Final Report

Hukai Luo

17 November 2018

1 Introduction

The goal of this project is to predict the *duration of taxi rides in NYC* based on features like trip coordinates or pickup date and time.

Since there are lots of parameters when we are estimating the trip duration, we will first study and visualise the original data, engineer new features, and examine potential outliers. Finally, we will choose some important parameters to curve fit the data using the least square method.

First of all, let's load the given data

```
library('tibble')
library('data.table')
train <- as.tibble(fread("/Users/luohukai/Documents/GitHub/final-project-hul17011/train.csv"))
test <- as.tibble(fread("/Users/luohukai/Documents/GitHub/final-project-hul17011/test.csv"))
sample <- as.tibble(fread("/Users/luohukai/Documents/GitHub/final-project-hul17011/sample.csv"))
```

Then find the structure of the data

```
library('dplyr')
summary(train)
combine <- bind_rows(train %>% mutate(dset = "train"),
                     test %>% mutate(dset = "test",
                                     dropoff_datetime = NA,
                                     trip_duration = NA))
combine <- combine %>% mutate(dset = factor(dset))
```

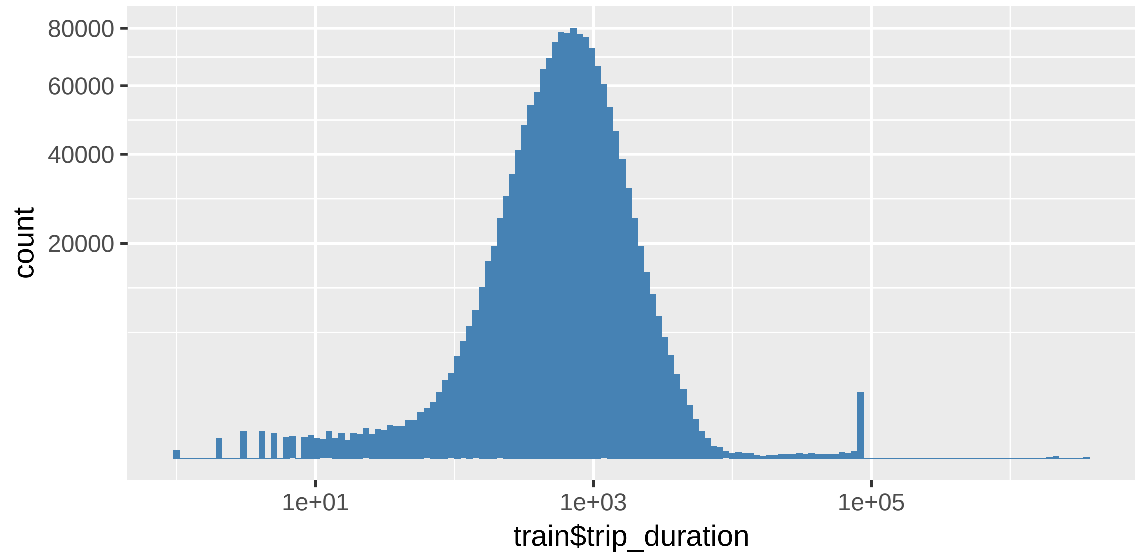
We find the data contains several factors: **vender_id** takes only 1 or 2 which represents two taxi companies; *pickup_datetime*; *dropoff_datetime*; *passenger_count*; *pickup_longitude*; *pickup_latitude*; *dropoff_longitude*; *dropoff_latitude*; *store_and_fwd_flag*; *trip_duration* which is measured in seconds.
In order to make the data easy to use, we will make some change to the data.

```
library('lubridate')
train <- train %>%
  mutate(pickup_datetime = ymd_hms(pickup_datetime),
         dropoff_datetime = ymd_hms(dropoff_datetime),
         vendor_id = factor(vendor_id),
         passenger_count = factor(passenger_count))
```

