Capstone Milestone Report - Data Wrangling

Suresh
Date- 2018-01-30

Introduction

New York City taxi rides paint a vibrant picture of life in the city. The millions of rides taken each month can provide insight into traffic patterns, road blockage, or large-scale events that attract many New Yorkers. With ridesharing apps gaining popularity, it is increasingly important for taxi companies to provide visibility to their estimated fare and ride duration, since the competing apps provide these metrics upfront. Predicting fare and duration of a ride can help passengers decide when is the optimal time to start their commute.

The primary goal of this project is to predict trip duration of NYC Taxis based on features like trip coordinates, duration date and time. The data for this project is from Kaggle, NYC Taxi Ride Duration competition.

The Data

We are using totally 4 datasets for our Exploratory analysis and modeling. 2 data namely Train and Test are published by Kaggle. We are also using OSRM Open Source Routing Machine for analysis. We are not going to use OSRM in initial Data Wrangling. Let's dive in to our data sets to understand the features.

Load Required Packages

Before we start analyzing dataset features, lets load the packages needed for Data Wrangling. Predominently we have used dplyr, ggplot2, ggmap and lubridate thoughtout our analysis.

```r
library(dplyr)
library(tidyr)
library(lubridate)
library(ggplot2)
library(ggmap)
library(stringr)
library(RColorBrewer)
library(geosphere)
library(tibble)
library(forcats)
library(maps)
library(readr)
```

Data Wrangling and Cleaning

For our intial Data Wrangling we are using the training and test datasets provided by Kaggle. Training dataset has close to 1.5 Million and 630k records in test dataset. Each row contains one taxi trip.

```r
taxi <- read_csv("train.csv")
test <- read_csv("test.csv")
glimpse(taxi)
```

```
## Observations: 1,458,644
## Variables: 11
## $ id                 <chr> "id2875421", "id2377394", "id3858529", "id3...
## $ vendor_id          <int> 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2...
## $ pickup_datetime    <dttm> 2016-03-14 17:24:55, 2016-06-12 00:43:35, ...
## $ dropoff_datetime   <dttm> 2016-03-14 17:32:30, 2016-06-12 00:54:38, ...
## $ passenger_count    <int> 1, 1, 1, 1, 1, 6, 4, 1, 1, 1, 1, 4, 2, 1, 1...
## $ pickup_longitude   <dbl> -73.98215, -73.98042, -73.97903, -74.01004,...
## $ pickup_latitude    <dbl> 40.76794, 40.73856, 40.76394, 40.71997, 40....
## $ dropoff_longitude  <dbl> -73.96463, -73.99948, -74.00533, -74.01227,...
## $ dropoff_latitude   <dbl> 40.76560, 40.73115, 40.71009, 40.70672, 40....
## $ store_and_fwd_flag <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N"...
## $ trip_duration      <int> 455, 663, 2124, 429, 435, 443, 341, 1551, 2...
```

```r
glimpse(test)
```

```
## Observations: 625,134
## Variables: 9
## $ id                 <chr> "id3004672", "id3505355", "id1217141", "id2...
## $ vendor_id          <int> 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 2, 1...
## $ pickup_datetime    <dttm> 2016-06-30 23:59:58, 2016-06-30 23:59:53, ...
## $ passenger_count    <int> 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 4, 1, 1, 1, 1...
## $ pickup_longitude   <dbl> -73.98813, -73.96420, -73.99744, -73.95607,...
## $ pickup_latitude    <dbl> 40.73203, 40.67999, 40.73758, 40.77190, 40....
## $ dropoff_longitude  <dbl> -73.99017, -73.95981, -73.98616, -73.98643,...
## $ dropoff_latitude   <dbl> 40.75668, 40.65540, 40.72952, 40.73047, 40....
## $ store_and_fwd_flag <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N"...
```

```r
taxi <- data.frame(taxi)
test <- data.frame(test)
```

We see below observations on Taxi and Test Records

- Trip *Id* is unique identification of a trip
- *vendor_id* field has only 2 values "1" or "2" asuming two taxi companies
- *pickup/dropoff_datetime* - holds date and time of pickup and dropoff. we need to mutate the fields to get date and time seperately.
- *pickup/dropoff_longitute/latitute* - hold values of geographical coordinates where the meter was activate/deactivated.
- *store_and_fwd_flag* is a flag that indicates whether the trip data was sent immediately to the vendor ("N") or held in the memory of the taxi because there was no connection to the server ("Y"). Maybe there could be a correlation with certain geographical areas with bad reception?
- *trip_duration* hold the duration in seconds and its our target prediction of ths project.
- Please note Test data will not have actual trip duration data. We have to submit the data in Kaggle to know the model score.

Check blank values in TAXI

```
sum(is.na(taxi))
```

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

```
sum(is.na(test))
```

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

Reformating data

For our analysis, we will date fields from characters into date objects. We also change vendor_id as a factor. This makes it easier to visualise relationships that involve these features.

```
taxi <- taxi %>% mutate(
  pickup_datetime = ymd_hms(pickup_datetime),
  dropoff_datetime = ymd_hms(dropoff_datetime),
  vendor_id = factor(vendor_id),
  passenger_count = factor(passenger_count)
)
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.2
```

Consistency check

This code is to check *trip_durations* are consistent with the intervals between *pickup_datetime* and *dropoff_datetime*.

Actual count of Taxi file is 1458644. Count of below check should the same else the records are inconsistent.

