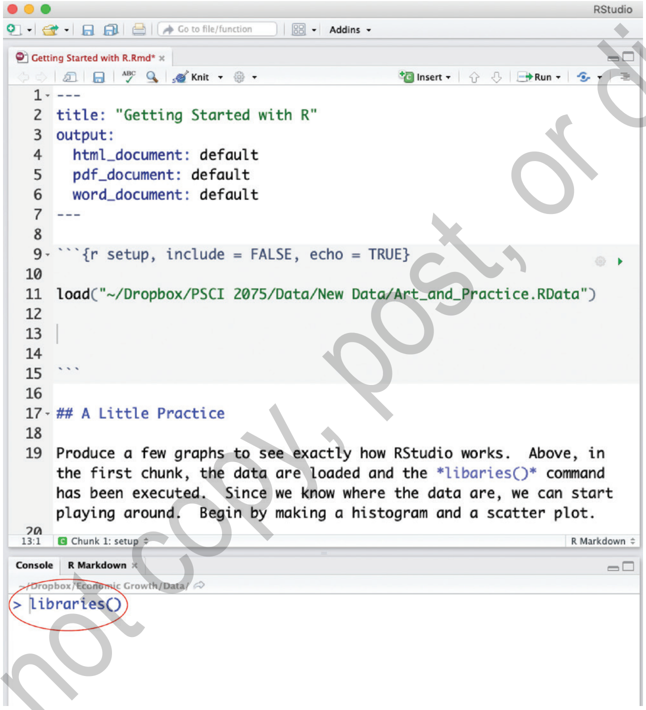
To review, the first time you open RStudio, you need to type *installD()* in the console in order to install the packages for the book. The *installD()* command grabs all of the packages you need from the Internet. As you progress beyond this book, installing additional packages will be necessary. Fortunately, the RStudio interface makes installing packages easy. The process involves nothing more than clicking the *Packages* tab in the lower-right pane, clicking the

■ **FIGURE 1-5A** Install the Packages for the Book



```
> load("~/Dropbox/PSCI 2075/Data/New Data/Art_and_Practice.RData")
> installD()
```

■ **FIGURE 1-5B** Loading Packages



~~~
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---

```{r setup, include = FALSE, echo = TRUE}

load("~/Dropbox/PSCI 2075/Data/New Data/Art_and_Practice.RData")



```

## A Little Practice

Produce a few graphs to see exactly how RStudio works.  Above, in
the first chunk, the data are loaded and the *libaries()* command
has been executed.  Since we know where the data are, we can start
playing around.  Begin by making a histogram and a scatter plot.
~~~

```
> libraries()
```

install button Install, and typing in which package you want to install. RStudio does the rest.

Remember that once RStudio stops installing packages (the cursor should come back to the '>' prompt in the console), you need to type *libraries()* and press enter. The libraries command (a function I've defined for the book) makes the packages you've installed available. While you should only have to run *installD()* once, you'll need to type *libraries()* every time you restart RStudio. To run all of the commands in the setup chunk, click the green arrow ▶ that appears in the upper-right part of the code chunk. If everything has gone according to plan, this will load the data and execute the *libraries()* command.

Before we continue, a note about packages is in order. One of R's strengths is that there are thousands of packages available. At the same time, that is one of its weaknesses. With so many packages written by so many different contributors, some packages have functions with the same name. For example, the *select()* function exists in the 'dplyr' package and in the package called *MASS*. Sometimes one package will "mask" another. When you call a function, R will simply run the function with the package last loaded. In order to specify the function you want, sometimes it's necessary to type the package name followed by two colons ("::"). For example, in Chapter 3 you'll see that the *select()* command is preceded by *dplyr::* in order for the function to run properly. In this book, there are just a few code chunks where this happens. In each case, I will draw your attention to those instances.

KNOWLEDGE CHECK: Download and install both R and RStudio.

6. Which of the following accurately describes the *installD()* command?
   a. It installs and loads R packages.
   b. It is a base R command that loads all packages.
   c. It is an author-defined command that loads all packages for this book.
   d. It is an author-defined command that installs all packages for this book.
7. Which of the following is true about the *libraries()* command?
   a. It installs and loads R packages.
   b. It is a base R command that loads all packages.
   c. It is an author-defined command that loads all packages required for this book.
   d. It is an author-defined command that installs all packages required for this book.
8. Which of the following statements are true?
   a. The *installD()* command must be typed in at the beginning of every session.
   b. The *libraries()* command must be typed in at the beginning of every session.
   c. The file *Art_and_Practice.RData* contains the data you'll need for this class.
   d. You should only need to type the *installD()* command once in this class.

Navigating RStudio With R Markdown

There are many different ways to work with RStudio. While this book shows how to work with R Markdown, you can follow along by simply copying or typing in commands to the console pane. I use R Markdown for several reasons. First, it places your code next to the images produced, providing a direct connection between your code and analysis. Consequently, it encourages good note-taking and careful analysis. Second, it helps us and others replicate our work, an important aspect of the scientific process. With the notes and code placed together, others can follow how you arrived at your conclusions. Perhaps even more importantly, it makes it easy for *you* to remember how you arrived at your conclusions. Finally, R Markdown generates professional-looking documents in a variety of formats. How you present your findings can be just as important as how you generated them in the first place. This section provides a brief introduction on how to use R Markdown in RStudio.