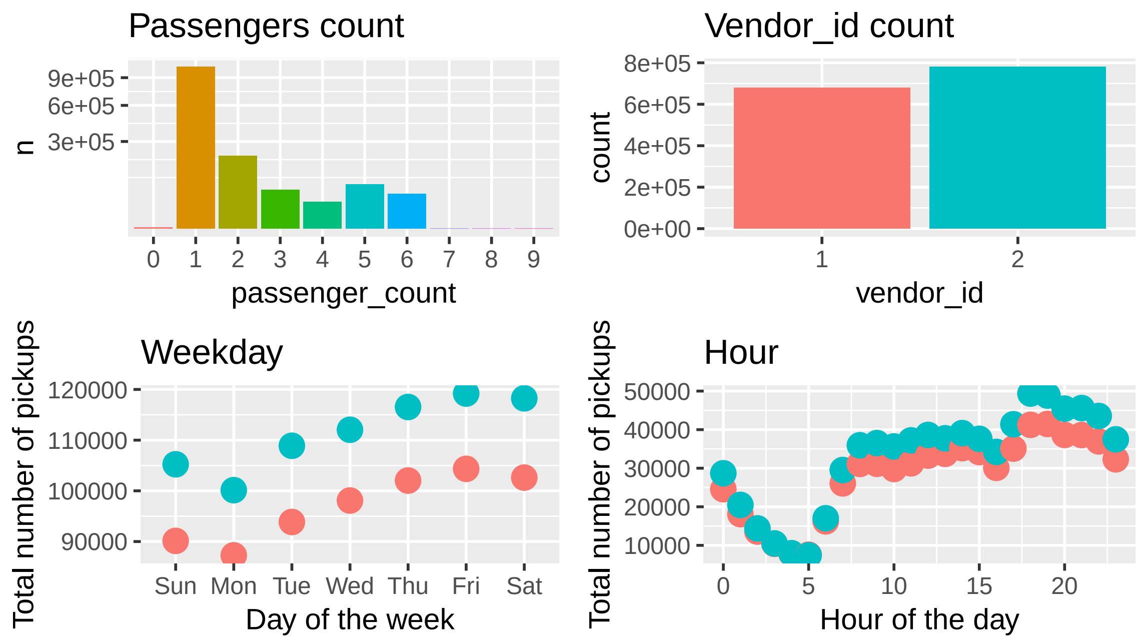
2 Plot by single parameter

Now in order for us to get a better understanding of the data, we will begin by having a look at the distributions of the individual data features. First of all, let's plot the target feature trip_duration.

```r
library('ggplot2')
p1 <- ggplot(train, aes(train$trip_duration)) +
  geom_histogram(fill = "steelblue", bins = 150) +
  scale_x_log10() +
  scale_y_sqrt()
p1
```



Comments: Most trips will ends in nearly 1000 seconds, but there will also be some exceptions.
Then we can also plot the distribution of passenger_count,Vendor_id,day of the week, hour of the day



Comments: Most trips only have 1 passenger; Thursday,Friday and Satuaday are the most busy days; there is a strong dip during the early morning hours and another dip around 4pm.

3 Relations

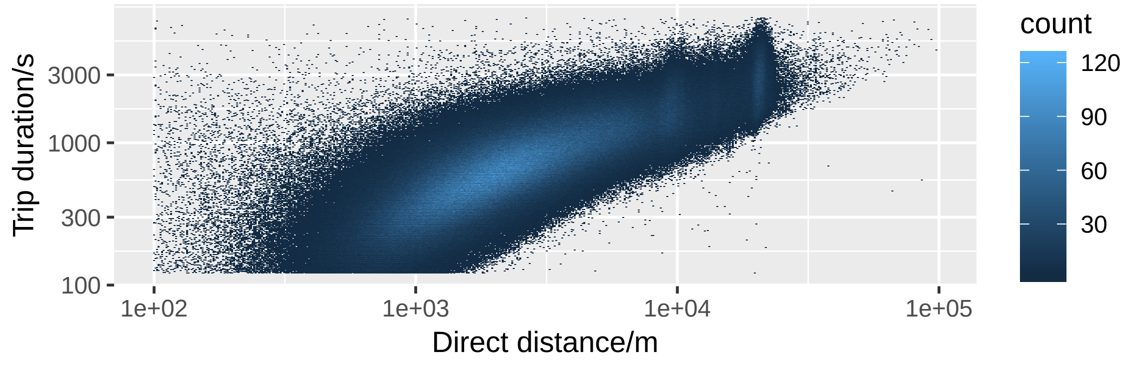
While the previous section looked primarily at the distributions of the individual features, here we will examine in more detail how those features are related to each other and to our target trip_duration. In this project, we will assume that the trip_duration is only related to **Trip distance**, **passenger numbers**, **vender_id**, **day of the week**, **hour of the day**.

