R_intermediate

Julius Alipala

January 14, 2019

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

```r
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
```

```r
#setosa, versicolor, virginica
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)) {
    #subset; i = species
    iris_sp = subset(iris, subset=Species == sp_ids[i], select=-Species)
    #loop through columns(features)
    for(j in 1:(ncol(iris_sp))) {
        x = 0
        y = 0
        #check if subset is not empty
        if (nrow(iris_sp) > 0) {
            for(k in 1:nrow(iris_sp)) {
                #sum all rows in j col
                x = x + iris_sp[k, j]
                #number of rows
                y = y + 1
            }
            #avg sepal length, sepal width, petal length, petal width
            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 loops summed each column value and divided by the number of rows to calculate a mean. The mean of each column for each subspecies is stored in 'output'.

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

```
loop through ...
   loop through ...
      And so on ...
```

```
loop through each subspecies
  loop through each column (feature)
    loop through rows (samples)
      sum column values
      sum number of rows
    output = mean of summed column values
```

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.

x: sum_values
y: num_samples
output: avg_trait

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.

Replace the for loop in the if statement with the mean() function.

```r
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
```

```r
#setosa, versicolor, virginica
sp_ids = unique(iris$Species)

avg_trait = 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)) {
    #subset; i = species
    iris_sp = subset(iris, subset=Species == sp_ids[i], select=-Species)
    for(j in 1:(ncol(iris_sp))) {
        if (nrow(iris_sp) > 0) {
          avg_trait[i,j] = mean(iris_sp[,j])
        }
    }
}
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
```

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.

```r
x = 1:10
y = vector("integer",length(x))
total = 0
for (i in x) {
  total = total + x[i]
  y[i] = total
}

x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
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

```r
x = 1:10
y = vector("integer",length(x))
total = 0
for (i in x) {
  total = total + x[i]
  if (total > 10) {
    y[i] = NA
  }
  else {
    y[i] = total
  }
}

x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
y
```

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

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

```r
vector_sum = function(vector) {
  y = vector("integer",length(vector))
  total = 0
  for (i in vector) {
    total = total + vector[i]
    y[i] = total

  }
  return(y)
}

x1 = 1:10
x2 = 1:15
x3 = 1:5

vector_sum(x1)
```

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

```r
vector_sum(x2)
```

```
##  [1]   1   3   6  10  15  21  28  36  45  55  66  78  91 105 120
```

```r
vector_sum(x3)
```

```
## [1]  1  3  6 10 15
```