Alonso\_Week 3 Homework Assignment

Martin Alonso

7/31/2018

# IST687 Introduction to Data Science: Week 3 Homework.

**Part 1**

#Cleaning/munging Dataframes

#Data is often not in the format that you want/need. So, by "data munging", you have # to refine the dataset into something more useful.

#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

#This function calls the URL for the US government census, calls the url through the read.csv() function, and returns the loaded csv, without tidying it.

readStates <- function(){

url <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est2011-01.csv"

states\_csv <- read.csv(url)

return(states\_csv)

}

#Let’s use the function and create the census dataframe

census <- readStates()

#Step 2: Clean the dataframe

#First, let’s remove the first 8 rows that contain the table information, and the #population data for the general areas.

#We’ll also remove the last seven rows of the dataset that contain the data for Puerto Rico and additional table information.

census <- census[c(-1:-8, -60:-66), ]

#With the rows being dropped, it’s now time to clean the columns that contain NAs.

census <- census[, -6:-10]

#Now we’ll change the column names.

colnames(census) <- c('stateName','Jul2010','Jul2011','base2010','base2011')

#Let’s make sure that we have 51 rows and 5 columns; and check the first five rows of #the dataset.

dim(census)

head(census)

#A function was created to clean the numerical census data, taking each column, #removing commas and spaces, and converting to number.

numberize <- function(dataset){

cols <- colnames(dataset[, -1])

for (i in cols){

dataset[[i]] <- gsub(',', '', dataset[[i]])

dataset[[i]] <- gsub(' ', '', dataset[[i]])

dataset[[i]] <- as.numeric(as.character(dataset[[i]]))

}

return(dataset)

}

census <- numberize(census)

str(census)

#Step 3: Store and explore the dataset

#Store the census data in the data.frame dfStates and check the mean of #dfStates$Jul2010

dfStates <- census

mean(dfStates$Jul2011)

#Step 4: Find the State with the Highest Population

#What is the highest population and to what state does it belong to?

max(dfStates$Jul2011) #max population is 37,253,956

dfStates$stateName[which.max(dfStates$Jul2011)] #the population of California

#Arrange the states by dfStates$Jul2011 in increasing order

dfStates[order(dfStates$Jul2011),]

#Step 5: Explore the distribution of the states.

#Create a function that takes two parameters (a vector and a number) and returns the percentage of elements in the vector that are equal or below the number.

perc\_below <- function(vector, number){

dat <- data.frame(dfStates[,vector][order(dfStates[, vector])])

colnames(dat) <- vector

tot <- nrow(dat)

under <- sum(dat[, vector] <= number)

perc\_under <- under/tot

return(perc\_under)

}

perc\_below('Jul2011', mean(dfStates$Jul2011))

**Part 2**

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

> #This function calls the URL for the US government census, calls the url through the read.csv() function, and returns the loaded csv, without tidying it.

> readStates <- function(){

+ url <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est2011-01.csv"

+ states\_csv <- read.csv(url)

+ return(states\_csv)

+ }

>

> #Let’s use the function and create the census dataframe

> census <- readStates()

>

> #Step 2: Clean the dataframe

> #First, let’s remove the first 8 rows that contain the table information, and the #population data for the general areas.

> #We’ll also remove the last seven rows of the dataset that contain the data for Puerto Rico and additional table information.

> census <- census[c(-1:-8, -60:-66), ]

> #With the rows being dropped, it’s now time to clean the columns that contain NAs.

> census <- census[, -6:-10]

>

> #Now we’ll change the column names.

> colnames(census) <- c('stateName','Jul2010','Jul2011','base2010','base2011')

>

> #Let’s make sure that we have 51 rows and 5 columns; and check the first five rows of #the dataset.

> dim(census)

[1] 51 5

> head(census)

stateName Jul2010 Jul2011 base2010 base2011

9 .Alabama 4,779,736 4,779,735 4,785,401 4,802,740

10 .Alaska 710,231 710,231 714,146 722,718

11 .Arizona 6,392,017 6,392,013 6,413,158 6,482,505

12 .Arkansas 2,915,918 2,915,921 2,921,588 2,937,979

13 .California 37,253,956 37,253,956 37,338,198 37,691,912

14 .Colorado 5,029,196 5,029,196 5,047,692 5,116,796

>

> #A function was created to clean the numerical census data, taking each column, #removing commas and spaces, and converting to number.

> numberize <- function(dataset){

+ cols <- colnames(dataset[, -1])

+ for (i in cols){

+ dataset[[i]] <- gsub(',', '', dataset[[i]])

+ dataset[[i]] <- gsub(' ', '', dataset[[i]])

+ dataset[[i]] <- as.numeric(as.character(dataset[[i]]))

+ }

+ return(dataset)

+ }

>

> census <- numberize(census)

> str(census)

'data.frame': 51 obs. of 5 variables:

$ stateName: Factor w/ 65 levels "",".Alabama",..: 2 3 4 5 6 7 8 9 10 11 ...

