R intermediate

Dan McGlinn

January 15, 2016

> Premature optimization is the root of all evil – Donald Knuth

The humble for loop is often considered distasteful by seasoned programmers because it is inefficient; however, the for loop is one of the most useful and generalizable programming structures in R. If you can learn how to construct and understand for loops then you can code almost any iterative task. Once your loop works you can always work to optimize your code and increase its efficiency.

Before attempting these exercises you should review the lesson R intermediate in which loops were covered.

Examine the following for loop, and then complete the exercises

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
sp_ids = unique(iris$Species)

output = matrix(0, nrow=length(sp_ids), ncol=ncol(iris)-1)
rownames(output) = sp_ids
colnames(output) = names(iris[ , -ncol(iris)])

for(i in seq_along(sp_ids)) {
    iris_sp = subset(iris, subset=Species == sp_ids[i], select=-Species)
    for(j in 1:(ncol(iris_sp))) {
        x = 0
        y = 0
        if (nrow(iris_sp) > 0) {
            for(k in 1:nrow(iris_sp)) {
                x = x + iris_sp[k, j]
                y = y + 1
            }
            output[i, j] = x / y
        }
    }
}
output
```

```
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa            5.006       3.428        1.462       0.246
## versicolor        5.936       2.770        4.260       1.326
## virginica         6.588       2.974        5.552       2.026
```

Excercises

Iris loops

1. Describe the values stored in the object `output`. In other words what did the loops create?

"The output the loops created is the average of each column for each species. The loop shows the number of rows and keeps adding a row untill it averages out by the total number of rows to provide and average."

2. Describe using pseudo-code how `output` was calculated, for example,

```
Loop from 1 to length of species identities
   Take a subset of iris data
   Loop from 1 to number of columns of the iris data
      If number of rows of iris data is greater than zero occurs then do a for loop for every row
      Add the values under varriable x from iris data for individual species, add new row each
      time to aquire total value under each species and divide
        output is the average of the data set for each species
```

3. The variables in the loop were named so as to be vague. How can the objects `output`, `x`, and `y` could be renamed such that it is clearer what is occurring in the loop.

"Objects changed to: output (avg_trait), X (i), y (j)"

4. It is possible to accomplish the same task using fewer lines of code? Please suggest one other way to calculate `output` that decreases the number of loops by 1.

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
sp_ids = unique(iris$Species)

avg_trait = matrix(0, nrow=length(sp_ids), ncol=ncol(iris)-1)
rownames(avg_trait) = sp_ids
colnames(avg_trait) = names(iris[ , -ncol(iris)])

for(i in seq_along(sp_ids)) {
    iris_sp = subset(iris, subset=Species == sp_ids[i], select=-Species)
    for(j in 1:(ncol(iris_sp))) {
        avg_trait[i, j] = mean(iris_sp[ , j])
      }
}
avg_trait
```

```
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa            5.006       3.428        1.462       0.246
## versicolor        5.936       2.770        4.260       1.326
## virginica         6.588       2.974        5.552       2.026
```

Sum of a sequence

5. You have a vector `x` with the numbers 1:10. Write a for loop that will produce a vector `y` that contains the sum of `x` up to that index of `x`. So for example the elements of `x` are 1, 2, 3, and so on and the elements of `y` would be 1, 3, 6, and so on.

```
x = c(1:10)
y = NULL
  for (i in x) {
    y[i] = sum(x[1:i])
  }
print(y)
```

```
##  [1]  1  3  6 10 15 21 28 36 45 55
```

6. Modify your for loop so that if the sum is greater than 10 the value of `y` is set to NA

```
x = c( 1:10 )
y = NULL
  for (i in x ) {
    y[i] = sum(x[1:i])
        if (y[i] > 10) {
          print('NA')
        } else {
            print(y[i])
      }
  }
```

```
## [1] 1
## [1] 3
## [1] 6
## [1] 10
## [1] "NA"
## [1] "NA"
## [1] "NA"
## [1] "NA"
## [1] "NA"
## [1] "NA"
```

7. Place your for loop into a function that accepts as its argument any vector of arbitrary length and it will return `y`.

```
x = c(1:10)
y = NULL
Akash <- function(z){
  for (i in z) {
    y[i] = sum(z[1:i])
  }
      return(y)
}
Akash(1:8)
```

```
## [1]  1  3  6 10 15 21 28 36
```