Green Taxi

C941637

2018/11/8

Question 1

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

```
# load the data
green_data<-read.csv('/Users/zhejindong/Downloads/green_tripdata_2015-09.csv')
```

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

```
str(green_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 (
##  $ Lpep_dropoff_datetime: Factor w/ 1077210 levels "2015-09-01 00:00:00",..: 3 6 6 22 5 11 14 12 27 1
##  $ 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 ...
```

There are 1494926 rows and 21 columns.

Question 2

Plot a histogram of the number of the trip distance. trip dis: The elapsed trip distance in miles reported by the taximeter

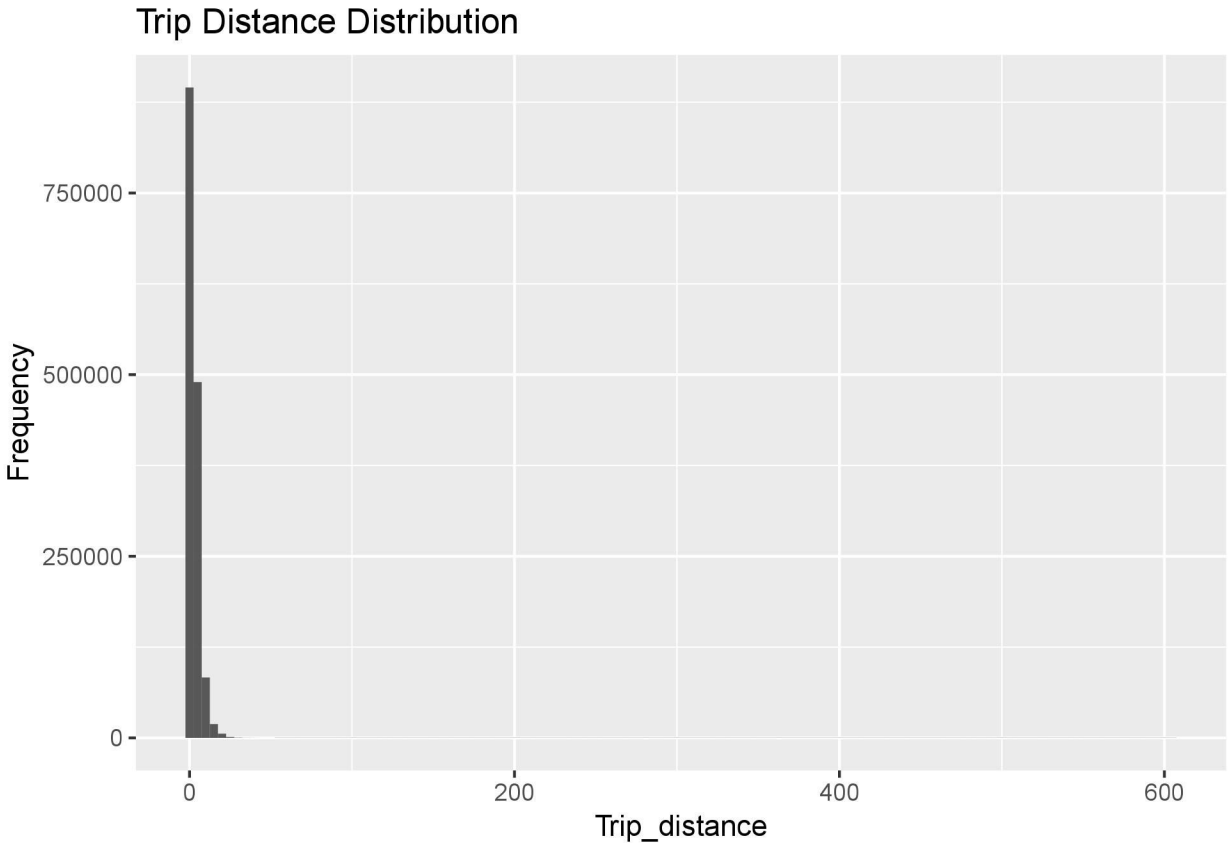
```
library(tidyverse)
```

```
## -- Attaching packages -------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

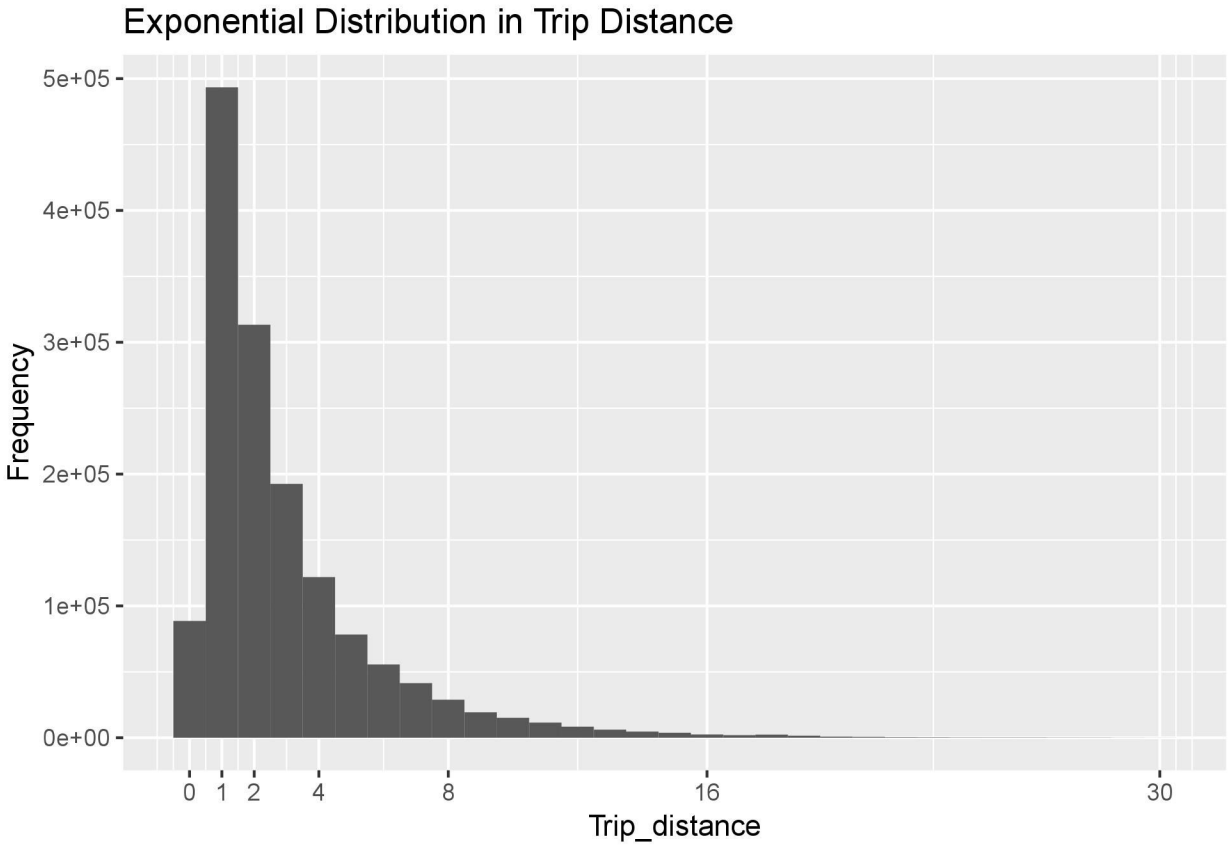
```
## -- Conflicts ----------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
ggplot(green_data, aes(x=Trip_distance))+geom_histogram(binwidth = 5)+ggtitle('Trip Distance Distributi
```



From the graph above, we notice most trip distance is less than 100 miles, except for some outliers. Considering we are concerned with the majority, we can delete the outliers.

