Lecture 25 - Custom functions

Why are custom functions useful?

Defining custom functions

What general components are involved in defining a custom function?

Defining a custom function

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

      return(average)
}
```

What "rules" were made when we defined this function?

Challenge:

- write a custom function that takes a series of numbers and returns half of the minimum value in the series

Challenge:

- change the function you just created to take an argument that allows the user to specify what to divide the minimum value in the series by; set a default value of 2 for that argument

functions and scoping

local v. global environment

if-else statements in functions

Remember you can specify any number of arguments in a custom function and you can also specify defaults for those arguments. Custom functions become especially powerful when you put `for loops` and `if-else` statements in them.

One important use of `if-else` statements is to provide the user of a function with warnings or errors when the information they pass to arguments isn't correct.

```
multiplyByTwo<-function(x){
     if(is.numeric(x)){
          return(x*2)
     }else{
          print("x is not numeric")
     }
}
```

Challenge

You have asked students in your biocomputing class to create a function that randomly creates a text file with any number of lines and a single number on each row, but the sum of numbers in the file must be less than 100. You, as the instructor, have to check each file, but want to do it in an automated way.

Write a function that determines whether a file (provided as an argument and assumed to be in the present working directory) contains numbers that sum to more or less than 100 and returns "The file is correct, A+!" or "Sorry but you have failed. :(" depending on whether the file meets the defined criteria.