Ryan Timbrook

IST687 Intro to Data Science, Spring 2019

Due Date: 04/24/2019

Homework: 3

NetID: RTIMBROO

SUID: 386792749

#R Code - unexecuted

---------- HW3: Cleaning/munging Dataframes ----------

```
# Clear objects from Memory
rm(list=ls())
# Clear Console:
cat("\014")

# Homework Week 3: Cleaning/munging Dataframes

##Objective:
##In this lab, you need to read in a dataset and work on that dataset (in a dataframe) so that it
##can be useful. Then, we will explore the distribution within the dataset.



#Step 1: Create a function (named readStates) to read a CSV file into R
readStates <- function(){
  ## Read a URL that has the U.S. State Population dataset
  state.pops <- read.csv(url("http://www2.census.gov/programs-surveys/popest/tables/2010-
2011/state/totals/nst-est2011-01.csv"), stringsAsFactors = FALSE)

  ## Call function to Clean dataframe
  clDf <- cleanDf(state.pops)

  return(clDf)
}

#Step 2: Function to Clean the dataframe
cleanDf <- function(df){
  if(!require("dplyr")) {install.packages("dplyr")}

  #Make all empty cells equal to NA
  df[df==""] <- NA

  #Clean NA Columns from Dataframe
  df <- df[ ,!apply(df,2,function(x) all(is.na(x)))]
```

```
  #Clean empty Rows from Dataframe
  df <- df[!apply(df,1,function(x) all(is.na(x))),]

  #Get column names from dataframe row 3
  #colNames <- as.character(as.vector(df[3,-1]))
  #colNames <- c("States",colNames)

  #Use Homework requirement, column names
  colNames <- c("stateName", "base2010", "base2011", "Jul2010", "Jul2011")

  #Rename colnames of the dataframe
  colnames(df) <- colNames

  #Get rows that contain state attributes
  df <- df[grep(pattern = "^\\.",df[,1]), ]

  #Renumber rownames
  rownames(df) <- NULL

  #Remove first character of State name that is a '.'
  df$stateName <- gsub("\\.","",df$stateName)

  #Convert column values from characters to numeric
  df$base2010 <- as.numeric(gsub(",","",df$base2010))
  df$base2011 <- as.numeric(gsub(",","",df$base2011))
  df$Jul2010 <- as.numeric(gsub(",","",df$Jul2010))
  df$Jul2011 <- as.numeric(gsub(",","",df$Jul2011))

  return(df)
}

#Step 3: Store and Explore the dataset
dfStates <- readStates()
str(dfStates)

#Test dataframe by calculating the mean for July2011 data
mean(dfStates$Jul2011)

#Step 4: Find the state with the Highest Population
state.max.pop <- dfStates[which.max(dfStates$Jul2011),1]
state.max.pop

## Sort the data, in increasing order, based on the July2011 data
orderedDfStates <- dfStates[order(dfStates$Jul2011),]
orderedDfStates
```

```
#Step 5: Explore the distribution of the states
## Functions
# input vect takes a vector
# input threshold takes a number
distOfStates <- function(vect, threshold){

  #return the percentage of the elements within the vector that is less than the same
  perc <- length(vect[vect < threshold])/length(vect)
  return(perc)
}

## Test Functions
#test with vector 'dfStates$Jul2011Num', and the mean of 'dfStates$Jul2011Num'

percentLess <- distOfStates(dfStates$Jul2011,mean(dfStates$Jul2011))
percentLess

## Additional Exploratory Methods ###
summary(dfStates$Jul2011)

require(stats)
plot(dfStates$Jul2011)
lines(lowess(dfStates$Jul2011))

with(dfStates,hist(Jul2011, breaks = "FD", col = "green"))
box()
```