```
green_subdata<-subset(green_data,Trip_distance<30)
ggplot(green_subdata, aes(x=Trip_distance))+geom_histogram(binwidth=1)+ggtitle('')+scale_x_continuous(b
```



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

1 The majority of trip distance distributes from 0 to 16 miles.

2 From 1 mile to 16 miles, the frequency exponentially decreases.

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
```

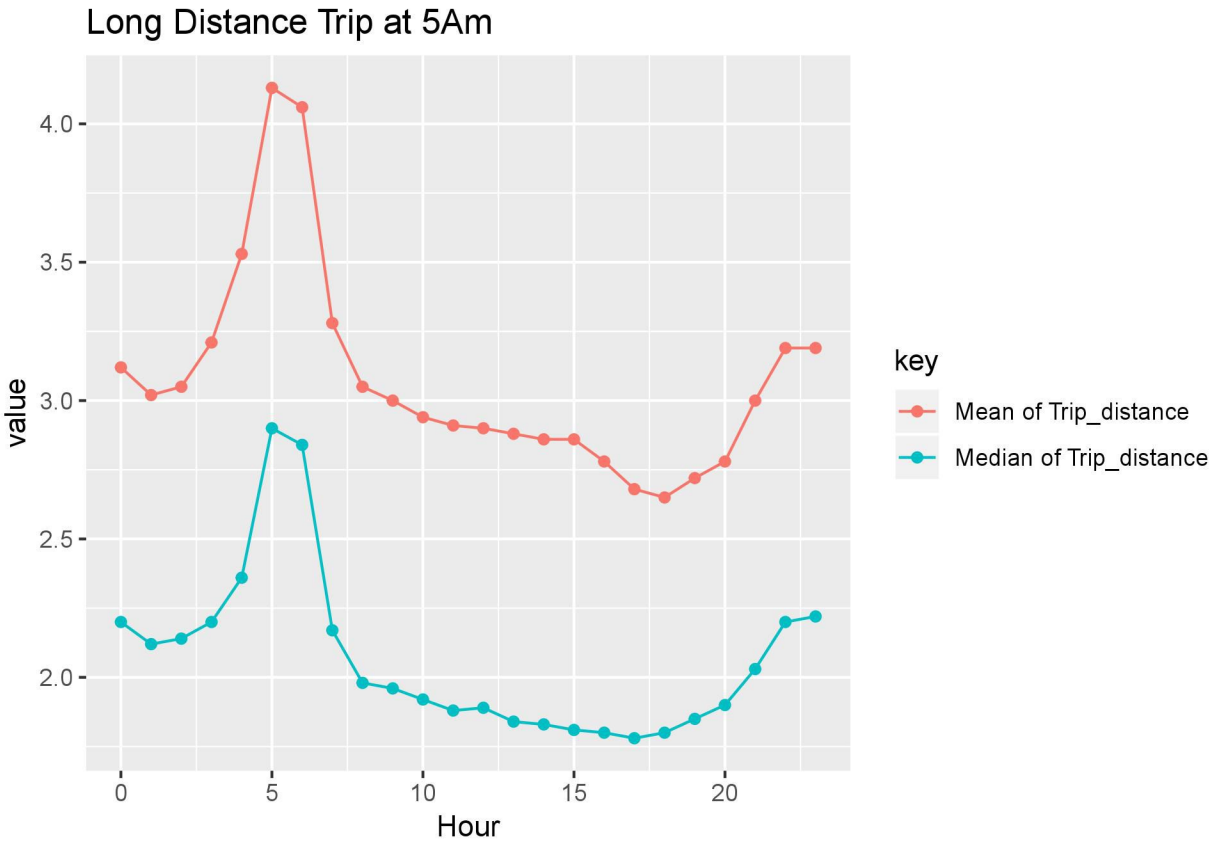
```
library(dplyr)
green_data<-mutate(green_data, Hour=hour(green_data$lpep_pickup_datetime))
```

```
Mean<-setNames(aggregate(green_data$Trip_distance, by=list(green_data$Hour), FUN=mean),c('Hour','Mean o
Mean$'Mean of Trip_distance'<-round(Mean$'Mean of Trip_distance',2)
Median<-setNames(aggregate(green_data$Trip_distance, by=list(green_data$Hour), FUN=median),c('Hour','Me
report<-full_join(Mean, Median,by='Hour')
report
```

```
##    Hour Mean of Trip_distance Median of Trip_distance
```

```
## 1      0                3.12                2.20
## 2      1                3.02                2.12
## 3      2                3.05                2.14
## 4      3                3.21                2.20
## 5      4                3.53                2.36
## 6      5                4.13                2.90
## 7      6                4.06                2.84
## 8      7                3.28                2.17
## 9      8                3.05                1.98
## 10     9                3.00                1.96
## 11    10                2.94                1.92
## 12    11                2.91                1.88
## 13    12                2.90                1.89
## 14    13                2.88                1.84
## 15    14                2.86                1.83
## 16    15                2.86                1.81
## 17    16                2.78                1.80
## 18    17                2.68                1.78
## 19    18                2.65                1.80
## 20    19                2.72                1.85
## 21    20                2.78                1.90
## 22    21                3.00                2.03
## 23    22                3.19                2.20
## 24    23                3.19                2.22
```

```
tidyreport <- report %>% gather(key, value, -Hour)
ggplot(tidyreport,aes(x=Hour,y=value,group=key,color=key))+geom_line()+geom_point()+ggtitle('Long Distar
```



```
# Using longitude and latitude to locate JFK airport.

green_data$jfk_drop_airtport <- ifelse((green_data$Dropoff_latitude < 40.645 & green_data$Dropoff_latit

green_data$jfk_pick_airtport <- ifelse((green_data$Pickup_latitude < 40.645 & green_data$Pickup_latitud

print("New York Airport")
```

```
## [1] "New York Airport"
```

```
Airport<-filter(green_data,(green_data$jfk_drop_airtport==1|jfk_pick_airtport==1|green_data$RateCodeID==
str(Airport$Fare_amount)
```

```
##  num [1:10193] 52 52 22.5 52 -52 52 52 52 42 52 ...
```

```
mean(abs(Airport$Fare_amount))
```

```
## [1] 45.99419
```

```
print("JFK Airport")
```

```
## [1] "JFK Airport"
```

```
JFK<-filter(green_data,(green_data$jfk_drop_airtport==1|jfk_pick_airtport==1|green_data$RateCodeID==3))
str(JFK$Fare_amount)
```

```
##  num [1:10193] 52 52 22.5 52 -52 52 52 52 42 52 ...
```

```
mean(abs(JFK$Fare_amount))
```

```
## [1] 45.99419
```

```
print("Newark Airport")
```

```
## [1] "Newark Airport"
```

```
Newark<-filter(green_data,RateCodeID==3)
str(Newark$Fare_amount)
```

```
##  num [1:1117] 48 72 85.5 62 20 20 20 20 20 80.5 ...
```

```
mean(abs(Newark$Fare_amount))
```

```
## [1] 49.69024
```