Before we continue, let's get oriented with the basics: the YAML, the code chunk, and how to execute commands in R Markdown. Let's pick up where we left off in the last section. Your screen should resemble Figure 1-6a. The R Markdown file *Getting_Started_with_R.Rmd* is open and we already added the *load()* command that brings in our data file and the *libraries()* command that loads the packages we'll need.

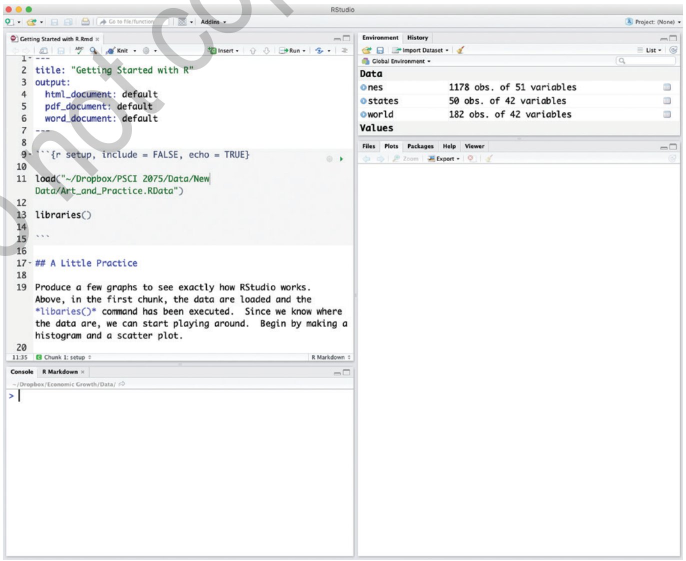
For future reference, to create a new R Markdown file, simply click on the arrow next to the file icon with the green plus sign and select R Markdown. It is located at the very top left of the RStudio window.

At the beginning of the R Markdown file, demarcated by three horizontal dashes, there are instructions that specify the main title and a list of the documents we can create: PDF, Word, and HTML (see Figure 1-6b). This section is referred to as the **YAML**, which stands for *YAML Ain't Markup Language*. It contains commands that set features for the entire document. When you open a new R Markdown file, it will have basic YAML instructions already included. We're going to keep the YAML simple for now, but if you want to take advantage of additional commands, consult the *R Markdown Cheat Sheet* available at the RStudio website (https://resources.rstudio.com/rstudio-developed/rmarkdown-2-0).

In addition to downloading the cheat sheet from the RStudio website, you can access the entire set of cheat sheets by simply going to the RStudio help menu at the very top of the screen. In Figure 1-6c, you can see how to access the various cheat sheets without leaving RStudio.

The more important part of the R Markdown document is the gray area called the *code chunk* (see Figure 1-6d). Code chunks include the actual R code that executes commands. They are created by typing three apostrophes and then a bracketed {r}, which marks the

FIGURE 1-6A RStudio With R Markdown File

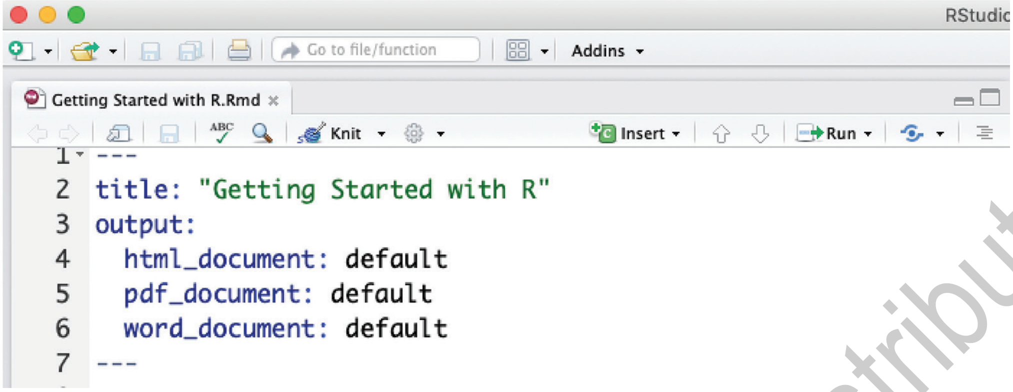


~~~
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---

```{r setup, include = FALSE, echo = TRUE}

load("~/Dropbox/PSCI 2075/Data/New
Data/Art_and_Practice.RData")

libraries()

```

## A Little Practice

Produce a few graphs to see exactly how RStudio works.
Above, in the first chunk, the data are loaded and the
*libaries()* command has been executed.  Since we know where
the data are, we can start playing around.  Begin by making a
histogram and a scatter plot.

