Capital One Data Science Challenge: Trips | Submitted by Saurav Kaushik

Question 1

• *Programmatically download and load into your favorite analytical tool the trip data for September 2015.*

The data dictionary can be found here:
http://www.nyc.gov/html/tlc/downloads/pdf/data_dictionary_trip_records_green.pdf
(http://www.nyc.gov/html/tlc/downloads/pdf/data_dictionary_trip_records_green.pdf)

```
url<-"https://s3.amazonaws.com/nyc-tlc/trip+data/green_tripdata_2015-09.csv"

download.file(url = url, destfile = "dataset.csv", mode = "wb")
```

Loading the downloaded data:

```
data<-read.csv("dataset.csv")
```

• *Report how many rows and columns of data you have loaded.*

```
print("Structure of data:")
```

```
## [1] "Structure of data:"
```

```
str(data)
```

```
## 'data.frame':    1494926 obs. of  21 variables:
##  $ VendorID             : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ lpep_pickup_datetime : Factor w/ 1079075 levels "2015-09-01 00:00:00",..: 58 97 43 60 5 15
18 52 60 50 ...
##  $ Lpep_dropoff_datetime: Factor w/ 1077210 levels "2015-09-01 00:00:00",..: 3 6 6 22 5 11 14
12 27 28 ...
##  $ Store_and_fwd_flag   : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RateCodeID           : int  5 5 1 1 1 1 1 1 1 1 ...
##  $ Pickup_longitude     : num  -74 -74 -73.9 -73.9 -74 ...
##  $ Pickup_latitude      : num  40.7 40.9 40.8 40.8 40.7 ...
##  $ Dropoff_longitude    : num  -74 -74 -73.9 -73.9 -73.9 ...
##  $ Dropoff_latitude     : num  40.7 40.9 40.8 40.8 40.7 ...
##  $ Passenger_count      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Trip_distance        : num  0 0 0.59 0.74 0.61 1.07 1.43 0.9 1.33 0.84 ...
##  $ Fare_amount          : num  7.8 45 4 5 5 5.5 6.5 5 6 5.5 ...
##  $ Extra                : num  0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ MTA_tax              : num  0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ Tip_amount           : num  1.95 0 0.5 0 0 1.36 0 0 1.46 0 ...
##  $ Tolls_amount         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Ehail_fee            : logi  NA NA NA NA NA NA ...
##  $ improvement_surcharge: num  0 0 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 ...
##  $ Total_amount         : num  9.75 45 5.8 6.3 6.3 8.16 7.8 6.3 8.76 6.8 ...
##  $ Payment_type         : int  1 1 1 2 2 1 1 2 1 2 ...
##  $ Trip_type            : int  2 2 1 1 1 1 1 1 1 1 ...
```

```
print("Dimensions of data:")
```

```
## [1] "Dimensions of data:"
```

```
dim(data)
```

```
## [1] 1494926      21
```

So, the data contains approximately 1.5 million observations (rows) of with 21 varaibles (columns).

Question 2

• *Plot a histogram of the number of the trip distance ("Trip Distance").*

Plotting an interactive histogram of the number of the trip distance:

```
library("plotly")
```

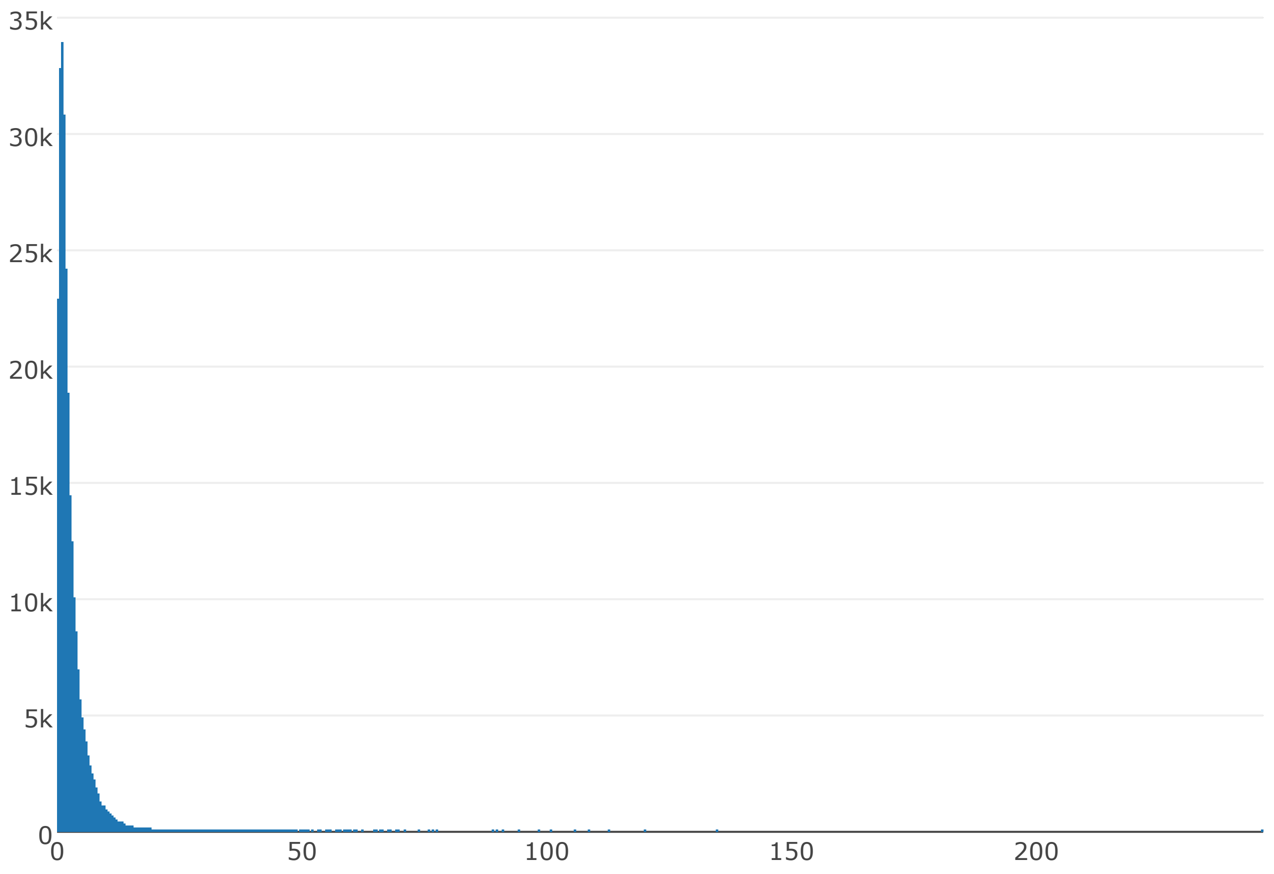
```
## Loading required package: ggplot2
```

```
## 
## Attaching package: 'plotly'
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

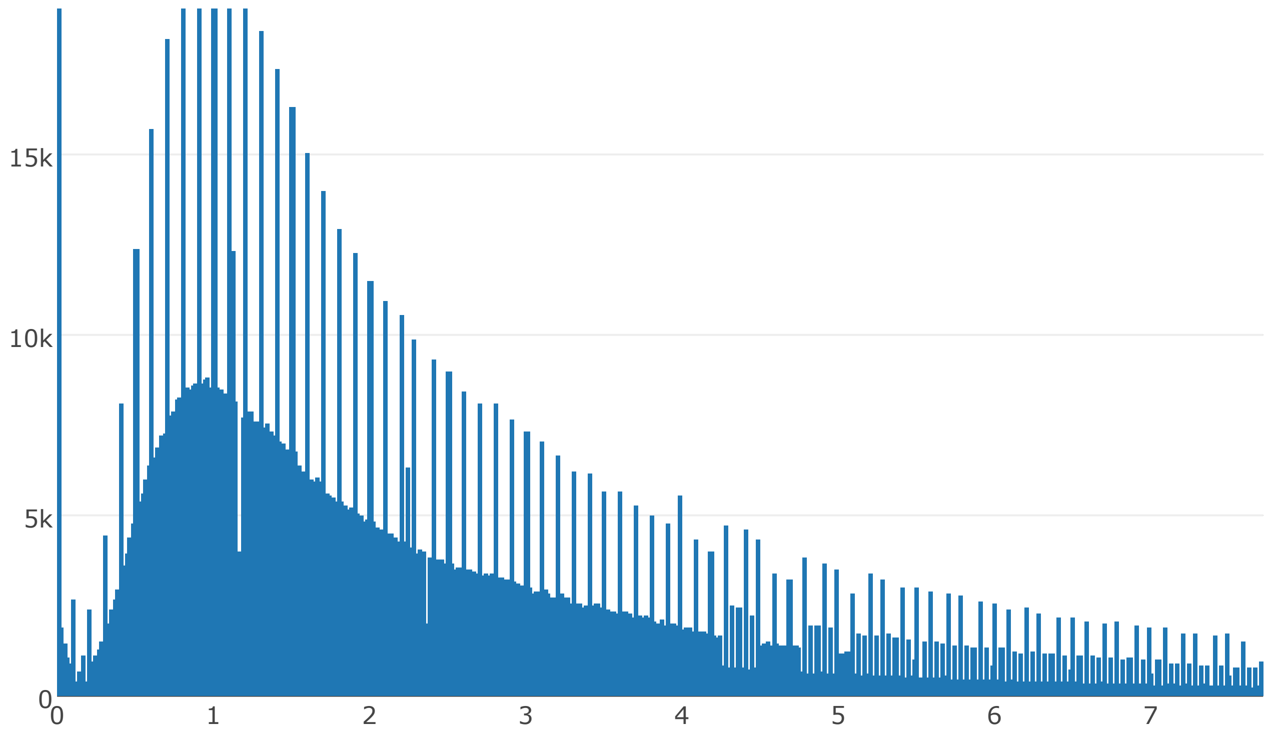
```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
plot_ly(x = data$Trip_distance, type = "histogram")
```



Removing outliers:

```
plot_ly(x = data$Trip_distance[!data$Trip_distance %in% boxplot.stats(data$Trip_distance)$out]
, type = "histogram")
```



Significantly large no. of people are travelling distances like X.Y9 (ex: 1.09, 1.19, etc) This might be the case because of:

1. There might exist different charge buckets that changes at every 0.09 miles.
2. Meters in significant no. of taxi's change value at intervals of 0.09, 0.19, 0.29, etc miles.

• *Report any structure you find and any hypotheses you have about that structure.*

i. We had close to 1.5 million observations of 21 variables but we had only 20 variables/ columns in the data dictionary.

So, Let's also look at the mising value count and structure.

```
print("Count of missing values")
```

```
## [1] "Count of missing values"
```

```
sum(is.na(data))
```

```
## [1] 1494930
```

```
print("Proportion of missing values per feature:")
```

```
## [1] "Proportion of missing values per feature:"
```

```
sort(sapply(data, function(x){sum(is.na(x))/length(x)}),decreasing = T)
```

```
##             Ehail_fee             Trip_type              VendorID 
##          1.000000e+00          2.675718e-06          0.000000e+00 
##  lpep_pickup_datetime Lpep_dropoff_datetime    Store_and_fwd_flag 
##          0.000000e+00          0.000000e+00          0.000000e+00 
##            RateCodeID      Pickup_longitude       Pickup_latitude 
##          0.000000e+00          0.000000e+00          0.000000e+00 
##     Dropoff_longitude      Dropoff_latitude       Passenger_count 
##          0.000000e+00          0.000000e+00          0.000000e+00 
##         Trip_distance           Fare_amount                 Extra 
##          0.000000e+00          0.000000e+00          0.000000e+00 
##               MTA_tax            Tip_amount          Tolls_amount 
##          0.000000e+00          0.000000e+00          0.000000e+00 
## improvement_surcharge          Total_amount          Payment_type 
##          0.000000e+00          0.000000e+00          0.000000e+00
```

```
print("No. of missing values in Trip_type:")
```

```
## [1] "No. of missing values in Trip_type:"
```

```
sum(is.na(data$Trip_type))
```

```
## [1] 4
```

Aha! As suspected, the column not present in the data dictionary: ‘Ehail_fee’ has 100% missing values. This is a garbage feature and we should remove it from our data.

Removing Ehail_fee from data which contains 100% missing values:

```
data$Ehail_fee<-NULL
```

ii. Also, there is a very small proportion (i.e. 4 in total) of missing values in the column Trip_type which is a code indicating whether the trip was a street-hail or a dispatch. This is automatically assigned based on the metered rate in use but can be altered by the driver. It can contain two values: 1 and 2:

1= Street-hail 2= Dispatch

Possible hypothisis of having missing values in Trip_type are:

1. Being a rule based system at the backend, it might be possible that no value of Trip_type is defined in the system for certain combinations of other variables. We can explore the other features in this case to figure out these combinations. In this case, filling these missing values with some aritiriary numbers makes more sense.
2. There might be some human error from the driver side by entering an invalid value as the driver also has the right to change this variable. In this case, using mode for fillin the missing values wil make more sense.

Testing:

1. While exploring other features for the observations where null values exists in Trip_type, I found that, all null values/ NA’s in Trip_type are characterised by the following occurances:

a. 99 RateCodeID, which is invalid according to the data dictionary.

b. 0 Passenger_count.
c. Signifancty high Total_ammount with Trip_distance 0.
d. Also, there are other interesting occourances like Pickup_latitude and Pickup_longitude present but o’s in Dropoff_latitude and Dropoff_longitude, but these are more general occurances than the onses mentioned above.

You can view the data with NA in Trip_type below:

```
data[is.na(data$Trip_type),]
```

```
##         VendorID lpep_pickup_datetime Lpep_dropoff_datetime
## 984681         1  2015-09-20 05:49:23   2015-09-20 09:49:23
## 985600         1  2015-09-20 06:27:40   2015-09-20 10:27:40
## 985621         1  2015-09-20 06:22:12   2015-09-20 10:22:12
## 1070055        1  2015-09-22 06:43:33   2015-09-22 10:43:33
##         Store_and_fwd_flag RateCodeID Pickup_longitude Pickup_latitude
## 984681                   Y         99        -73.93222        40.80048
## 985600                   Y         99        -73.89917        40.74610
## 985621                   Y         99        -73.95770        40.71779
## 1070055                  Y         99        -73.92852        40.69253
##         Dropoff_longitude Dropoff_latitude Passenger_count Trip_distance
## 984681                  0                0               0             0
## 985600                  0                0               0             0
## 985621                  0                0               0             0
## 1070055                 0                0               0             0
##         Fare_amount Extra MTA_tax Tip_amount Tolls_amount
## 984681        17.30     0       0          0            0
## 985600        12.35     0       0          0            0
## 985621        23.15     0       0          0            0
## 1070055       20.30     0       0          0            0
##         improvement_surcharge Total_amount Payment_type Trip_type
## 984681                      0        17.30            1        NA
## 985600                      0        12.35            1        NA
## 985621                      0        23.15            1        NA
## 1070055                     0        20.30            3        NA
```

By the looks of it, this looks like a machine error and we should fill these missing vales with some arbitiriary values.