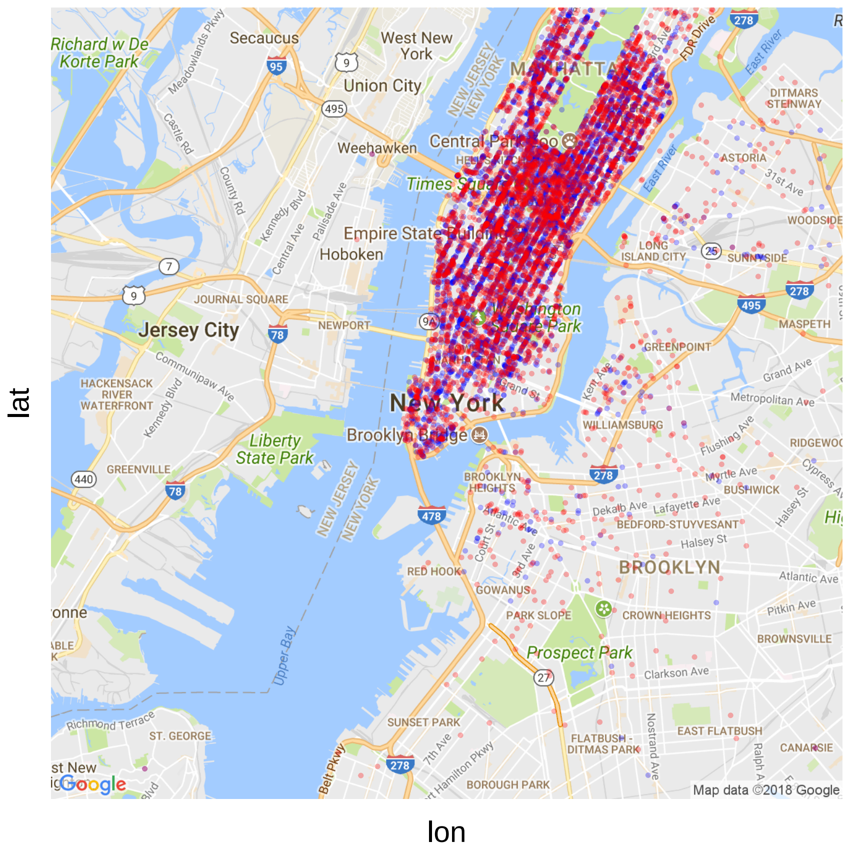
```
taxi %>%
  mutate(check = abs(int_length(interval(dropoff_datetime, pickup_datetime)) + trip_duration) > 0) %>%
  select(check, pickup_datetime, dropoff_datetime, trip_duration) %>%
  group_by(check) %>%
  count()
```

```
## # A tibble: 1 x 2
## # Groups:   check [1]
##   check       n
##   <lgl>   <int>
## 1 FALSE 1458644
```

Feature Visualizations

We are starting with NYC Map to check where the most of pickup and dropoff are happening. We took 5000 samples from Taxi file and plotted. It turns out that almost all of our trips were in fact taking place in Manhattan only.

```
my_map <- get_map(location = "New York City", zoom = 12, maptype = "roadmap", source = "google", color =
set.seed(1234)
tax_samp <- sample_n(taxi, 5000)
ggmap(my_map) +
  geom_point(data = tax_samp, aes(x = pickup_longitude, y = pickup_latitude), size = 0.3, alpha = 0.3, c
  geom_point(data = tax_samp, aes(x = dropoff_longitude, y = dropoff_latitude), size = 0.3, alpha = 0.3
  theme(axis.ticks = element_blank(), axis.text = element_blank())
```



Below analysis to check if any abnormal trip duration exists in our data.

```r
taxi %>%
  arrange(desc(trip_duration)) %>%
  select(trip_duration, pickup_datetime, dropoff_datetime) %>%
  head(5)
```

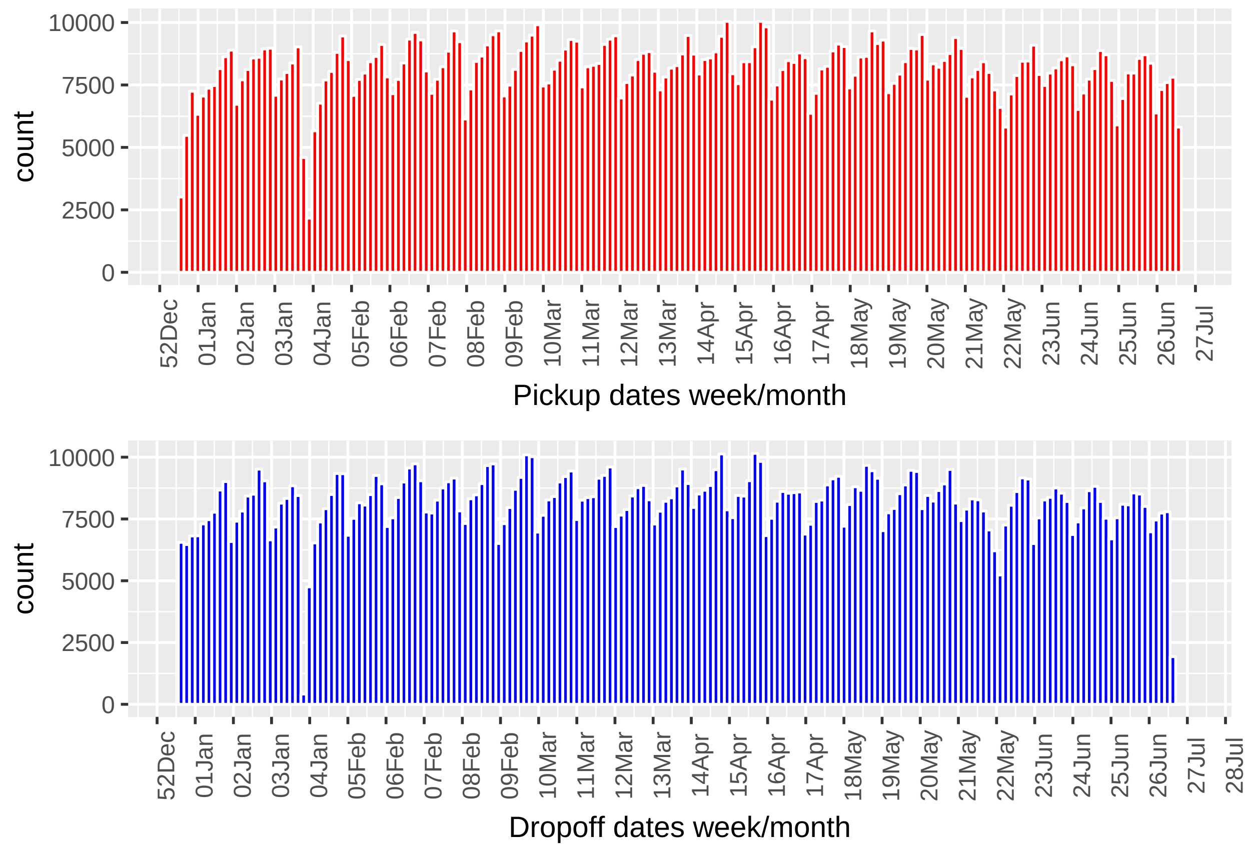
```
##   trip_duration     pickup_datetime    dropoff_datetime
## 1       3526282 2016-02-13 22:46:52 2016-03-25 18:18:14
## 2       2227612 2016-01-05 06:14:15 2016-01-31 01:01:07
## 3       2049578 2016-02-13 22:38:00 2016-03-08 15:57:38
## 4       1939736 2016-01-05 00:19:42 2016-01-27 11:08:38
## 5         86392 2016-02-15 23:18:06 2016-02-16 23:17:58
```

```r
taxi %>%
  arrange(desc(trip_duration)) %>%
  select(trip_duration, pickup_datetime, dropoff_datetime) %>%
  tail(5)
```