~~~

```
Data
nes         1178 obs. of 51 variables
states      50 obs. of 42 variables
world       182 obs. of 42 variables
Values
```

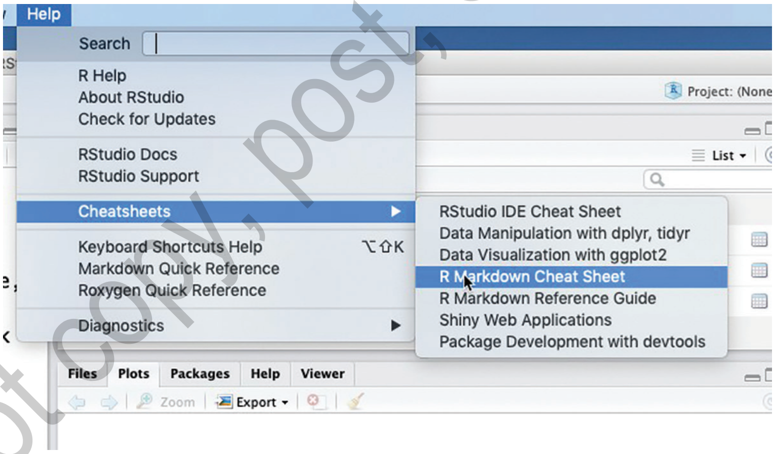
```
> 
```

■ FIGURE 1-6B YAML

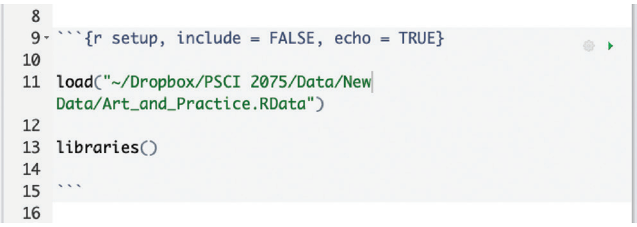


```
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---
```

■ FIGURE 1-6C Accessing the Cheat Sheet for R Markdown



■ FIGURE 1-6D The Code Chunk



~~~
```{r setup, include = FALSE, echo = TRUE}

load("~/Dropbox/PSCI 2075/Data/New Data/Art_and_Practice.RData")

libraries()

```
~~~

beginning. Three more apostrophes indicate where the code chunk ends. You can also create a code chunk by typing *control+command+i* on a Mac or *control+alt+i* on a Windows machine.

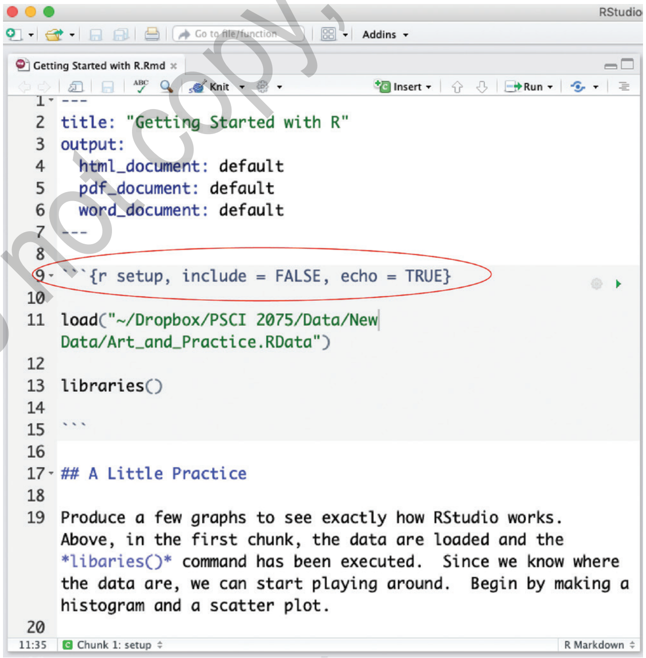
The code located at the top of the code chunk (in between the brackets) tells R Markdown to do several things. First, the *include = FALSE* statement tells R Markdown to not include any extraneous output from that specific code chunk in the document. For example, if the *include* statement is set on *TRUE*, the knitted document will show all of the output produced when the *libraries()* command was executed. We don't want all of that to appear in our document, so we type "FALSE".

Second, the *echo = TRUE* instruction tells R Markdown to include the code you entered in the code chunk, which is a good way to show your work (see Figure 1-6e). In each case, if you change the capitalized TRUE and FALSE, the opposite will happen.

The instructions (the actual R code) you either type or paste are placed inside the gray area (see Figure 1-6f). The command *load()* indicates, for example, what data file you are going to use. For this book, the name of the file is *Art_and_Practice.RData*. Once the command is executed, the data are loaded into RStudio.

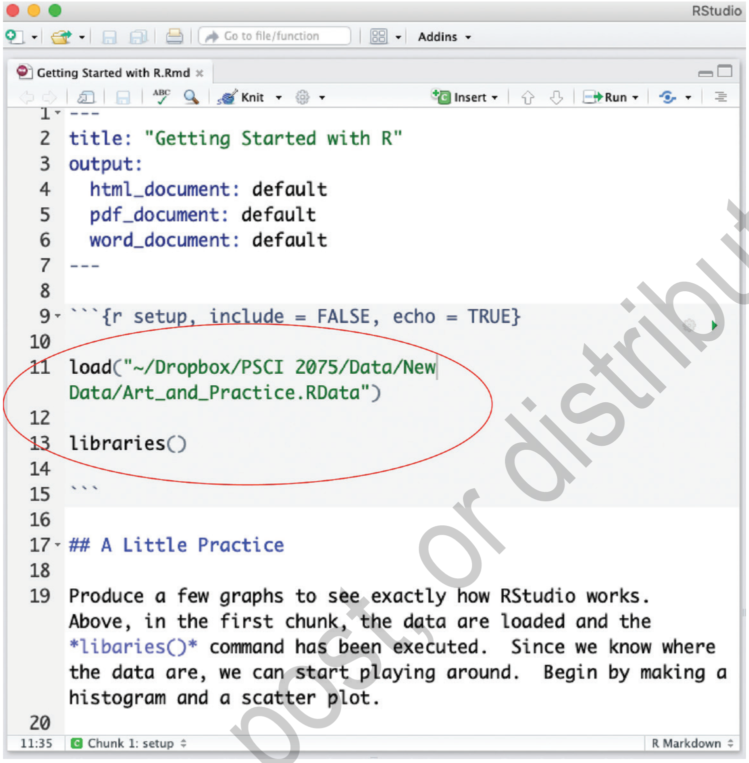
Now that we've specified the code chunk options we want and we have the actual commands entered into the code chunk, we can execute them by clicking on the *Run* button ➡ Run ▾ at the top of the upper-left pane or by clicking on the green arrow ▶ in the code chunk (see Figure 1-6g). Again, I've placed red ellipses marking the two locations where you can execute the code. You can also execute each line of code separately by simply placing the cursor anywhere in the line and typing *command+enter* on a Mac or *control+enter* on a Windows machine.

