STA234: Homework 2

Derin Gezgin

2025-02-18

# Problem #1

Using [Ozone hourly 2024 data](https://aqs.epa.gov/aqsweb/airdata/download_files.html) compare the EDA findings with those done in class for the 2020 data.

Our Exploratory Data Analysis (EDA) for the USEPA hourly ozone data for 2020 had the following questions:

* Which counties in the U.S. have the highest levels of ambient ozone pollution? (Lecture-4 Page 6)
* Are hourly ozone levels on average higher in New York state than are in California? (Lecture-4 Page 25)
* Are hourly ozone levels on average higher in New York City then are in Los Angeles? (Lecture-4 Page 5)

To answer these questions for the 2024 data, we can follow a similar path. We can start by reading the data from the .csv file

ozone.2024 = read.csv("../../DATA/hourly\_44201\_2024.csv", header = TRUE)

The first research question looks for the counties in the US with the highest levels of ambient ozone pollution. To answer this question, we would first need the list of counties within the states.

group.counties = aggregate(ozone.2024$Sample.Measurement,  
 by = list(ozone.2024$State.Name,  
 ozone.2024$County.Name),  
 FUN = mean)

As we group the data by counties and took the mean ozone level for each county, now we can sort them by the ozone level.

sorted.counties = group.counties[order(group.counties$x,  
 decreasing=TRUE),]

To view the counties with the highest ambient ozone pollution, we can check the head of the sorted.counties list.

head(sorted.counties, 10)

## Group.1 Group.2 x  
## 148 Colorado Clear Creek 0.05206572  
## 10 Wyoming Albany 0.05101239  
## 263 Colorado Gilpin 0.04829935  
## 261 Arizona Gila 0.04784383  
## 607 Utah San Juan 0.04716827  
## 751 Arizona Yavapai 0.04668674  
## 281 Colorado Gunnison 0.04667247  
## 736 Nevada White Pine 0.04654307  
## 336 Colorado Jefferson 0.04649084  
## 156 Arizona Cochise 0.04591013

This would give us the counties with the highest ozone pollution. Compared to the 2020 analysis -which I checked from the lecture slides- we can see similar counties (Colorado-Clear Creek, Wyoming-Albany, Colorado-Gilpin) in top-5.

To determine if the hourly ozone levels on average are higher in New York then are in California, we can group our data by state and apply the summary function.

group.states.summary = aggregate(ozone.2024$Sample.Measurement,  
 by = list(ozone.2024$State.Name),  
 FUN=summary)

Following this, we can extract and combine New York and California from this to be able to compare them

NY.summary = group.states.summary[group.states.summary$Group.1 == "New York",]  
CA.summary = group.states.summary[group.states.summary$Group.1 == "California",]  
NY.CA.summary = rbind(NY.summary, CA.summary)  
NY.CA.summary

## Group.1 x.Min. x.1st Qu. x.Median x.Mean x.3rd Qu.  
## 34 New York 0.00000000 0.02300000 0.03200000 0.03103199 0.03900000  
## 5 California -0.00400000 0.02200000 0.03300000 0.03299081 0.04300000  
## x.Max.  
## 34 0.11500000  
## 5 0.14700000

When we look at the summary output for New York and California, we can see that California has slightly higher mean hourly ozone levels compared to New York.

To determine if the hourly ozone levels on average are higher in New York City then are in LA using the 5-number summary, we can group the data-frame by counties using the summary function.

group.counties.summary = aggregate(ozone.2024$Sample.Measurement,  
 by = list(ozone.2024$State.Name,  
 ozone.2024$County.Name),  
 FUN = summary)

Following this, we can simply extract and combine the summaries for New York City and Los Angeles.

