Chicago Taxi Market Analysis

Welcome Taxi

Welcome Taxi – Parth Gandhi, Rinkesh Patel, Sharath Kumar Reddy Guddati, Mustafa Fnu

24 April, 2021

Project Part 1

1. Introduction:

In a large metropolitan city like Chicago, plenty of taxi service companies work alongside with each other, some companies continue to do well by making smart strategies while some lose to competition and goes out of business. Year 2020 is very different than the rest because in 2020, COVID pandemic got introduced in everyone lives because of which plenty of companies went out of business and that also includes many taxi companies. Taxi companies went out of business due to effects of lockdown that started in March until June, during that time everyone was advised by health authority to stay at home unless it was necessary to go out, and even after lockdown got lifted in July, people didn't ride taxi much as before COVID since everyone was working from home and some people were simply scared to travel due to which taxi companies slowly went out of business that survived lockdown until June.

Also, for your information, the data is real, but the story is made up. We do not have enough data like companies profit and loss information to back up that taxi companies slowly went out of business, our team is just assuming that taxi companies went out business that survived lockdown since even they must have difficult time pay for their employees, taxis, and other operations during lockdown with no source of income coming.

Our team wants to start a new taxi service called "Welcome Taxi" for resident, tourists & employees who travels around Chicago. We think this is good time to enter the competitive taxi service market since some old competitors have left the market while others are still struggling to come up with strategy to survive during the pandemic. We want to provide good service at best rate possible which our passengers could also afford while we also make some profit from it. Since the COVID19 pandemic has taken over the world, it has also made changes in Taxi Service market that we want to enter, so our team thinks its best if we first analyze the Taxi market before, we start our taxi service business to better understand the changes market that have occurred during 2020 & make smart business strategies so we can start on the right foot and also be a profitable business even when the pandemic has not ended yet.

Dataset description:

The dataset is available on the City of Chicago Data Portal. The dataset represents the details of the taxi trips reported to the City of Chicago Data Portal in the year of 2020. The data shares information related with every single trip made during that year that were reported to City of Chicago Data Portal.

The link is: (https://data.cityofchicago.org/Transportation/Taxi-Trips-2020/r2u4-wwk3))

Column Name	**Data Type**	**Description**
trip_id	**chr**	A unique identifier for the trip
trip_timestamp (month)	**int**	Recorded start/end time for a trip
trip_timestamp (date)	**int**	Recorded start/end time for a trip
trip_timestamp (year)	**int**	Recorded start/end time for a trip
trip_timestamp(hour)	**int**	Recorded start/end time for a trip
trip_timestamp(min)	**int**	Recorded start/end time for a trip
weekday	**fctr**	Day of the week
trip_seconds	**int**	Time of the trip in seconds
trip_miles	**dbl**	Distance of the trip in miles
pickup_communities_area_code	**fctr**	The community area where the trip began
dropoff_communities area_code	**fctr**	The community area where the trip ended
fare	**dbl**	A standard fare for the trip
tips	**dbl**	Tips paid by a passenger for the trip
tolls	**dbl**	The tolls for the trip
extras	**dbl**	Surge price during the trip
trip_total	**dbl**	The total cost of the trip (Sum of the previous columns)
payment_type	**fctr**	The payment method for the trip
company	**fctr**	Name of the taxi company
pickup_centroid_latitude	**dbl**	The location of a pickup as shown in a map
Pickup_centroid_longitude	**dbl**	The location of a pickup as shown in a map
dropoff_centroid_latitude	**dbl**	The location of the dropoff as shown in a map
dropoff_centroid_longtitude	**dbl**	The location of a dropoff as shown in a map

Since we are a new company, we want to use all the columns related to taxi service industry to come up with visualizations and perform statistical analysis that would help us answer our questions to start operation at optimal time.

Data Import:

The complete 2020 chicago taxi trips is loading into R using read_csv fundtion which directly loads the data into R in a Tibble format.

```
### Loading Complete Taxi_Trips_2020 data
```

```
taxi_trips_2020_part1_csv <- read_csv("C:\\Users\\shara\\Desktop\\OMIS_665_RS
tudio\\OMIS665_R_Studio_Project\\OMIS665_Project\\taxi_trips_2020.csv")
```

2. Body

After importing the original dataset, we found some columns such as taxi ID, pickup centroid location and dropoff centroid location that were not going to help us in our analysis because taxi ID was not going to provide meaningful information to us, and pickup centroid location and Dropoff Centroid Location were redundant since we already had pickup centroid latitude, pickup centroid longitude, dropoff centroid latitude and dropoff centroid longitude columns in our dataset. Next, we came across some columns such as trip start timestamp & trip end timestamp that needed to be separated into multiple columns to get good analysis because if we want to do analysis on month, date or time it would be easier for us to do it.

Cleaning the dataset:

Since our team felt it was necessary to delete some columns while separating one column into couple columns, we cleaned the dataset, changed the data types of the columns and created a new taxi_trips_2020_final table on which we are going to do our analysis on. For example, we found that we do not need column taxi ID, pickup centroid location, & dropoff centroid location. We deleted taxi ID column since it is a randomly generated number which does not provide much insight into our analysis while we deleted pickup and drop-off centroid location since they were repetitive since the csv file already has pickup and drop-off centroid latitude and longitude columns separately, so this column does not help in our analysis. While we felt it was necessary for us to separate trip start timestamp into 4 separate columns which are month, date, year, hour, minute, and weekday respectively. Lastly, our final dataset has total of 22 columns which is same as original dataset we downloaded from City of Chicago Data Portal but some columns are separated into multiple columns as shown in glimpse of dataset. Also, our original dataset had over 1 million observations which would have made it difficult for our group to load the dataset and perform analysis, so we decided to decrease the observations to 504,000 to be exact which was within the acceptable requirement to do our project. Later in the report, you will find that we have taken 42,000 observations for each month so with 12 months it will equal to 504,000 for this reason. We thought to mention it here to follow along easily. Also, later in the report, you will find that we have separated the date & time column into separate columns since originally date & time column was in “MM/DD/YYY HH:MM” format to perform analysis easier.

