ForteLab8.R

andre

2022-06-11

# Function 1: Create a function called "readStates":  
#Step 1: Create a function (named readStates) to read a CSV file into R: within the Function 1  
#Q1. You need to read a URL, not a local file to your computer.  
#Q2. 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-est2011-01.csv  
# In case you have trouble using this url, use this csv file nst-est2011-01.csv Download nst-est2011-01.csv and read the file locally using read.csv function.  
#Step 2: Clean the dataframe: within Function 1  
#Q3. Note the issues that need to be fixed (removing columns, removing rows, removing dots in front of the state names, changing column names).  
#Q4. Within your function, make sure there are 51 rows (one per state + the district of Columbia). Make sure there are only 5 columns with the columns having the following names (stateName, Census, Estimates, Pop2010, Pop2011).  
#Q5. Make sure the last four columns are numbers (i.e. not strings).  
  
readStates <- function(urlToRead){  
 censusFrame <- read.csv(url(urlToRead))  
 censusFrame <- censusFrame[-1:-8,]  
 censusFrame <- censusFrame[,1:5]  
 censusFrame <- censusFrame[-52:-58,]  
 censusFrame$stateName <- censusFrame[,1]  
 censusFrame <- censusFrame[,-1]  
 censusFrame$stateName <- gsub("\\.","", censusFrame$stateName)  
 censusFrame$Census <- gsub(",", "", censusFrame$X)  
 censusFrame$Estimates <-gsub(",", "", censusFrame$X.1)  
 censusFrame$Pop2010 <- gsub(",", "", censusFrame$X.2)  
 censusFrame$Pop2011 <- gsub(",", "", censusFrame$X.3)  
 censusFrame$Census <- as.numeric(gsub("", "", censusFrame$Census))  
 censusFrame$Estimates <- as.numeric(gsub("", "", censusFrame$Estimates))  
 censusFrame$Pop2010 <- as.numeric(gsub("", "", censusFrame$Pop2010))  
 censusFrame$Pop2011 <- as.numeric(gsub("", "", censusFrame$Pop2011))  
 censusFrame <- censusFrame[,-1:-4]  
 rownames(censusFrame) <- NULL  
 return(censusFrame)  
}  
  
#Step 3: Store and explore the dataset: outside of Function 1  
#Q6. Store the dataset into a dataframe, called dfStates.  
# When you run the following, it should print a clean dataframe. Please include the output of "dfStates" in the compiled file by running dfStates as below.   
  
urlToRead <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est2011-01.csv"  
dfStates <- readStates(urlToRead)  
dfStates

## stateName Census Estimates Pop2010 Pop2011  
## 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

#Q7. Test your dataframe by calculating the mean for the 2011 data, by doing (include your output):  
  
mean(dfStates$Pop2011)

## [1] 6109645

#Step 4: Find the state with the highest population: outside the Function 1  
#Q8. Based on the 2011 data, what is the population of the state with the highest population? What is the name of that state, and what is the value of the population?  
  
highestPop <- max(dfStates$Pop2011)  
index <- which.max(dfStates$Pop2011)  
highestPopName <- dfStates[index, 1]  
highestPop

## [1] 37691912

highestPopName

## [1] "California"

#Q9. Sort the data, in decreasing order, based on the 2011 data.  
  