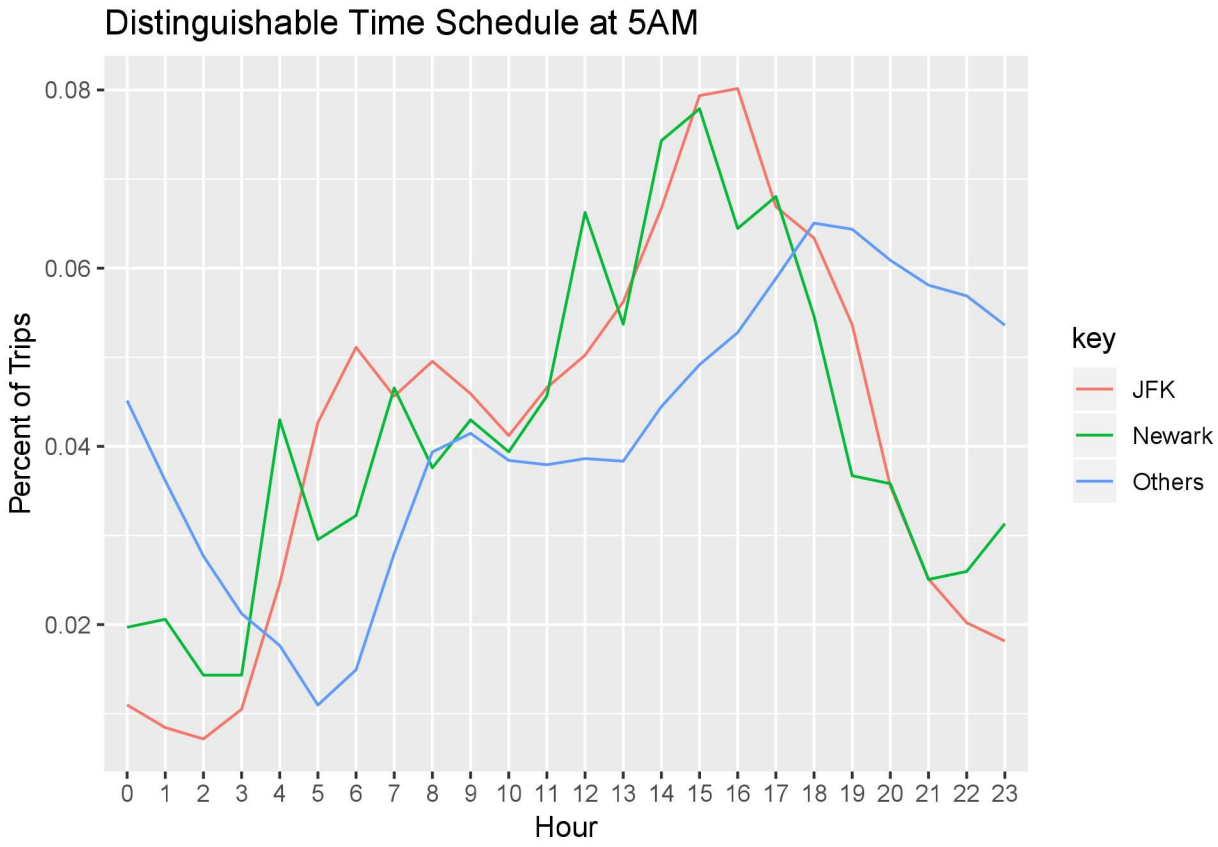
We'd like to get a rough sense of identifying trips that originate or terminate at one of the NYC area airports. Can you provide a count of how many transactions fit this criteria, the average fare, and any other interesting characteristics of these trips?

In September, there are 9077 total transactions that originate or terminate at JFK airport, and 1117 transactions from and to Newark. The average fare amount for each trip to airport is about 45.99419$.

Annotation: I utilized longitude and latitude to approximately locate JFK airport, rather than only RateCodeID. According to the data dictionary, if RateCodeID equals to 'JFK', we can only get trips between Manhattan and JFK, and the fare amount is fixed at 52 dollars, which will underrepresent the total transactions involving JFK. Besides, the negative fare amount in data needed to be transformed before calculating the average fare.

```
library(tidyverse)

None_Air<-filter(green_data,RateCodeID!=2 & jfk_drop_airtport!=1&jfk_pick_airtport!=1)
None_Air<-data.frame(table(None_Air$Hour))
```

```r
names(None_Air)<-c("Hour","Others")
None_Air$Others<-None_Air$Others/sum(None_Air$Others)

JFK2<-data.frame(table(JFK$Hour))
names(JFK2)<-c("Hour","JFK")
JFK2$JFK<-JFK2$JFK/sum(JFK2$JFK)



Newark2<-data.frame(table(Newark$Hour))
names(Newark2)<-c("Hour","Newark")
Newark2$Newark<-Newark2$Newark/sum(Newark2$Newark)

A1<-full_join(None_Air,JFK2,by='Hour')
Time<-full_join(A1,Newark2,by='Hour')


tidyTime <- Time %>% gather(key, value, -Hour)

ggplot(tidyTime, aes(x=Hour, y=value, group=key,color=key))+geom_line()+ylab('Percent of Trips')+ggtitl
```