```
data$Trip_type[is.na(data$Trip_type)]<-99
```

iii. Also, let’s correct the data types baesd on the data dictionary we have.

iii.a Converting VendorID to factors in R:

1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.

```
data$VendorID<-as.factor(data$VendorID)
```

iii.b Converting pickup and drop datetime to date-time types in R:

```
data$lpep_pickup_datetime<-as.POSIXct(data$lpep_pickup_datetime,tz="EST")

data$Lpep_dropoff_datetime<-as.POSIXct(data$Lpep_dropoff_datetime,tz="EST")
```

iii.c Convering RateCodeID to factor in R:

1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride

```
data$RateCodeID<-as.factor(data$RateCodeID)
```

iii.d Convering Payment_type to factor in R:

1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip

```
data$Payment_type<-as.factor(data$Payment_type)
```

iii.e Convering Trip_type to factor in R:

1= Street-hail 2= Dispatch 99 = Missing values

```
data$Trip_type<-as.factor(data$Trip_type)
```

Finally, let’s have a look athe structure of the data again:

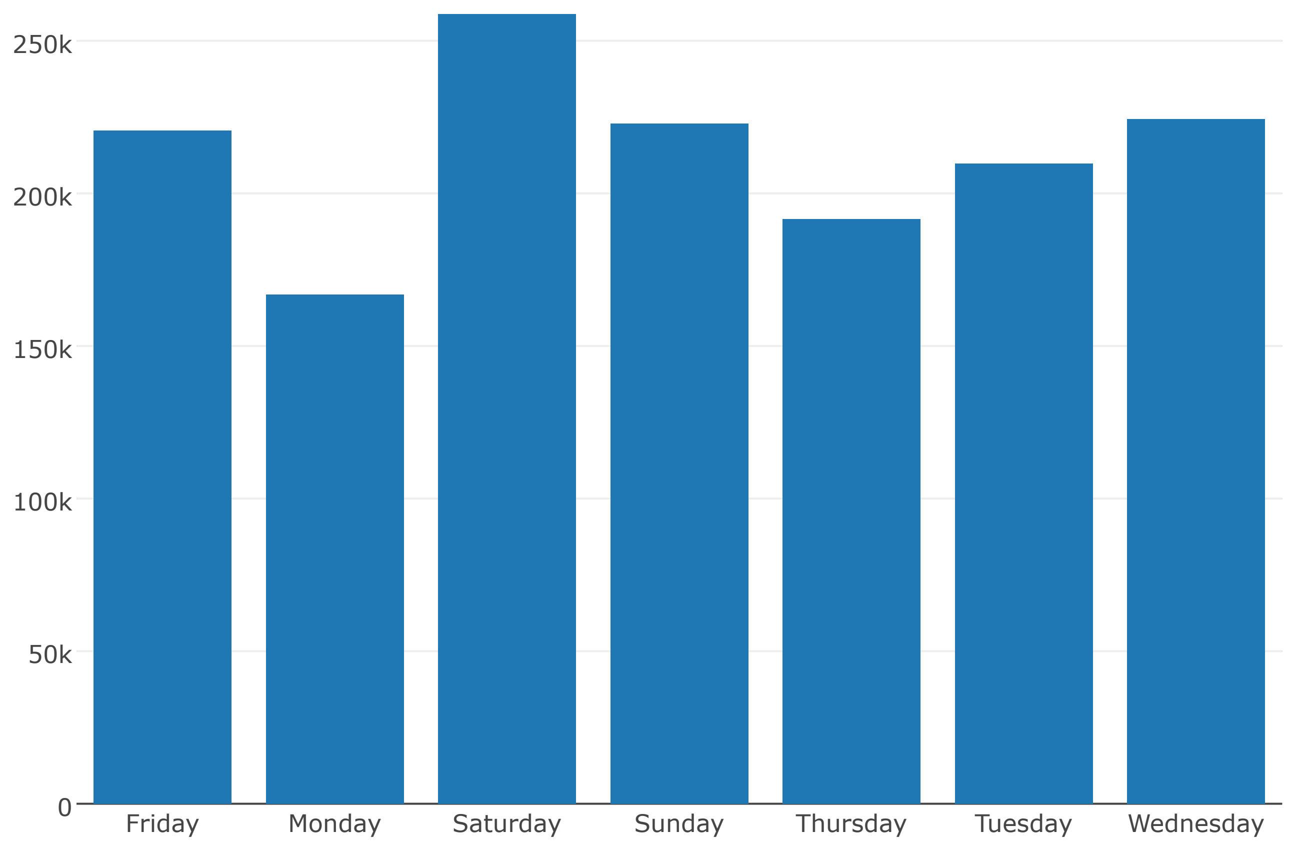
```
str(data)
```

```
## 'data.frame':    1494926 obs. of  20 variables:
##  $ VendorID             : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ lpep_pickup_datetime : POSIXct, format: "2015-09-01 00:02:34" "2015-09-01 00:04:20" ...
##  $ Lpep_dropoff_datetime: POSIXct, format: "2015-09-01 00:02:38" "2015-09-01 00:04:24" ...
##  $ Store_and_fwd_flag   : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RateCodeID           : Factor w/ 7 levels "1","2","3","4",..: 5 5 1 1 1 1 1 1 1 1 ...
##  $ Pickup_longitude     : num  -74 -74 -73.9 -73.9 -74 ...
##  $ Pickup_latitude      : num  40.7 40.9 40.8 40.8 40.7 ...
##  $ Dropoff_longitude    : num  -74 -74 -73.9 -73.9 -73.9 ...
##  $ Dropoff_latitude     : num  40.7 40.9 40.8 40.8 40.7 ...
##  $ Passenger_count      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Trip_distance        : num  0 0 0.59 0.74 0.61 1.07 1.43 0.9 1.33 0.84 ...
##  $ Fare_amount          : num  7.8 45 4 5 5 5.5 6.5 5 6 5.5 ...
##  $ Extra                : num  0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ MTA_tax              : num  0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ Tip_amount           : num  1.95 0 0.5 0 0 1.36 0 0 1.46 0 ...
##  $ Tolls_amount         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ improvement_surcharge: num  0 0 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 ...
##  $ Total_amount         : num  9.75 45 5.8 6.3 6.3 8.16 7.8 6.3 8.76 6.8 ...
##  $ Payment_type         : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 1 1 2 1 2 ...
##  $ Trip_type            : Factor w/ 3 levels "1","2","99": 2 2 1 1 1 1 1 1 1 1 ...
```

iv. Let’s look at the no. of trips distribution based on day of week.

Assumption: Almost all Pick-ups and drops happen on the same day i.e. assuming no cases where pickup is at like Mon, 23:50:00 and drop off is at Tue, 01:02:20

```
data$weekday<-weekdays(data$Lpep_dropoff_datetime)

plot_ly(x = data$weekday, type = "histogram")
```



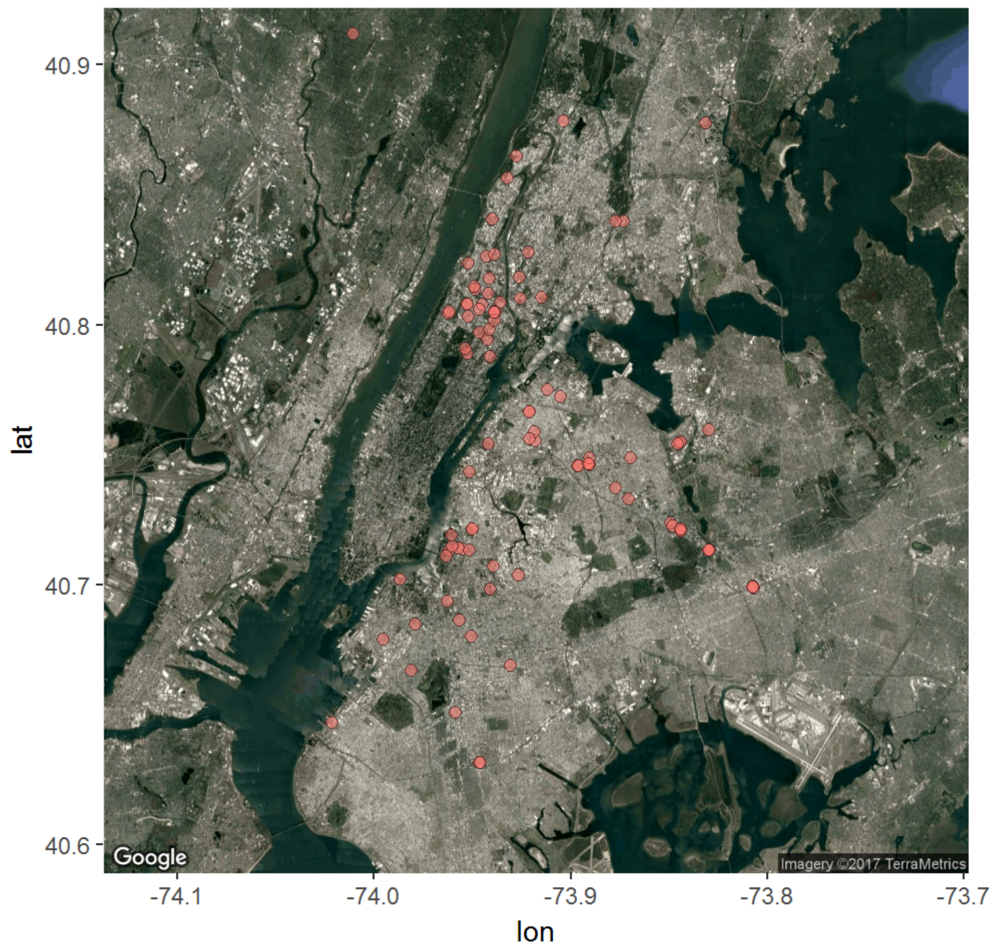
The peak of number of rides is at weekends being highest at Saturday (with ~258 K) followed by Wednesday (with ~224 k) and Sunday (with ~222.9 K). Monday is the day with least number of rides (with ~166.8 k).

iv.a. Count of rides on Weekdays is significantly lesser than in weekends. Possibly, People prefer taxi's for travelling in weekends over personal vehicles maybe because of drinking habits.

iv.b. Monday has signifantly less count of trips possibly because Green taxis are only allowed to provide services in the outer areas of New York and most of the people are moving in towards the central NY for thier work rather than out.

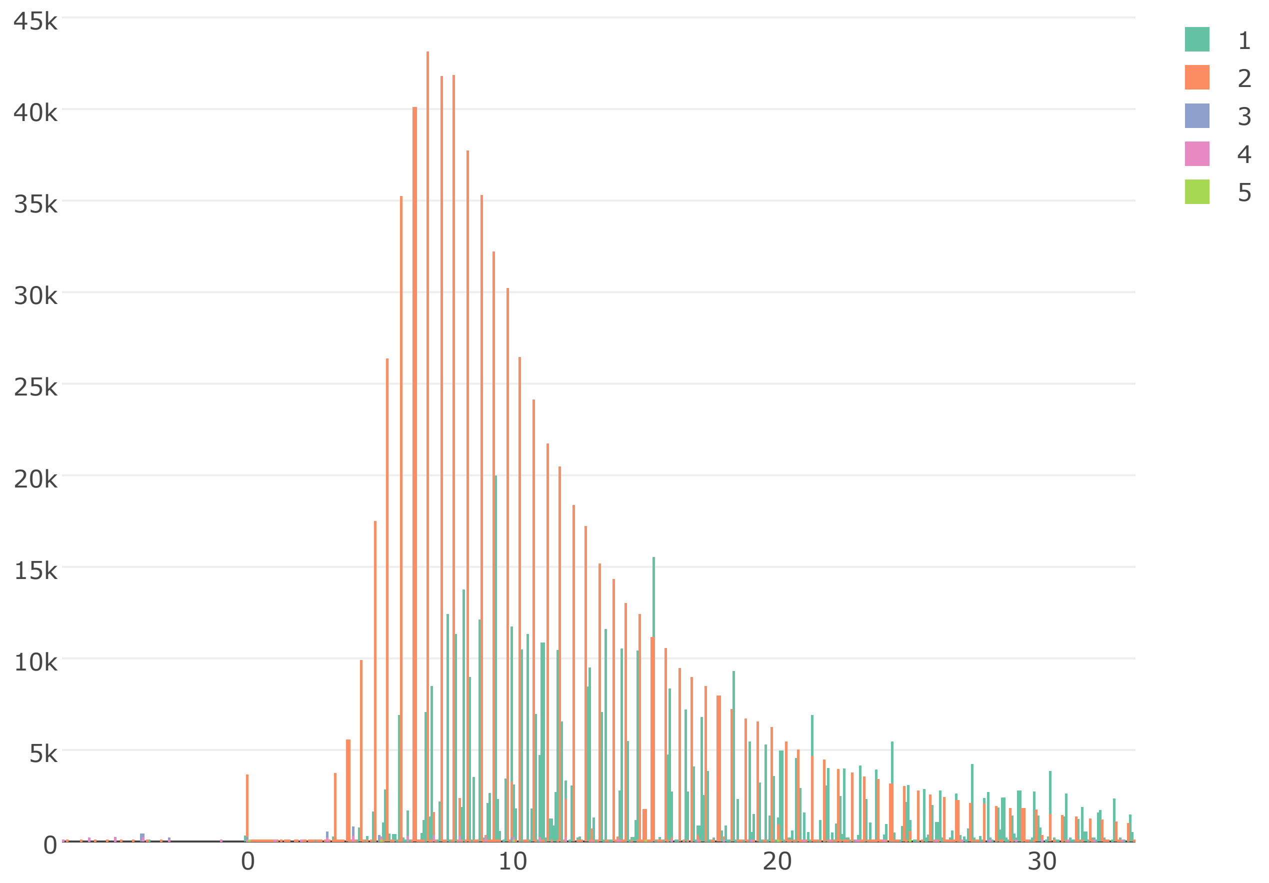
v. let's have a look at sample of the pickup latitudes and longitudes on map:

```r
# Loading the required packages
library(ggplot2)
library(ggmap)

# creating a sample data.frame with your lat/lon points
lon <- data$Pickup_longitude[1:100]
lat <- data$Pickup_latitude[1:100]
df <- as.data.frame(cbind(lon,lat))

# getting the map
map <- get_map(location = c(lon = mean(df$lon), lat = mean(df$lat)), zoom = 11,
               maptype = "satellite", scale = 2)

# plotting the map with some points on it
ggmap(map) +
    geom_point(data = df, aes(x = lon, y = lat, fill = "red", alpha = 0.8), size = 2, shape =
21) +
    guides(fill=FALSE, alpha=FALSE, size=FALSE)
```



vi. Let’s also have a look at the final fare distribution and its assiciation with payment type.

```r
plot_ly(
  x = data$Total_amount[!data$Total_amount %in% boxplot.stats(data$Total_amount)$out]
, color = data$Payment_type[!data$Total_amount %in% boxplot.stats(data$Total_amount)$out]
, type = "histogram")
```



For Reference:

1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip

vi.a Total ammount less than 0 is most possibly because of machine errors.

vi.b Most of the total ammounts are peaked at around 8 USD.

vi.c People generally prefer credit cards for higher payments.

This was a very interesting section and it would have been interesting to do much exploration of features and thier relationships provided with more time.

Question 3

• *Report mean and median trip distance grouped by hour of day.*

```
library(lubridate)
```

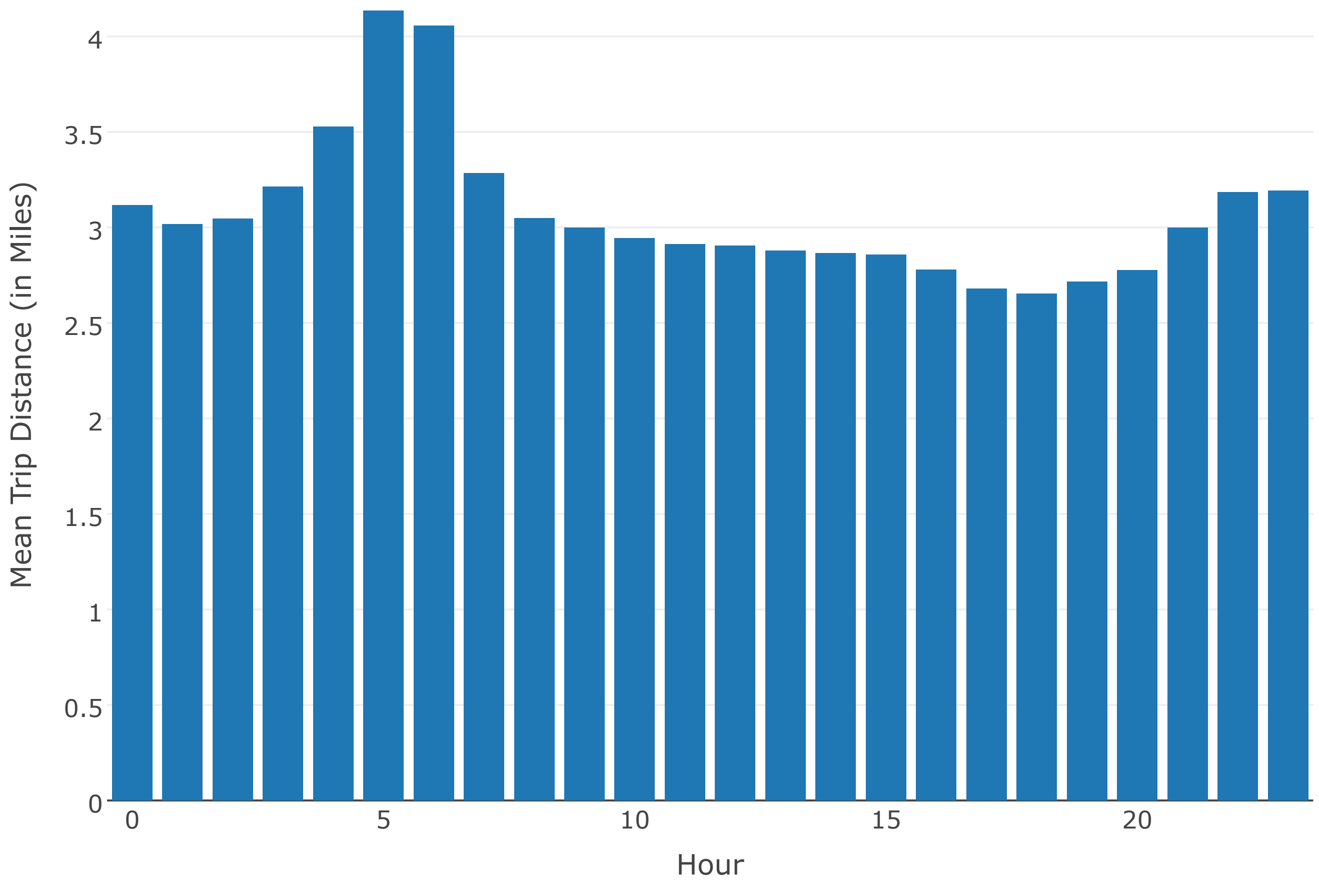
```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
data$hour<-hour(data$lpep_pickup_datetime)

mean_df<-as.data.frame(aggregate(data$Trip_distance, list(data$hour), mean))

colnames(mean_df)<-c("Hour","Mean_dist")

median_df<-as.data.frame(aggregate(data$Trip_distance, list(data$hour), median))

colnames(median_df)<-c("Hour","Median_dist")
```

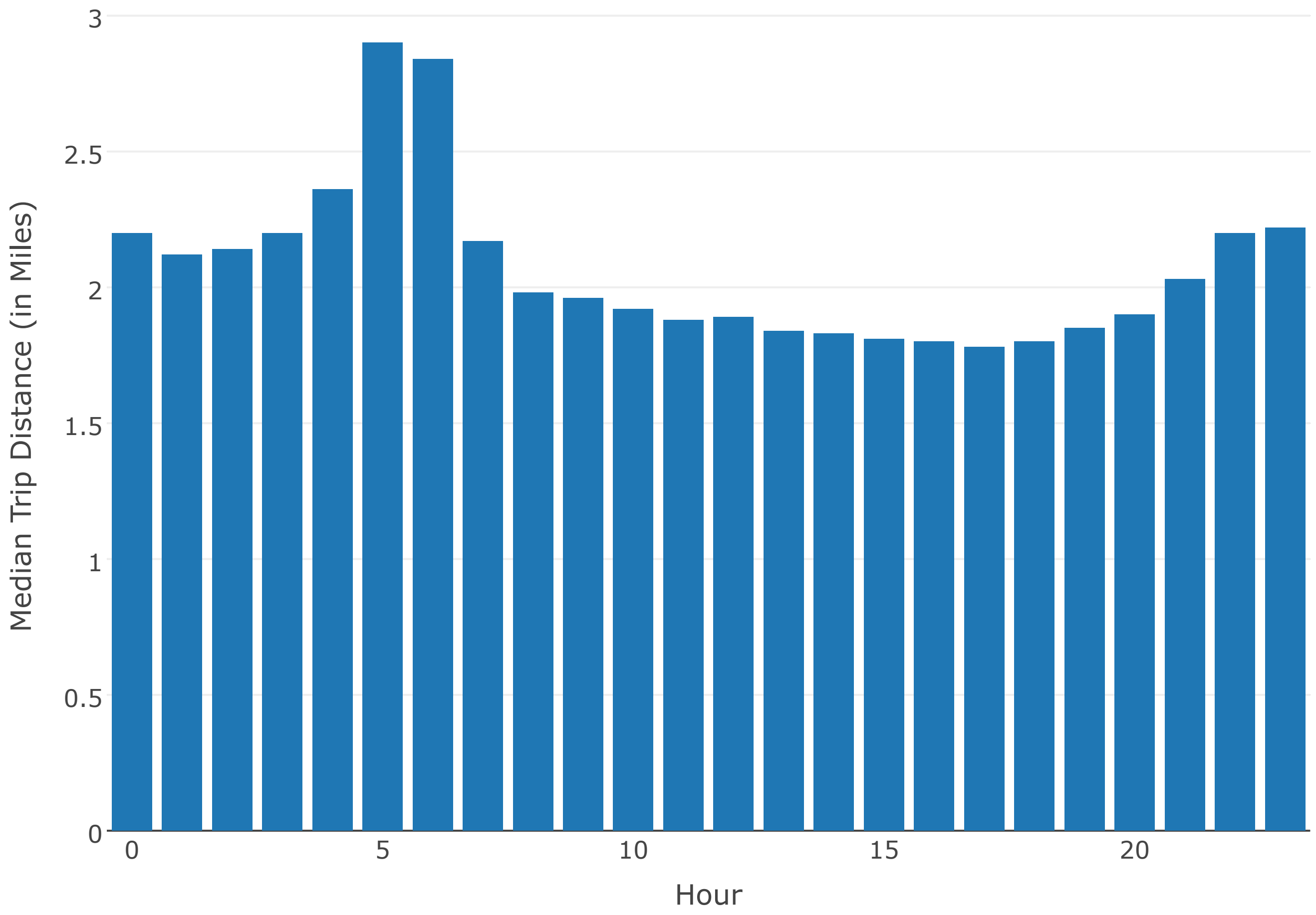
The distrirution of mean trip distance per hour is as:

```
plot_ly(x = mean_df$Hour,
        y = mean_df$Mean_dist,
        type = 'bar')%>%
        layout(xaxis = list(title = "Hour"),
               yaxis = list(title = "Mean Trip Distance (in Miles)"))
```



The distrirution of median trip distance per hour is as:

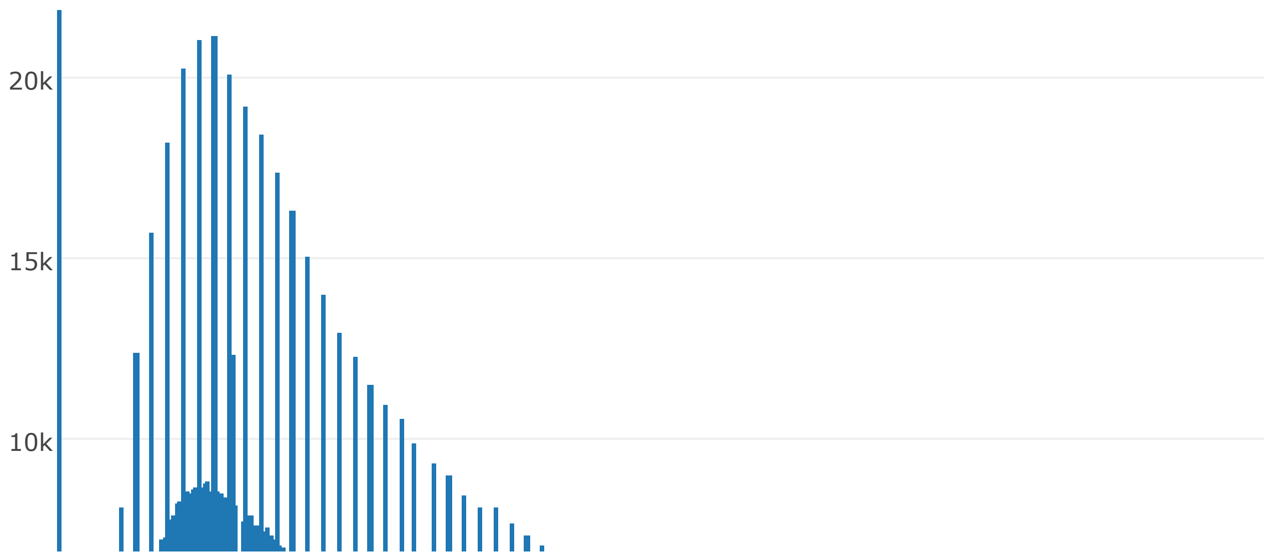
```
plot_ly(x = median_df$Hour,
        y = median_df$Median_dist,
        type = 'bar')%>%
        layout(xaxis = list(title = "Hour"),
               yaxis = list(title = "Median Trip Distance (in Miles)"))
```

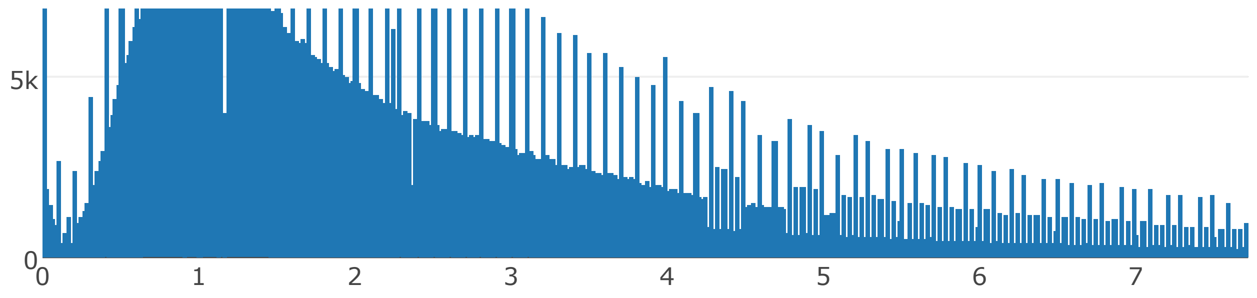


i. The hourly mean trip distance follows the same trend as hourly median trip distance but the hourly mean trip distance is significantly above the hourly median trip distance by close to one mile.

Having mean greater than median represents represents right skewness of distribution. Let’s confirm it by checking the distribution of trip distance varaible:

```
plot_ly(
  x = data$Trip_distance[!data$Trip_distance %in% boxplot.stats(data$Trip_distance)$out]
, type = "histogram")
```





ii. Also, we see that the two peaks occour at the morning hours and at the evening hours, which looks like commuting hours. However, the morning peak is taller that the evening peak possibly because Green taxi's are not permitted to pick up from the central NY where most of the offices are situated.

• *We'd like to get a rough sense of identifying trips that originate or terminate at one of the NYC area airports. Can you provide a count of how many transactions fit this criteria, the average fair, and any other interesting characteristics of these trips.*

Going through the data dictionary again, I'm able to figure out that the RateCodeID variable looks to be the appropriate way to seprate airport trips frpm non-airort trips.

Other was can be to use the pickup drop - latitudes and longitudes to create this separationn, but that would be time consuming and might be slighly unaccurate as wel.

So, in RateCodeID variable, can be used to find the 4 major airports:

2=JFK : John F. Kennedy International Airport 3=Newark : Newark Liberty International Airport 4=Nassau or Westchester : Lynden Pindling International Airport or Westchester County Airport

```
data$Is_Airport_Trip<-NA

data$Is_Airport_Trip<-as.numeric(ifelse(data$RateCodeID == 2 | data$RateCodeID == 3 | data$RateC
odeID == 4, 1, 0))
```

i. The no. of airport rides/ transactions of the total data in absolute numbers as well as percentage is as:

```
print("Absolute no:")
```

```
## [1] "Absolute no:"
```

```
nrow(data[data$Is_Airport_Trip==1,])
```

```
## [1] 6477
```

```
print("Percentage of all transactions:")
```

```
## [1] "Percentage of all transactions:"
```

```
print(paste ( (nrow(data[data$Is_Airport_Trip==1,])/nrow(data))*100, " %"))
```

```
## [1] "0.433265593079524  %"
```

ii. The average fair for airport trips is:

```
print("Average Fair for Airport trips:")
```

```
## [1] "Average Fair for Airport trips:"
```

```
print( paste( mean(data$Fare_amount[data$Is_Airport_Trip==1]), " USD"))
```

```
## [1] "50.5747290412228  USD"
```

```
print("Average Fair for Non - Airport trips:")
```

```
## [1] "Average Fair for Non - Airport trips:"
```

```
print( paste( mean(data$Fare_amount[data$Is_Airport_Trip==0]), " USD"))
```

```
## [1] "12.3777029444744  USD"
```

The avearge fare for airport trips is more than 4 times the average fare for the non - airport trips.

iii. Let’s also look at the count of passangers for airport and non-airport trips:

```
print("Average trip distance for Airport trips:")
```

```
## [1] "Average trip distance for Airport trips:"
```

```
print( paste( mean(data$Trip_distance[data$Is_Airport_Trip==1]), " miles"))
```

```
## [1] "11.0243106376409  miles"
```

```
print("Average trip distance for Non - Airport trips:")
```

```
## [1] "Average trip distance for Non - Airport trips:"
```

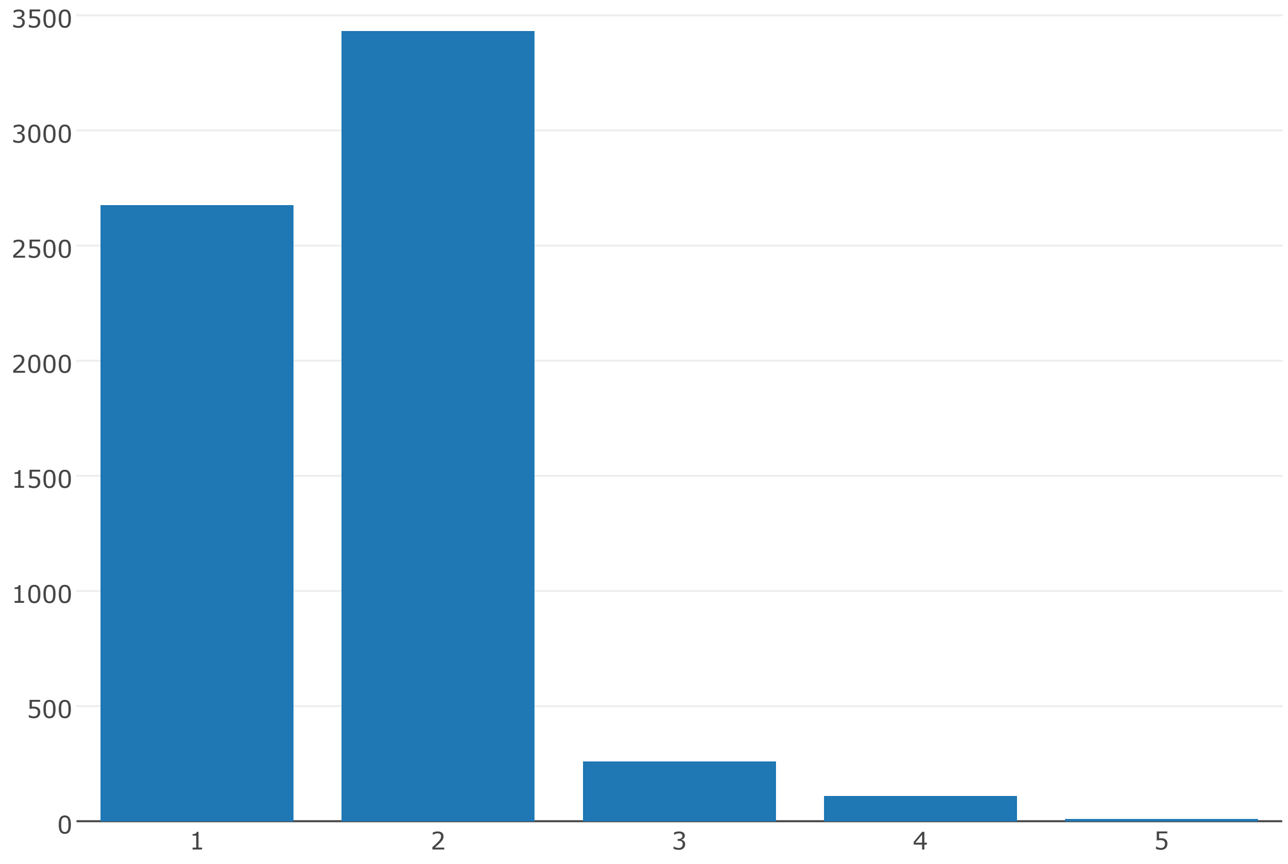
```
print( paste( mean(data$Trip_distance[data$Is_Airport_Trip==0]), " miles"))
```

```
## [1] "2.93308435156327  miles"
```

Looks fair enough now, even if the airport trips/ transactions has average fair ammount more 4 times than the nor-airport trips/ transactions, the trip distance is also 3.8 times.

iv. Let's also look at the payment method prefernces of the airport trips:

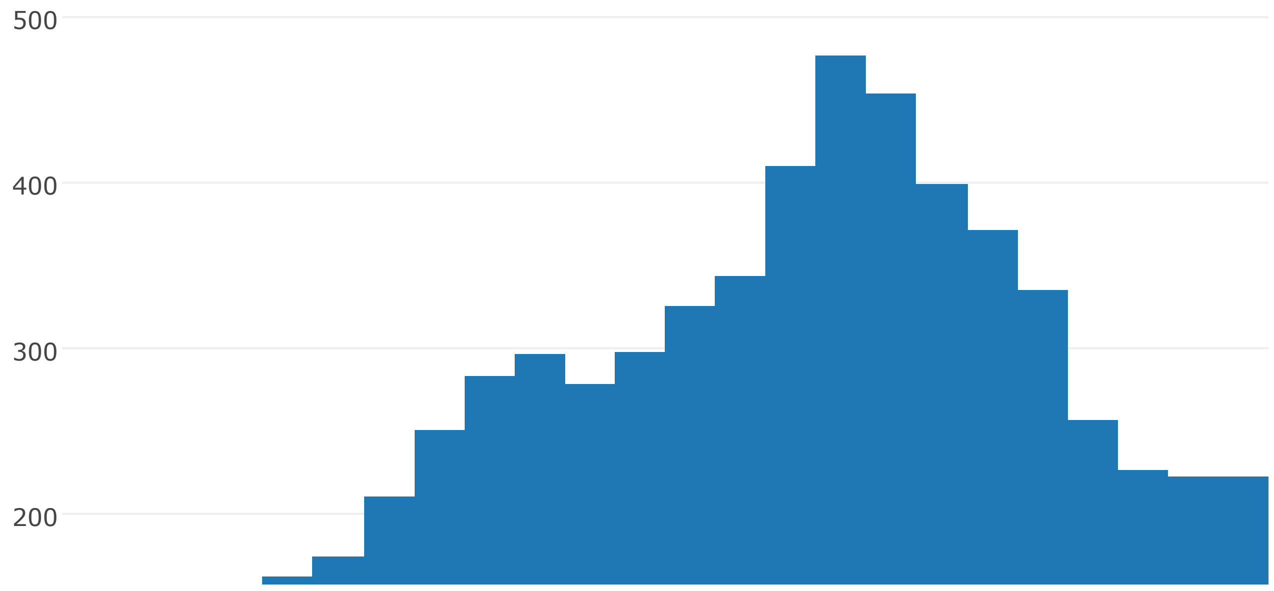
```
plot_ly(x = data$Payment_type[data$Is_Airport_Trip == 1], type = "histogram")
```

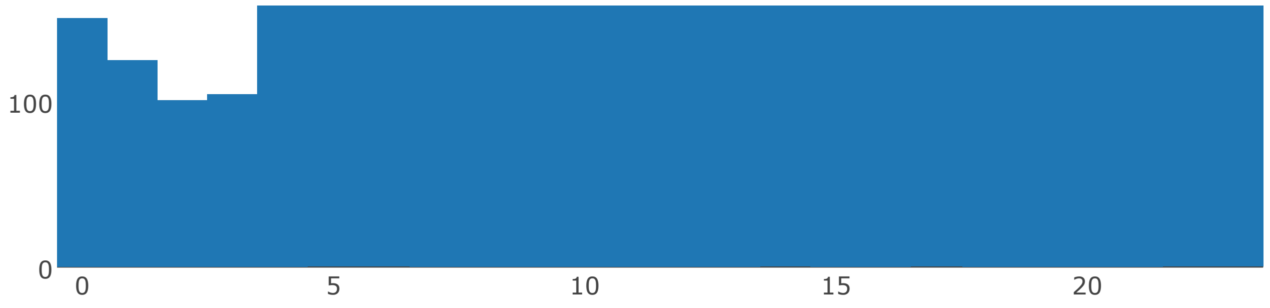


Therefore, The passangers in airport trips likes to pay by cash followed by credit cards.

v. Let's also look at the houry trend of airport trips.

