Chapter 22

Conditional Execution

In addition to the capabilities of R that have already been described, it can also serve as a programming language, as outlined in the next three chapters. This chapter introduces conditional execution; the next chapter introduces looping; the subsequent chapter introduces recursion. The topics considered in this chapter are (a) the if statement, and (b) applications of conditional execution.

An algorithm is a step-by-step procedure for performing a task, such as baking a cake. On a computer, algorithms are generally associated with calculations or data manipulation. The programming capabilities that will be introduced in this chapter and the next two chapters can be used to define an algorithm as a sequence of instructions for problem solving. Many programmers like to draw out the algorithm using a flow chart or d-chart prior to coding up the algorithm in a particular language. This important step of planning is encouraged, particularly for large or complicated algorithms. R supports “structured programming,” which means that R commands can be indented in the code when contained within an if statement or a loop for clarity.

22.1 The if statement

This chapter introduces conditional execution, which can be thought of as an introduction to the if statement. The syntax for the if statement in its simplest form is

```
if {condition} command
```

The condition is a single logical value (a vector of length one whose element is logical) or if condition is numeric, then 0 is FALSE and non-zero is TRUE. The execution of the if statement in this simplest form is that if the condition is TRUE, then the command will be executed. If the condition is FALSE, then the command will not be executed. If the condition or the command is lengthy, it is acceptable to write the if statement involving a single command in the form

```
if {condition} command
```

These forms of the if statement work fine if there is just a single command to be executed. But if there are two or more commands to be executed, the syntax is

```
if {condition} {
 commands
 }
```
It is good programming practice to left-align all of the R commands contained in the curly braces and to indent them two spaces (some programmers prefer four spaces, but whatever you choose, be consistent) as shown above for readability. This practice also emphasizes the logical structure of the program. The curly braces serve as delimiters to enclose the R commands that are to be executed if the condition is TRUE. R uses the + prompt as the R commands are entered from the keyboard, then returns to the > prompt after the closing curly brace has been entered.

It is oftentimes the case that there are also one or more commands that should be executed if the condition is FALSE. If there is just a single command to be executed when the condition is TRUE and another single command to be executed when the condition is FALSE, then the if statement can be written on a single line with the syntax

```
if (condition) command1 else command2
```

If the condition is TRUE, command1 will be executed; if the condition is FALSE, command2 will be executed. When there are two or more commands to execute when the condition is TRUE or FALSE, the curly braces are again used to enclose the commands:

```
if (condition) {
  commands1
} else {
  commands2
}
```

It is good programming practice to left-align all of the R commands within each block of R commands and indent two spaces for readability. This particular style is fine when the R code has been stored in a file and is being executed with the source function. But if you are keying in the code line-by-line, R will assume that the if statement has concluded when it encounters the first closing curly brace. So when keying these commands into an R session, the syntax is

```
if (condition) {
  commands1
} else {
  commands2
}
```

We begin the illustration of the if statement with a simple illustration that prints the character string "x is negative" when x assumes a negative value.

```
> x = -3
> if (x < 0) print("x is negative")
[1] "x is negative"
```

The condition x < 0 evaluates to TRUE because x = -3, so the call to the print function gets executed.

As a second example, an if statement is used to replace x by its opposite if it is negative. The R commands to accomplish this are

```
> x = -3
> if (x < 0) x = -x
> x
[1] 3
```