Intermediate- Caughron

```r
#load data
data("iris")
```

```r
#unique species IDs
sp_ids = unique(iris$Species)

#creating output matrix that stores average values
output = matrix(0, nrow=length(sp_ids), ncol=ncol(iris)-1)
rownames(output) = sp_ids
colnames(output) = names(iris[ , -ncol(iris)])
```

```r
#subset for each species
#j is run through each trait of iris
#if number of rows for a species ID is greater then zero calculate average of those rows for given trait
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
```

Exercises Iris Loops 1. Describe the values stored in the object output. In other words, what did the loops create?

The average of each parameter for the three species.

2. Describe using pseudo code how output was created.

Loop through indices for parameter one for rows associated with each species

Each time add to running sum

Compute ratio of running sum over length of x

Loop through indices for parameter two for rows associate with each species

Each time add to running sum

Compute ratio of running sum over length of x

Loop through indices for parameter three for rows associated with each species Each time add to running sum

Compute ratio of running sum over length of x

Loop through indices for parameter four for rows associated with each species

Each time add to running sum

Compute ratio of running sum over length of x

**Simplified

Loop through indices for each parameter for rows associated with each species

Each time add to running sum

Compute ratio of running sum over length of x

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

Output can be renamed to indicate what the output table is showing. An example is avg_trait or trait_averages. An alternative for x could be TraitSum and y could be TraitLength.

4. Is it possible to accomplish the same task using fewer lines of codes? Please suggest one other way to calculate output that decreases the numer of loops by one.

```
#unique species IDs
sp_ids = unique(iris$Species)

#creating output matrix that stores average values
avg_trait = matrix(0, nrow=length(sp_ids), ncol=ncol(iris)-1)
rownames(avg_trait) = sp_ids
colnames(avg_trait) = names(iris[ , -ncol(iris)])
```

```
#subset for each species
#j is run from 1 through each trait of iris
#if number of rows for a species ID greater is greater then zero calculate average of those rows for given trait
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<-(1:10)
y<-NULL
for (i in x){
 y[i]<-sum(x[1:i])
}
y
```

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

```
1+2+3+4+5+6+7+8+9+10
```

```
## [1] 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<-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 in its arguments any vector of arbitrary length and it will return y.

```r
CumulativeSum<-function(x){
y<-NULL
for (i in length(x)){
 y<-sum(x[1:i])
 }
 return(y)
}
```

Test

```r
vec<-(1:25)
CumulativeSum(vec)
```

```
## [1] 325
```

```r
1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23+24+25
```

```
## [1] 325
```