NYC.summary = group.counties.summary[group.counties.summary$Group.2 == "New York", ]  
LA.summary = group.counties.summary[group.counties.summary$Group.2 == "Los Angeles", ]  
NYC.LA.summary = rbind(NYC.summary, LA.summary)  
NYC.LA.summary

## Group.1 Group.2 x.Min. x.1st Qu. x.Median x.Mean  
## 501 New York New York 0.00000000 0.01900000 0.02800000 0.02886334  
## 413 California Los Angeles -0.00400000 0.02000000 0.03300000 0.03326734  
## x.3rd Qu. x.Max.  
## 501 0.03700000 0.10800000  
## 413 0.04400000 0.14300000

From this table, we can see that Los Angeles has a slightly higher average hourly ozone level compared to the New York City. 20/20

# Problem #2

Collecting data is often a messy process resulting in multiple errors in the data. Consider the following small vector representing the weights of 10 adults in pounds.

my.weights <- c(150, 138, 289, 239, 12, 103, 310, 200, 218, 178)

As far as we know, it’s not possible for an adult to weigh 12 pounds, so that is most likely an error. Change this value to NA, and then find the standard deviation of the weights after removing the NA value.

To replace the value 12 with NA we can use the *exactly equal to* operator.

my.weights[my.weights == 12] = NA  
my.weights

## [1] 150 138 289 239 NA 103 310 200 218 178

Before calculating the standard deviation, we should remove the NA value from the vector

my.weights = my.weights[!is.na(my.weights)]  
my.weights

## [1] 150 138 289 239 103 310 200 218 178

Finally, we can use the sd function to calculate the standard deviation

sd(my.weights)

## [1] 68.95067 5/5

# Problem #3

Consider the following variables: age and income:

age <- c("middle age", "senior", "middle age", "senior", "senior", "senior", "senior", "middle age")  
income <- c("lower", "lower", "upper", "middle", "upper", "lower", "lower", "middle")

### What is the class of each variable?

To find out what class is age and income, we can use the class function.

class(age)

## [1] "character"

class(income)

## [1] "character"

We can see that both age and income are in the character class.

### Change the age variable to a factor with levels for age as: youth, young adult, middle age, and senior.

To change the age variable in a factor, we can simply use the factor function

age.factor = factor(age,  
 levels = c("youth", "young adult", "middle age", "senior"))  
age.factor

## [1] middle age senior middle age senior senior senior senior   
## [8] middle age  
## Levels: youth young adult middle age senior

### Change the income variable to a factor with levels as lower, middle, and upper.

Same as the previous one, to change the income variable to a factor, we can use the factor function

income.factor = factor(income,  
 levels = c("lower", "middle", "upper"))  
income.factor

## [1] lower lower upper middle upper lower lower middle  
## Levels: lower middle upper 10/10

# Problem #4

Suppose you keep track of the mileage each time you fill up. At your last 8 fill-ups the mileage was: 65311, 65624, 65908, 66219, 66499, 66821, 67145, 67447.

Enter this data into R. Use the function diff on the data (use ?diff). Use the documentation for diff function to learn about it and then explain briefly what does this function give you?

To enter this mileage data into R, we can basically save these values in a vector

mileage = c(65311, 65624, 65908, 66219, 66499, 66821, 67145, 67447)

When I checked the hep documentation and examples, I saw that the diff function basically takes the difference between each consecutive element in a vector. For example, in this case the elements of diff(mileage) would be

| Index | Difference (Next - Current) |
| --- | --- |
| 1 | 65624 - 65311 = 313 |
| 2 | 65908 - 65624 = 284 |
| 3 | 66219 - 65908 = 311 |
| 4 | 66499 - 66219 = 280 |
| 5 | 66821 - 66499 = 322 |
| 6 | 67145 - 66821 = 324 |
| 7 | 67447 - 67145 = 302 |

From the R output, we can see these values as well.

diff(mileage)

## [1] 313 284 311 280 322 324 302 5/5

# Problem #5

Create the following data frame:

my\_data <- data.frame(student\_id = c(100234, 132454, 453123),  
 test\_1\_grade = c(82, 93, 87),  
 hw\_1\_grade = c(92, 89, 98),  
 session = c("7 AM", "7 PM", "7 AM"))  
my\_data

## student\_id test\_1\_grade hw\_1\_grade session  
## 1 100234 82 92 7 AM  
## 2 132454 93 89 7 PM  
## 3 453123 87 98 7 AM

Obtain the column names of our data frame.