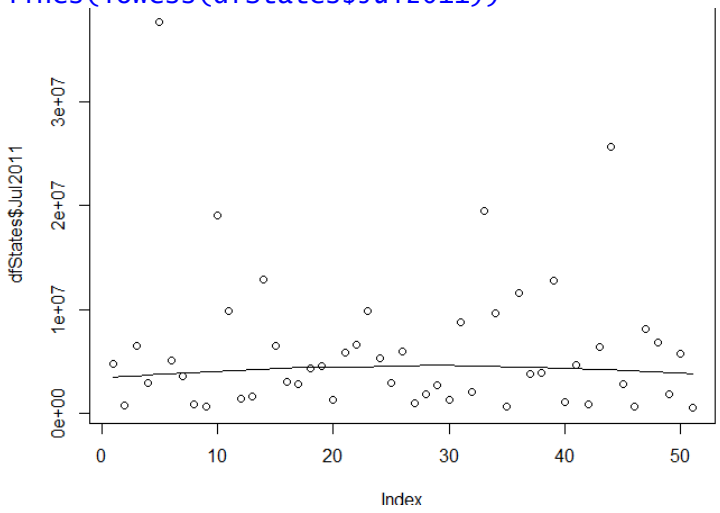
#R Code – executed

```
> ##Objective:
> ##In this lab, you need to read in a dataset and work on that dataset (in a
dataframe) so that it
> ##can be useful. Then, we will explore the distribution within the dataset.
>
>
> #Step 1: Create a function (named readStates) to read a CSV file into R
> readStates <- function(){
+   ## Read a URL that has the U.S. State Population dataset
+   state.pops <- read.csv(url("http://www2.census.gov/programs-surveys/popes
t/tables/2010-2011/state/totals/nst-est2011-01.csv"), stringsAsFactors = FALS
E)
+
+   ## Call function to Clean dataframe
+   clDf <- cleanDf(state.pops)
+
+   return(clDf)
+ }
>
>
> #Step 2: Function to Clean the dataframe
> cleanDf <- function(df){
```