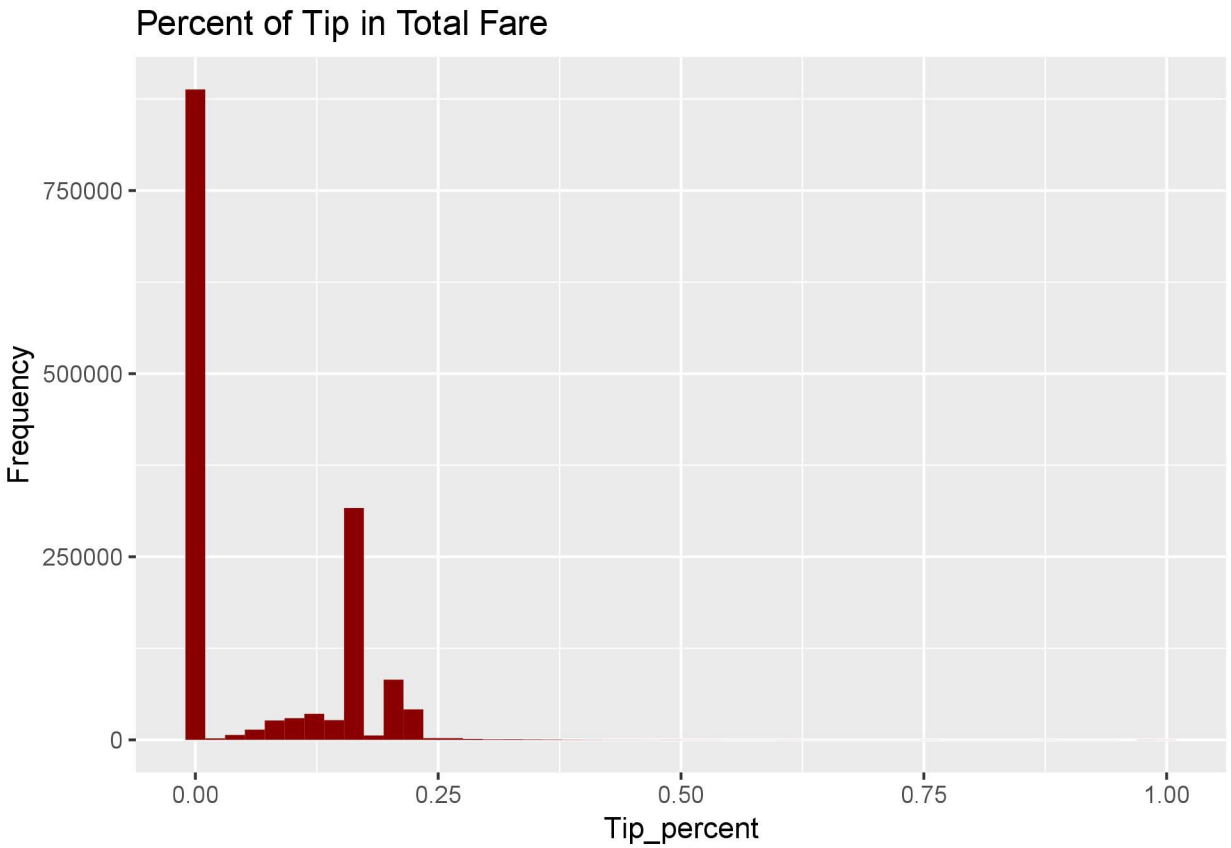


The picture above shows three time schedules for JFK, Newark, and others. In comparison with other type of trips, trips originate or terminate at airports have apparently opposite time sechedule from 0AM to 5AM. A suggestion for taxi driver: more consumers at 5AM at NYC airport.

Question 4

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

```
green_data1<-subset(green_data,green_data$Tip_amount>=0&green_data$Total_amount>0) #remove data with ne
green_data1$Tip_percent<-green_data1$Tip_amount/green_data1$Total_amount
ggplot(green_data1,aes(x=Tip_percent))+geom_histogram(fill='darkred',bins=50)+ggtitle("Percent of Tip i
```



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). Provide an estimate of performance using an appropriate sample, and show your work.

Data Preprocessing

```
green_data1$lpep_pickup_datetime<-as.Date(green_data1$lpep_pickup_datetime)
green_data1$Weekday<-wday(green_data1$lpep_pickup_datetime) # Extract the weekday of time
x<-select(green_data1,Tip_percent,RateCodeID,Pickup_longitude,Pickup_latitude,Dropoff_longitude,Dropoff
```

```
x<-scale(x, scale = TRUE) #scale data
smp_size <- floor(0.8* nrow(x))
set.seed(3)
train_ind <- sample(seq_len(nrow(x)), size = smp_size) # split testing dataset and training dataset
test_x <- x[-train_ind, ][,2:18]
test_y<-x[-train_ind, ][,1]
train_data<- x[train_ind,]
```

Data Preprocessing module finishes there tasks:

1 Remove data with negative fare amount.

2 Extract weekday of time as new feature.

3 Scale data.

4 Split training data and testing data with ratio 8:2

Using Gradient Boosting Machine to predict tip amount

```
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:lubridate':
##
##     day, hour, month, week, year
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc, ||
```

```
h2o.init(
  nthreads=-1,
  max_mem_size = "8G")
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 hours 16 minutes
##     H2O cluster timezone:       America/New_York
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.20.0.8
##     H2O cluster version age:    1 month and 23 days
##     H2O cluster name:           H2O_started_from_R_zhejindong_tfd562
##     H2O cluster total nodes:    1
```

```
##     H2O cluster total memory:   6.60 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.5.1 (2018-07-02)
```

```
h2o.removeAll()
```

```
## [1] 0
```

```
gbm <- h2o.gbm(training_frame = as.h2o(train_data),x=2:18,y=1,
               ntrees = 100, learn_rate = 0.2, max_depth = 8)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |=                                                                     |   1%
  |
  |=                                                                     |   2%
  |
  |==                                                                    |   3%
  |
  |===                                                                   |   4%
  |
  |===                                                                   |   5%
  |
  |====                                                                  |   6%
  |
  |=====                                                                 |   7%
  |
  |=====                                                                 |   8%
  |
  |======                                                                |   9%
  |
  |======                                                                |  10%
  |
  |=======                                                               |  11%
  |
  |========                                                              |  12%
  |
  |========                                                              |  13%
  |
  |=========                                                             |  14%
  |
```

```
## ==============
##
## H2ORegressionModel: gbm
## Model Key:  GBM_model_R_1542166921737_4
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1             100                     100              223445         8
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1         8    8.00000         17        232   173.01000
##
## H2ORegressionMetrics: gbm
## ** Reported on training data. **
##
## MSE:  0.002126677
## RMSE:  0.04611591
## MAE:  0.02099863
## RMSLE:  NaN
## Mean Residual Deviance :  0.002126677
##
##
##
##
##
## Scoring History:
##             timestamp   duration number_of_trees training_rmse
## 1 2018-11-13 23:59:35  0.001 sec               0       1.00087
## 2 2018-11-13 23:59:36  1.127 sec               1       0.85160
## 3 2018-11-13 23:59:37  2.121 sec               2       0.73719
## 4 2018-11-13 23:59:38  3.099 sec               3       0.64896
## 5 2018-11-13 23:59:39  3.926 sec               4       0.57447
##   training_mae training_deviance
## 1      0.90081           1.00174
## 2      0.75704           0.72523
## 3      0.64302           0.54344
## 4      0.55221           0.42115
## 5      0.47499           0.33002
##
## ---
##              timestamp          duration number_of_trees training_rmse
## 22 2018-11-14 00:00:55  1 min 19.383 sec              77       0.05445
## 23 2018-11-14 00:00:59  1 min 23.436 sec              81       0.05235
## 24 2018-11-14 00:01:03  1 min 27.881 sec              86       0.05037
## 25 2018-11-14 00:01:07  1 min 32.264 sec              91       0.04847
## 26 2018-11-14 00:01:12  1 min 36.932 sec              96       0.04708
## 27 2018-11-14 00:01:16  1 min 40.728 sec             100       0.04612
##    training_mae training_deviance
## 22      0.02531           0.00297
## 23      0.02423           0.00274
## 24      0.02318           0.00254
## 25      0.02221           0.00235
## 26      0.02147           0.00222
## 27      0.02100           0.00213
##
## Variable Importances: (Extract with `h2o.varimp`)
```

```
## ============================================
##
## Variable Importances:
##                 variable relative_importance scaled_importance percentage
## 1           Payment_type      2112318.500000          1.000000   0.638891
## 2           Total_amount       560080.000000          0.265149   0.169401
## 3            Fare_amount       528306.437500          0.250107   0.159791
## 4                  Extra        40760.378906          0.019297   0.012328
## 5          Trip_distance        17374.460938          0.008225   0.005255
## 6             RateCodeID        16578.824219          0.007849   0.005014
## 7           Tolls_amount        15786.778320          0.007474   0.004775
## 8       Dropoff_latitude         6599.124023          0.003124   0.001996
## 9                MTA_tax         4078.112061          0.001931   0.001233
## 10     Dropoff_longitude         2501.654541          0.001184   0.000757
## 11 improvement_surcharge          928.950684          0.000440   0.000281
## 12             Trip_type          414.390411          0.000196   0.000125
## 13                  Hour          247.666931          0.000117   0.000075
## 14       Pickup_latitude           98.145477          0.000046   0.000030
## 15               Weekday           85.018181          0.000040   0.000026
## 16       Passenger_count           43.763599          0.000021   0.000013
## 17      Pickup_longitude           26.081717          0.000012   0.000008
```

```
gbm_model_predictions <- predict(gbm,as.h2o(test_x))
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

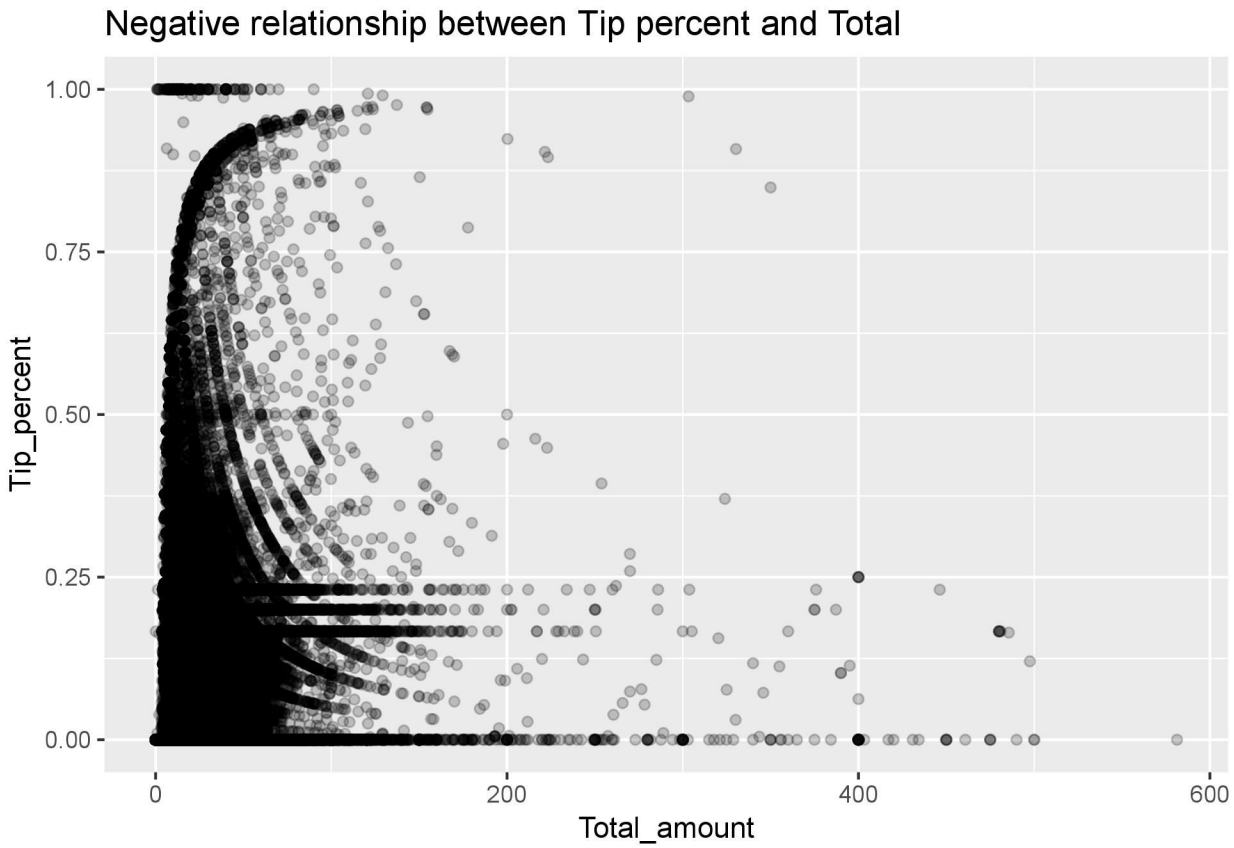
```
gbm_model_residue<-mean(abs((as.data.frame(gbm_model_predictions)-as.data.frame(test_y))$predict))
gbm_model_residue  #0.02167418
```