To get the column names, we can basically use the colnames function.

colnames(my\_data)

## [1] "student\_id" "test\_1\_grade" "hw\_1\_grade" "session"

This function returns the column names which are student\_id, test\_1\_grade, hw\_1\_grade, and session.

Get the number of rows or columns in a data frame, try nrow(), ncol(), or dim() functions.

nrow(my\_data)

## [1] 3

ncol(my\_data)

## [1] 4

dim(my\_data)

## [1] 3 4

nrow() returns the row count of our data-frame

ncol() returns the column count of our data-frame

dim() returns the dimensions (row count x column count) of our data-frame.

To subset rows and columns of a data frame we can use the following syntax:

my\_data\_frame[row condition, column condition]. The row/column conditions may be either numeric indexes, logical expressions, or vectors

Explain the subset you get from the following code:

my\_data\_frame <- data.frame(make = c("Toyota","Honda","Ford", "Toyota", "Ford", "Honda"),  
 mpg = c(34, 33, 22, 32, 29, 27),  
 cylinders = c(4, 4, 8, 6, 6, 8))  
my\_data\_frame

## make mpg cylinders  
## 1 Toyota 34 4  
## 2 Honda 33 4  
## 3 Ford 22 8  
## 4 Toyota 32 6  
## 5 Ford 29 6  
## 6 Honda 27 8

This code snippet creates a new data-frame with 3 columns (make, mpg, cylinders) and 6 rows. This data-frame gives information about the brand, distance the car can go on one gallon of fuel, and the number of cylinders the car has for 6 different cars.

Explain the subset you get from the following code:

my\_data\_frame[1:3,1:2]

## make mpg  
## 1 Toyota 34  
## 2 Honda 33  
## 3 Ford 22

This code snippet selects rows 1 to 3 and first two columns.

my\_data\_frame[c(1, 2, 3), c(1, 2)]

## make mpg  
## 1 Toyota 34  
## 2 Honda 33  
## 3 Ford 22

Similar to the previous example, this code snippet also selects the first 3 rows and first 2 columns. 1:3 is equal to creating a vector from 1 to 3 which the second code snippet does.

my\_data\_frame[2, c(1, 3)]

## make cylinders  
## 2 Honda 4

This code snippet specifically selects the second row of the data-frame. At the same time, it selects the first and the third columns. Different than the previous examples this is **not** a range in form of 1:3, it is a vector with 1 **and** 3.

my\_data\_frame[1:2, ]

## make mpg cylinders  
## 1 Toyota 34 4  
## 2 Honda 33 4

This example takes the first two rows of the data-frame. At the same time, it selects all the columns as nothing was specified in the column selection part.

logical\_condition <- my\_data\_frame$mpg >= 30  
my\_data\_frame[logical\_condition, ]

## make mpg cylinders  
## 1 Toyota 34 4  
## 2 Honda 33 4  
## 4 Toyota 32 6

This code snippet first defines a logical condition which would return TRUE or FALSE for each row if that the mpg value of that row is more than or equal to 30. The logical\_condition variable is a vector with TRUE and FALSE values.

In the second part of the code, this vector is used to filter the rows which has an mpg value less than 30. Similar to the previous example, as the column is not specified, it returns all 3 columns.

my\_data\_frame[my\_data\_frame$mpg >= 30, ]

## make mpg cylinders  
## 1 Toyota 34 4  
## 2 Honda 33 4  
## 4 Toyota 32 6

This example is exactly identical as the previous example as it again checks if the mpg value for each row is more then or equal to 30. Similarly, it selects all the columns as it does not specify anything related to columns.

my\_data\_frame[my\_data\_frame$mpg >= 32, c(2, 3)]

## mpg cylinders  
## 1 34 4  
## 2 33 4  
## 4 32 6

This example merges the previous examples together. We select the rows with an mpg value larger than or equal to 32. At the same time, it selects the 2nd and the 3rd columns by specifying them in a vector.

my\_data\_frame[my\_data\_frame$mpg >= 32, c("mpg", "cylinders")]

## mpg cylinders  
## 1 34 4  
## 2 33 4  
## 4 32 6

In this example, it first selects the rows with an mpg value larger then or equal to 32. At the same time, it specifically selects the columns named mpg and cylinders.