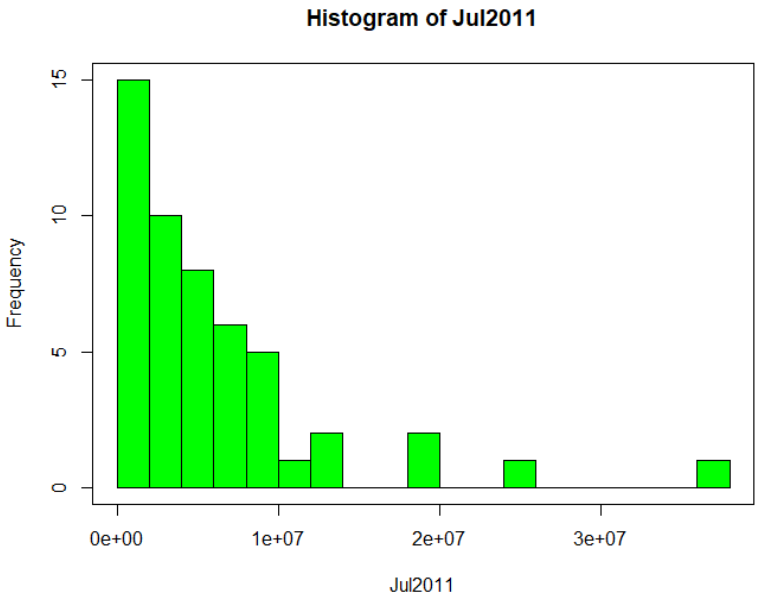
```
+   if(!require("dplyr")) {install.packages("dplyr")}
+
+   #Make all empty cells equal to NA
+   df[df==""] <- NA
+
+   #Clean NA Columns from Dataframe
+   df <- df[ ,!apply(df,2,function(x) all(is.na(x)))]
+
+   #Clean empty Rows from Dataframe
+   df <- df[!apply(df,1,function(x) all(is.na(x))),]
+
+   #Get column names from dataframe row 3
+   #colNames <- as.character(as.vector(df[3,-1]))
+   #colNames <- c("States",colNames)
+
+   #Use Homework requirement, column names
+   colNames <- c("stateName", "base2010", "base2011", "Jul2010", "Jul2011")
+
+   #Rename colnames of the dataframe
+   colnames(df) <- colNames
+
+   #Get rows that contain state attributes
+   df <- df[grep(pattern = "^\\.",df[,1]), ]
+
+   #Renumber rownames
+   rownames(df) <- NULL
+
+   #Remove first character of State name that is a '.'
+   df$stateName <- gsub("\\.","",df$stateName)
+
+   #Convert column values from characters to numeric
+   df$base2010 <- as.numeric(gsub(",","",df$base2010))
+   df$base2011 <- as.numeric(gsub(",","",df$base2011))
+   df$Jul2010 <- as.numeric(gsub(",","",df$Jul2010))
+   df$Jul2011 <- as.numeric(gsub(",","",df$Jul2011))
+
+   return(df)
+ }
> #Step 3: Store and Explore the dataset
> dfStates <- readStates()
> str(dfStates)
'data.frame':  51 obs. of  5 variables:
 $ stateName: chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
 $ base2010 : num  4779736 710231 6392017 2915918 37253956 ...
 $ base2011 : num  4779735 710231 6392013 2915921 37253956 ...
 $ Jul2010  : num  4785401 714146 6413158 2921588 37338198 ...
 $ Jul2011  : num  4802740 722718 6482505 2937979 37691912 ...
> #Test dataframe by calculating the mean for July2011 data
> mean(dfStates$Jul2011)
[1] 6109645
> #Step 4: Find the state with the Highest Population
> state.max.pop <- dfStates[which.max(dfStates$Jul2011),1]
> state.max.pop
[1] "California"
> ## Sort the data, in increasing order, based on the July2011 data
> orderedDfStates <- dfStates[order(dfStates$Jul2011),]
> orderedDfStates
```

```
               stateName base2010 base2011  Jul2010  Jul2011
51               Wyoming   563626   563626   564554   568158
9  District of Columbia   601723   601723   604912   617996
46               Vermont   625741   625741   625909   626431
35          North Dakota   672591   672591   674629   683932
2                 Alaska   710231   710231   714146   722718
42          South Dakota   814180   814180   816598   824082
8               Delaware   897934   897934   899792   907135
27               Montana   989415   989415   990958   998199
40          Rhode Island  1052567  1052567  1052528  1051302
30         New Hampshire  1316470  1316472  1316807  1318194
20                 Maine  1328361  1328361  1327379  1328188
12                Hawaii  1360301  1360301  1363359  1374810
13                 Idaho  1567582  1567582  1571102  1584985
28              Nebraska  1826341  1826341  1830141  1842641
49         West Virginia  1852994  1852996  1854368  1855364
32            New Mexico  2059179  2059180  2065913  2082224
29                Nevada  2700551  2700551  2704283  2723322
45                  Utah  2763885  2763885  2775479  2817222
17                Kansas  2853118  2853118  2859143  2871238
4               Arkansas  2915918  2915921  2921588  2937979
25           Mississippi  2967297  2967297  2970072  2978512
16                  Iowa  3046355  3046350  3050202  3062309
7            Connecticut  3574097  3574097  3575498  3580709
37              Oklahoma  3751351  3751354  3760184  3791508
38                Oregon  3831074  3831074  3838332  3871859
18              Kentucky  4339367  4339362  4347223  4369356
19             Louisiana  4533372  4533372  4545343  4574836
41        South Carolina  4625364  4625364  4637106  4679230
1                 Alabama  4779736  4779735  4785401  4802740
6                Colorado  5029196  5029196  5047692  5116796
24             Minnesota  5303925  5303925  5310658  5344861
50             Wisconsin  5686986  5686986  5691659  5711767
21              Maryland  5773552  5773552  5785681  5828289
26              Missouri  5988927  5988927  5995715  6010688
43             Tennessee  6346105  6346110  6357436  6403353
3                Arizona  6392017  6392013  6413158  6482505
15               Indiana  6483802  6483800  6490622  6516922
22         Massachusetts  6547629  6547629  6555466  6587536
48            Washington  6724540  6724540  6742950  6830038
47              Virginia  8001024  8001030  8023953  8096604
31             New Jersey  8791894  8791894  8799593  8821155
34        North Carolina  9535483  9535475  9560234  9656401
11                Georgia  9687653  9687660  9712157  9815210
23               Michigan  9883640  9883635  9877143  9876187
36                   Ohio 11536504 11536502 11537968 11544951
39           Pennsylvania 12702379 12702379 12717722 12742886
14               Illinois 12830632 12830632 12841980 12869257
10                Florida 18801310 18801311 18838613 19057542
33               New York 19378102 19378104 19395206 19465197
44                  Texas 25145561 25145561 25253466 25674681
5              California 37253956 37253956 37338198 37691912
> #Step 5: Explore the distribution of the states
> ## Functions
> # input vect takes a vector
> # input threshold takes a number
> distOfStates <- function(vect, threshold){
```

```
+
+   #return the percentage of the elements within the vector that is less tha
n the same
+   perc <- length(vect[vect < threshold])/length(vect)
+   return(perc)
+ }
> ## Test Functions
> #test with vector 'dfStates$Jul2011Num', and the mean of 'dfStates$Jul2011N
um'
>
> percentLess <- distOfStates(dfStates$Jul2011,mean(dfStates$Jul2011))
> percentLess
[1] 0.6666667
## Additional Exploratory Methods ###
> summary(dfStates$Jul2011)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  568158  1713813  4369356  6109645  6708787 37691912
>
> require(stats)
> plot(dfStates$Jul2011)
> lines(lowess(dfStates$Jul2011))
```




```
>
> with(dfStates,hist(Jul2011, breaks = "FD", col = "green"))
> box()
```


Histogram of Jul2011
Frequency
0
5
10
15
0e+00
1e+07
2e+07
3e+07
Jul2011