sortedStates <- dfStates[order(-dfStates$Pop2011), ]  
sortedStates

## stateName Census Estimates Pop2010 Pop2011  
## 5 California 37253956 37253956 37338198 37691912  
## 44 Texas 25145561 25145561 25253466 25674681  
## 33 New York 19378102 19378104 19395206 19465197  
## 10 Florida 18801310 18801311 18838613 19057542  
## 14 Illinois 12830632 12830632 12841980 12869257  
## 39 Pennsylvania 12702379 12702379 12717722 12742886  
## 36 Ohio 11536504 11536502 11537968 11544951  
## 23 Michigan 9883640 9883635 9877143 9876187  
## 11 Georgia 9687653 9687660 9712157 9815210  
## 34 North Carolina 9535483 9535475 9560234 9656401  
## 31 New Jersey 8791894 8791894 8799593 8821155  
## 47 Virginia 8001024 8001030 8023953 8096604  
## 48 Washington 6724540 6724540 6742950 6830038  
## 22 Massachusetts 6547629 6547629 6555466 6587536  
## 15 Indiana 6483802 6483800 6490622 6516922  
## 3 Arizona 6392017 6392013 6413158 6482505  
## 43 Tennessee 6346105 6346110 6357436 6403353  
## 26 Missouri 5988927 5988927 5995715 6010688  
## 21 Maryland 5773552 5773552 5785681 5828289  
## 50 Wisconsin 5686986 5686986 5691659 5711767  
## 24 Minnesota 5303925 5303925 5310658 5344861  
## 6 Colorado 5029196 5029196 5047692 5116796  
## 1 Alabama 4779736 4779735 4785401 4802740  
## 41 South Carolina 4625364 4625364 4637106 4679230  
## 19 Louisiana 4533372 4533372 4545343 4574836  
## 18 Kentucky 4339367 4339362 4347223 4369356  
## 38 Oregon 3831074 3831074 3838332 3871859  
## 37 Oklahoma 3751351 3751354 3760184 3791508  
## 7 Connecticut 3574097 3574097 3575498 3580709  
## 16 Iowa 3046355 3046350 3050202 3062309  
## 25 Mississippi 2967297 2967297 2970072 2978512  
## 4 Arkansas 2915918 2915921 2921588 2937979  
## 17 Kansas 2853118 2853118 2859143 2871238  
## 45 Utah 2763885 2763885 2775479 2817222  
## 29 Nevada 2700551 2700551 2704283 2723322  
## 32 New Mexico 2059179 2059180 2065913 2082224  
## 49 West Virginia 1852994 1852996 1854368 1855364  
## 28 Nebraska 1826341 1826341 1830141 1842641  
## 13 Idaho 1567582 1567582 1571102 1584985  
## 12 Hawaii 1360301 1360301 1363359 1374810  
## 20 Maine 1328361 1328361 1327379 1328188  
## 30 New Hampshire 1316470 1316472 1316807 1318194  
## 40 Rhode Island 1052567 1052567 1052528 1051302  
## 27 Montana 989415 989415 990958 998199  
## 8 Delaware 897934 897934 899792 907135  
## 42 South Dakota 814180 814180 816598 824082  
## 2 Alaska 710231 710231 714146 722718  
## 35 North Dakota 672591 672591 674629 683932  
## 46 Vermont 625741 625741 625909 626431  
## 9 District of Columbia 601723 601723 604912 617996  
## 51 Wyoming 563626 563626 564554 568158

# Function 2: Create a function called "Distribution"  
#Step 5: Explore the distribution of the states: You need to create a new function called "Distribution"  
  
#Q10. You will write a function to calculate percentage of states that have population that is lower than the average. The function (function name: "Distribution") takes two parameters. The first is a vector and the second is a number. For example, Distribution <- function(vector, number). This step is just a setup for the following instruction.   
  
# The function will return the percentage of elements within the vector that is less than the number (i.e. cumulative distribution below the value provided). For example,   
  
# (1) Think about this: You only keep the elements within the vector that are less than the number, and store the number of eligible elements into the variable "count". Populate XXXX to complete this line of code:   
# (2) Then, you will calculate the percentage and return the results. Populate XXXX to complete this line of code:  
# (3) Test the function with the vector “dfStates$Pop2011”, and the mean of “dfStates$Pop2011.” \*\*\* you should get 66.66667 as a result. There are many ways to write this function (described in point 10) – so please try to write multiple versions of this function – which do you think is the best?  
  
Distribution <- function(vector, number){  
 count <- length(vector[vector < number])  
 return(count / length(vector) \* 100)  
}  
  
Distribution(dfStates$Pop2011, mean(dfStates$Pop2011))

## [1] 66.66667