Now try to subset my\_data\_frame to only include rows that have a cylinders value of 4.

my\_data\_frame[my\_data\_frame$cylinders == 4, ]

## make mpg cylinders  
## 1 Toyota 34 4  
## 2 Honda 33 4

To do this, we can basically use the *equal to* operator. We have to select the rows with cylinder values equal to 4. At the same time, we should ensure that we select all the columns.

From the result, we can see that the first and second vehicles (branded Toyota and Honda with 34 and 33 mpg respectively) has 4 cylinders. 15/15

# Problem #6

Create data as the following list:

my\_list <- list(classes\_offered = c("MIS 431", "MIS 310", "MIS 410", "MIS 412"),  
 student\_data = data.frame(student\_id = c(54, 100, 32, 423, 2, 19, 39),  
 age = c(18, 22, 27, 18, 29, 22, 20),  
 gpa = c(3.1, 2.8, 3.7, 3.4, 3.2, 3.4, 3.2),  
 stringsAsFactors = FALSE))

Write the R code that calculates the median value (use the median() function) of the gpa variable in student\_data. All you need to do is pass the student\_id vector into the median() function.

This code snippet creates a list of a vector and a data-frame. We should first access to the student\_data data-frame which is inside my\_list. Following this, we should access to the GPA column in order to pass it to the median function and calculate the median GPA. We can achieve this chain access using the $ operator.

gpa.values = my\_list$student\_data$gpa # Saving the GPA values  
median(gpa.values) # Calculating the median

## [1] 3.2 10/10

# Problem #7

Let us first create dataframes.

Feature1A <- c("A", "B", "C", "D")  
Feature2A <- c(1000, 2000, 3000, 4000)  
Feature3A <- c(25.5, 35.5, 45.5, 55.5)  
Feature4A <- c(10, 34, 78, 3)  
Dataframe1 <- data.frame(Feature1A, Feature2A, Feature3A, Feature4A)  
colnames(Dataframe1) <- c("Feature1", "Feature2", "Feature3", "Feature4")  
Dataframe1

## Feature1 Feature2 Feature3 Feature4  
## 1 A 1000 25.5 10  
## 2 B 2000 35.5 34  
## 3 C 3000 45.5 78  
## 4 D 4000 55.5 3

# creating Dataframe2  
Feature1B <- c("E", "F", "G", "H")  
Feature2B <- c(5000, 6000, 7000, 8000)  
Feature3B <- c(65.5, 75.5, 85.5, 95.5)  
Dataframe2 <- data.frame(Feature1B, Feature2B, Feature3B)  
colnames(Dataframe2) <- c("Feature1", "Feature2", "Feature3")  
Dataframe2

## Feature1 Feature2 Feature3  
## 1 E 5000 65.5  
## 2 F 6000 75.5  
## 3 G 7000 85.5  
## 4 H 8000 95.5

Merge merges Features 1-3 of the two data frames and called the resulting dataframe as Output. Use function merge().

output = merge(Dataframe1, Dataframe2, all = TRUE)  
output

## Feature1 Feature2 Feature3 Feature4  
## 1 A 1000 25.5 10  
## 2 B 2000 35.5 34  
## 3 C 3000 45.5 78  
## 4 D 4000 55.5 3  
## 5 E 5000 65.5 NA  
## 6 F 6000 75.5 NA  
## 7 G 7000 85.5 NA  
## 8 H 8000 95.5 NA

In this case, we can directly use the merge function on both data-frames and set the all argument to TRUE in order to keep all rows from the both data-frames and fill the unavailable values with NA. only merge feature 1-3. 7/10

# Problem #8

## Part A

Import the data from Moodle or shared Google drive, it is called pima.csv. Change the name of the nine columns to preg\_times, glucose\_test, blood\_press, tsk\_thickness, serum, bm\_index, pedigree\_fun, age, class.

pima = read.csv("../../DATA/pima.csv")  
colnames(pima) = c("preg\_times",  
 "glucose\_test",  
 "blood\_press",  
 "tsk\_thickness",  
 "serum",  
 "bm\_index",  
 "pedigree\_fun",  
 "age",  
 "class")

This code snippet will simply read the pima dataset from my data folder and change the column names to the requested values.

