R Notebook

Title: "IST687 – Cleaning/munging Dataframes"
Name: Sathish Kumar Rajendiran
Week: 3
Date: 04/20/2020

Exercise: Data munging on state populations (within the United States).

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

```
  * Note that you are to read a URL, not a file local to your computer.
  * The file is a dataset on state populations (within the United States).
  * The URL is: http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est201
```

```
#   1. Note that you are to read a URL, not a file local to your computer.
#   2. The file is a dataset on state populations (within the United States).

      #file path
      fpath <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est201

      # function readStates
      readStates <- function(fpath)
        {
        dftemp <- read.csv(url(fpath),skip=3)
        #head(dfStates,10)
        #colnames(dfStates)
        return(dftemp)
        }

      dftemp <- readStates(fpath)
      head(dftemp)
```

```
##               X      Census Estimates.Base       X2010       X2011 X.1 X.2 X.3
## 1 United States 308,745,538    308,745,538 309,330,219 311,591,917  NA  NA  NA
## 2     Northeast  55,317,240     55,317,244  55,366,108  55,521,598  NA  NA  NA
## 3       Midwest  66,927,001     66,926,987  66,976,458  67,158,835  NA  NA  NA
## 4         South 114,555,744    114,555,757 114,857,529 116,046,736  NA  NA  NA
## 5          West  71,945,553     71,945,550  72,130,124  72,864,748  NA  NA  NA
## 6      .Alabama   4,779,736      4,779,735   4,785,401   4,802,740  NA  NA  NA
##   X.4 X.5
## 1  NA  NA
## 2  NA  NA
## 3  NA  NA
## 4  NA  NA
## 5  NA  NA
## 6  NA  NA
```

Step 2: Clean the dataframe

```
#   3. Note the issues that need to be fixed (removing columns, removing rows, changing column names).
```

```
    head(dftemp)
```

```
##               X      Census Estimates.Base       X2010       X2011 X.1 X.2 X.3
## 1 United States 308,745,538    308,745,538 309,330,219 311,591,917  NA  NA  NA
## 2     Northeast  55,317,240     55,317,244  55,366,108  55,521,598  NA  NA  NA
## 3       Midwest  66,927,001     66,926,987  66,976,458  67,158,835  NA  NA  NA
## 4         South 114,555,744    114,555,757 114,857,529 116,046,736  NA  NA  NA
## 5          West  71,945,553     71,945,550  72,130,124  72,864,748  NA  NA  NA
## 6      .Alabama   4,779,736      4,779,735   4,785,401   4,802,740  NA  NA  NA
##   X.4 X.5
## 1  NA  NA
## 2  NA  NA
## 3  NA  NA
## 4  NA  NA
## 5  NA  NA
## 6  NA  NA
```

```
    tail(dftemp)
```

```
##
## 58
## 59 Note: The April 1, 2010 Population Estimates base reflects changes to the Census 2010 population
## 60
## 61
## 62
## 63
##       Census Estimates.Base     X2010     X2011 X.1 X.2 X.3 X.4 X.5
## 58 3,725,789      3,725,789 3,721,978 3,706,690  NA  NA  NA  NA  NA
## 59                                               NA  NA  NA  NA  NA
## 60                                               NA  NA  NA  NA  NA
## 61                                               NA  NA  NA  NA  NA
## 62                                               NA  NA  NA  NA  NA
## 63                                               NA  NA  NA  NA  NA
```

```
    summary(dftemp)
```

```
##            X            Census     Estimates.Base       X2010          X2011
##              : 1            : 6            : 6              : 6            : 6
##  .Alabama    : 1   1,052,567: 1   1,052,567: 1     1,052,528: 1   1,051,302: 1
##  .Alaska     : 1   1,316,470: 1   1,316,472: 1     1,316,807: 1   1,318,194: 1
##  .Arizona    : 1   1,328,361: 1   1,328,361: 1     1,327,379: 1   1,328,188: 1
##  .Arkansas   : 1   1,360,301: 1   1,360,301: 1     1,363,359: 1   1,374,810: 1
##  .California: 1   1,567,582: 1   1,567,582: 1     1,571,102: 1   1,584,985: 1
##  (Other)    :57   (Other)  :52   (Other)  :52     (Other)  :52   (Other)  :52
##    X.1            X.2            X.3            X.4            X.5
##  Mode:logical   Mode:logical   Mode:logical   Mode:logical   Mode:logical
##  NA's:63        NA's:63        NA's:63        NA's:63        NA's:63
##
```

```
##
##
##
##
```

```
#   4. Within your function, make sure there are 51 rows (one per state + the district of following nam
#      (stateName,base2010,base2011,Jul2010,Jul2011)

    # remove empty column
    dftemp <- dftemp[,c(1,2,3,4,5)]
    #head(dfStates)

    colnames(dftemp)<- c('stateName', 'base2010', 'base2011','Jul2010', 'Jul2011')
    #summary(dfStates)

    # Keep only rows corresponding to the states & remove empty rows
    dftemp <- dftemp[6:56,]
    #head(dfStates)

#   5. Make sure the last four columns are numbers (i.e. not strings)
    dftemp$base2010 <- as.integer(gsub(',','',dftemp$base2010))
    dftemp$base2011 <- as.integer(gsub(',','',dftemp$base2011))
    dftemp$Jul2010 <- as.integer(gsub(',','',dftemp$Jul2010))
    dftemp$Jul2011 <- as.integer(gsub(',','',dftemp$Jul2011))

    # remove the special character "." infront of stateName
    dftemp$stateName <- gsub('^[.]{1}','',dftemp$stateName)

    # reset Row Index
    row.names(dftemp) <- NULL

    summary(dftemp)
```

```
##   stateName           base2010           base2011           Jul2010
##  Length:51          Min.   :  563626   Min.   :  563626   Min.   :  564554
##  Class :character   1st Qu.: 1696962   1st Qu.: 1696962   1st Qu.: 1700622
##  Mode  :character   Median : 4339367   Median : 4339362   Median : 4347223
##                     Mean   : 6053834   Mean   : 6053834   Mean   : 6065298
##                     3rd Qu.: 6636084   3rd Qu.: 6636084   3rd Qu.: 6649208
##                     Max.   :37253956   Max.   :37253956   Max.   :37338198
##     Jul2011
##  Min.   :  568158
##  1st Qu.: 1713813
##  Median : 4369356
##  Mean   : 6109645
##  3rd Qu.: 6708787
##  Max.   :37691912
```

```
    head(dftemp)
```

```
##    stateName base2010 base2011  Jul2010  Jul2011
## 1    Alabama  4779736  4779735  4785401  4802740
## 2     Alaska   710231   710231   714146   722718
```

```
## 3    Arizona  6392017  6392013  6413158  6482505
## 4   Arkansas  2915918  2915921  2921588  2937979
## 5 California 37253956 37253956 37338198 37691912
## 6   Colorado  5029196  5029196  5047692  5116796
```