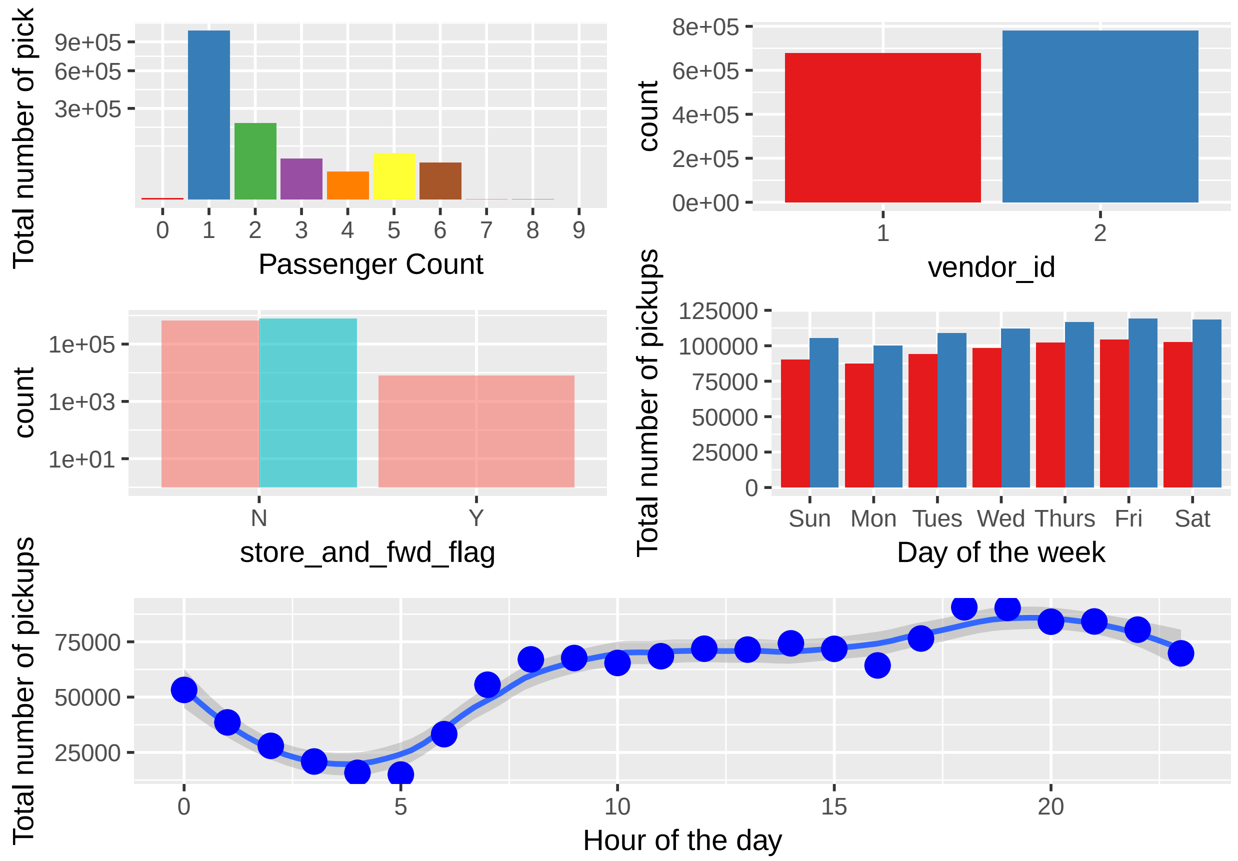
```
##         trip_duration     pickup_datetime    dropoff_datetime
## 1458640             1 2016-02-22 00:40:25 2016-02-22 00:40:26
## 1458641             1 2016-03-12 02:15:31 2016-03-12 02:15:32
## 1458642             1 2016-01-14 12:33:28 2016-01-14 12:33:29
## 1458643             1 2016-02-06 13:40:27 2016-02-06 13:40:28
## 1458644             1 2016-01-03 16:55:44 2016-01-03 16:55:45
```