■ FIGURE 1-6E Chunk Options



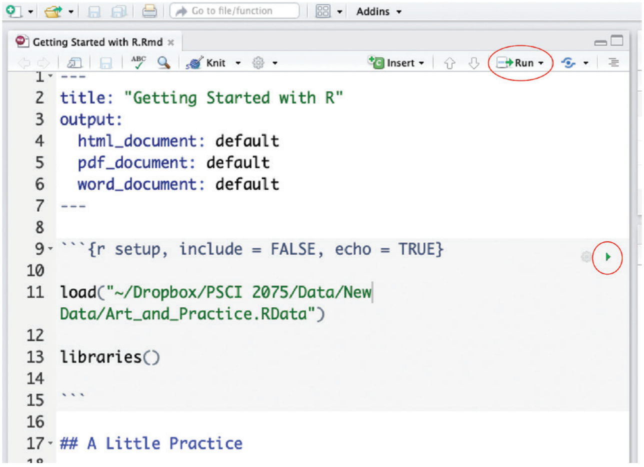
~~~
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---

```{r setup, include = FALSE, echo = TRUE}

load("~/Dropbox/PSCI 2075/Data/New Data/Art_and_Practice.RData")

libraries()

```

## A Little Practice

Produce a few graphs to see exactly how RStudio works.
Above, in the first chunk, the data are loaded and the
*libaries()* command has been executed.  Since we know where
the data are, we can start playing around.  Begin by making a
histogram and a scatter plot.

~~~

FIGURE 1-6F The R Code That Is Executed



~~~
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---

```{r setup, include = FALSE, echo = TRUE}

load("~/Dropbox/PSCI 2075/Data/New
Data/Art_and_Practice.RData")

libraries()

```

## A Little Practice

Produce a few graphs to see exactly how RStudio works.
Above, in the first chunk, the data are loaded and the
*libaries()* command has been executed.  Since we know where
the data are, we can start playing around.  Begin by making a
histogram and a scatter plot.

~~~

FIGURE 1-6G Executing Commands

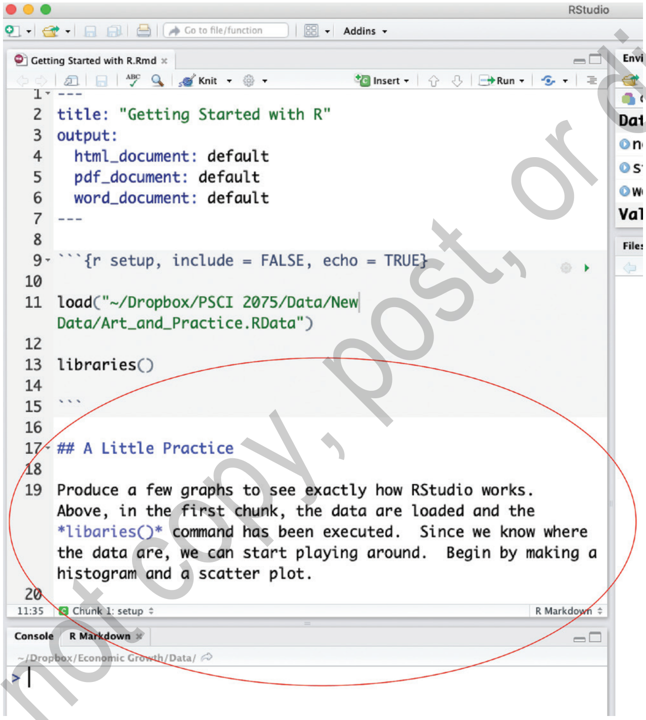


~~~
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---

```{r setup, include = FALSE, echo = TRUE}

load("~/Dropbox/PSCI 2075/Data/New
Data/Art_and_Practice.RData")

libraries()

```

## A Little Practice
~~~

Finally, anything appearing in the white areas is simply the text you'd like to include in your document (your notes). Type anything you'd like into this space (see Figure 1-6h). R Markdown is a stripped-down version of more elaborate markup languages. This is intentional so that you don't have to spend time learning more code. All of the commands you'll need to produce a professional-looking document are available from the drop-down menu that selects the R Markdown cheat sheet (refer back to Figure 1-6c).

There are a number of ways to work with RStudio. Described below are a set of best practices to get started. Once you are more comfortable with executing commands, analyzing output, and knitting documents, develop your own workflow when analyzing data. Before we proceed, let's review the different panes in the RStudio window.

