Tutorial 11 - Really programming in R

Indexing Review

- To index a vector or matrix, you need square brackets. For example...

  ```
  myVector[2]
  ```

  ```
  myMatrix[3,]
  ```

- To index a data frame, you need square brackets and/or a dollar sign. For example...

  ```
  myDataFrame[3,2]
  ```

  or

  ```
  myDataFrame$column2name[3]
  ```

Indexing Review

Given a data frame called `data` that has 4 rows and 3 columns (with column names `A`, `B`, `C`), what code, without using any loops, would:

- print the 2nd column
- print each row, one at a time
- print each element of the 3rd column, one at a time

for loops in R

In Bash, we looped through files in directories, with code like

```
for file in $@
```

In R, we commonly load full files into R and need to combine sets of integers with subsetting to loope through these data structures.

```
for(line in 1:10){
     print(myDF[line,])
}
```

for loop Review

Given a data frame called `data` that has 4 rows and 3 columns (with column names `A`, `B`, `C`), what code, USING LOOPS, would:

- print each row, one at a time
- print each element of the 3rd column, one at a time

Tips for working with loops

- when writing a loop, work on a single case, get it to work, and then generalize for each row
- think of the index variable as placeholder for each integer in the set you'll loop through
- use print statements to figure out which parts are working and which are not

if-else

This is a useful way to let your code make decisions for you

Given the outcome of a logic test, one or more behaviors can occur

```
if(x > 0){
     print("x is positive")
}else if(x < 0){
     print("x is negative")
}else{
     print("x is equal to zero")
}
```

if-else in a loop

You can even use if-else statements in loops. Actually this is where they are most useful!

Challenge:

Use a `for loop` and `if-else` statement to calculate the sum of wages for males and females in `wages.csv`.

Defining a custom function

```
myMean<-function(x){
     n=length(x)
     total=sum(x)
     average=total/n

     return(average)
}
```

you can even put a loop in a function

Let's say we want to generate a count of the number of observations for each species in the iris dataset.

```
speciesCount<-function(x){
     speciesList<-unique(x$Species)
     obsCounts<-numeric(length(speciesList))
     names(obsCounts)<-speciesList
     for(i in 1:length(speciesList)){
          obsCounts[i]<-sum(x$Species==speciesList[i])
     }
     return(obsCounts)
}
```

Challenge:

- Use a for loop to calculate the average sepal length for each species in the iris data set. Don't cheat and use the `mean()` function!

- Use a for loop and if-else statement to find the minimum petal width of Setosa iris in the iris data set. Don't cheat and use the `min()` function!

Advanced Challenge

Define a function that takes the name of a text file that contains a series of integers and reports the index for the element at the beginning of runs of repeated values and the length of the runs. For example, in the vector 0, 1, 2, 2, 3, 4 your script should store and/or return the values 3 (because the repeated 2's begin in element 3 of the vector) and 2 because there are two 2's in a row. Make sure your solution uses a for loop and at least one if-else statement. Test your function on “findRuns.txt” available on Sakai.