```
## [1] 0.02168183
```

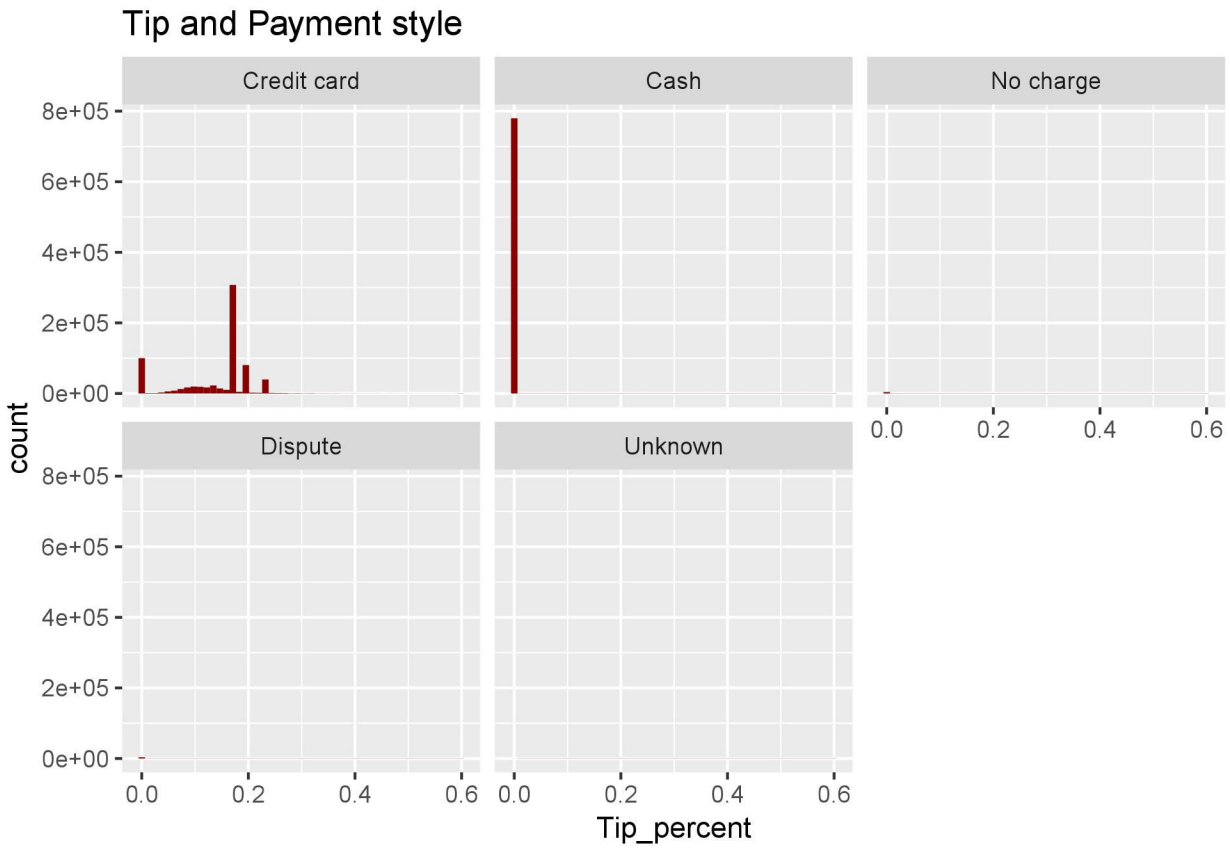
I chose Gradient Boosted Regression Trees because of its three obvious advantages: natural handling of heterogeneous features, predictive power and robustness to outliers in output space. The mean residual is 0.02167418, which performs much better than linear regression model.

Besides, after analyzing the feature importance, I notice variable 'Total_amount', 'Payment_type' and 'Fare_amount' are three determining factors for tip percent prediction. We can visualize these three factors to explore the relationship between them and tip amount.

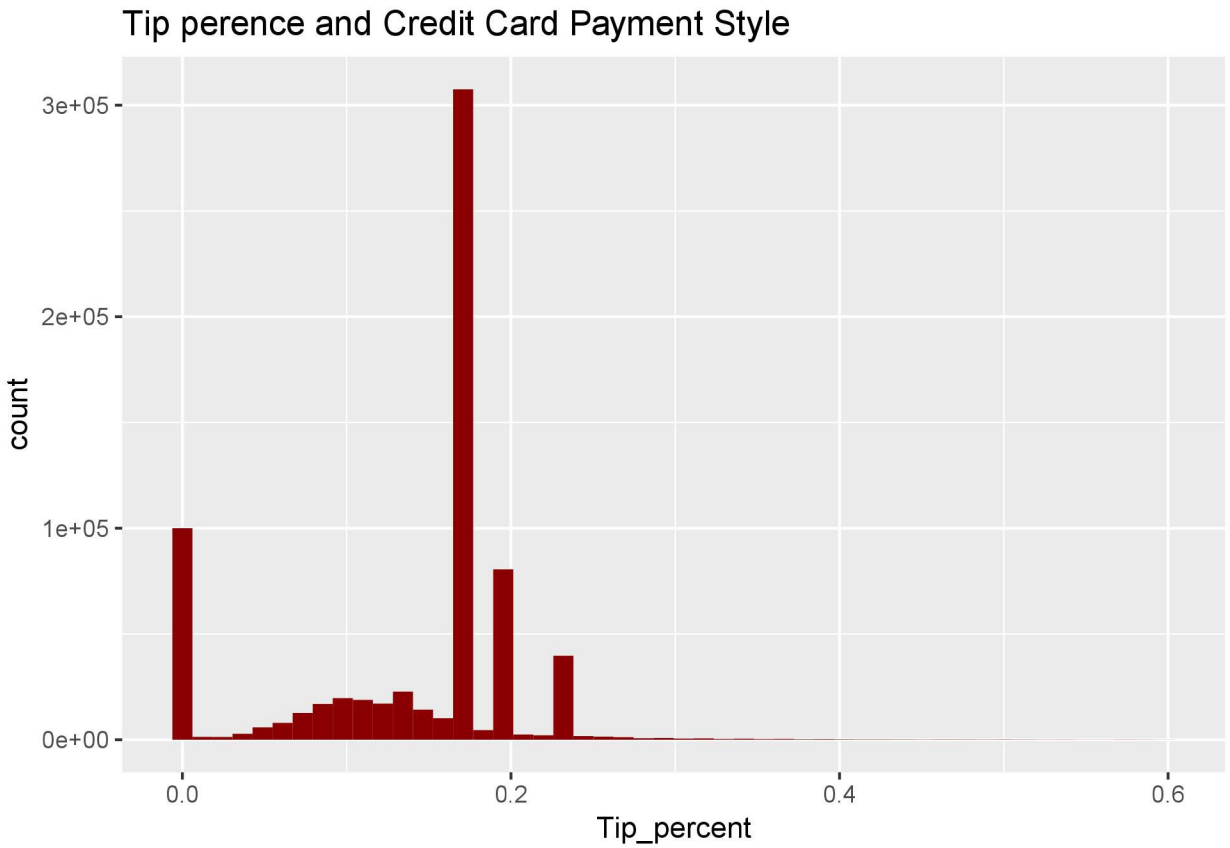
```
ggplot(green_data1,aes(x=Total_amount,y=Tip_percent))+geom_point(alpha=0.2)+ggtitle("Negative relationsh
```