FIGURE 1-6H Where to Type Text



~~~
---
title: "Getting Started with R"
output:
  html_document: default
  pdf_document: default
  word_document: default
---

```{r setup, include = FALSE, echo = TRUE}

load("~/Dropbox/PSCI 2075/Data/New
Data/Art_and_Practice.RData")

libraries()

```

## A Little Practice

Produce a few graphs to see exactly how RStudio works.
Above, in the first chunk, the data are loaded and the
*libaries()* command has been executed.  Since we know where
the data are, we can start playing around.  Begin by making a
histogram and a scatter plot.

~~~

```
~/Dropbox/Economic Growth/Data/
>
```

Art and Practice of Data Visualization

A MULTITUDE OF OPTIONS

There is no single correct way to operate in RStudio. There are multiple options available to achieve the same estimates, pictures, and ways to present the results. More than anything, a healthy curiosity and sense of humor are required to develop your own effective workflow that originates with a data set and ends with important insights.

- *The upper-left pane.* Most of your work will occur in the upper-left pane of RStudio. This is where you can create R Markdown files that record your work, notes, and observations. It's a good idea to develop the habit of executing commands in this manner, since they are all saved and easy to access.
- *The lower-left pane (the console).* In the console pane, you can observe the commands being executed. You can type commands directly into the console if you don't necessarily need to record them. For example, before running more sophisticated analyses, you may want to see a simple picture of the data before you proceed.[4]
- *The lower-right pane.* The pane in the lower-right corner is used to manage packages and to view the figures and tables you create. If you're using an R Markdown file and you want tables and figures to appear in the lower-right pane, find the gear wheel ⚙ ▾ in the upper-left pane and select Chunk Output in Console. Once they appear in the lower-right pane, they can be enlarged, saved as files, and exported as individual items.
- *The upper-right pane (the global environment).* This pane shows what data sets are available and what functions you've defined. It helps you keep track of the data and objects you are working with.

Using R Markdown Files Versus R-Scripts

As noted earlier, this book was written in R Markdown. All of the code you'll need to execute is contained in the sections I've labeled as "code chunks" in the book. They proceed each table and figure. Before each code chunk, I provide descriptions of the code to explain what is going on. You'll simply need to become familiar with where that code is, type it into your own R Markdown file, make any necessary adjustments, and execute. To aid in locating the code you'll need, at the end of each chapter is a guide that lists all of the R functions I use in the chapter and in what labeled code chunk they first appear.

R-scripts are files that contain code you want to save. You can't generate nice-looking documents from them, but they serve as more informal ways to save your code for a specific task so that you can retrace your steps. If you don't plan on writing in-depth commentary or producing a document, operating with R-scripts is highly recommended. Note that they don't include code chunks like R Markdown. The entire R-script document can be thought of as one big code chunk in R Markdown. As long as there's no hashtag in front of the command, it can be executed as if it were in the console. As mentioned earlier, this book is structured to follow along using R Markdown rather than R-scripts. Nevertheless, operating RStudio with R-scripts and following this book is easily done if that's what you prefer.

KNOWLEDGE CHECK: Identify the main features of R Markdown.

9. Indicate which statements describe the YAML section of an R Markdown document.
   a. It provides a place to execute commands in R.
   b. It contains commands that set features for the entire document.

[4] Note that you can generate the previous command in the console pane by typing the up arrow on your keyboard.

   c. It is where you record the most often-used commands.

   d. It always appears at the end of the R Markdown document.

10. Which of the following features of a code chunk is true?

    a. It provides a place to execute commands in R.

    b. Code chunks appear as gray boxes in an R Markdown file.

    c. The only way to execute a code chunk is by clicking the green arrow.

    d. You should indicate which data file to use in the first code chunk.

11. Which of the following describes navigation in RStudio?

    a. You can only execute commands in the upper-left pane.

    b. You can execute commands in the console pane.

    c. Typing *command+enter* (Mac) or *control+enter* (Windows) executes commands.

    d. Typing *control+command+i* (Mac) or *control+alt+i* (Windows) generates a code chunk.

12. What is the difference between an R Markdown file and an R-script?

    a. R Markdown files should be used for producing documents for presentation.

    b. R-scripts should be used for producing documents for presentation.

    c. R Markdown files should only be used for saving often-repeated or complex combinations of code.

    d. R-scripts files should only be used for saving often-repeated or complex combinations of code.

A Little Practice

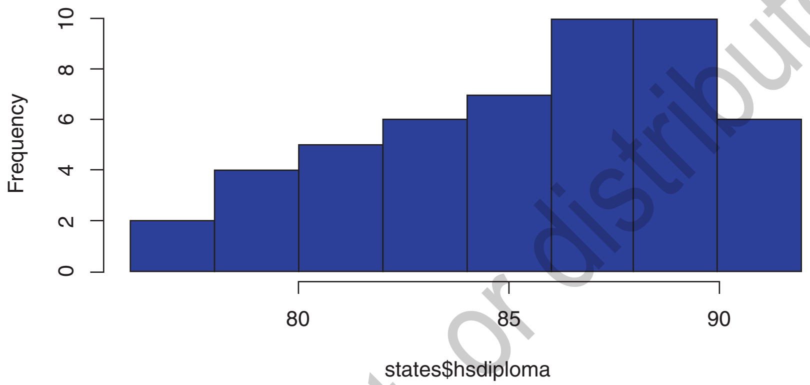
Let's produce a picture of our data to see how everything works. To review, we loaded the data, executed the *libraries()* command, and opened up the R Markdown file *Getting_Started_with_R.Rmd*. Now we can start playing around. Let's start by making a histogram that reveals the data's distribution. Suppose we're interested in educational attainment. We have a **variable**, a list of numbers, that records the percentage of a state's adult population with a high school degree (*hsdiploma*). A collection of variables is called a **data set**. Stated properly, *hsdiploma* is a variable in the *states* data set. I give a more complete description of variables and data sets in Chapter 2. For now, let's see what the variable called *hsdiploma* actually looks like.