Lets check the distributions of pickup_datetime and dropoff_datetime by year.

```
p1 <- taxi %>%
  ggplot(aes(x = pickup_datetime)) +
  geom_histogram(fill = "red", colour = "white", bins = 180) +
  scale_x_datetime(date_breaks = "1 week", date_labels = "%W%b") +
  labs(x = "Pickup dates week/month") +
  theme(axis.text.x = element_text(angle = 90))

p2 <- taxi %>%
  ggplot(aes(dropoff_datetime)) +
  geom_histogram(fill = "blue", colour = "white", bins = 180) +
  scale_x_datetime(date_breaks = "1 week", date_labels = "%W%b") +
  labs(x = "Dropoff dates week/month") +
  theme(axis.text.x = element_text(angle = 90))

layout <- matrix(c(1, 2), 2, 1, byrow = FALSE)
multiplot(p1, p2, layout = layout)
```



In the below plot we are checking passenger count, vendor_id, total number of pickups on hour/day distribution.



- We found some abnormal trip with zero passenger and more 7 passengers

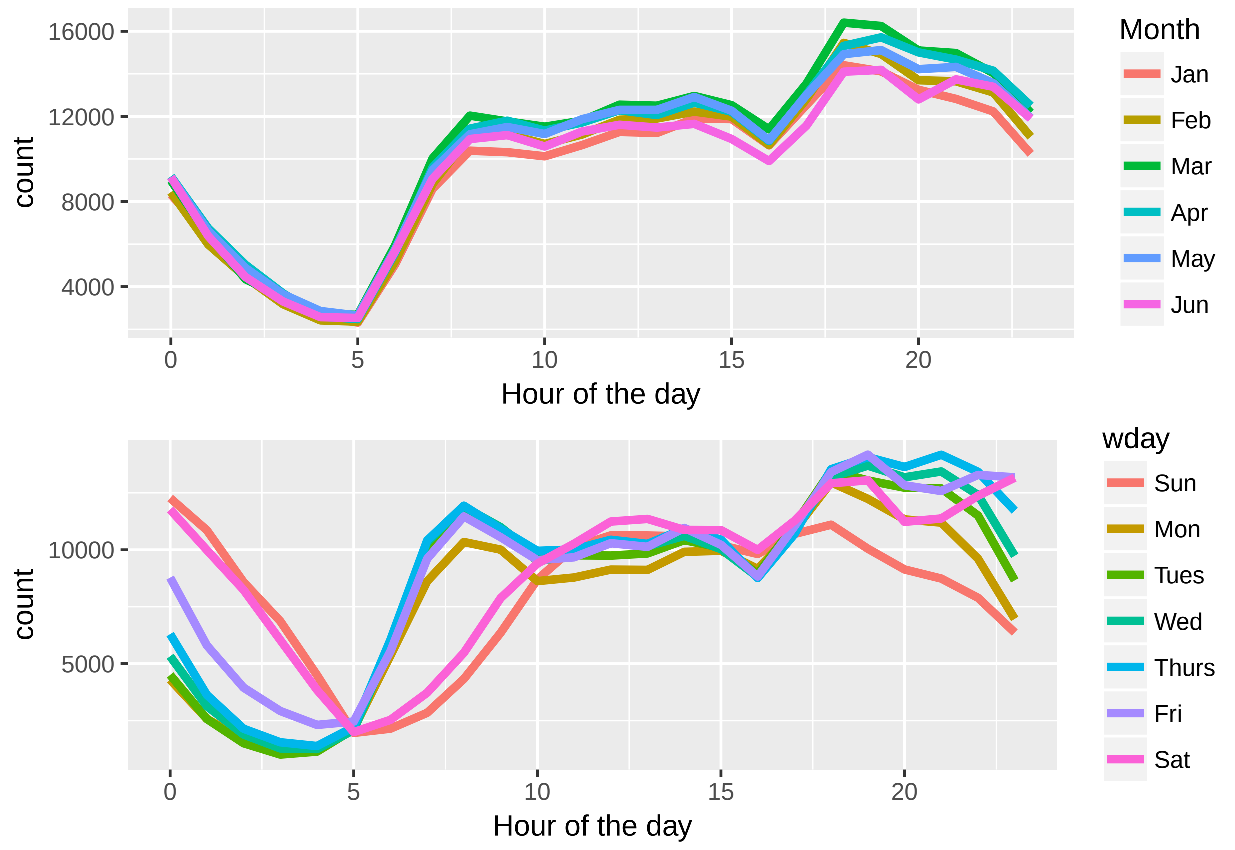
- We find an interesting pattern with Monday being the quietest day and Friday very busy.

- we find evening hours are busiest hours of the day.

Now lets check how the trends in different vizualization.

