1.7 R IN FOCUS

Entering and Defining Variables

Throughout this book, we present instructions for using R by showing you how you can use functions and code to solve problems instead of doing them by hand. Before you read this section, please take the time to read the section titled "How to Use R With This Book" at the beginning of this book. That section provides an overview of what R is and how to use it.

In this chapter, we discussed how variables are defined, coded, and measured. We can expand on what we learned in the introductory section of "How to Use R With This Book" to reinforce the different ways we can get data into R. We first talked about entering data into Excel, saving it as a .csv file, and reading it into R. We can also import data from other software programs, such as SPSS, by using the foreign package. Finally, we discussed how we can create our own dataframe by typing our data in R. The biggest challenge is making sure you enter the data correctly. Entering even a single value incorrectly can alter your results. For this reason, always double-check the data to make sure the correct values have been entered.

Let us create a dataframe of data again, but this time using a different example and a different approach. Suppose you record the average GPA of students in one of three statistics classes. You record the following GPA scores for each class, given in Table 1.4.

Table 1.4 GPA Scores in Three Statistics Classes

Class 1	Class 2	Class 3
3.3	3.9	2.7
2.9	4.0	2.3
3.5	2.4	2.2
3.6	3.1	3.0
3.1	3.0	2.8

There are two ways you can enter these data: by column or by row. To enter data by *column,* use the following code:

```
GPA.Scores <- data.frame("Class 1" = c(3.3,2.9,3.5,3.6,3.1), "Class 2"
= c(3.9,4.0,2.4,3.1,3.0), "Class 3" = c(2.7,2.3,2.2,3.0,2.8))

GPA.Scores
```

In this example, remember you should be typing your code in your R Script instead of typing directly into the Console. If you did this, and then ran the code from the R Script, your output should look identical (or nearly identical) to the output below. Notice the "+" sign. In R, if you

execute a function (in this case, the data.frame function) and all the code does not fit on one line in the Console window, R will let you know by using this symbol. In other words, you should not type the "+" sign in your code.

```
> GPA.Scores <- data.frame("Class 1" = c(3.3,2.9,3.5,3.6,3.1),
+ "Class 2" = c(3.9,4.0,2.4,3.1,3.0),
+ "Class 3" = c(2.7,2.3,2.2,3.0,2.8))
> GPA.Scores
 Class.1 Class.2 Class.3
1     3.3     3.9     2.7
2     2.9     4.0     2.3
3     3.5     2.4     2.2
4     3.6     3.1     3.0
5     3.1     3.0     2.8
```

Notice the output—the data for each group are now listed down each column. Also, notice how R changed "Class 1" to Class.1. It does this to get rid of any spaces, which may cause confusion. Alternatively, you could have used the following code to label your Class variable:

```
> GPA_Scores <- data.frame(Class1 = c(3.3,2.9,3.5,3.6,3.1),
+                          Class2 = c(3.9,4.0,2.4,3.1,3.0),
+                          Class3 = c(2.7,2.3,2.2,3.0,2.8))
> GPA_Scores
 Class1 Class2 Class3
1    3.3    3.9    2.7
2    2.9    4.0    2.3
3    3.5    2.4    2.2
4    3.6    3.1    3.0
5    3.1    3.0    2.8
```

There is another way to enter these data in R: You can enter data by *row*. This requires coding the data:

```
> GPA.Scores.Row <- data.frame(Class = c(rep(1,5),rep(2,5),rep(3,5)),
GPA = c(3.3,2.9,3.5,3.6,3.1,3.9,4.0,2.4,3.1,3.0,2.7,2.3,2.2,3.0,2.8))
```

```
> GPA.Scores.Row
   Class GPA
1      1 3.3
2      1 2.9
3      1 3.5
4      1 3.6
5      1 3.1
6      2 3.9
7      2 4.0
8      2 2.4
9      2 3.1
10     2 3.0
11     3 2.7
12     3 2.3
13     3 2.2
14     3 3.0
15     3 2.8
```

In this example, we introduced a new function, rep(). This is the repeat function, and we supplied it with two arguments. First, we told it what number to repeat, and second, we told it how many times to repeat that number. So rep(1,5) means repeat the number 1 five times:

```
> rep(1,5)
[1] 1 1 1 1 1
```

This is a shortcut and saves us time so we do not have to type out a lot of numbers. For this example, we are not saving ourselves too much time, but imagine if we had a dataset with 500 observations! That most certainly will save us time.