3.1 Trip distance vs trip_duration

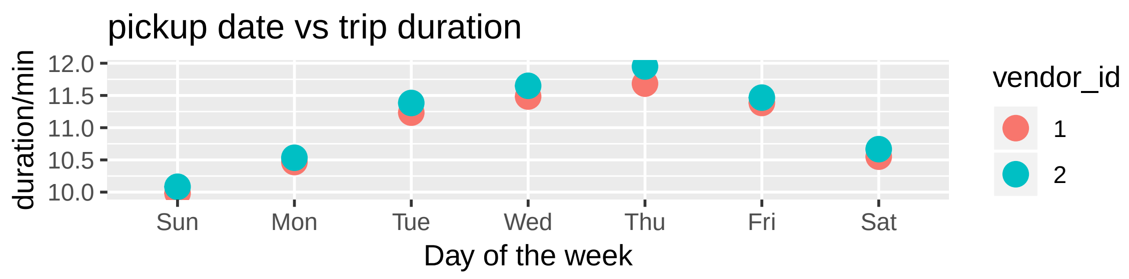
First, we need to calculate the exact trip distance by the pickup and dropoff location.

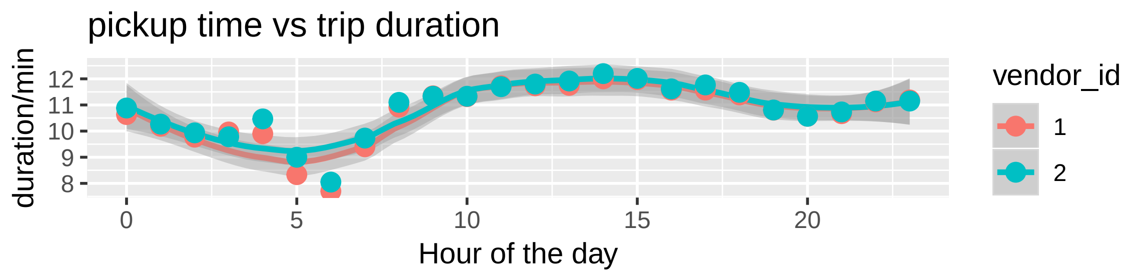
```r
library('geosphere')
pick_coord <- train %>%
  select(pickup_longitude, pickup_latitude)
drop_coord <- train %>%
  select(dropoff_longitude, dropoff_latitude)
train$dist <- distCosine(pick_coord, drop_coord)
```

Then plot the Trip distance vs trip_duration distribution.

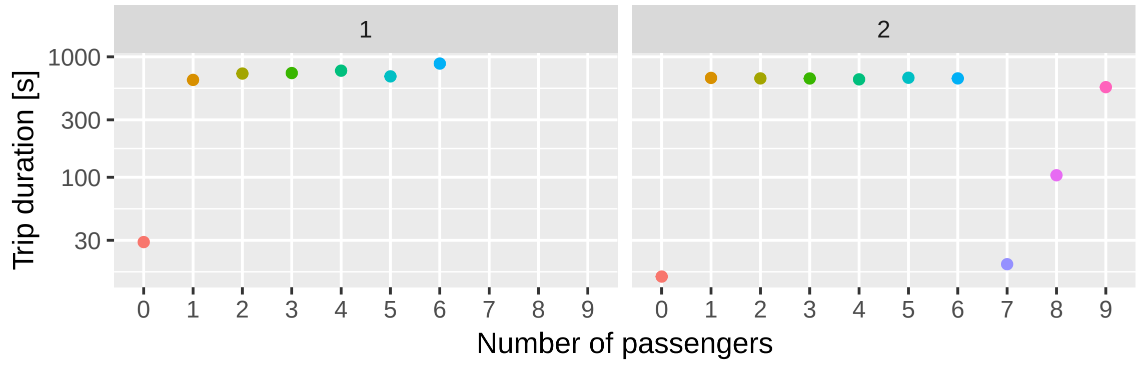


3.2 Pickup date/time vs trip_duration





3.3 Passenger number vs trip_duration



From this plot, we can find the trip_duration doesn't have a strong relationship with the passenger numbers, the difference in the picture may only reveal the impact of different distance. For those passenger numbers = 7, 8, 9, we don't think that they are reasonable, so we just won't consider them.