```
plot_ly(x = data$hour[data$Is_Airport_Trip == 1], type = "histogram")
```

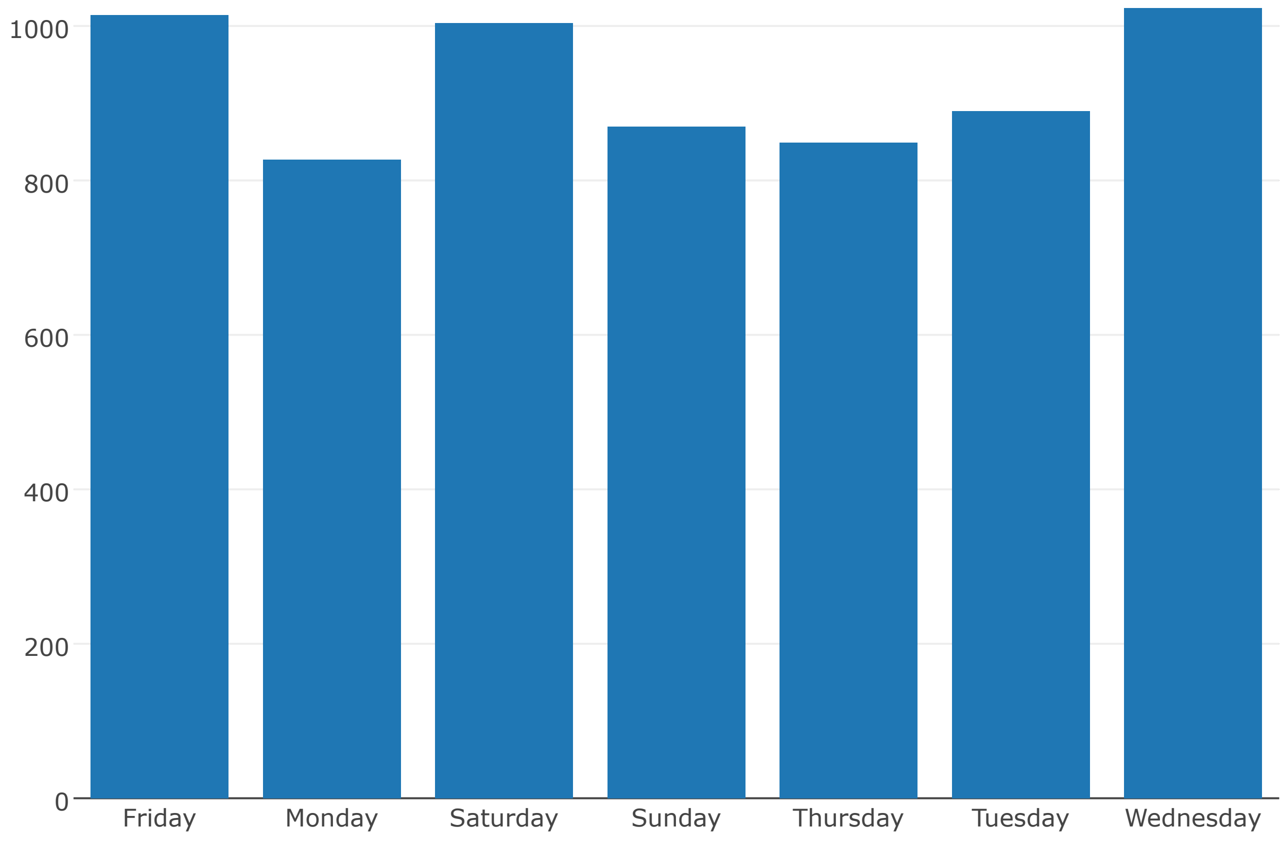




Most of the airport taxis pickups are in afternoon time.

vi. let's also have a look at the weekly distriution of airport trips.

```
plot_ly(x = data$weekday[data$Is_Airport_Trip == 1], type = "histogram")
```



Most airport rides are made mid-week on wednesday followed by saturday and friday.

Question 4

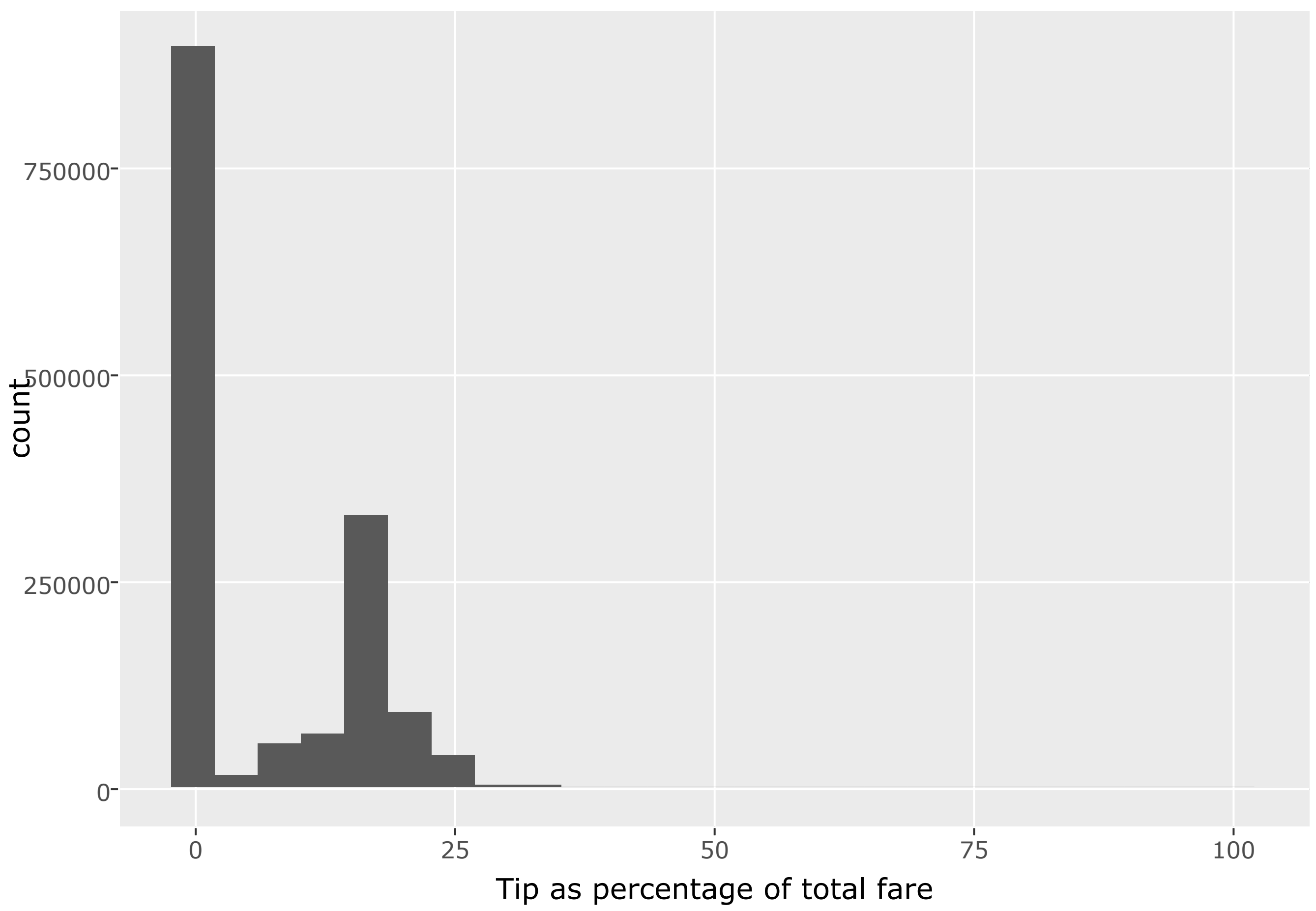
• *Build a derived variable for tip as a percentage of the total fare.*

Creating the derived varaible:

```
data$tip_perc<-(data$Tip_amount/data$Total_amount)*100

data$tip_perc[is.na(data$tip_perc)]<-0
```

Studying the distribution of the derived variable:

```
ggplotly(qplot(data$tip_perc,bins = 25, xlab = "Tip as percentage of total fare"))
```



• *Build a predictive model for tip as a percentage of the total fare. Use as much of the data as you like (or all of it). We will validate a sample.*

This predictive model is supposed to provide what is the percentage of total tip that can be expected as tip given a certain set of features. Since the outcome is derived from Tip_amount and Total_amount, so they should be removed.

```
data$Tip_amount<-NULL

data$Total_amount<-NULL
```

Now, let’s start the predictive modelling:

i. Data Cleaning.

We have already treated the missing values. Also, removing observations with negative fare ammount, negative improvement surcharge and negative toll amount as they seem logically flawed.

```
data<-data[data$Fare_amount>=0,]

data<-data[data$improvement_surcharge>=0,]

data<-data[data$Tolls_amount>=0,]
```

ii. Feature Engineering.

ii.a Let’s create some basic datetime based features from the drop-off datetime. As most of the tips are made at the dropoff instead of pickup. Since the data is only for one month, making features like hour of the day and weekday only makes sense.

```
data$hour<-as.numeric(hour(data$Lpep_dropoff_datetime))

data$weekday<-as.factor(weekdays(data$Lpep_dropoff_datetime))
```

ii.b Trip time:

```
data$Trip_time_mins<-as.numeric(difftime(data$Lpep_dropoff_datetime, data$lpep_pickup_datetime,
units = "mins"))
```

ii.c Average Trip speed (miles/mins):

```
data$Trip_avg_speed<-data$Trip_distance/data$Trip_time_mins
```

ii.d Tax as percentage of fare ammount:

```
data$Trip_toll_perc<-data$Tolls_amount/data$Fare_amount
```

I would have liked to create additional features if I had more time, but I’ll need to stop due to time-constraint. Anyway, I beleive we have a few really good set of features.

iii. Data Prepration.

For, data prepration, let’s convert a few factors that we have in our data into numeric by using one-hot encoding:

```
#Removing date columns

cols<-c("VendorID", "Store_and_fwd_flag", "RateCodeID", "Pickup_longitude", "Pickup_latitude",
"Dropoff_longitude", "Dropoff_latitude", "Passenger_count", "Trip_distance", "Fare_amount", "Ext
ra", "MTA_tax", "Tolls_amount", "improvement_surcharge", "Payment_type", "Trip_type", "weekday",
 "hour", "Is_Airport_Trip", "tip_perc", "Trip_time_mins", "Trip_avg_speed", "Trip_toll_perc")

library("caret")
```

```
## Loading required package: lattice
```

```
dmy <- dummyVars(" ~ .", data = data[,cols],fullRank = F)
data_new <- data.frame(predict(dmy, newdata = data[,cols]))

data_new$lpep_pickup_datetime<-data$lpep_pickup_datetime
```

Now, Let’s create the training, validation and testing set. Since this predictive model is supposed to be used in the real world for predicting for future percentage of total ammounts given as tips, therefore, the train, validation and test set should be created based on time instead of randomly.

We shall use the first 20 days of the mont as training data, next five days as validation set and the rest of days as testig data.

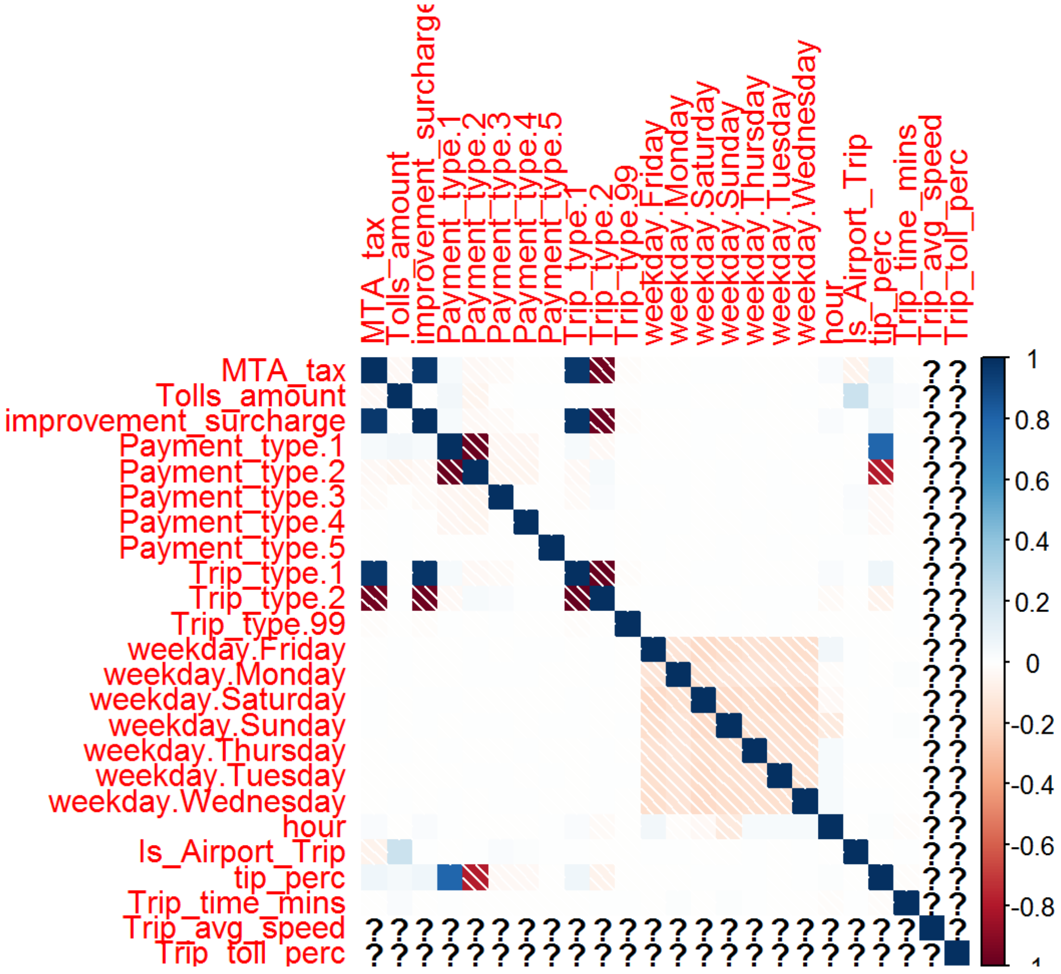
```
train<-data_new[data_new$lpep_pickup_datetime<"2015-09-20 00:00:00",]

val<-data_new[data_new$lpep_pickup_datetime>="2015-09-20 00:00:00" & data_new$lpep_pickup_dateti
me<"2015-09-25 00:00:00",]

test<-data_new[data_new$lpep_pickup_datetime>="2015-09-25 00:00:00",]

#Ensuring the sanity:

nrow(train)+nrow(val)+nrow(test)==nrow(data_new)
```

```
## [1] TRUE
```

iv. Exploratory analysis:

Let’s strudy the linear corelation of the independent varaibles with the continous outcome variable:

```
library(corrplot)

cols<-names(data_new)[!names(data_new) %in% 'lpep_pickup_datetime']

corrplot(cor(data_new[,cols[20:43]]), method = "shade")
```



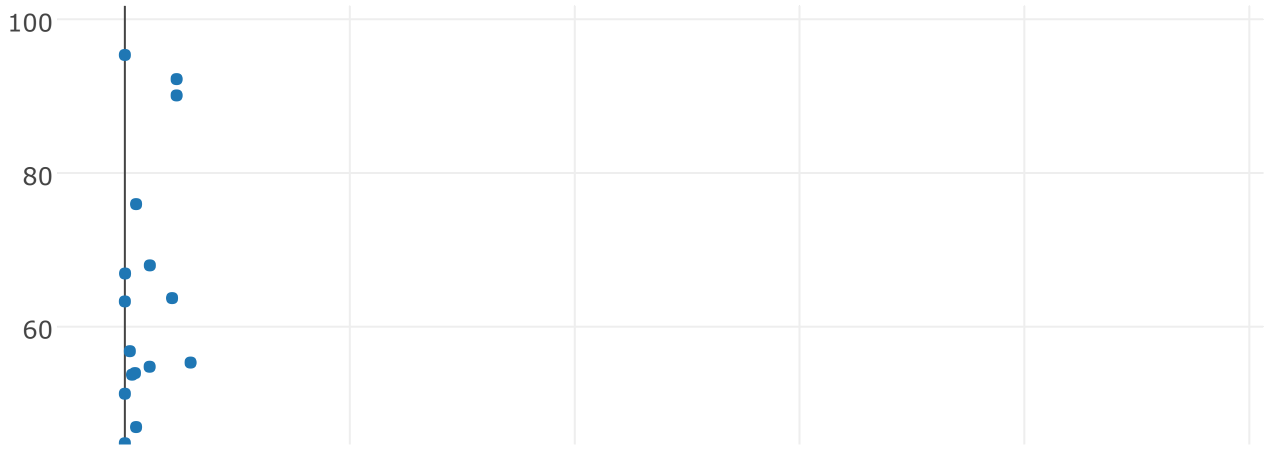
On an in-depth inspection, I found that there's very weak linear corelation between the tip_perc (outcome varaible) and other independent varaibles.

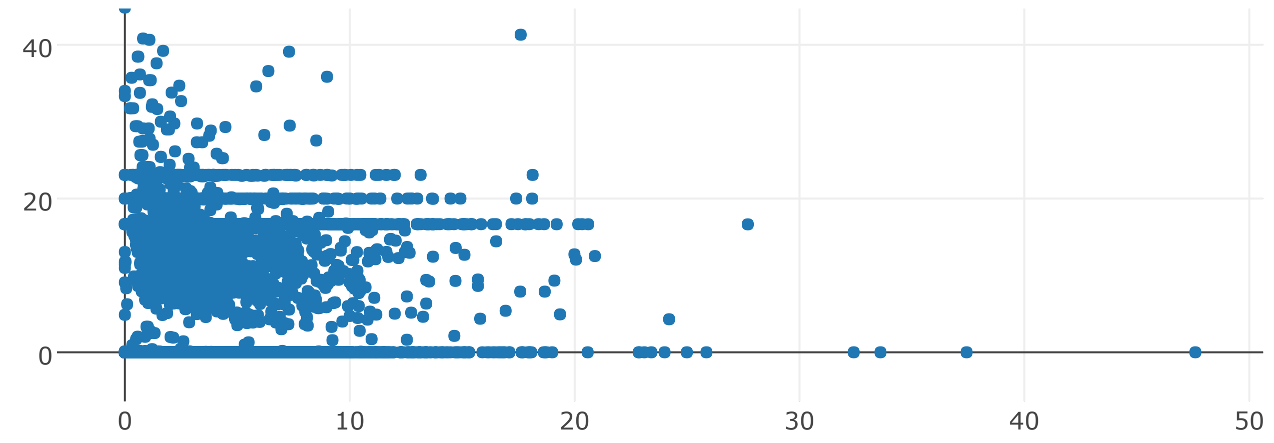
Let's check if there exists a stronger non linear corelation:

With Trip Distance:

```
plot_ly(x = train$Trip_distance[1:10000], y = train$tip_perc[1:10000], type = "scatter")
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

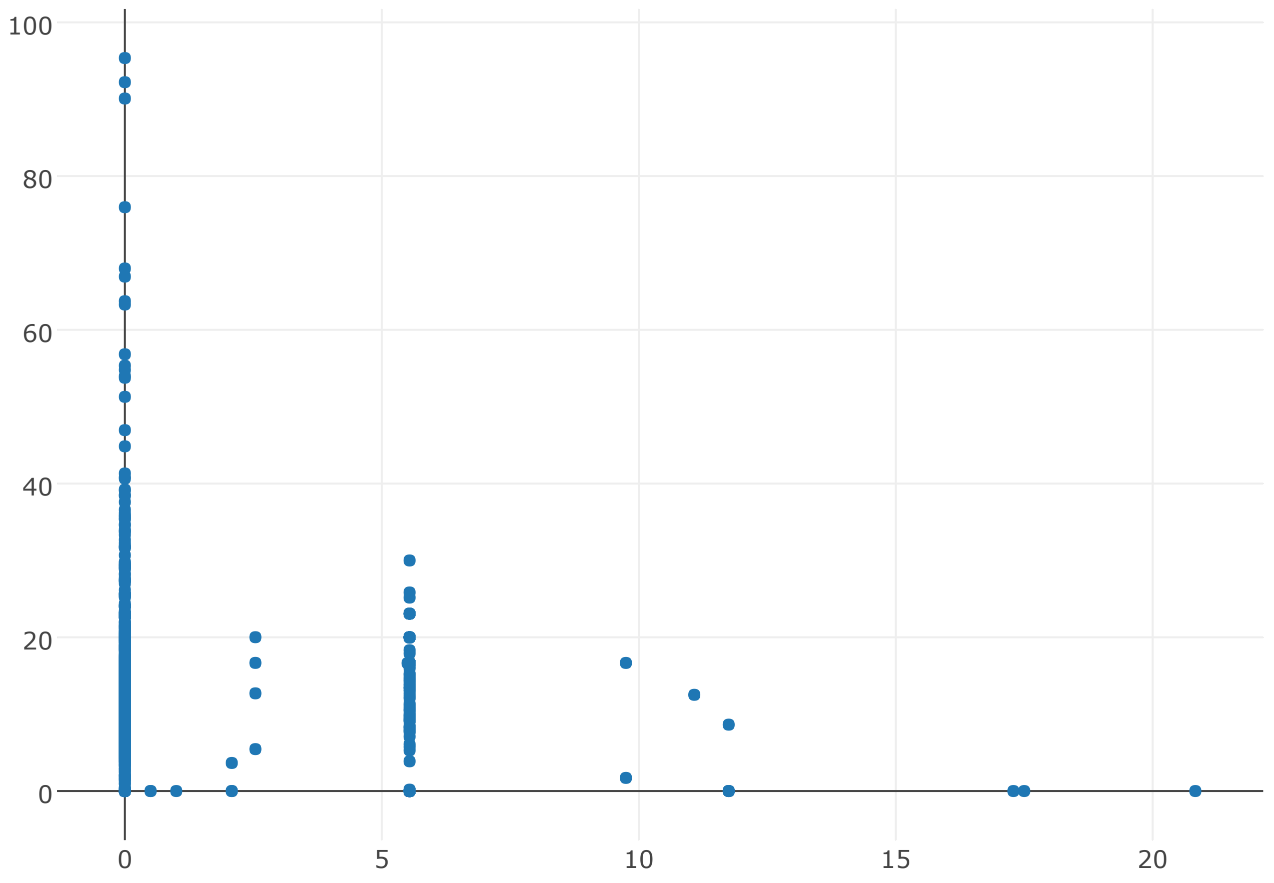




With Tolls amount:

```
plot_ly(x = train$Tolls_amount[1:10000], y = train$tip_perc[1:10000], type = "scatter")
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

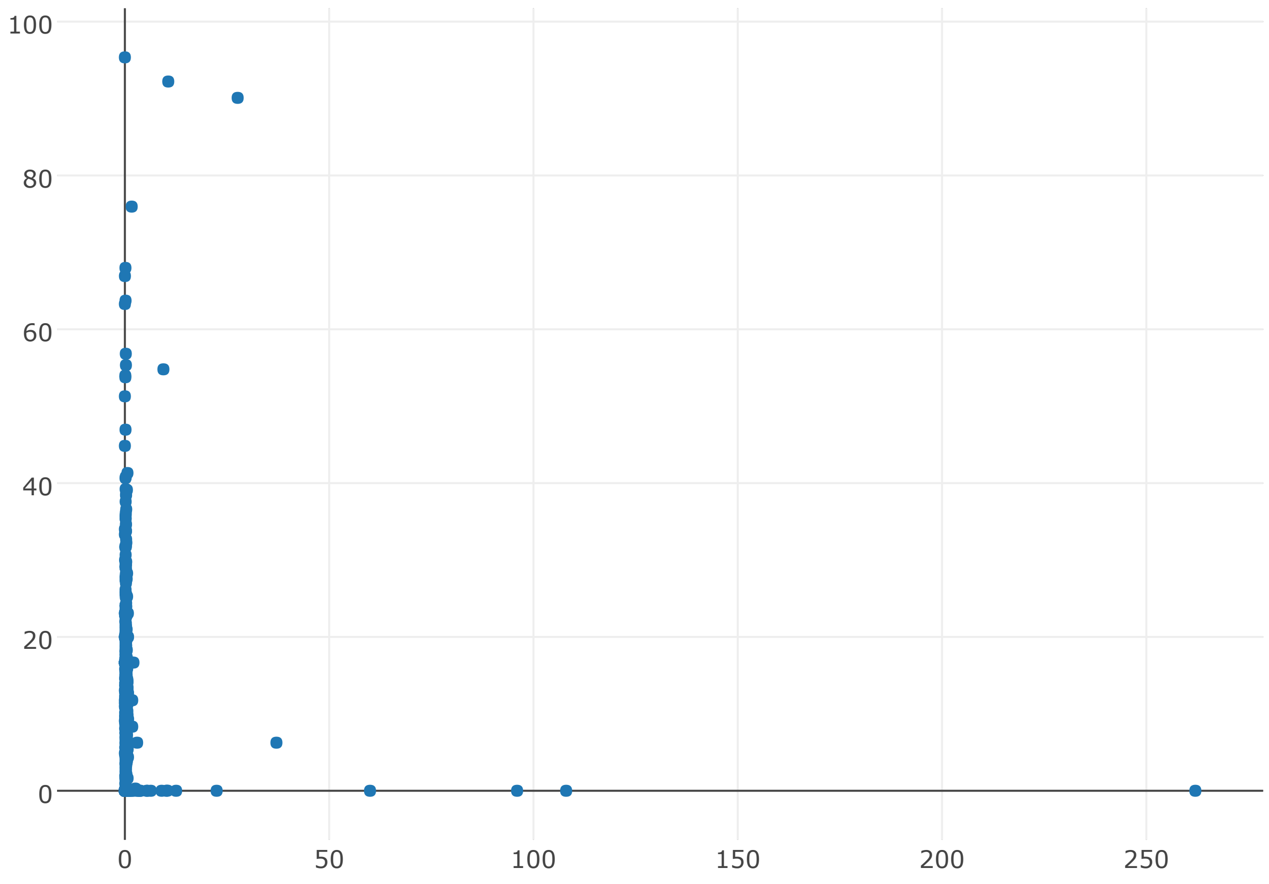


With Trip average speed

```
plot_ly(x = train$Trip_avg_speed[1:10000], y = train$tip_perc[1:10000], type = "scatter")
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

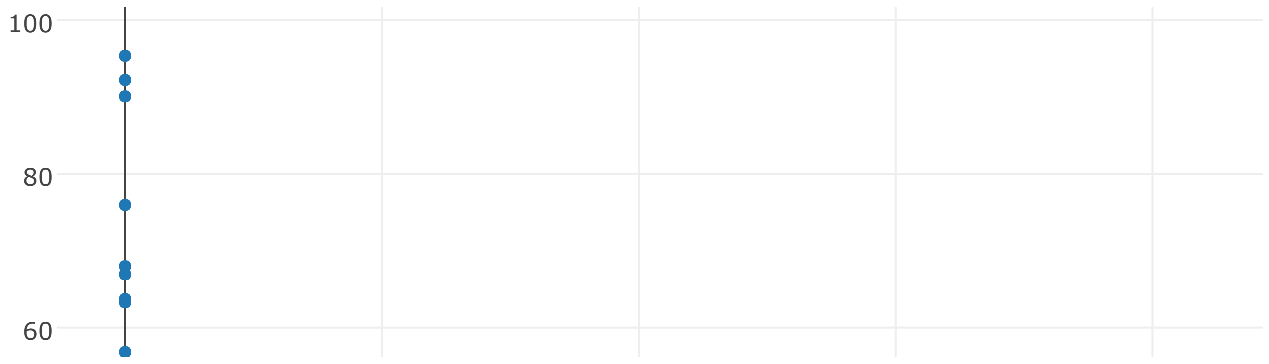
```
## Warning: Ignoring 5 observations
```

