Stat 2994: Statistical Computing

Introduction to R: 5

Looping in R

There are three kinds of looping constructs in R: The `for` loop, the `while` loop, and the `repeat` loop. Three reasons to use a loop in your computations are:

- repeating the same computation on every element of a data structure, like a matrix,
- forming sums, such as in the case of our computational formula for the sample variance, and
- implementing iterative methods.

(a) *for* loops

This type of a loop construct is suitable when the sequence of values of a variable through which a loop is repeated is known in advance. The general form of the statement is

for(*name* in *seq*) *expr*

where *seq* is a sequence of values usually in a vector, *name* is a name of an object usually placeholder for a counter, and *expr* is an R expression. Each value of *seq* is assigned to the *name* in turn and the *expr* is evaluated.

```
> x <- 10:20
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20
> for(i in 1:11) x[i] <- x[i]^2
> x
 [1] 100 121 144 169 196 225 256 289 324 361 400
```

In the example above, the variable `i` is used for extracting successive elements of a vector using subscripting. As demonstrated below, the use of a loop is unnecessary in this example as the exponentiation operation is vectorized.

```
> x <- 10:20
> x^2
 [1] 100 121 144 169 196 225 256 289 324 361 400
```

Many times a loop is used with a conditional statement in the interior. You can also use functions in the *seq* argument if you may not know how long the argument will be.

```
> y <- c(0,-1,2,-3,4,-5,6,-7,8,-9,10)
> for(i in 1:length(y)) {
+   if (y[i]<0) y[i] <- -y[i] else y[i] <- y[i]
+ }
> y
 [1]  0  1  2  3  4  5  6  7  8  9 10
```

(b) *while* loops

This kind of looping structure is suitable when the number of times the computations are repeated are not known in advance, and the termination of the loop is dependent on some criteria. The general form of the while loop is:

while(*cond*) *expr*

The R expression *expr*, is repeatedly executed until the logical expression *cond* evaluates to a FALSE value. Then, the loop it exited and the next statement (if there is one) is executed. The value of the *cond* logical expression must necessarily depend on computations carried out in *expr*, if not the *cond* will never change to FALSE and the loop runs indefinitely (or until you manually stop it). Typically while loops use a **counter**, an object that starts at one value and increases by increments until the *cond* is FALSE.

In the example below, the while loop is taking the difference of two vectors and placing the values into a third vector. It is important to know that the third vector must be created beforehand, the counter value must also be initiated beforehand.

```
> a <- rep(50,100)
> b <- 1:100
> counter <- 0
> diff <- NULL
> while (counter < 100) {
+   counter <- counter + 1
+   diff[counter] <- a[counter]-b[counter]
+ }
```

In practice, a while loop is preferred when the *cond* expression is used for other purposes than just counting. It can be used, for example, to determine if a specific value has been met or an error term has been decreases to a tolerable level. For example the following loop accumulates a sum to compute exp(5) using the power series $1 + x + x^2/2! + \ldots$

```
> i <- 0
> term <- 1
> sum <- 1
> x <- 5
> while(term > 0.0001){
+   i <- i+1
+   term <- x^i / factorial(i)
+   sum <- sum + term
+ }
> sum
[1] 148.4131
> exp(5)
[1] 148.4132
```

The loop is terminated when the value of the next term to be added to the series is less than or equal to 0.0001. After that, the term is not adding any significant value so we can stop the summation.

(c) *repeat* loops

The repeat loop is similar to the while loop except that the condition for termination is tested inside the loop. The loop is only exited when a condition inside it is satisfied. The general form of the repeat loop is:

repeat *expr*

and the break statement used to tell it when to stop repeating is:

if (*cond*) break

any number of these statements may appear at different places in the loop with different *cond* logical expressions. For example, the while loop in the previous example may be rewritten as follows:

```
> i <- 0
> sum <- 1
> x <- 5
> repeat{
+   i <- i+1
+   term <- x^i / factorial(i)
+   if (term <= 0.0001) break
+   sum <- sum + term
+ }
> sum
[1] 148.4131
```

Thus, there is not too much of a difference between a repeat and a while loop. The choice depends on how the user wants to organize the computations within the loop, and when the condition for exiting the loop is checked.