## Part B

All patients (768 Observations) in this dataset contains are females at least 21 years old of Pima Indian heritage. All zero values for the biological variables other than number of times pregnant should be treated as missing values. Count how many zeros are there in each variable (column). For any 0 in the data (except for class and preg\_times) assign it as an NA.

summary(pima == 0)

## preg\_times glucose\_test blood\_press tsk\_thickness   
## Mode :logical Mode :logical Mode :logical Mode :logical   
## FALSE:657 FALSE:763 FALSE:733 FALSE:541   
## TRUE :111 TRUE :5 TRUE :35 TRUE :227   
## serum bm\_index pedigree\_fun age   
## Mode :logical Mode :logical Mode :logical Mode :logical   
## FALSE:394 FALSE:757 FALSE:768 FALSE:768   
## TRUE :374 TRUE :11   
## class   
## Mode :logical   
## FALSE:268   
## TRUE :500

When we take the summary of the values equal to 0, we will have the dataset converted to TRUE and FALSE logical values. In the summary, we can see how many TRUE values we have which shows us the count of 0 values.

exclude.columns = c("class", "preg\_times")  
selected.cols = !(colnames(pima) %in% exclude.columns)  
pima[, selected.cols][pima[, selected.cols] == 0] = NA

To replace the 0 values with NA, we should first save the columns we would like to exclude from this process in a vector. We should also create another vector with selected columns, so that we can apply the procedure only to these columns.

After we selected the column we would like to work with, we should use these column names to select the 0 values in these respective columns and replace them with NA value.

summary(pima == 0)

## preg\_times glucose\_test blood\_press tsk\_thickness   
## Mode :logical Mode :logical Mode :logical Mode :logical   
## FALSE:657 FALSE:763 FALSE:733 FALSE:541   
## TRUE :111 NA's :5 NA's :35 NA's :227   
## serum bm\_index pedigree\_fun age   
## Mode :logical Mode :logical Mode :logical Mode :logical   
## FALSE:394 FALSE:757 FALSE:768 FALSE:768   
## NA's :374 NA's :11   
## class   
## Mode :logical   
## FALSE:268   
## TRUE :500

We can call the summary function again to see that the values are not TRUE anymore and they are set to NA.

## Part C

For class variable, check if it is a factor and if not, then make it a factor with levels 0 replaced with neg (for negative diabetic) and 1 replicated with pos (for positive diabetic).

class(pima$class)

## [1] "integer"

When we check the class of the class variable, we can see that it belongs to the integer class.

pima$class = factor(pima$class,  
 levels = c(0, 1),  
 labels = c("neg", "pos"))

We can replace this column with a factor using the factor function. In this function, we can determine the levels as 0 and 1. At the same time, we can set the labels of these levels as “neg” and “pos” as requested in the prompt.

class(pima$class)

## [1] "factor"

levels(pima$class)

## [1] "neg" "pos"

We can check the class and the levels of the variable again and see that it now belongs to the factor class with levels of “pos” and “neg”.