```
x3<-filter(green_data1,Tip_percent<0.6 &Tip_amount>=0)
x3$Payment_type<-factor(x3$Payment_type,level=c(1,2,3,4,5),
labels =c('Credit card','Cash','No charge','Dispute','Unknown'))
ggplot(x3,aes(x=Tip_percent))+geom_histogram(fill='darkred',bins=50)+facet_wrap(~Payment_type)+ggtitle(
```

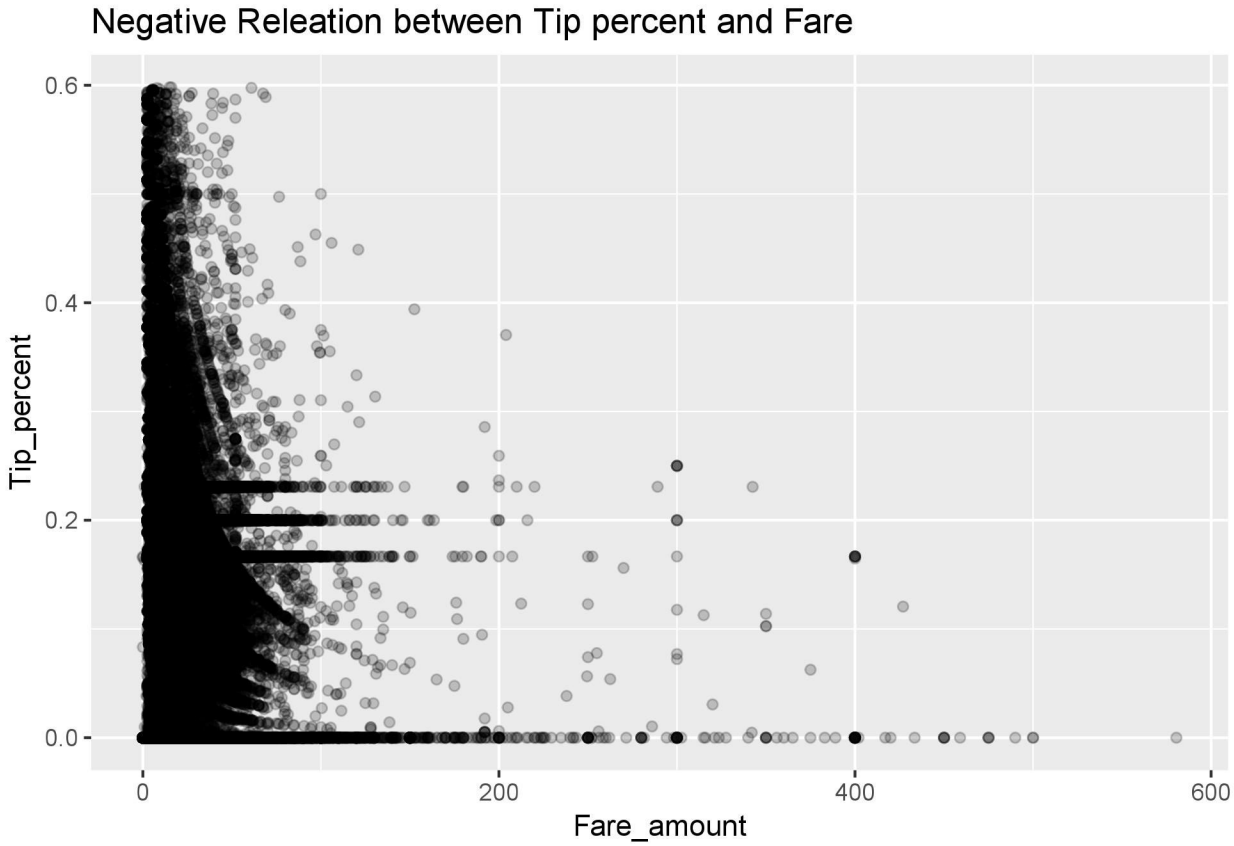



