HW3 : Cleaning/munging Dataframes : Thulasi Ram Ruppa Krishnan

```
#Step   1:  Function    (named  readStates) to  read    a   CSV file    into    R
readStates <- function(file,ex_rows,in_rows,header_flg,in_cols,in_col_nms)
{
# This function accepts a file name, number of rows to be imported in a dataset, number of cols,
 column names and the number of rows that needs to be excluded as parameters and create a data f
rame

census <- data.frame(read.csv(file,header = header_flg, nrows = in_rows,skip = ex_rows,na.string
s = "NA", strip.white = TRUE, stringsAsFactors = FALSE,blank.lines.skip = TRUE,col.names = in_co
l_nms)[,in_cols])

return(census)
}
```

```
#Step   2:  Function to Clean   the census dataframe
clean_dataframe <- function(my_census)
{
# This function cleans the input census data frame
ds <- my_census[- c(1:5,57:58),]
ds$stateName <- sub(".","",ds$stateName)
ds$base2010 <- as.numeric(gsub(",","",ds$base2010))
ds$base2011 <- as.numeric(gsub(",","",ds$base2011))
ds$Jul2010 <- as.numeric(gsub(",","",ds$Jul2010))
ds$Jul2011 <- as.numeric(gsub(",","",ds$Jul2011))
row.names(ds)<-1:nrow(ds)
return(ds)
}
```

```
# load the census data from http://www2.census.gov/programs-surveys/popest/tables/2010-2011/stat
e/totals/nst-est2011-01.csv into my_census data frame
my_census<-readStates("http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/tot
als/nst-est2011-01.csv",4,58,FALSE,c(1:5),c("stateName", "base2010", "base2011","Jul2010","Jul20
11","","","","",""))
```

```
# Step  3:  Store   and Explore the dataset
# Clean the data frame by removing  unwanted columns and  rows, change column   names, reset ind
ex, change mode
dfStates <- clean_dataframe(my_census)
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           Connecticut  3574097  3574097  3575498  3580709
## 8              Delaware   897934   897934   899792   907135
## 9  District of Columbia   601723   601723   604912   617996
## 10              Florida 18801310 18801311 18838613 19057542
## 11              Georgia  9687653  9687660  9712157  9815210
## 12               Hawaii  1360301  1360301  1363359  1374810
## 13                Idaho  1567582  1567582  1571102  1584985
## 14             Illinois 12830632 12830632 12841980 12869257
## 15              Indiana  6483802  6483800  6490622  6516922
## 16                 Iowa  3046355  3046350  3050202  3062309
## 17               Kansas  2853118  2853118  2859143  2871238
## 18             Kentucky  4339367  4339362  4347223  4369356
## 19            Louisiana  4533372  4533372  4545343  4574836
## 20                Maine  1328361  1328361  1327379  1328188
## 21             Maryland  5773552  5773552  5785681  5828289
## 22        Massachusetts  6547629  6547629  6555466  6587536
## 23             Michigan  9883640  9883635  9877143  9876187
## 24            Minnesota  5303925  5303925  5310658  5344861
## 25          Mississippi  2967297  2967297  2970072  2978512
## 26             Missouri  5988927  5988927  5995715  6010688
## 27              Montana   989415   989415   990958   998199
## 28             Nebraska  1826341  1826341  1830141  1842641
## 29               Nevada  2700551  2700551  2704283  2723322
## 30        New Hampshire  1316470  1316472  1316807  1318194
## 31           New Jersey  8791894  8791894  8799593  8821155
## 32           New Mexico  2059179  2059180  2065913  2082224
## 33             New York 19378102 19378104 19395206 19465197
## 34       North Carolina  9535483  9535475  9560234  9656401
## 35         North Dakota   672591   672591   674629   683932
## 36                 Ohio 11536504 11536502 11537968 11544951
## 37             Oklahoma  3751351  3751354  3760184  3791508
## 38               Oregon  3831074  3831074  3838332  3871859
## 39         Pennsylvania 12702379 12702379 12717722 12742886
## 40         Rhode Island  1052567  1052567  1052528  1051302
## 41       South Carolina  4625364  4625364  4637106  4679230
## 42         South Dakota   814180   814180   816598   824082
## 43            Tennessee  6346105  6346110  6357436  6403353
## 44                Texas 25145561 25145561 25253466 25674681
## 45                 Utah  2763885  2763885  2775479  2817222
## 46              Vermont   625741   625741   625909   626431
## 47             Virginia  8001024  8001030  8023953  8096604
## 48           Washington  6724540  6724540  6742950  6830038
## 49        West Virginia  1852994  1852996  1854368  1855364
## 50            Wisconsin  5686986  5686986  5691659  5711767
## 51              Wyoming   563626   563626   564554   568158
```

```
#mean   for the July2011    data
mean(dfStates$Jul2011)
```

```
## [1] 6109645
```

```
# Step  4:   Find   the state   with    the Highest Population
# Population of the State with Highest Population
dfStates[which.max(dfStates$Jul2011),5]
```

```
## [1] 37691912
```

```
# Name of the State with Highest Population
dfStates[which.max(dfStates$Jul2011),1]
```

```
## [1] "California"
```

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

```
#Step   5:   Explore    the distribution    of  the states
# Method 1: function    that    takes   two parameters. The first   is  a   vector  and the seco
nd  is  a   number

StatesDist <- function (x,numbr)
{
if (is.vector(x))
  { if (is.numeric(numbr))
    {
      return(length(which(x<numbr))/length(x))
    }
  else return("Incorrect argument, Expected is a Number")
  } else
  {
        if (is.numeric(numbr))
        {
          return("Incorrect argument, Expected is a vector")
        } else return("Incorrect arguments, Expected is a vector and a number")
  }

}
```

```
#Step   5:   Explore    the distribution    of  the states
# Method 2: function    that    takes   two parameters. The first   is  a   vector  and the seco
nd  is  a   number

StatesDist2 <- function (x,numbr)
{
if (is.vector(x))
  { if (is.numeric(numbr))
    {
      return(ecdf(x)(numbr))
    }
  else return("Incorrect argument, Expected is a Number")
  } else
  {
        if (is.numeric(numbr))
        {
          return("Incorrect argument, Expected is a vector")
        } else return("Incorrect arguments, Expected is a vector and a number")
  }

}
```

```
# Test both the function using method 1 and 2   with    the vector  'dfStates$Jul2011Num',  and
 the mean    of  dfStates$Jul2011Num'
# Percentage of elements in dfStates$Jul2011 which are less than its mean

StatesDist(dfStates$Jul2011,mean(dfStates$Jul2011))
```

```
## [1] 0.6666667
```

```
StatesDist2(dfStates$Jul2011,mean(dfStates$Jul2011))
```

```
## [1] 0.6666667
```

```
# From the above two methods of deriving percentage of elements within the vector that is less t
han the mean of the vector, It appears that method 2 is the best as it is uses inbuilt function
 ecdf whereas we are trying to dervie the formula in the method 1.
```