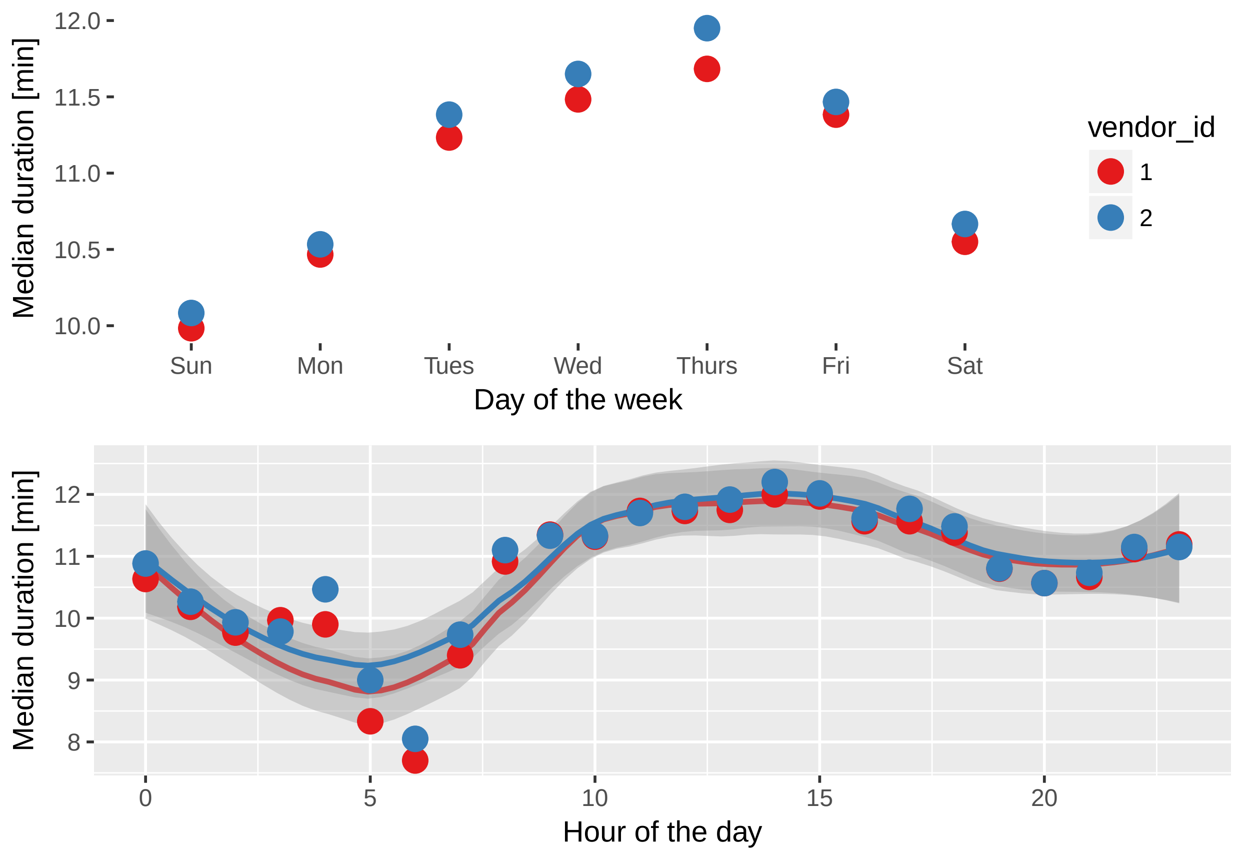
- We find Jan and June has less number of trips
- We find During weekends early morning are busy

```
p1 <- taxi %>%
  mutate(
    hpick = hour(pickup_datetime),
    Month = factor(month(pickup_datetime, label = TRUE))
  ) %>%
  group_by(hpick, Month) %>%
  count() %>%
  ggplot(aes(hpick, n, color = Month)) +
  geom_line(size = 1.5) +
  labs(x = "Hour of the day", y = "count")

p2 <- taxi %>%
  mutate(
    hpick = hour(pickup_datetime),
    wday = factor(wday(pickup_datetime, label = TRUE))
  ) %>%
  group_by(hpick, wday) %>%
  count() %>%
  ggplot(aes(hpick, n, color = wday)) +
  geom_line(size = 1.5) +
  labs(x = "Hour of the day", y = "count")

layout <- matrix(c(1, 2), 2, 1, byrow = FALSE)
multiplot(p1, p2, layout = layout)
```