Code Chunk 1-2 draws a histogram, a useful picture that indicates the lowest, highest, and most common values of a variable (Figure 1-7). Code Chunk 1-2 tells R to draw a histogram. Remember, the code can either be typed into an R Markdown code chunk or executed in the console pane. Since we're using *Getting_Started_with_R.Rmd*, the code is included in the code chunk. Execute the code by clicking on the green arrow ▶ found at the upper-right corner of the code chunk to produce Figure 1-7.

Code Chunk 1-2

```
hist(states$hsdiploma, main = “Figure 1-7: This Is My First Plot!”,
       font.main = 1, col = “#0000bf”)
```

■ **FIGURE 1-7** This Is My First Plot!



As I explained earlier when introducing functions and objects, all functions in R begin with the name of the function and a set of parentheses where you insert information that the function needs (the object). Note that we use the dollar sign ($) to separate the data set name from the variable name. By specifying the data set and the variable name, R knows which data set you're using and specifically which variable you'd like to see. In Code Chunk 1-2, we use the function *hist()* to draw the histogram. Specify which variable to use (*states$hsdiploma*) within the parentheses. You can also add a title (*main* =), specify the font for the title (*font.main* =), and indicate a color (*col* =).

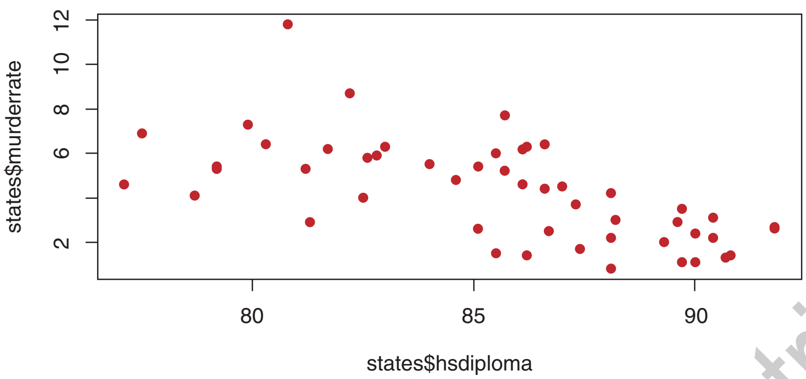
By clicking on the right-pointing green arrow ▶ in the upper-right corner of the code chunk, R studio will execute the command and draw the histogram.

Let's draw another picture, a scatter plot of the two variables *murderrate* and *hsdiploma* from the *states* data set. Scatter plots show how two different variables relate to one another. In the following example, we'll look at educational attainment and murder rates (the number of homicides per 100,000 population) in the U.S. states. Note the appearance of *pch* =, which specifies the shape of the dots.

Code Chunk 1-3

```
plot(states$hsdiploma, states$murderrate,
      main = “Figure 1-8: Education Reduces Violence \n”,
      font.main = 1, col = “#bf0000”, pch = 19)
```

■ **FIGURE 1-8** Education Reduces Violence



You can change the settings so that your output will appear in the bottom-right pane for closer inspection. To do that, go to the gear icon at the top of this window ⚙ ▾ and choose *Chunk Output in Console*. Now that we've loaded the data, loaded the packages, and produced a few plots, let's *knit* the document.

Once you've checked to make sure all of the code chunks in the R Markdown file run properly by clicking the green button in each one, choose the small arrow by the *Knit* icon Knit ▾ at the top of this window and choose either *Word* or *HTML*. Once you've done that, you'll see RStudio going through the process of generating a document (you'll see its progress in the console window). If done correctly, you should have a nice-looking HTML or Word document of this brief analysis.

As an additional example, let's make a picture of some data we've created by hand to demonstrate the power of R as an object-oriented programming language. Earlier, we discussed the difference between executing a function and defining an object. We've executed two functions already to create a histogram and a scatter plot.

For clarity, we'll use the example from our discussion on functions and objects. In *Getting_Started_with_R.Rmd*, you'll see a code chunk that defines an object called *myobject* as a list of 12 numbers (the example we used earlier). Once that object is defined, we can put it inside the parentheses for the function to act on it. R knows whether it's the right kind of object or not. If you try to execute a function on an object that is inappropriate, R will object!

To explain why R might object, let's refer back to our example *eat(broccoli)*. R expects some kind of food when you're using the command *eat()*. If you typed in *eat(glass)*, R would spit it out (give you an error message). It knows *glass* is not the right kind of object given the function *eat*.