## Part D

Make data subsets for four age groups: 21-36, 37-51, 52-66 and 67-81.

subset.21.36 = subset(pima, age %in% 21:36)  
subset.37.51 = subset(pima, age %in% 37:51)  
subset.52.66 = subset(pima, age %in% 52:66)  
subset.67.81 = subset(pima, age %in% 67:81)

We can directly use the %in% operator to select the age values by checking if they are in a specific range or not. We can also check the answers by looking at the min/max values in the numerical summary of the age column of these subsets.

summary(subset.21.36$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 21.00 22.00 25.00 26.22 29.00 36.00

summary(subset.37.51$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 37.00 39.00 42.00 42.63 45.00 51.00

summary(subset.52.66$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 52.00 54.00 58.00 57.91 61.25 66.00

summary(subset.67.81$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 67 67 69 70 70 81

## Part E

Create a new factor vector called age.factor, with age in pima data replaced with the age group.

pima$age.factor[pima$age %in% 21:36] = "21-36"  
pima$age.factor[pima$age %in% 37:51] = "37-51"  
pima$age.factor[pima$age %in% 52:66] = "52-66"  
pima$age.factor[pima$age %in% 67:81] = "67-81"  
  
pima$age.factor = factor(pima$age.factor)  
class(pima$age.factor)

## [1] "factor"

summary(pima$age.factor)

## 21-36 37-51 52-66 67-81   
## 514 181 64 9

In this example we can set the values of the new column age.factor to the specific age groups, using the same selection method we used in the previous part.

After setting the values, we can directly convert this column into a factor. Lastly, we can check the class of the age.factor to ensure the type is correct. 25/25

# Project Problem

Do the following for the approved dataset(s):

### Part (A) Read the data here in R.

traffic.data = read.csv("../../DATA/ca\_san\_francisco\_2020\_04\_01.csv")

### Part (B) Show the structure of data.

str(traffic.data)

## 'data.frame': 905070 obs. of 22 variables:  
## $ raw\_row\_number : chr "869921" "869922" "869923" "869924" ...  
## $ date : chr "2014-08-01" "2014-08-01" "2014-08-01" "2014-08-01" ...  
## $ time : chr "00:01:00" "00:01:00" "00:15:00" "00:18:00" ...  
## $ location : chr "MASONIC AV & FELL ST" "GEARY&10TH AV" "SUTTER N OCTAVIA ST" "3RD ST & DAVIDSON" ...  
## $ lat : num 37.8 37.8 37.8 37.7 37.8 ...  
## $ lng : num -122 -122 -122 -122 -122 ...  
## $ district : chr NA NA NA NA ...  
## $ subject\_age : int NA NA NA NA NA NA NA NA NA NA ...  
## $ subject\_race : chr "asian/pacific islander" "black" "hispanic" "hispanic" ...  
## $ subject\_sex : chr "female" "male" "male" "male" ...  
## $ type : chr "vehicular" "vehicular" "vehicular" "vehicular" ...  
## $ arrest\_made : logi FALSE FALSE FALSE FALSE FALSE FALSE ...  
## $ citation\_issued : logi FALSE TRUE TRUE FALSE TRUE TRUE ...  
## $ warning\_issued : logi TRUE FALSE FALSE TRUE FALSE FALSE ...  
## $ outcome : chr "warning" "citation" "citation" "warning" ...  
## $ contraband\_found : logi NA NA NA NA NA NA ...  
## $ search\_conducted : logi FALSE FALSE FALSE FALSE FALSE FALSE ...  
## $ search\_vehicle : logi FALSE FALSE FALSE FALSE FALSE FALSE ...  
## $ search\_basis : chr NA NA NA NA ...  
## $ reason\_for\_stop : chr "Mechanical or Non-Moving Violation (V.C.)" "Mechanical or Non-Moving Violation (V.C.)" "Mechanical or Non-Moving Violation (V.C.)" "Mechanical or Non-Moving Violation (V.C.)" ...  
## $ raw\_search\_vehicle\_description : chr "No Search" "No Search" "No Search" "No Search" ...  
## $ raw\_result\_of\_contact\_description: chr "Warning" "Citation" "Citation" "Warning" ...

This output shows us the variable names and the way they are recognized by R. This shows me that in the future parts of my project, I might have to convert some of the columns to appropriate data types as date/time are recognized as strings (they should be date and time objects, some of the categorical variables (such as subject sex/age/race) are not recognized a factors, etc.

