Chapter 23

Iteration

This is the second chapter on programming in R. The previous chapter introduced conditional execution with the `if` statement. This chapter introduces *iteration*, which is often known as *looping*. A loop is generally used to execute R commands repeatedly, oftentimes with conditions that vary with each pass through the loop. The four topics introduced in this chapter are (a) the `while` loop, (b) the `for` loop, (c) the `repeat` loop, and (d) debugging.

23.1 The `while` loop

The first of the three loops introduced in this chapter is the `while` loop. The syntax for the `while` loop when there is a single R command to be executed in the loop can be written in a single line:

```
while (condition) command
```

As was the case with the `if` statement, the `condition` is a single logical value (a vector of length one whose element is `logical`) or if the `condition` is numeric, then 0 is `FALSE` and non-zero is `TRUE`. The `while` loop executes in the following fashion. The `condition` is evaluated. If the `condition` is `TRUE`, then the `command` is executed. The `condition` is then re-evaluated, and the `command` is again executed if it is `TRUE`. This process continues until the `condition` becomes `FALSE`, and at that point execution continues with the command that follows the loop. The `while` loop evaluates the condition at the "top of the loop," which means that the `command` is not executed if the `condition` is initially `FALSE`. The `while` loop is appropriately named because the `command` is executed repeatedly *while* the `condition` remains `TRUE`. If the `condition` or the `command` is lengthy, it is acceptable to input the `while` loop involving a single command in the form

```
while (condition)
  command
```

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

```
while (condition) {
  commands
}
```

As it was with the `if` statement, it is good programming practice to left-align all of the R commands contained in the curly braces and to indent them two spaces for readability. The curly braces serve as delimiters to enclose the R commands that are to be executed as long as the `condition` remains `TRUE`. R uses the `+` prompt as the R commands in the `while` loop are entered from the keyboard, then returns to the `>` prompt after the closing curly brace has been entered.

The first programming exercise involving a `while` loop is to compute the first ten numbers in the Fibonacci sequence, which are

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55.$$

The first two numbers in this sequence are defined to be

$$x_1 = 1 \qquad \text{and} \qquad x_2 = 1,$$

and subsequent numbers in the sequence are found using the formula

$$x_i = x_{i-1} + x_{i-2}$$

for $i = 3, 4, \ldots$. In other words, each number in the sequence is simply the sum of the previous two numbers. The Fibonacci sequence appears in Pascal's triangle, models the population growth in an idealized population of rabbits, can be used to form a golden spiral, appears in numerous applications in number theory and computer science, and appears in biological applications involving trees, flowers, pine cones, artichokes, and pineapples. The first R command below establishes `Fib` as a vector of 10 numeric elements that are each initialized to zero. The second and third commands set the first two elements of `Fib` to `1`. The fourth command initializes an index named `i` to `3`. The fifth command begins a `while` loop. The condition `i <= 10` evaluates to `TRUE` on the first pass through the `while` loop, so the two R commands delimited by the curly braces are executed. The first command in the `while` loop calculates the *i*th element of the `Fib` vector. The second command in the `while` loop increments `i`. After the closing curly brace, the prompt returns to `>` and the `Fib` vector is displayed by typing its name. The `while` loop is executed eight times as `i` ranges from `3` to `10`. When `i` is incremented to `11`, the condition `i <= 10` evaluates to `FALSE`, and control is passed to the R command following the closing curly brace.

```
> Fib = numeric(10)
> Fib[1] = 1
> Fib[2] = 1
> i = 3
> while (i <= 10) {
+   Fib[i] = Fib[i - 1] + Fib[i - 2]
+   i = i + 1
+ }
> Fib
 [1]  1  1  2  3  5  8 13 21 34 55
```

As the size of a program grows, it is beneficial to write the R code in an external file, then execute the code either using a cut and paste operation, using the `source` function, or using an integrated development environment such as RStudio.

The second example of using the `while` loop in an algorithm concerns the calculation of the greatest common divisor (gcd) of two integers. The gcd of two positive integers *a* and *b* is the largest integer that divides both evenly. An algorithm for calculating the gcd of two positive integers