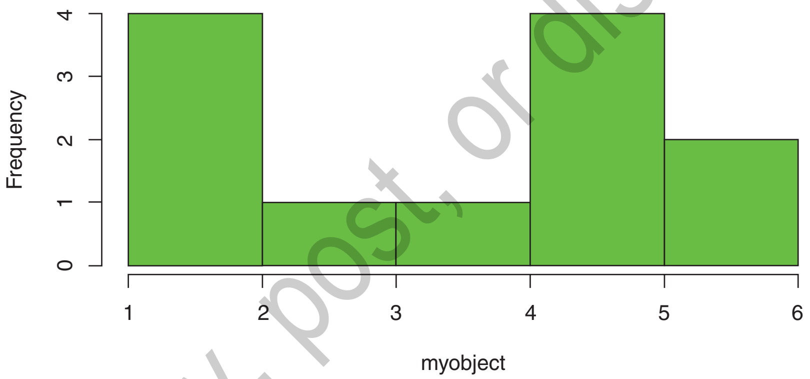
Since the commands are already included in the code chunk, simply click the green arrow ▶ on the upper-right side of the chunk to execute. You should see a histogram that looks like the one in Figure 1-9. This is the beauty of object-oriented languages: once we've defined an object, there are a multitude of functions we can apply to it.

In Code Chunk 1-4, there are two commands. First, an object (*myobject*) is defined. The function *c()* tells R to combine the elements separated by commas into a list (a variable in this case). Having defined an object called *myobject* that contains 12 numbers, we draw a histogram of that variable. We use the same *hist()* function as in Code Chunk 1-2, only this time we color it green.

Code Chunk 1-4

```
myobject <- c(1,1,1,1,3,4,5,5,5,5,6,6)

hist(myobject, main = “Figure 1-9: Object that You Defined”,
    font.main = 1, col = “#00ff00”)
```

■ **FIGURE 1-9** Object That You Defined



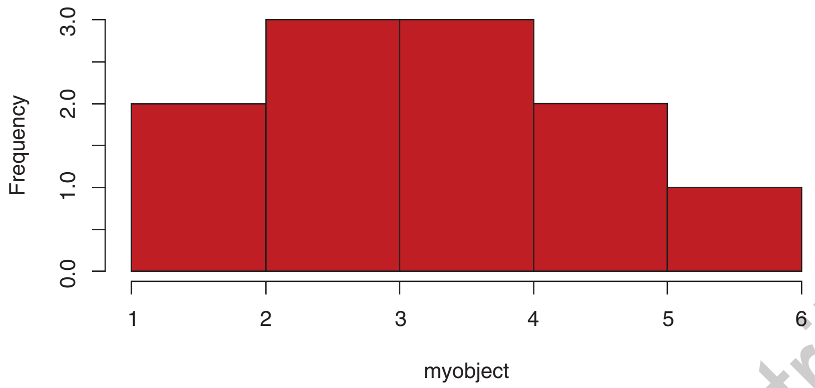
As we discussed earlier, the purpose of this chapter is to get you up and running with RStudio. Just copying the code and running it yourself is an important step. But don't stop there. See how small changes to the code change the output. Gaining the facility necessary to be a good data analyst comes with practice. Let's take the previous example and make some modifications.

Take the code in the last code chunk and alter it. Just to get you started, I copied the previous code chunk, pasted it into Code Chunk 1-5, and played around with the numbers in the variable *myobject* to change the shape of the histogram. I converted numbers to threes and fours to emphasize the middle of the range. I also took the color option and changed it from green (#00ff00) to red (#bf0000). The result is displayed in Figure 1-10.

Code Chunk 1-5

```
myobject <- c(1,2,3,3,3,4,4,4,5,5,6)

hist(myobject, main = “Figure 1-10: Object that You Defined”,
    font.main = 1, col = “#bf0000”)
```

FIGURE 1-10 Object That You Defined



As you'll learn, colors and shapes can easily be overdone, so much so that they distract from understanding what the data actually say. We'll pay more attention to that as we progress through the chapters. For now, to quote famous school teacher, Miss Frizzle:

> *Take chances. Make mistakes. Get messy!*

KNOWLEDGE CHECK: Create a Word or HTML file from R Markdown.

13. Identify which commands will likely produce an error message.
    a. play(football)
    b. play(homework)
    c. eat(pizza)
    d. eat(spoon)
14. What does *knitting* do?
    a. Executes the R commands
    b. Generates an HTML document
    c. Generates a Word document
    d. Generates a PDF
15. Which of the following statements are true?
    a. Used within the *hist()* command, *main* = allows you to specify the color of a histogram.
    b. Used within the *hist()* command, *states$hsdiploma* specifies the variable to be plotted.
    c. Used within the *hist()* command, *col* = specifies the column number to use.
    d. Used within the *plot()* command, *pch* = specifies the shape of each point to be plotted.

SUMMARY

This chapter introduced R, RStudio, and R Markdown. You also learned how to load data, load the packages necessary for executing commands, and generate some simple pictures of the data. While this chapter is no substitute for helpful books devoted to learning the R programming language, you now have enough skill to start exploring.

If you play around with RStudio in these beginning chapters, you'll quickly develop the skills necessary to do some amazing work and to make some interesting discoveries. A few short commands and basic knowledge of RStudio change everything.