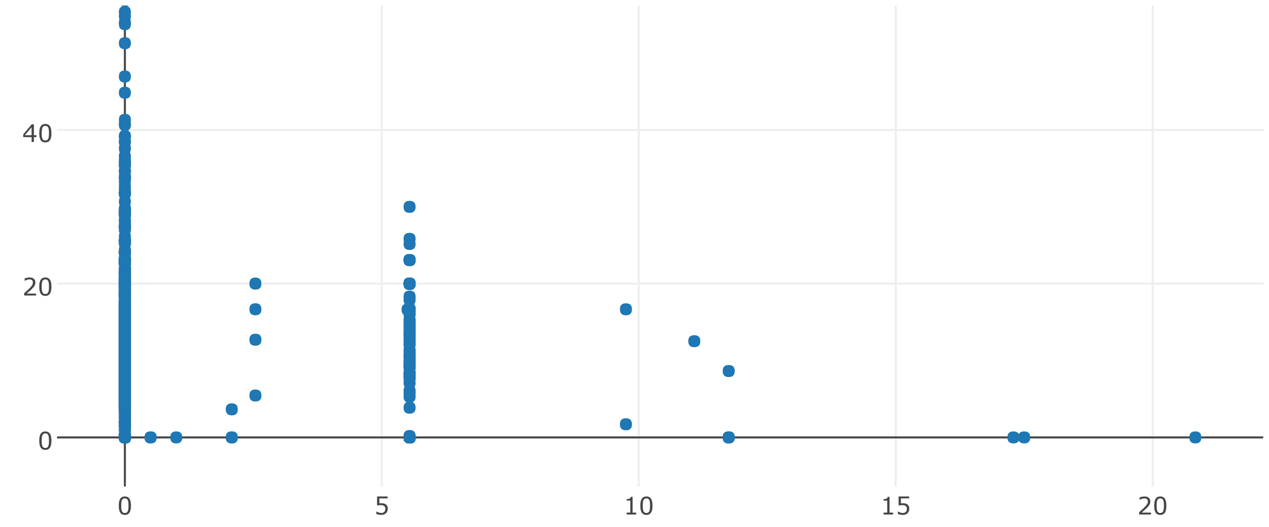


With Tolls amount:

```
plot_ly(x = train$Tolls_amount[1:10000], y = train$tip_perc[1:10000], type = "scatter")
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

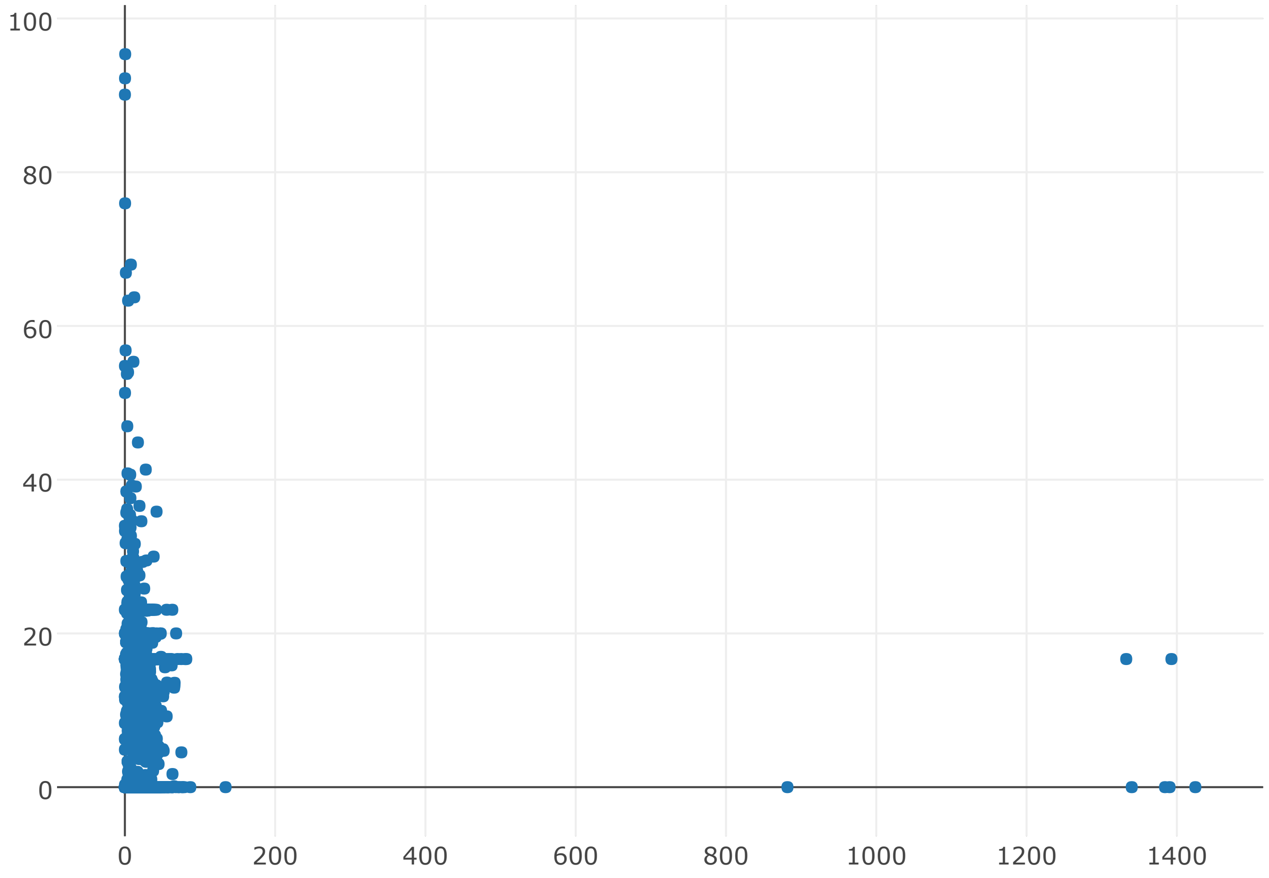




With Trip time in mins:

```
plot_ly(x = train$Trip_time_mins[1:10000], y = train$tip_perc[1:10000], type = "scatter")
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```



There’s a signifanct proof that the independent variables lack linear corelation with the dependent varaible. Therefore, we’ll use tree-based ensembling models.

I’ll be using XGBoost because of the following reasons:

1. It has its implementation in C++ which makes it faster.
2. Boosting arcitecture is suited to this dataset as there does not exist much linear corelation but only weak non-linear corelation.
3. R implementation of XGBoost allows for custom cost function.
4. R implementation of XGBoost allows for watchlist method for validation.

V. Modelling:

v.a Data Prepration

```
library("xgboost")
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:plotly':
##
##     slice
```

```
outcome<-'tip_perc'

predictors<-names(train)[!names(train) %in% c(outcome, 'lpep_pickup_datetime')]

dtrain <- xgb.DMatrix(data=as.matrix(train[,predictors]), label=train[,outcome])

dval <- xgb.DMatrix(data=as.matrix(val[,predictors]), label=val[,outcome])

dtest <- xgb.DMatrix(data=as.matrix(test[,predictors]))
```

v.b Custom cost function:

Cost function: If the end user of this predictive model are the taxi drivers, then we should be using a cost function like Mean Absolute error for thier better understanding as it’s more intutive than cost functions like Root Mean Square error.

```
mae <- function(preds, dtrain) {
  labels <- getinfo(dtrain, "label")
  error<-mean(abs(preds-labels))
  return(list(metric = "MAE", value = error))
}
```

v.c Time based validation:

Validation: We’ll use trainin data (<20 Sep, 2015) for training, validation data (20 - 25 Sep, 2015) for validation and test data (>25 Sep, 2015) for evaluation of the model.

```
watchlist <- list(train=dtrain, validation=dval)
```

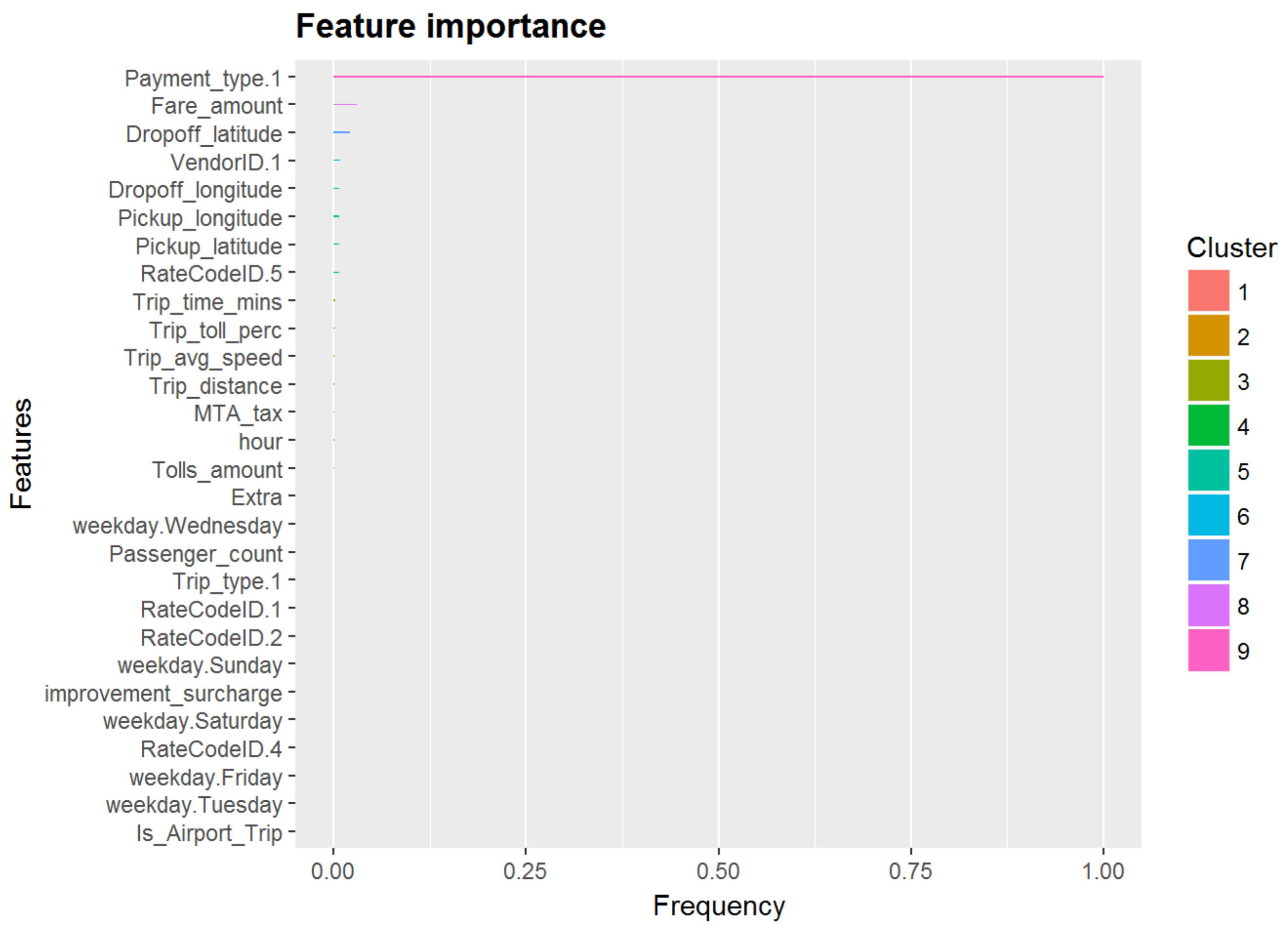
v.d Training:

```
set.seed(1)

param <- list(
  objective = "reg:linear",
  eval_metric = mae,
  eta = 0.1,
  max_depth = 5,
  lambda = 2
)

model <- xgb.train(
  params = param,
  data = dtrain,
  nrounds = 500,
  watchlist = watchlist,
  maximize = FALSE,
  early_stopping_rounds = 20,
  print_every_n = 5
)
```

```
## [1]  train-MAE:6.147661  validation-MAE:6.413619
## Multiple eval metrics are present. Will use validation_MAE for early stopping.
## Will train until validation_MAE hasn't improved in 20 rounds.
##
## [6]  train-MAE:4.411841  validation-MAE:4.564191
## [11] train-MAE:3.503199  validation-MAE:3.593277
## [16] train-MAE:3.044391  validation-MAE:3.102969
## [21] train-MAE:2.800682  validation-MAE:2.842951
## [26] train-MAE:2.664702  validation-MAE:2.698418
## [31] train-MAE:2.586776  validation-MAE:2.615809
## [36] train-MAE:2.540212  validation-MAE:2.566669
## [41] train-MAE:2.511768  validation-MAE:2.537010
## [46] train-MAE:2.493968  validation-MAE:2.518508
## [51] train-MAE:2.482828  validation-MAE:2.506821
## [56] train-MAE:2.479751  validation-MAE:2.503540
## [61] train-MAE:2.476451  validation-MAE:2.500497
## [66] train-MAE:2.474082  validation-MAE:2.498091
## [71] train-MAE:2.473077  validation-MAE:2.497008
## [76] train-MAE:2.473804  validation-MAE:2.498185
## [81] train-MAE:2.473138  validation-MAE:2.497741
## [86] train-MAE:2.472894  validation-MAE:2.497753
## [91] train-MAE:2.472359  validation-MAE:2.497560
## Stopping. Best iteration:
## [71] train-MAE:2.473077  validation-MAE:2.497008
```

v.e Feature Importance:

```
#Feature Importance
gg <- xgb.ggplot.importance(xgb.importance(model = model,feature_names =  predictors), rel_to_fi
rst = TRUE)
gg + ggplot2::ylab("Frequency")
```



```
y_pred<-predict(object = model,newdata = dtest)

y_pred[y_pred<0]<-0

y_pred[y_pred>100]<-100

print(paste("The mean absolute error for the prediction of percentage of total fare given as tip
 is: ", mean(abs(y_pred-test[,outcome]))))
```

```
## [1] "The mean absolute error for the prediction of percentage of total fare given as tip is:
2.50690055456054"
```

Great! So, we are predicting the percentage of total fare given as tip with an accuracy of +- 2.5%.

It looks decent but could be improved by:

1. Creating more geographical features using latitudes and longitude varaibles.
2. Creating more logical features.

3. Using transformation (like using log transformation for right skewed data or square root transformation for left skewed data) of continous independent variables to ensure normal distribution.
4. Using transformation (like using log transformation for right skewed data or square root transformation for left skewed data) of continous dependent caraible to ensure normal distribution.
5. Grid search for oprimizing the parameters.
6. Stacking with varied models to ensure reduce varaince.

However, I’m restricting myself from trying out these due to the time constraint.

