Control Structures

Biostatistics 140.776

September 27, 2016

Control Structures

Control structures in R allow you to control the flow of execution of a series of R expressions. Commonly used control structures are

- `if` and `else`: testing a condition and acting on it
- `for`: execute a loop a fixed number of times
- `break`: break the execution of a loop
- `next`: skip an interation of a loop

Most control structures are not used in interactive sessions, but rather when writing functions or longer expresisons.

if

Test a condition and act on it depending on whether it's true or false.

```
if(<condition>) {
        ## do something
}
## Continue with rest of code
```

if-else

The previous code does nothing if the condition is false. If you have an action you want to execute when the condition is false, then you need an `else` clause.

```
if(<condition>) {
        ## do something
}
else {
        ## do something else
}
```

if-else

You can have a series of tests by following the initial `if` with any number of `else ifs`.

```
if(<condition1>) {
        ## do something
} else if(<condition2>)  {
        ## do something different
} else {
        ## do something different
}
```

Alternate if-else'

Here is an example of a valid if/else structure.

```
## Generate a uniform random number
x <- runif(1, 0, 10)
if(x > 3) {
        y <- 10
} else {
        y <- 0
}
```

The value of `y` is set depending on whether `x > 3` or not.

if-else

You could have a series of if clauses that always get executed if their respective conditions are true.

```
if(<condition1>) {

}

if(<condition2>) {

}
```

for Loops

For loops take an **interator variable** and assign it successive values from a **sequence** or **vector**.

- Most commonly used for iterating over the elements of an object (list, vector, etc.)

```
numbers <- rnorm(5)
for(i in 1:5) {
        print(numbers[i])
}
```

This loop takes the `i` variable and in each iteration of the loop gives it values 1, 2, 3, 4, 5, executes the code within the curly braces, and then the loop exits.

for Loops

There are many ways to specify a for loop.

```r
x <- c("a", "b", "c", "d")

for(i in 1:4) {
        ## Print out each element of 'x'
        print(x[i])
}
```

```
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

for Loops

The seq_along() function is commonly used in conjunction with for loops in order to generate an integer sequence based on the length of an object (in this case, the object x).

```
x <- c("a", "b", "c", "d")

## Generate a sequence based on length of 'x'
for(i in seq_along(x)) {
        print(x[i])
}
```

```
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

for Loops

It is not necessary to use an index-type variable.

```
x <- c("a", "b", "c", "d")

for(letter in x) {
        print(letter)
}
```

```
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

for Loops

For one line loops, the curly braces are not strictly necessary.

```r
x <- c("a", "b", "c", "d")
for(i in 1:4)
        print(x[i])
```

```
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

for Loops

Or simply

```
x <- c("a", "b", "c", "d")
for(i in 1:4) print(x[i])
```

```
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

Curly braces are sometimes useful even for one-line loops, because that way if you decide to expand the loop to multiple lines, you won't be burned because you forgot to add curly braces (and you *will* be burned by this).

Nested `for` loops

`for` loops can be nested inside of each other.

```r
x <- matrix(1:6, 2, 3)

for(i in seq_len(nrow(x))) {
        for(j in seq_len(ncol(x))) {
                print(x[i, j])
        }
}
```

- Nested loops are commonly needed for multidimensional or hierarchical data structures (e.g. matrices, lists).
- Nesting beyond 2 to 3 levels often makes it difficult to read or understand the code.
- If you find yourself in need of a large number of nested loops, you may want to break up the loops by using functions

next, break

next is used to skip an iteration of a loop.

```
for(i in 1:100) {
        if(i <= 20) {
                ## Skip the first 20 iterations
                next
        }
        ## Do something here
}
```

next, break

break is used to exit a loop immediately, regardless of what iteration the loop may be on.

```
for(i in 1:100) {
      print(i)

      if(i > 20) {
              ## Stop loop after 20 iterations
              break
      }
}
```

Summary

- Control structures like `if-else` and `for` allow you to control the flow of an R program
- Control structures mentiond here are primarily useful for writing programs; for command-line interactive work, the "apply" functions are typically more useful.