Step 3: Store and Explore the dataset Putting all together into one function

```
#     6. Store the dataset into a dataframe, called dfStates

      # file path
      fpath <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est201

      # function readStates
      readStates <- function(fpath)
        {
        dftemp <- read.csv(url(fpath),skip=3)
        dftemp <- dftemp[,c(1,2,3,4,5)]
        colnames(dftemp)<- c('stateName', 'base2010', 'base2011','Jul2010', 'Jul2011')
        dftemp <- dftemp[6:56,]
        dftemp$base2010 <- as.integer(gsub(',','',dftemp$base2010))
        dftemp$base2011 <- as.integer(gsub(',','',dftemp$base2011))
        dftemp$Jul2010 <- as.integer(gsub(',','',dftemp$Jul2010))
        dftemp$Jul2011 <- as.integer(gsub(',','',dftemp$Jul2011))
        dftemp$stateName <- gsub('^[.]{1}','',dftemp$stateName)

        row.names(dftemp) <- NULL
        return(dftemp)
        }

      dfStates <- readStates(fpath)
      head(dfStates)
```

```
##    stateName base2010 base2011  Jul2010  Jul2011
## 1    Alabama  4779736  4779735  4785401  4802740
## 2     Alaska   710231   710231   714146   722718
## 3    Arizona  6392017  6392013  6413158  6482505
## 4   Arkansas  2915918  2915921  2921588  2937979
## 5 California 37253956 37253956 37338198 37691912
## 6   Colorado  5029196  5029196  5047692  5116796
```

```
#     7. Test your dataframe by calculating the mean for the July2011 data, by doing:

      Jul2011Mean <- mean(dfStates$Jul2011)
      sprintf("%.0f is the Jul2011 mean",Jul2011Mean)
```

```
## [1] "6109645 is the Jul2011 mean"
```

Step 4: Find the state with the Highest Population

```
#     8. Based on the July2011 data, what is the population of the state with the highest population? W

      maxPopulation <- max(dfStates$Jul2011)
      sprintf("%.0f is the highest population in July 2011",maxPopulation)
```

```
## [1] "37691912 is the highest population in July 2011"
```

```
      maxPopState_index <- which.max(dfStates$Jul2011)
      maxPopState <- dfStates[maxPopState_index,1]
      print(paste("State with highest population in July 2011 is:",maxPopState),max.levels = 0)
```

```
## [1] "State with highest population in July 2011 is: California"
```

```
#     9. Sort the data, in increasing order, based on the July2011 data

      dfStates <- dfStates[order(dfStates$Jul2011),]
      head(dfStates)
```

```
##               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
```

```
      tail(dfStates)
```

```
##       stateName base2010 base2011  Jul2010  Jul2011
## 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

```
#     10. Write a function that takes two parameters. The first is a vector and the second is a number.
#     11. The function will return the percentage of the elements within the vector that is less than t
#     12. For example, if the vector had 5 elements (1,2,3,4,5), with 2 being the number passed into th

# function readStates
      cumDist <- function(v,n)
        {

        v <- c(1,2,3,4,5)
        #distValue <- mean(v<n)  #--option using mean
        distValue <- prop.table(table(v < n))[2]  #-- option using proportion
        #distValue <- length(which(v<n))/length(v)  # option using length
        return(distValue)
        }

      distValue <- cumDist(v,2)
      distValue_percent <- distValue*100
      sprintf("%0.0f Percentage of population are less than the average population in July 2011",distVal
```

```
## [1] "20 Percentage of population are less than the average population in July 2011"
```

13. Test the function with the vector 'dfStates$Jul2011Num', and the mean of dfStates$Jul2011Num'. There are many ways to write this function (described in #10 above) – so please try to write multiple versions of this function – which do you think is best?

```
#     13. Test the function with the vector 'dfStates$Jul2011Num', and the mean of dfStates$Jul2011Num'
        cumDist <- function(v,n)
          {
            #distValue <- prop.table(table(v < n))[2]  #-- option using proportion
            #distValue <- length(which(v<n))/length(v)  # option using length
            distValue <- mean(v<n)  #--option using mean
            return(distValue)
          }
        meanJul2011 <- mean(dfStates$Jul2011)
        distValue <- cumDist(dfStates$Jul2011,meanJul2011)
        distValue_percent <- distValue*100
        sprintf("%.0f Percentage of population are less than the average population in July 2011",distV
```

```
## [1] "67 Percentage of population are less than the average population in July 2011"
```

```
         #67 Percentage of population are less than the average population in July 2011" This option lo
```