```
x4<-filter(green_data1,Tip_percent<0.6 &Tip_amount>=0&Payment_type==1)
ggplot(x4,aes(x=Tip_percent))+geom_histogram(fill='darkred',bins=50)+ggtitle("Tip perence and Credit Ca
```



The graph above shows that customers paying fare with credit card more possible to give tip than cash. Besides, we also notice for credit card payment, the majority of people tend to give tip as 18% of the total fare.

```
ggplot(x3,aes(x=Fare_amount,y=Tip_percent))+ggtitle("Negative Releation between Tip percent and Fare")+
```

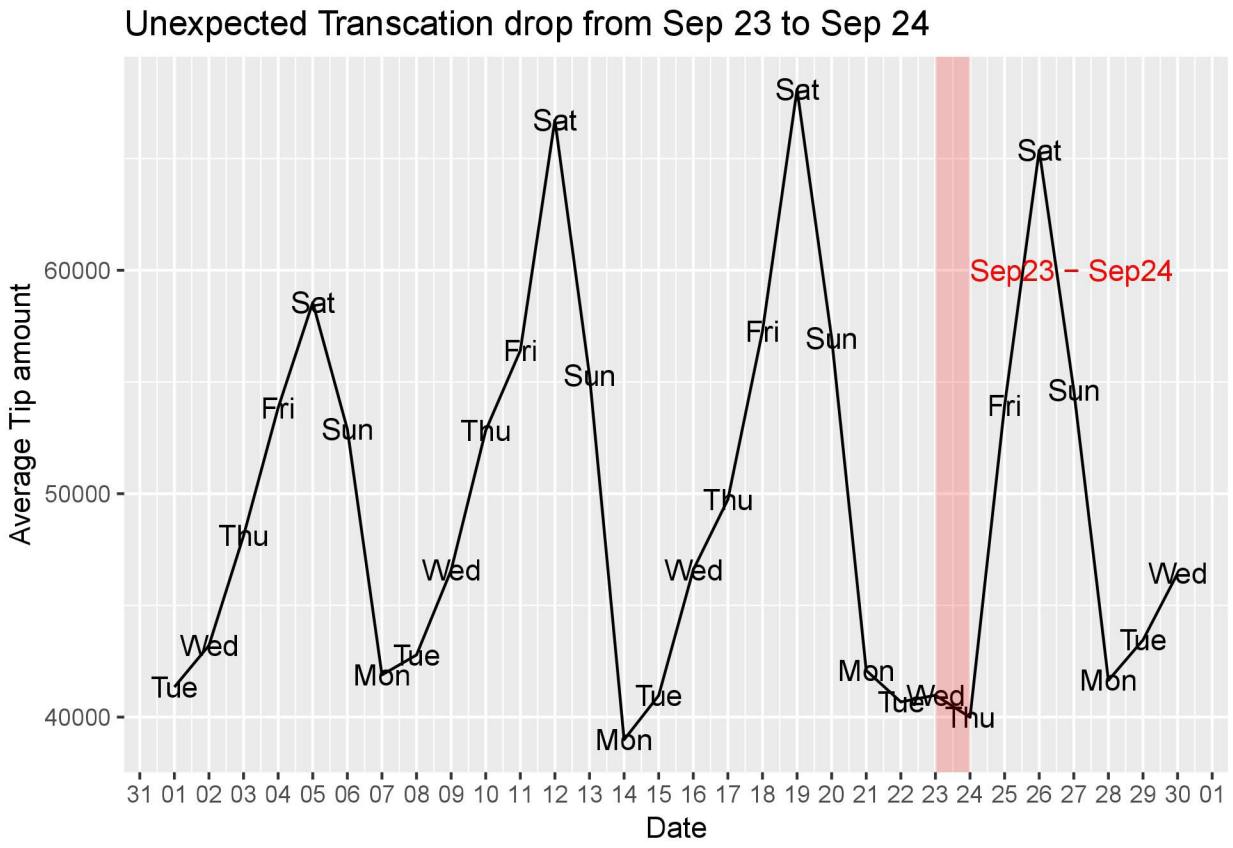


Question 5

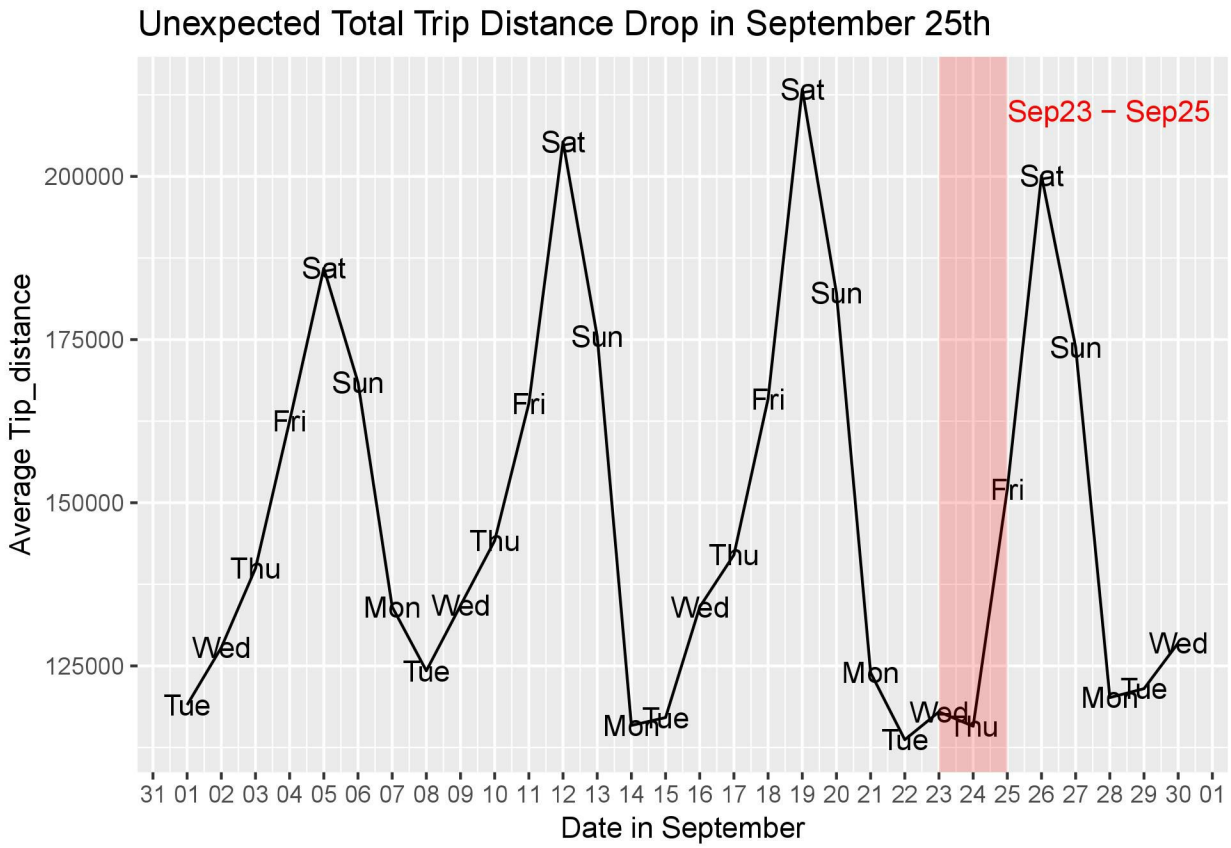
Anomaly Detection

