Assignment 2

Zach Proux

1/18/2018

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

- The loop created a matrix with the average values of each column for each species.

2) Describe using pseudo-code how `output` was calculated.

```
sp_ids = unique(iris$Species)
# Create an object of unrepeated species names in the iris data frame.
output = matrix(0, nrow=length(sp_ids), ncol=ncol(iris)-1)
# Create an object, specifically a matrix called 'output', with the number of rows defined by
# the number of unique species ids and the number of columns defined by the number of columns
# in iris except for the last one which was "Species."  The 0 indicates that the matrix is
# empty at this point.
rownames(output) = sp_ids
colnames(output) = names(iris[ , -ncol(iris)])
# Populates 'output' with row names from 'sp_ids' and column names from 'iris' except for the
# 5th column - species.

for(i in seq_along(sp_ids)) {
# Generate a regular sequence of species ids and assign i as the variable to refer to that sequence.
# "Do the following for each species one at a time."
    iris_sp = subset(iris, subset=Species == sp_ids[i], select=-Species)
# Create a data frame for each unique species id that excludes the Species column.
    for(j in 1:(ncol(iris_sp))) {
# Create a vector of 1:4 (number of columns in object 'iris_sp') and use j to refer to it.
        x = 0
        y = 0
# Create two objects, x and y, both defined as equal to 0.
        if (nrow(iris_sp) > 0) {
# Perform the function that follows if there are more than 0 rows in 'iris_sp'
            for(k in 1:nrow(iris_sp)) {
# Create a vector of 1:50 (number of rows in 'iris_sp') and use k to refer to it.
                x = x + iris_sp[k, j]
# Define x as the sum of all numbers in 'iris_sp'.
                y = y + 1
# Define y such that it is a running count of how many values were added together.
            }
            output[i, j] = x / y
# Populate 'output' with the averages for each of the 3 species in each of the 4 columns
        }
    }
}
output
```

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?

- output could be renamed 'averages' so the user knows what was calculated. x could be renamed 'sum' to specify it's adding values together and y could be renamed 'count' to indicate it is keeping count of how many values were added.

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)

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)))
            output[i, j] = sum(iris_sp[, j]) / nrow(iris_sp)
        }
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
```

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 = 1:10
y = vector("integer",10)
for(l in 1:length(y)) {
  y[l] = sum(x[1:l])
}
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 = 1:10
y = vector("integer",10)
for(l in 1:length(y)) {
  y[l] = sum(x[1:l])
      if(y[l] > 10) {
        y[l] = NA
      }
}
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.

```
sum_vec = function(x) {
  for(l in 1:length(x)) {
  y[l] = sum(x[1:l])
  }
     return(y)
```

```
}
# Sum_vec functionality test
test = 1:20
sum_vec(test)
```

```
##  [1]   1   3   6  10  15  21  28  36  45  55  66  78  91 105 120 136 153
## [18] 171 190 210
```