$ Jul2010 : num 4779736 710231 6392017 2915918 37253956 ...

$ Jul2011 : num 4779735 710231 6392013 2915921 37253956 ...

$ base2010 : num 4785401 714146 6413158 2921588 37338198 ...

$ base2011 : num 4802740 722718 6482505 2937979 37691912 ...

>

> #Step 3: Store and explore the dataset

> #Store the census data in the data.frame dfStates and check the mean of #dfStates$Jul2010

> dfStates <- census

> mean(dfStates$Jul2011)

[1] 6053834

>

> #Step 4: Find the State with the Highest Population

> #What is the highest population and to what state does it belong to?

> max(dfStates$Jul2011) #max population is 37,253,956

[1] 37253956

> dfStates$stateName[which.max(dfStates$Jul2011)] #the population of California

[1] .California

65 Levels: .Alabama .Alaska .Arizona .Arkansas .California .Colorado .Connecticut ... West

>

> #Arrange the states by dfStates$Jul2011 in increasing order

> dfStates[order(dfStates$Jul2011),]

stateName Jul2010 Jul2011 base2010 base2011

59 .Wyoming 563626 563626 564554 568158

17 .District of Columbia 601723 601723 604912 617996

54 .Vermont 625741 625741 625909 626431

43 .North Dakota 672591 672591 674629 683932

10 .Alaska 710231 710231 714146 722718

50 .South Dakota 814180 814180 816598 824082

16 .Delaware 897934 897934 899792 907135

35 .Montana 989415 989415 990958 998199

48 .Rhode Island 1052567 1052567 1052528 1051302

38 .New Hampshire 1316470 1316472 1316807 1318194

28 .Maine 1328361 1328361 1327379 1328188

20 .Hawaii 1360301 1360301 1363359 1374810

21 .Idaho 1567582 1567582 1571102 1584985

36 .Nebraska 1826341 1826341 1830141 1842641

57 .West Virginia 1852994 1852996 1854368 1855364

40 .New Mexico 2059179 2059180 2065913 2082224

37 .Nevada 2700551 2700551 2704283 2723322

53 .Utah 2763885 2763885 2775479 2817222

25 .Kansas 2853118 2853118 2859143 2871238

12 .Arkansas 2915918 2915921 2921588 2937979

33 .Mississippi 2967297 2967297 2970072 2978512

24 .Iowa 3046355 3046350 3050202 3062309

15 .Connecticut 3574097 3574097 3575498 3580709

45 .Oklahoma 3751351 3751354 3760184 3791508

46 .Oregon 3831074 3831074 3838332 3871859

26 .Kentucky 4339367 4339362 4347223 4369356

27 .Louisiana 4533372 4533372 4545343 4574836

49 .South Carolina 4625364 4625364 4637106 4679230

9 .Alabama 4779736 4779735 4785401 4802740

14 .Colorado 5029196 5029196 5047692 5116796

32 .Minnesota 5303925 5303925 5310658 5344861

58 .Wisconsin 5686986 5686986 5691659 5711767

29 .Maryland 5773552 5773552 5785681 5828289

34 .Missouri 5988927 5988927 5995715 6010688

51 .Tennessee 6346105 6346110 6357436 6403353

11 .Arizona 6392017 6392013 6413158 6482505

23 .Indiana 6483802 6483800 6490622 6516922

30 .Massachusetts 6547629 6547629 6555466 6587536

56 .Washington 6724540 6724540 6742950 6830038

55 .Virginia 8001024 8001030 8023953 8096604

39 .New Jersey 8791894 8791894 8799593 8821155

42 .North Carolina 9535483 9535475 9560234 9656401

19 .Georgia 9687653 9687660 9712157 9815210

31 .Michigan 9883640 9883635 9877143 9876187

44 .Ohio 11536504 11536502 11537968 11544951

47 .Pennsylvania 12702379 12702379 12717722 12742886

22 .Illinois 12830632 12830632 12841980 12869257

18 .Florida 18801310 18801311 18838613 19057542

41 .New York 19378102 19378104 19395206 19465197

52 .Texas 25145561 25145561 25253466 25674681

13 .California 37253956 37253956 37338198 37691912

>

> #Step 5: Explore the distribution of the states.

> #Create a function that takes two parameters (a vector and a number) and returns the percentage of elements in the vector that are equal or below the number.

> perc\_below <- function(vector, number){

+ dat <- data.frame(dfStates[,vector][order(dfStates[, vector])])

+ colnames(dat) <- vector

+ tot <- nrow(dat)

+ under <- sum(dat[, vector] <= number)

+ perc\_under <- under/tot

+ return(perc\_under)

+ }

> perc\_below('Jul2011', mean(dfStates$Jul2011))

[1] 0.6666667