Jess Daly intermediate hw

R Intermedaite Homework

```
library(datasets)
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
```

1. The output function here has taken the average sepeal length, sepal width, petal length, and petal width for each of the 3 species of iris (setosa, versicolor, and virginica). The 0 creates a blank matrix which is then filled.
2. There are 3 for loops nested here, “i”, “j”, and “k”.

Line 15: sp_ids is defined as the unique iris species, meaning that regardless of how many times each species appears in the dataset, each will only appear once.

Line 17: output is defined as a matrix where the rows are the unique iris species and the columns are the same as the coluns in the original dataset (sepal.length, sepal.width, petal.length, and petal.width).

Lines 18+19: rownames(output) and colnames(output) define the row and column names for the matrix output as the iris species and the column names from the iris dataset, respectively.

Line 21: for() shows we are beginning the for loop for seq_along(sp_ids), the length of the sequence of the unique iris species.

Line 22: iris_sp is defined as a subset of the iris dataset which excludes the column species (the only non-numeric column in the set).

Lines 23: sets up for loop [j] involving the columns from subset iris_sp

Lines 24&25: x and y are set up to equal 0, or null values, so they can be defined later

Line 26: an if statement reading that if the row number within the sunbet iris_sp is greater than 0, then for loop “k” should be entered

Line 27: sets up for loop [k] involving the rows from subset iris_sp

Line 28: x is redefined as a matrix containing the rows ([k] loop) and columns ([j] loop) from the iris_sp subset.

Line 29: y is redefined as 1

Line 31: The output of the function should be a matrix where the rows are the species [i] and the columns are the other columns from the dataset, [k]. The numbers within the matrix are x/y.

3. Output could be renamed something like ‘means matrix’, or something to indicate that mean is what’s being calculated. X is the sum of the numbers in the column for each species and could be called ‘sum.’ Y is the total number of values in the column for each species +1 and could be called ‘count.’

4.

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

```
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa                0           0            0           0
## versicolor            0           0            0           0
## virginica             0           0            0           0
```

5.

```
x<-1:10
y<-length(x)
```

```
y[1]<-1+x[1]-1
for (i in 2:length(x))
{
  y[i]<-x[i]+y[i-1]
}
y
```

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

6.

```
x<-1:10
y<-length(x)
y[1]<-1+x[1]-1
for (i in 2:length(x))
{
  y[i]<-x[i]+y[i-1]
  if(y[i]>10) {
    print('NA')
  }
  else {
    print(y[i])
  }
}
```

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

7.

```
x<-1:10
y<-length(x)
y[1]<-1+x[1]-1
while (i > 10:length(x))
{
  y[i]<-x[i]+y[i-1]
}
y
```

```
## [1] 1
```

Using a while loop instead of a for loop should give you the flexability to use the code even when you are unsure how long the input sequence is.