### Part (C) What is the dimension of your data?

dim(traffic.data)

## [1] 905070 22

From this output, we can see that the dataset has 22 columns and 905070 rows.

### Part (D) Show names of variables in the data.

colnames(traffic.data)

## [1] "raw\_row\_number" "date"   
## [3] "time" "location"   
## [5] "lat" "lng"   
## [7] "district" "subject\_age"   
## [9] "subject\_race" "subject\_sex"   
## [11] "type" "arrest\_made"   
## [13] "citation\_issued" "warning\_issued"   
## [15] "outcome" "contraband\_found"   
## [17] "search\_conducted" "search\_vehicle"   
## [19] "search\_basis" "reason\_for\_stop"   
## [21] "raw\_search\_vehicle\_description" "raw\_result\_of\_contact\_description"

Similar to the structure, this part gives us the variable names of the dataset.

### Part (E) Find easy answers to your research question (one of them) using the data

One of my research questions was: ***How do traffic stop rates vary by driver demographic characteristics?***

Before tackling the question and plotting the results, there are two data preparation steps we have to complete:

traffic.data$subject\_sex = factor(traffic.data$subject\_sex,  
 levels = c("male", "female"),  
 labels = c("Male", "Female"))  
  
traffic.data$date = as.Date(traffic.data$date)  
summary(traffic.data$date)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## "2007-01-01" "2009-02-02" "2011-03-16" "2011-06-13" "2013-11-16" "2016-06-30"

I converted the subject\_sex column to a R-factor. At the same time, I converted the date into a real date column in order to extract the year data from each entry.

Lastly, I checked the summary of the date column to see the starting and the end date of the data.

In this question -as an easy answer is asked- I am going to focus on the gender information of the traffic stops.

At first step, I am going to look into the gender information of stopped drivers for the whole time frame.

group.gender.total = aggregate(traffic.data$raw\_row\_number,  
 by = list(traffic.data$subject\_sex),  
 FUN = length)  
colnames(group.gender.total) = c("Gender", "Count")  
  
group.gender.total

## Gender Count  
## 1 Male 639219  
## 2 Female 265851

From this output, we can see that the amount male drivers stopped is much higher than (approximately 2.4 times higher) than the female drivers.

At the second stage of this problem, I grouped the data by year and also gender. This allowed me to see the gender distribution of the traffic stops throughout the years.

group.gender.year.total = aggregate(x = traffic.data$raw\_row\_number,  
 by = list(traffic.data$subject\_sex,  
 format(traffic.data$date, "%Y")),  
 FUN = length)  
  
colnames(group.gender.year.total) = c("Gender", "Year", "Count")  
  
group.gender.year.total

## Gender Year Count  
## 1 Male 2007 72339  
## 2 Female 2007 29857  
## 3 Male 2008 79638  
## 4 Female 2008 33487  
## 5 Male 2009 76942  
## 6 Female 2009 33670  
## 7 Male 2010 72194  
## 8 Female 2010 32576  
## 9 Male 2011 68768  
## 10 Female 2011 30739  
## 11 Male 2012 57758  
## 12 Female 2012 24633  
## 13 Male 2013 52997  
## 14 Female 2013 21165  
## 15 Male 2014 65666  
## 16 Female 2014 26295  
## 17 Male 2015 63182  
## 18 Female 2015 22816  
## 19 Male 2016 29735  
## 20 Female 2016 10613

From this table, we can see that throughout the years the trend of “male drivers being stopped more then female drivers” continued. It is important to note that 2016 data is available only until June which explains the less than usual amount of data points. In the future parts of my data exploration and modeling, I might consider to remove 2016 data from the dataset when comparing years.

From both of these simple data explorations, we can see that male drivers have been stopped significantly more compared to the female drivers in San Francisco between January 1st 2007 and June 30th 2016 in both total number of stops through the time period and also the total number of stops annually.

Another demographic information I would like to explore is the relationship between the age and the number of stops. As the question asked for easy answers I limited myself only to the gender part of this dataset. In the further assignments I can include subject age, and race to the demographic data exploration part of the project.