4 Prediction

In this project, we assume the trip_duration distribution function has three parameters: trip_distance d, pickup date w, pickup time t.

First of all, let's calculate the average driving speed V, let's delete some abnormal data.

```
train$speed <- train$dist/train$trip_duration*3.6
train$wday <- wday(train$pickup_datetime, label = FALSE)
train$hour <- hour(train$pickup_datetime)
train <- train %>%
  filter(trip_duration < 7600 & trip_duration > 40) %>% #delete abnormal trip_duration time: T>2hours a
  filter(dist > 100 & dist < 100e3) %>%                 #delete abnormal distance: d>100km and d<100m
  filter(speed < 100 & speed > 1)                       #deleta speed which is too fast or too slow
average_speed <- mean(train$speed)
average_speed
```

```
## [1] 14.52126
```

Now, we get the average speed $\bar{V} = 14.52126$ we assume that the trip_duration distribution density has this format below:

$$T = N[\frac{Distance}{\bar{V}}, \sigma^2] * Date[wday] * Hour[hour]$$

```
wdaydata <- train %>%                                     # get pickup date median duration
  mutate(wday = wday(pickup_datetime, label = FALSE)) %>%
  group_by(wday) %>%
  summarise(median_duration = median(trip_duration))
wdaydata
```

```
## # A tibble: 7 x 2
##    wday median_duration
##   <dbl>           <dbl>
## 1     1             604
## 2     2              632
## 3     3              681
## 4     4              696
## 5     5              712
## 6     6              688
## 7     7              638
```

```
hpickdata <- train  %>%                                  # get pickup hour median duration
  mutate(hour = hour(pickup_datetime)) %>%
  group_by(hour) %>%
  summarise(median_duration = median(trip_duration))
hpickdata
```

```
## # A tibble: 24 x 2
##     hour median_duration
##    <int>           <dbl>
##  1     0             649
##  2     1             616
##  3     2             594
##  4     3             597
##  5     4             617
##  6     5             525
##  7     6             476
##  8     7             576
##  9     8             662
## 10     9             682
## # ... with 14 more rows
```

Then, define the DATE and HOUR function below, then use them to generate the duration function:

```
DATE <- function(x){wdaydata$median_duration[x]/mean(wdaydata$median_duration)}
HOUR <- function(x){hpickdata$median_duration[x+1]/mean(hpickdata$median_duration)}
```

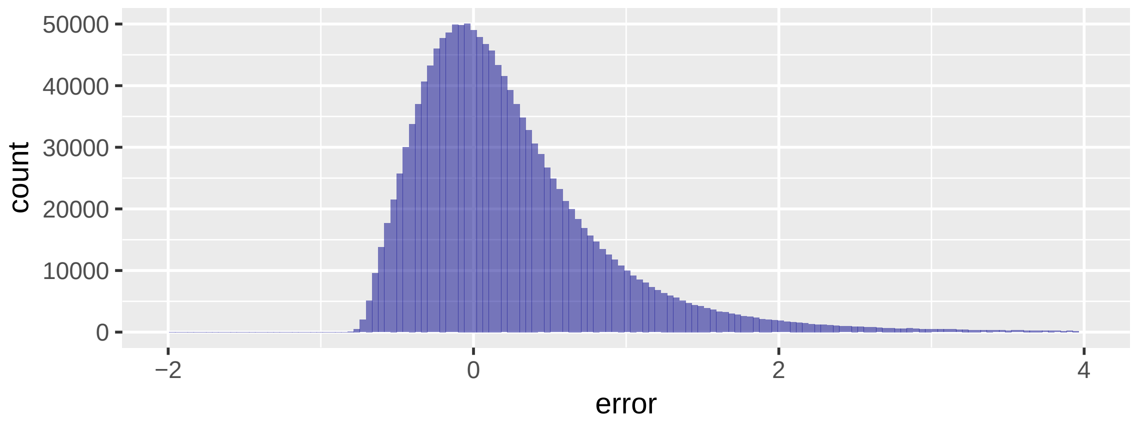
So in this situation

$$T = \frac{Distance}{\bar{V}}(1+r) * Date[wday] * Hour[hour]$$

we only need to estimate r to get the estimate function.