Question 5

Option A: Distributions

• *Build a derived variable representing the average speed over the course of a trip.*

Creating the derived varaible representing the average speed over the course of a trip in Km/ hr:

```
data$Trip_time_mins<-as.numeric(difftime(data$Lpep_dropoff_datetime, data$lpep_pickup_datetime,
units = "mins"))

#Replacing 0 mins with 0.1 to escape the divide by zero error

data$Trip_time_mins[data$Trip_time_mins==0]<-mean(data$Trip_time_mins)


data$Trip_avg_speed_miles_min<-data$Trip_distance/data$Trip_time_mins

data$Trip_avg_speed_km_hour<-data$Trip_avg_speed_miles_min * 1.6 / (1/60)

#Putting a maximum cap of 200 on the average trip speed
data$Trip_avg_speed_km_hour[data$Trip_avg_speed_km_hour>200]<-200

#Summary stats of the new derived varaible:
summary(data$Trip_avg_speed_km_hour)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   14.90   18.83   20.88   24.11  200.00
```

• *Can you perform a test to determine if the average trip speeds are materially the same in all weeks of September? If you decide they are not the same, can you form a hypothesis regarding why they differ?*

Creating the week of the month feature:

Assumption: Almost all Pick-ups and drops happen on the same day i.e. assuming no cases where pickup is at like Sun, 23:50:00 and drop off is at Mon, 01:02:20

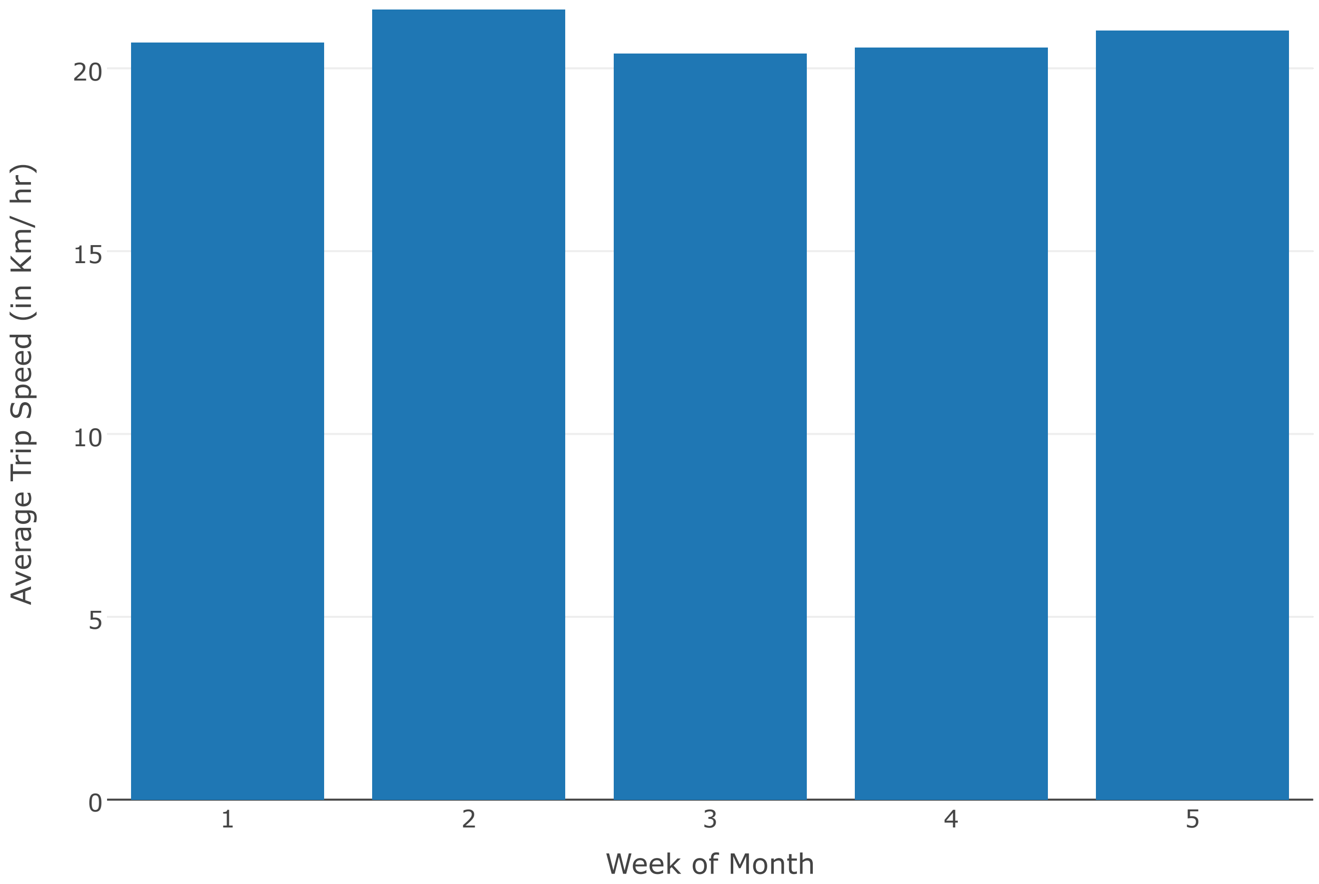
```
data$week_of_year<-as.numeric(week(data$lpep_pickup_datetime))

data$week_of_month<-as.numeric(data$week_of_year-min(data$week_of_year)+1)
```

Plotting average trip speeds for each week in september:

```
avg_speed_df<-as.data.frame(aggregate(data$Trip_avg_speed_km_hour, list(data$week_of_month), mea
n))

colnames(avg_speed_df)<-c("Week","Avg_trip_speed")
```

The distrirution of average trip speeds per week is as:

```
plot_ly(x = avg_speed_df$Week,
        y = avg_speed_df$Avg_trip_speed,
        type = 'bar')%>%
        layout(xaxis = list(title = "Week of Month"),
               yaxis = list(title = "Average Trip Speed (in Km/ hr)"))
```



The vizualization shows that the average trip speed peaks at the second week. Let’s further use anova statistical test for figure out whether the weekly average trip speeds in september 2015 are significantly different or not:

```
fit = lm(formula = data$Trip_avg_speed_km_hour ~ as.factor(data$week_of_month))

anova (fit)
```

```
## Analysis of Variance Table
##
## Response: data$Trip_avg_speed_km_hour
##                                   Df    Sum Sq Mean Sq F value    Pr(>F)
## as.factor(data$week_of_month)       4    309497   77374  500.22 < 2.2e-16
## Residuals                     1492504 230861478     155
##
## as.factor(data$week_of_month) ***
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since p-value < 0.05, we'll reject the null hypothesis H0. Hence, the four means are statistically different. Therefore, we can state that there is significant difference between the weekly average trip speeds in september 2015.

Possible hypothisis for this can be:

1. September 7, 2015 was celebrated as labour's day in USA which leads to long first weekend. This could have meant more than usual number of people in new york going to night-parties on week-1 especially Monday. Since the traffic at night is generally less as compared to the traffic at day. Therefore, it could have contributed to the increase in average trip speeds.
2. September 28, 2015 was celebrated as Native American Day. All the schools are closed on Native American Day. Therefore, lack of school buses could have led to lesser than ussual traffic especially in peak hours. Therefore, contributing towards the second highest peak in the 5th weak.

It would be great to study the weekly average trip speeds of September 2014, September 2013, August 2015 and October 2015 to get better understaning but I'm restricting myself due to the time constraint.

• *Can you build up a hypothesis of average trip speed as a function of time of day?*

Creating the hour of the day (used as a simulation for time of the day) feature:

Assumption: Almost all Pick-ups and drops happen on the same day i.e. assuming no cases where pickup is at like Tue, 23:50:00 and drop off is at Wed, 01:02:20

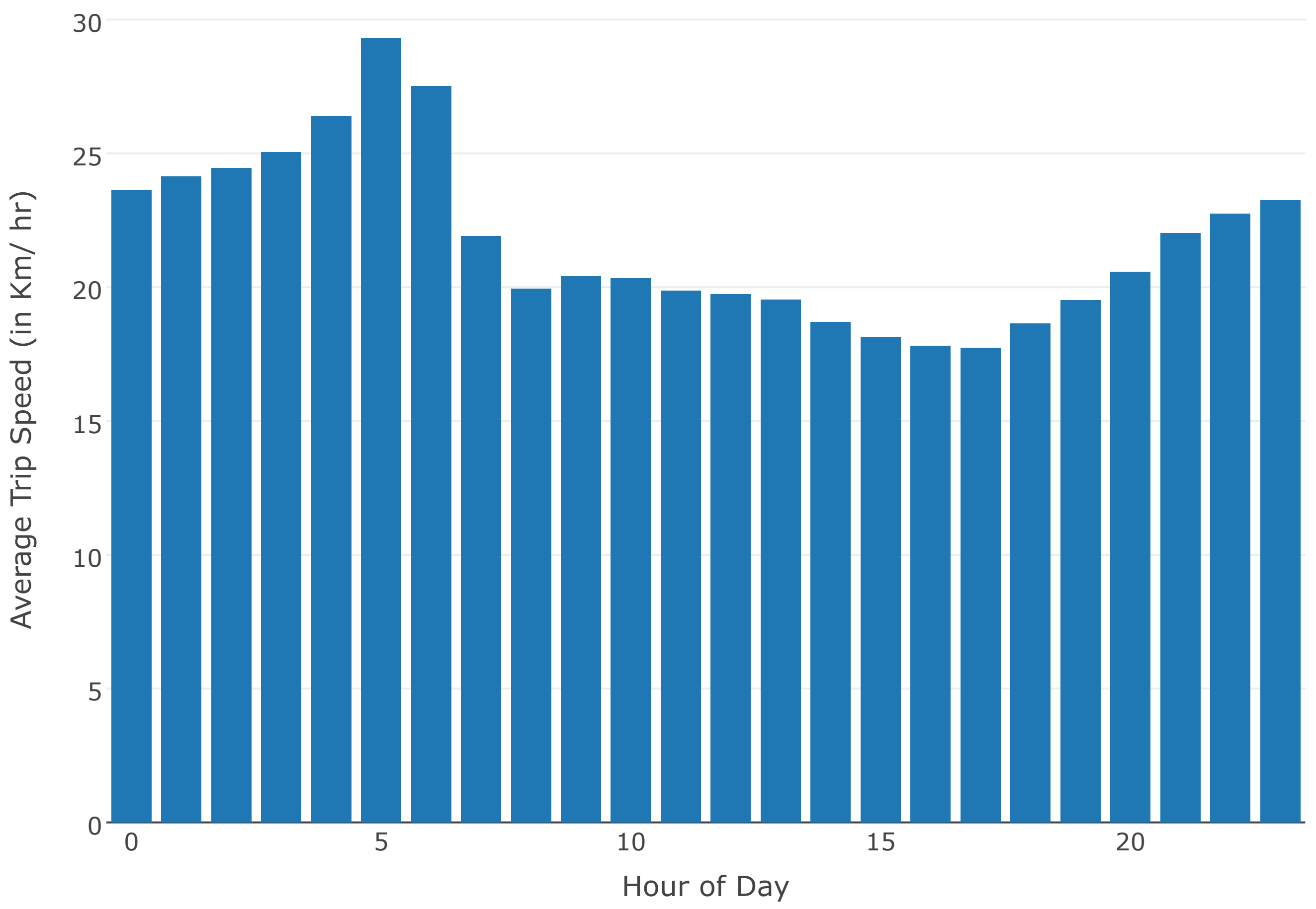
```
data$hour<-hour(data$lpep_pickup_datetime)

mean_df<-as.data.frame(aggregate(data$Trip_avg_speed_km_hour, list(data$hour), mean))

colnames(mean_df)<-c("Hour","Mean_Trip_speed")
```

Plotting average trip speeds for each hour in september:

```
avg_speed_df<-as.data.frame(aggregate(data$Trip_avg_speed_km_hour, list(data$hour), mean))

colnames(avg_speed_df)<-c("Hour","Avg_trip_speed")
```

The distrirution of average trip speeds per hour is as:

```
plot_ly(x = avg_speed_df$Hour,
        y = avg_speed_df$Avg_trip_speed,
        type = 'bar')%>%
        layout(xaxis = list(title = "Hour of Day"),
               yaxis = list(title = "Average Trip Speed (in Km/ hr)"))
```



The vizualization shows that the average trip speed peaks at 5 AM in the morining. Also, the valley is at 5 PM in the evening.

The following are the hypothesis of average trip speed as a function of time of day:

1. The traffic is least at around 5 AM in the morning. Therefore, the taxis have high average trip speeds of taxis at that time.
2. The traffic is at the peak in the evening hours starting from 5 PM, which leads low average trip speeds of taxis at that time.
3. The average trip speeds of taxis drops sharply from 6 AM to 7 AM in morning, suggesting that most of the offices starts early in the morning in USA at around 8 AM. This can also be because of additional traffic created by school busses which opens at around the same time.
4. The evening hours has more no. of low average trip speeds of taxis as compared to the morning hours.This suggests that people could be more likely to use public transport while returning from office as compared to while going to the office, which is understandable as they might not want to get late while trying to reach the office.

5. Also, this difference in average trip speeds of taxis in morning and evening hours can also be because most of the schools opens in mornings and closes in afternoon hours while the offices opens in mornings but closes in evenings. Hence, the traffic is sparse across the afternoon and evening hours while dense across the morning hours.

Lastly, I'll like to thank you for this wonderful opportunity. It was a really interesting real-world dataset which delivered some fairly great insights of how the taxi system and traffic works in New York city. For my codes in various other data science competitions and datasets, please view my GitHub profile and published articles from the links below. Thank you!

GitHub (https://github.com/sauravkaushik8) Articles (https://www.analyticsvidhya.com/blog/author/sauravkaushik8/)