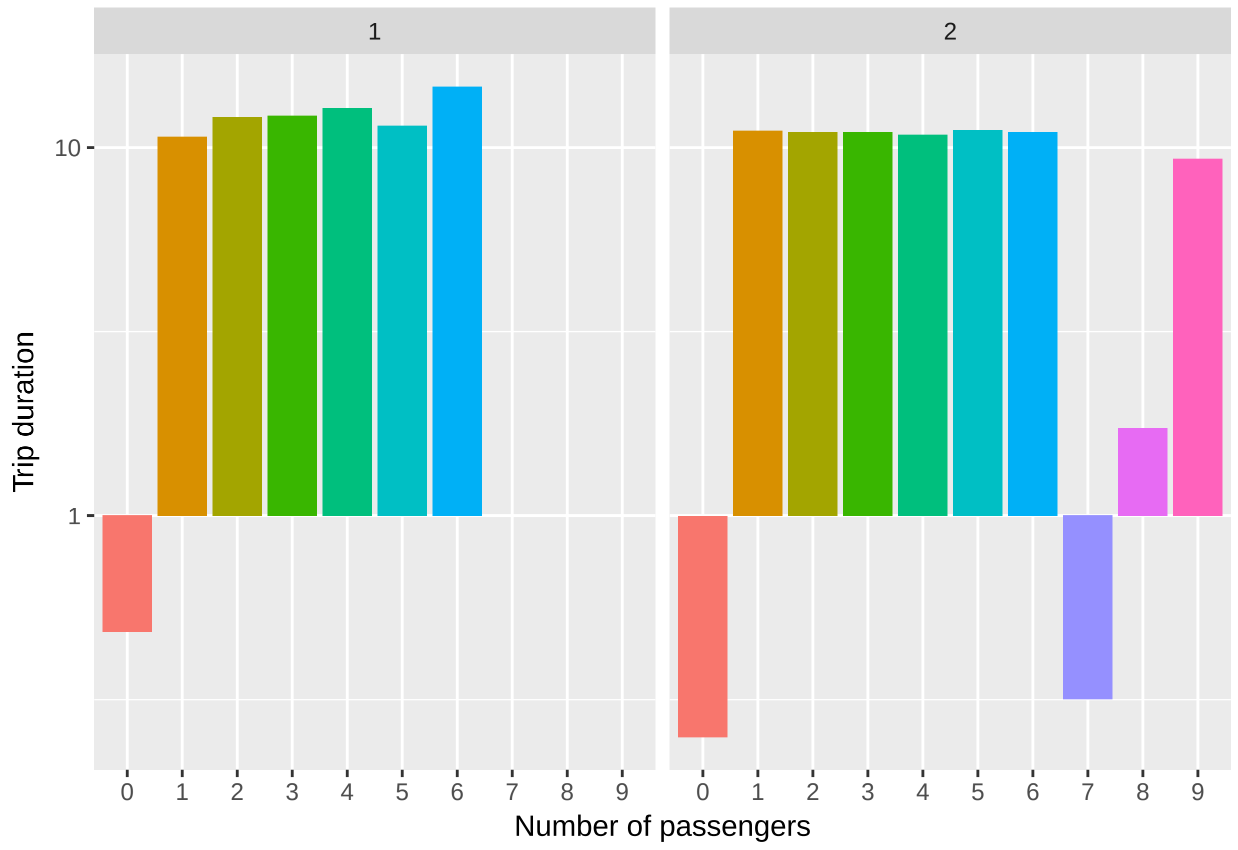
In the next slides we trying find the relation between the trip duration and picking up data & time. This will help us identify how strongly correlated to add them in the model.

```
p1 <- taxi %>%
  mutate(wday = wday(pickup_datetime, label = TRUE)) %>%
  group_by(wday, vendor_id) %>%
  summarise(median_duration = median(trip_duration) / 60) %>%
  ggplot(aes(wday, median_duration, color = vendor_id)) +
  geom_point(size = 4) +
  scale_colour_brewer(palette = "Set1") +
  labs(x = "Day of the week", y = "Median duration [min]") +
  theme(panel.background = element_rect(fill = "white"))

p2 <- taxi %>%
  mutate(pickt = hour(pickup_datetime)) %>%
  group_by(pickt, vendor_id) %>%
  summarise(median_duration = median(trip_duration) / 60) %>%
  ggplot(aes(pickt, median_duration, color = vendor_id)) +
  geom_smooth(method = "loess", span = 1 / 2) +
  geom_point(size = 4) +
  scale_colour_brewer(palette = "Set1") +
  labs(x = "Hour of the day", y = "Median duration [min]") +
  theme(legend.position = "none")

layout <- matrix(c(1, 2), 2, 1, byrow = FALSE)
multiplot(p1, p2, layout = layout)
```



In the our next slide, we are checking for any correlation between passenger count and trip duration.

```
taxi %>%
  group_by(passenger_count, vendor_id) %>%
  summarise(median_duration = median(trip_duration) / 60) %>%
  ggplot(aes(passenger_count, median_duration, fill = passenger_count)) +
  geom_bar(stat = "identity") +
  scale_y_log10() +
  theme(legend.position = "none") +
  facet_wrap(~ vendor_id) +
  labs(y = "Trip duration", x = "Number of passengers")
```

