IST 687
Homework 3
Due Date: 10/26

Step 1: Create a function (named readStates) to read a CSV file into R
Step 2: Clean the dataframe

```
#read in the census data set
readStates <- function() {
urlToRead <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est2
011-01.csv"
testFrame <- read.csv(url(urlToRead)) #read the data from the web
testFrame <- testFrame[-1:-8,] #remove the first 8 rows ('header information')
testFrame <- testFrame [,1:5] #only keep the first 5 columns
testFrame$stateName <- testFrame[,1] #rename the first column as new column
testFrame <- testFrame[,-1] #remove the original column
testFrame <- testFrame[-52:-58,] #remove the last rows (tail info)
testFrame$stateName <- gsub("\\.","", testFrame$stateName) #remove the period from the state name
testFrame$base2011 <- Numberize(testFrame$X) #convert column to number and rename (1)
testFrame$base2010 <- Numberize(testFrame$X.1) #convert column to number and rename (2)
testFrame$Jul2010 <- Numberize(testFrame$X.2) #convert column to number and rename (3)
testFrame$Jul2011 <- Numberize(testFrame$X.3) #convert column to number and rename (4)
testFrame <- testFrame[,-1:-4] #remove the
rownames(testFrame) <- NULL #remove the old rownames, which are now confusing
return(testFrame)}

#Numberize() function - gets rid of junk and converts to numbers
Numberize <- function(inputVector){
inputVector <- gsub(",","", inputVector) #get rid of commas
inputvector <- gsub(" ","", inputVector) #get rid of spaces
return(as.numeric(inputVector))}
```

Step 3: Store and Explore the dataset

```
#Store dataset into dataframe called dfStates
dfStates <- data.frame(readStates())

#Test dataframe by calculating mean for July2011
mean(dfStates$Jul2011)
## [1] 6109645
```

Step 4: Find the state with the Highest Population

```
#What is the state with the highest population?
dfStates_Sorted_Ascending_by_Jul2011 <- dfStates[order(dfStates$Jul2011),]
tail(dfStates_Sorted_Ascending_by_Jul2011,1)
##    stateName base2011 base2010  Jul2010  Jul2011
## 5 California 37253956 37253956 37338198 37691912
```

Step 5: Explore the distribution of the states

Version 1 of the requested function

```
DistributionV1 <- function(myVector, myNumber){
row_less_than_number_T_or_F <- as.numeric(myVector < myNumber) #create new column to tell if row i
n vector is less than argument
count_of_row_less_than_number <- tabulate(row_less_than_number_T_or_F) #tabulate all the values th
at are true
row_counter <- as.numeric(1) #added row counter as a helper column
percent_row_less_than_number <- count_of_row_less_than_number / 51 #calculate the percentage less
than argument
return(percent_row_less_than_number)}

#test out the function that was just created
DistributionV1(dfStates$Jul2011,mean(dfStates$Jul2011))
## [1] 0.6666667
```

Version 2 of the requested function

```
DistributionV2 <- function(myVector, myNumber){
row_less_than_number_T_or_F <- as.numeric(myVector < myNumber) #create new column to tell if row i
n vector is less than argument
count_of_row_less_than_number <- tabulate(row_less_than_number_T_or_F) #tabulate all the values th
at are true
row_counter <- as.numeric(1) #added row counter as a helper column
percent_row_less_than_number <- count_of_row_less_than_number / length(myVector) #calculate the
percentage less than argument
return(percent_row_less_than_number)}

#test out the functin that was just created
DistributionV2(dfStates$Jul2011,mean(dfStates$Jul2011))
## [1] 0.6666667
```

The difference between the two is that I forced the total rows to be 51 in the first one. In the second one I used a function rather to count the length. The second one is better because we don't always know how many rows there will be. For example, if I was using the first one for a different vector, the output might not be correct. For the purposes of answering this question, however, it is sufficient. That usually is not the case in the real world when we need something a little more dynamic and flexible enough to handle different data.