Art and Practice of Data Visualization

CURIOSITY AND HUMOR

Very much like writing, the way we go about coding and organizing our work reveals a lot about ourselves. Our level of patience will be on full display. One of the strengths and drawbacks of R is that there are a multitude of ways to do the same thing. While there is a lot of online help and advice, it will differ. Occasionally, since much of what we use in R is written by many different people, there are inconsistencies that can make it difficult to troubleshoot. Add to that, learning a new system of syntax can lead to despair. Learning the basics of R is not for the faint-hearted. The more you develop a sense of calm and a sense of humor throughout the enterprise, the faster you'll learn. Getting stuck and developing an effective way to get unstuck can be very rewarding and can be translated from this realm into many others. It's called critical thinking.

COMMON PROBLEMS

- Most of the problems I've seen at this stage include one of two things: (1) the versions of R, RStudio, and the computer's system software are not compatible; or (2) when the *installD()* command is executed, a clean and complete installation of the packages does not occur. So, pay attention to the version numbers of R and RStudio and make sure they are compatible with your system's software.
- Pay attention to what is happening once the *installD()* command is executed. Be on the lookout for packages that don't load or if the process ends abruptly. If the following *libraries()* command does not execute, go back and try the *installD()* command again. Sometimes, it's helpful to reinstall both R and RStudio and start again. Taking the extra 5–10 minutes to reinstall the applications and to install all of the packages properly will prevent problems down the road.
- Once you're up and running, you may find that something isn't working correctly. There are all sorts of sites to find help, but here are some of the main problems I've run across at this stage of the journey:
  - There was a typo in the command.
  - A copy of R from another class already existed on the computer.

- When installing packages, some files got corrupted.
- Not all of the packages from the *installD()* command got installed.
- Compatible versions of R and RStudio were not downloaded.
- The *libraries()* command was not executed when RStudio was opened.
- The *installD()* command was never executed.
- The versions of R and RStudio were not compatible with the computer's system software.
- The *Art_and_Practice.RData* file was not opened in the correct pane.

REVIEW QUESTIONS

1. What is RStudio?
2. What is R Markdown?
3. What does the console pane in RStudio do?
4. What is the difference between an object and a function in R?
5. How are commands executed in R Markdown?
6. When should you use an R-script?
7. What does the keystroke combination *command+enter* achieve? (Mac)
8. What are the advantages in using R Markdown when navigating RStudio?
9. What is *knitting*?
10. What color is #00ff00?

PRACTICE ON ANALYSIS AND VISUALIZATION

1. What is the difference between installing and loading packages?
   a. Loading packages has to be performed each time RStudio is reopened.
   b. Installing packages has to be performed each time RStudio is reopened.
   c. Installing packages occurs after they've been loaded.
   d. Loading packages occurs after they've been installed.
2. Which statements generally characterize navigating RStudio?
   a. There is a correct and widely accepted way to navigate RStudio.
   b. R-scripts are the best way to save your notes and commands as you are performing analysis.
   c. R Markdown files are the best way to save your notes and commands as you perform your analysis.
   d. Always execute commands in the console pane.
3. Which of the following would be likely functions?
   a. eat
   b. pray
   c. calculate
   d. Sally
4. Which of the following would be likely objects?
   a. eat
   b. pray
   c. prayer
   d. broccoli

5. Which commands make sense in the R context?
   a. eat(sandwich)
   b. broccoli(eat)
   c. eat(broccoli, steamed, everyday)
   d. eat(steamed, broccoli, everyday)

6. What is a code chunk in R Markdown?
   a. Defines features for the entire R Markdown document
   b. Contains executable code in R
   c. R code that is knitted in an R Markdown file
   d. A great place to write comments on your code

7. Which of the following statements are true?
   a. RStudio will run without having R open and running.
   b. R will run without RStudio open and running.
   c. RStudio will run if R has been installed on your computer.
   d. RStudio will run only if R is open first.

8. Which of the following statements are true?
   a. *installD()* installs all of the packages you'll need for the book.
   b. *libraries()* is an author-defined command that installs the packages you'll need for this book.
   c. *libraries()* is an author-defined command that loads the packages you'll need for this book.
   d. you need to execute *installD()* every time you open RStudio.

9. What best describes the activity in the console pane?
   a. Execution of single commands
   b. Execution of commands you don't need to save
   c. Execution of long combinations of code
   d. A good place to save commands for future use

10. What will you need to navigate RStudio successfully?
    a. A background in differential calculus
    b. At least one class in programming
    c. A previous class in statistics
    d. A sense of humor

ANNOTATED R FUNCTIONS

The following functions appear in this chapter. They are listed in order of their first appearance (with the code chunk number in parentheses) and annotated here to give a very brief description of their use. Some are not stand-alone functions and only work in combination with other commands. As a reminder, the code in every chapter will work properly if executed in the order it appears. Proper execution also depends on typing the author-defined *libraries()* command, which loads the required R packages.

plot(): basic R command to generate a plot. In this chapter, it is used to produce a scatter plot. (1-1)

hist(): basic R command to generate a histogram. (1-2)

c(): function used to combine elements separated by commas into a list. (1-4)

ANSWERS

KNOWLEDGE CHECK

1. a
2. b
3. b, c, d
4. b, c
5. a, d
6. b
7. a, b, d
8. b, c, d
9. a, d
10. b, d
11. b, c, d
12. b, d
13. b, d
14. b, c, d
15. b, d

PRACTICE ON ANALYSIS AND VISUALIZATION

1. a, d
2. c
3. a, b, c
4. c, d
5. a, c
6. b, c
7. a, b, c
8. a, c
9. a, b
10. d

online resources

Access digital resources, including datasets, at http://edge.sagepub.com/brownstats1e.