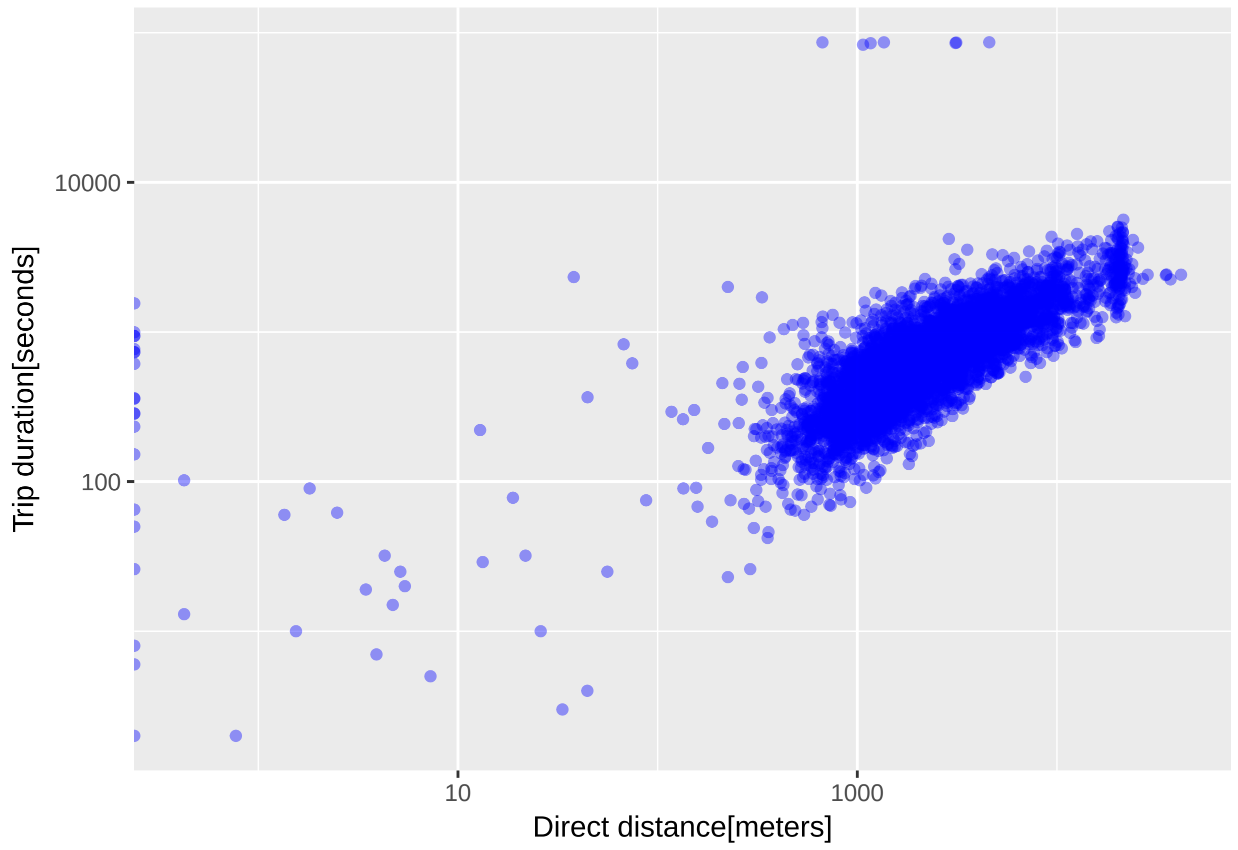


Relation between Time and Direct distance

In this section we are trying to find out the relation between drip duration and direct distance. To derive direct distance, we are using "Geosphere" package. Also, we are trying to find out any significance counts trips made out of Manhattan. Two major airports attract more taxi rides from city. We need to find how significant they are for our modelling.

Lets plot relationship trip duration and distance

```
set.seed(1234)
taxi %>%
  sample_n(5000) %>%
  ggplot(aes(dist, trip_duration)) +
  geom_point(color = "blue", alpha = 0.4) +
  scale_x_log10() +
  scale_y_log10() +
  labs(x = "Direct distance[meters]", y = "Trip duration[seconds]")
```

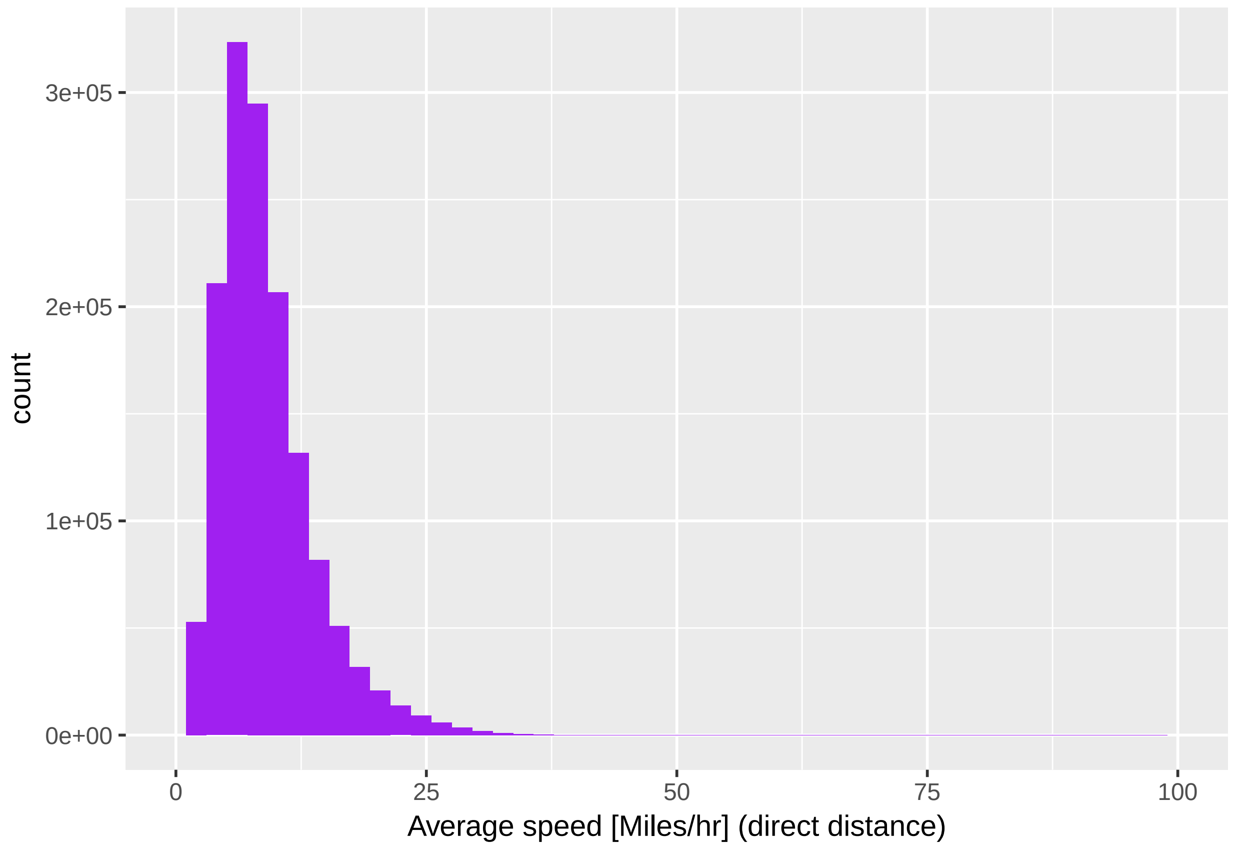


- Its clear observation, that distance increases trip duration.

Lets visualize how speed new york taxis travelling during peak hours and weekends. In order to find bogus values in the datasets, we can find extreme speed records and eliminate them