$$r = \frac{T\bar{V}}{Date[wday] * Hour[hour] * Distance} - 1$$

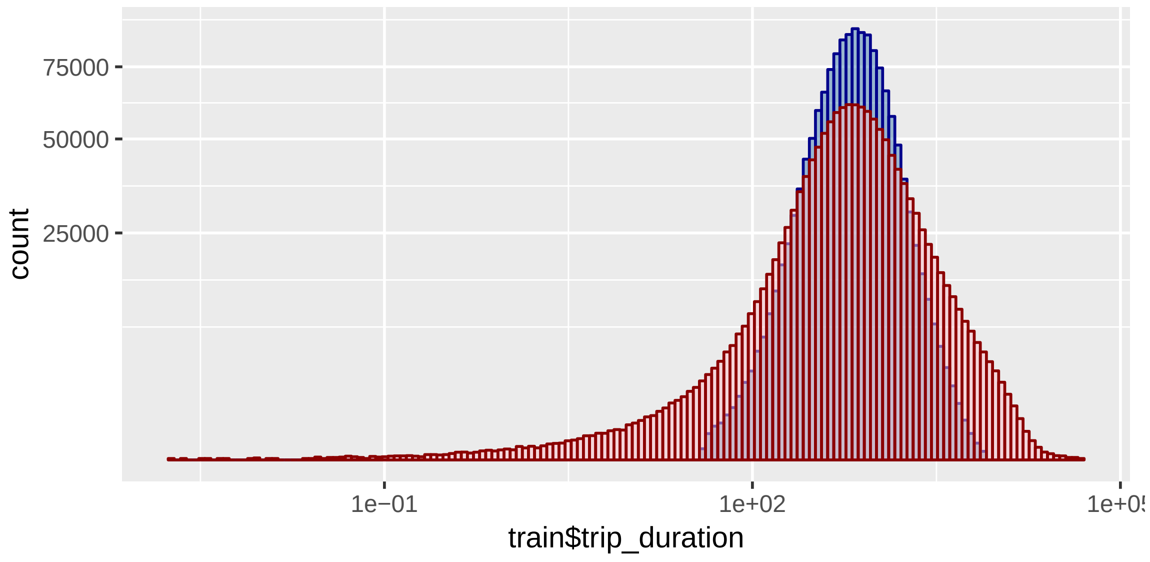
```
n <- function(data){
  (data$trip_duration/(DATE(data$wday)*HOUR(data$hour))-data$dist/average_speed*3.6)/(data$dist/average_
}
error <- n(train)
med <- median(n(train))
sd <- sd(n(train))
mean <- mean(n(train))
p10 <- ggplot(data.frame(x=error),aes(x=x)) +          # histogram of error
  geom_histogram(fill="darkblue", bins = 150,position="identity", alpha=0.5)+
  xlim(-2,4)+xlab("error")
p10
```



Now we get the r data, which reperents the error(pecentage) compared to $\frac{Distance}{\bar{V}}$

```
T <- function(data,mean,sd){
  res <- 0
  for(i in 1:length(data$dist)){
    res[i] <- data$dist[i]/average_speed*3.6*(1+rnorm(1,mean,sd))*(DATE(data$wday[i])*HOUR(data$hour[i]
  }
  res
}

train$estimate <- T(train,mean,sd)     # assume r=N[mean,sd]
p9 <- ggplot(train) +
  geom_histogram(aes(train$trip_duration),fill="steelblue",color="darkblue", bins = 150,position="identi
  geom_histogram(aes(train$estimate),fill="pink", color="darkred", bins = 150,position="identity", alpha
  scale_x_log10() +
  scale_y_sqrt()
p9
```



5 Comments and Improvement

To be continued.