```
# Daily transcations in September

Time2<-green_data1$lpep_pickup_datetime
Time2<-data.frame(table(Time2))
```

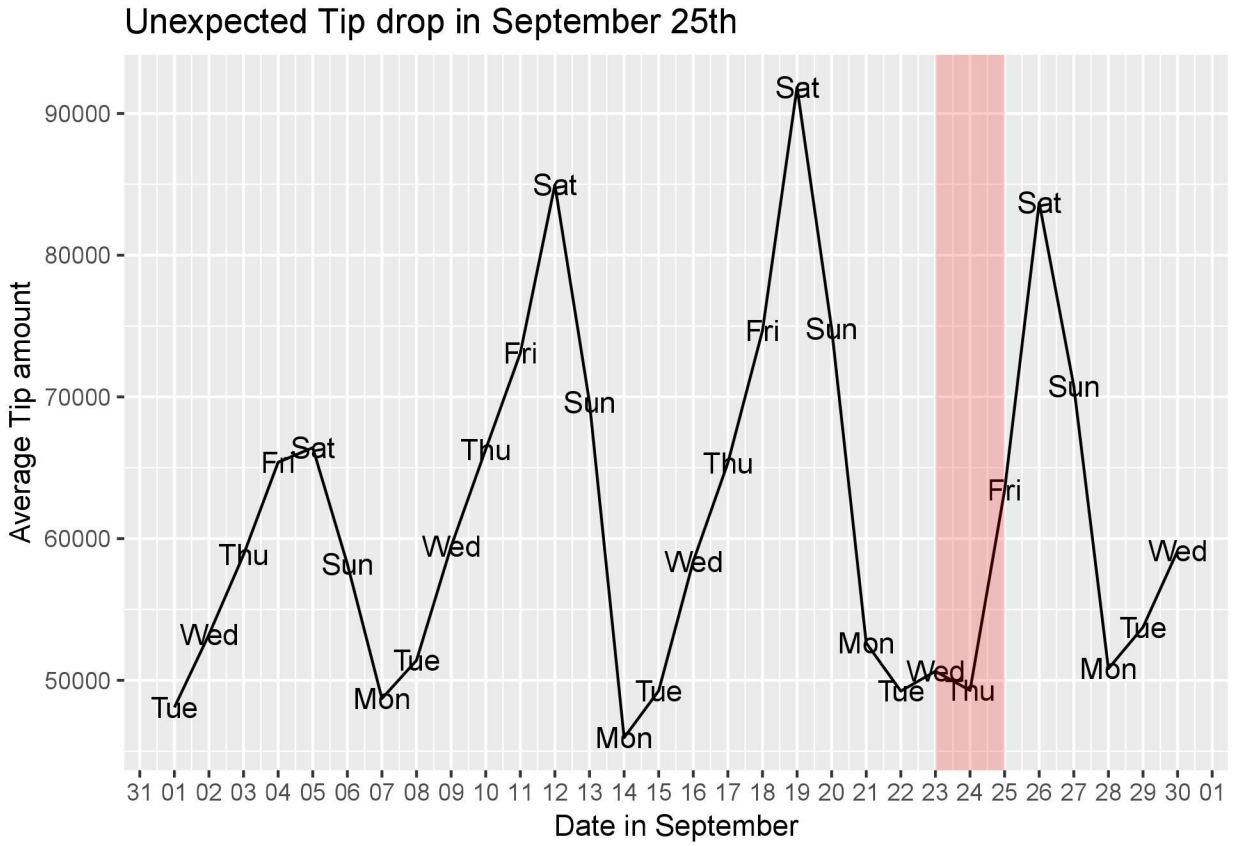
```
start <- as.Date("2015-09-23")
end <- as.Date("2015-09-24")
names(Time2)<-c('Date','Transactions')
Time2$Date<-as.Date(Time2$Date)
ggplot(Time2,aes(x=Time2$Date,y=Transactions,group=1))+geom_line()+geom_text(aes(label = wday(Time2$Dat
             ymin = -Inf, ymax = Inf, fill = "red",
             alpha = .2) +
    annotate("text", x = end,
             y =  60000, label = "Sep23 - Sep24",
             color = "red", hjust = 0)
```




```
start <- as.Date("2015-09-23")
end <- as.Date("2015-09-25")
green_data2<-filter(green_data1,Trip_distance>0)

Time3<-green_data2 %>% group_by(lpep_pickup_datetime) %>% summarise_at(vars(Trip_distance),sum)
ggplot(Time3,aes(x=Time3$lpep_pickup_datetime,y=Trip_distance))+geom_line()+geom_text(aes(label = wday('
              ymin = -Inf, ymax = Inf, fill = "red",
              alpha = .2) +
    annotate("text", x = end,
              y =  210000, label = "Sep23 - Sep25",
              color = "red", hjust = 0)
```




```
# Time analysis
green_data3<-filter(green_data1,Tip_amount>0)
Time<-green_data3 %>% group_by(lpep_pickup_datetime) %>% summarise_at(vars(Tip_amount),sum)
#aggregate(Green_data$Tip_amount,data=green_data,Fun="mean")
ggplot(Time,aes(x=Time$lpep_pickup_datetime,y=Tip_amount))+geom_line()+geom_text(aes(label = wday(Time$
             ymin = -Inf, ymax = Inf, fill = "red",
             alpha = .2)+scale_x_date(date_labels = "%d", date_breaks = '1 day')
```



The pictures above demonstrate the fluctuation of total daily transactions, total daily traveling distance, and total daily tip in September, we notice there is a natural weekly circle in Taxi trips, however we found from 09-23 to 09-25 there is a obvious drop. The transactions, traveling distance and tip are all negatively influenced during this period.

The reason for this anomaly was possibly because of Pope Francis' NYC Visit on 2015-09-25. According to NBC News, dozens of streets were closed from Thursday through Saturday for the pontiff???s visit. Here is the link for the news: https://www.nbcnewyork.com/news/local/Pope-Francis-Visit-Traffic-Transit-Parks-Changes-Delivery-What-To-Know-328675561.html

More ideas about the project

I would like to make a traffic animation of NYC green taxi, but I was still learning how to use Preprocessing and Transitflow.py to realize it. I plan to present a interactive daily traffic flow of green taxis.

Besides, I think the longitude and latitude should be taken advantage of to present more information, for example finding the regions with heaviest taxi traffic.