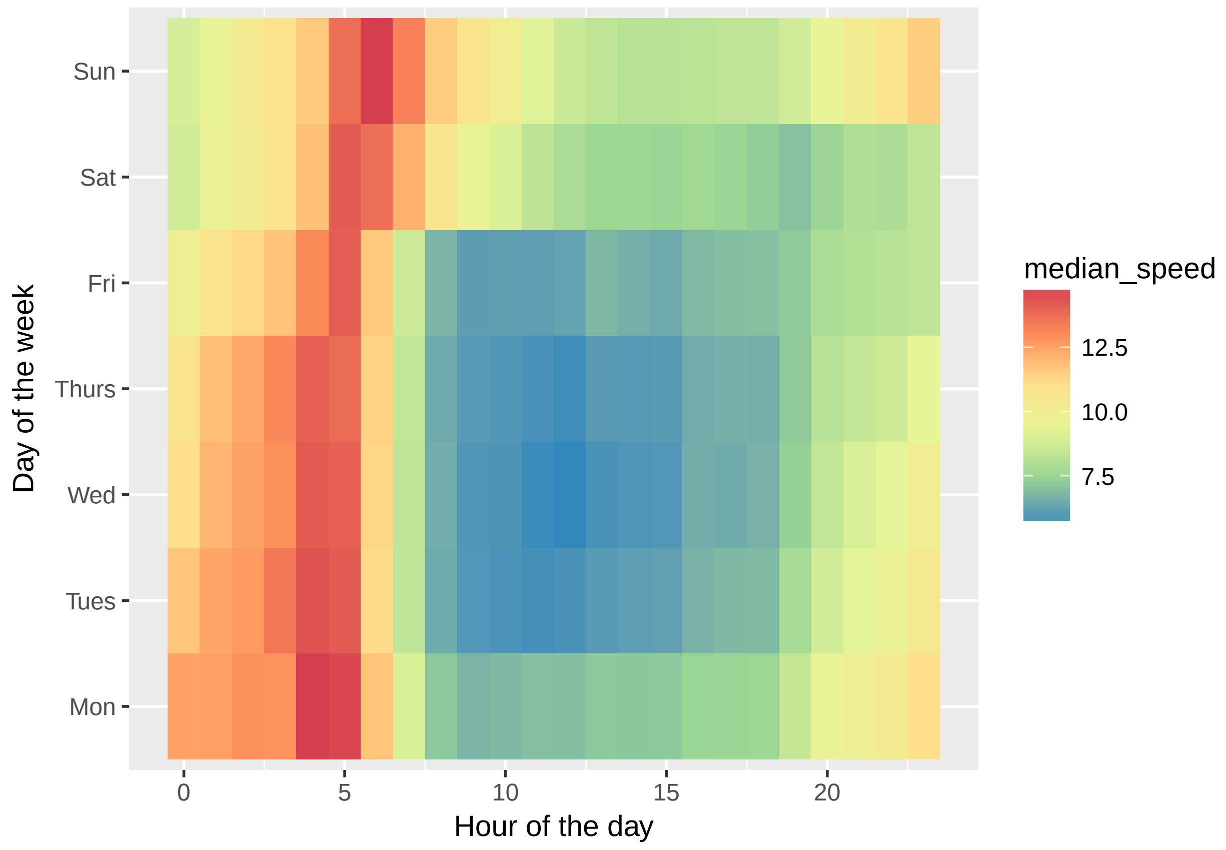
```
taxi %>%
  ggplot(aes(speed)) +
  geom_histogram(fill = "purple", bins = 50) +
  scale_x_continuous(limits = c(0, 100)) +
  labs(x = "Average speed [Miles/hr] (direct distance)")
```

```
## Warning: Removed 89 rows containing non-finite values (stat_bin).
```



Plotting speed on different times using heat map.

```
P1 <- taxi %>%
  group_by(wday, hour) %>%
  summarise(median_speed = median(speed)) %>%
  ggplot(aes(hour, wday, fill = median_speed)) +
  geom_tile() +
  labs(x = "Hour of the day", y = "Day of the week") +
  scale_fill_distiller(palette = "Spectral")
layout <- matrix(c(1, 1), 1, 1, byrow = TRUE)
multiplot(P1, layout = layout)
```



Data Cleaning.

The aim here is to remove trips that have improbable features, such as extreme trip durations or very low average speed.

** Filter trips more than 24 hours.

```
long_hrtrips <- taxi %>%
  filter(trip_duration > 24 * 3600)

long_hrtrips %>%
  arrange(desc(dist)) %>%
  select(pickup_datetime, dropoff_datetime, speed) %>%
  head(05)
```

```
##       pickup_datetime    dropoff_datetime       speed
## 1 2016-01-05 00:19:42 2016-01-27 11:08:38 0.023252072
## 2 2016-02-13 22:46:52 2016-03-25 18:18:14 0.012633059
## 3 2016-02-13 22:38:00 2016-03-08 15:57:38 0.006533943
## 4 2016-01-05 06:14:15 2016-01-31 01:01:07 0.001643123
```

** Filter trips shorter than a few minutes

```
min_trips <- taxi %>%
  filter(trip_duration < 5 * 60)

min_trips %>%
  arrange(dist) %>%
  select(dist, pickup_datetime, dropoff_datetime, speed) %>%
  head(05)
```

```
##   dist     pickup_datetime    dropoff_datetime speed
## 1    0 2016-02-29 18:39:12 2016-02-29 18:42:59     0
## 2    0 2016-01-27 22:29:31 2016-01-27 22:29:58     0
## 3    0 2016-01-22 16:13:01 2016-01-22 16:13:20     0
## 4    0 2016-01-18 15:24:43 2016-01-18 15:28:57     0
## 5    0 2016-05-04 22:28:43 2016-05-04 22:32:51     0
```

** Find trip with zero miles

```
zero_dist <- taxi %>%
  filter(near(dist, 0))
nrow(zero_dist)
```

```
## [1] 5897
```

Filter all bogus data from taxi dataset

```
taxi <- taxi %>%
  filter(
    trip_duration < 22 * 3600,
    dist > 0 | (near(dist, 0) & trip_duration < 60),
    jfk_dist_pick < 3e5 & jfk_dist_drop < 3e5,
    trip_duration > 10,
    speed < 100
  )
```

Write the records to output file

```
taxi %>% write_csv("Taxi_Clean.csv")
```

Next Steps

Above analysis helped us to understand the structure of data, values, trends in the data. We were able to filter the data for modelling. As a next step, we will introduce external data and try to find features will help model the data.