Changing the column names to ignore the spaces in column names and to maintain a single format for all the column names.

```
#Change the column names

colnames(taxi_trips_2020_part1_csv) <- c ("trip_id","taxi_id","trip_start_tim
estamp","trip_end_timesatmp","trip_seconds","trip_miles","pickup_census_tract
","dropoff_census_tract","pickup_community_area_code","dropoff_community_area
_code","fare","tips","tolls","extras","trip_total","payment_type","company","
pickup_centroid_latitude","pickup_centroid_longitude","pickup_centroid_locati
on","dropoff_centroid_latitude","dropoff_centroid_longitude","dropoff_centroi
d_location")
```

Changing the trip_start_timestamp date format into default format using as.POSIXct funtion to generate a weekday column to analyze the data on weekday basis.

```
#Changing the trip_start_timestamp date format

taxi_trips_2020_part1_csv$trip_start_timestamp <- as.POSIXct(taxi_trips_2020_
part1_csv$trip_start_timestamp, format = '%m/%d/%Y %I:%M:%S %p', tz = 'UTC')

#Adding weekday column to the table

taxi_trips_2020_part1_csv$weekday <- weekdays(taxi_trips_2020_part1_csv$trip_
start_timestamp)
```

Removing columns that are not useful for the analysis like taxi_id , trip_end_timesatmp (As we have start time and trip seconds),pickup and dropoff census tract and pickup and dropoff centroid location.

```
#Removing columns that are not required

taxi_trips_2020_tidy <- select(taxi_trips_2020_part1_csv, 1,3,24,5,6,9,10,11,
12,13,14,15,16,17,18,19,21,22)
```

Separating trip_start_timestamp column into month, days, year, hour and minutes columns which was in “MM/DD/YYYY HH:MM” format earlier for analyzing the data accordingly.

```
#Separating timestamp to individual columns

taxi_trips_2020_tidy <- taxi_trips_2020_tidy %>%
  separate(trip_start_timestamp, into = c("year", "month", "date", "hour", "m
inute"))
```

To meet the project requirements and have same number of observations for each month, we are filtering and taking random sample of 42,000 observations so our analysis won’t have any outliers if one month has more observations than another. Also, we have and loaded the results into individual sample tables.

```
#Filter and random sample records by month

jan_data <- taxi_trips_2020_tidy %>%
  filter(month=="01") %>%
  sample_n(42000)
```

After sampling the data each month, we are joining the months sample data that we separated earlier to get 42,000 observations for each month. By joining the months sample tables, we can have a final dataset with equal number of records from each month which would be better for our analysis.

```r
#joining  sample data
taxi_trips_sample <- jan_data %>%
 full_join(feb_data) %>%
 full_join(mar_data) %>%
 full_join(apr_data) %>%
 full_join(may_data) %>%
 full_join(jun_data) %>%
 full_join(jul_data) %>%
 full_join(aug_data) %>%
 full_join(sep_data) %>%
 full_join(oct_data) %>%
 full_join(nov_data) %>%
 full_join(dec_data)
```

After analyzing the data, we decided to change the datatypes of columns that are not right according to our analysis. Reason behind change in datatype is when we want to do a visualization, certain datatype won't give us the output we want. So, its better we change the datatype now so we don't run into errors later. Like weekday, pickup and dropoff community area code columns to factor which we are considering as categorical variables.

```r
#Changing the datatypes of columns

taxi_trips_sample_final <- taxi_trips_sample %>%
  mutate_at(c(2, 3, 4, 5, 6, 8), as.integer) %>%
  mutate_at(c(7, 10, 11, 17, 18), as.factor)
```

As we have community area codes instead of names, we are loading the community area codes and names csv file. So, we can update the community area names with related community_area_code.

```r
# loading the csv file to update the community area names with codes

community_areas <- read_csv("C:\\Users\\shara\\Desktop\\OMIS_665_RStudio\\OMI
S665_R_Studio_Project\\OMIS665_Project\\community_areas.csv")
```

Changing the datatype of the column from community areas csv file to factor to maintain the similar data type with taxi trips data.

```r
#changing the datatype

community_areas<-community_areas %>%
  mutate_at(c(1), as.factor)
```

Changing the names of columns, for example we are changing it from "pickup_area_number" to "pickup_community_area_code" and "community_name" to "pickup_community_area_name" which seems appropriate.

```r
#changing the column names

colnames(community_areas) = c("community_area_code", "community_area_name")
```

After loading the community area file, we are looking for community_area_name assigned to community_area_code from the community_areas table and replace the pickup_community_area_code and dropoff_community_area_code with community_area_name in the taxi_trips dataset.

```r
#looking up community_area_name from the community_areas and load into taxi_t
rips data accordingly

taxi_trips_2020_final <- taxi_trips_sample_final

taxi_trips_2020_final$pickup_community_area_code <- community_areas$community
_area_name[match(unlist(taxi_trips_2020_final$pickup_community_area_code), co
mmunity_areas$community_area_code )]

taxi_trips_2020_final$dropoff_community_area_code <- community_areas$communit
y_area_name[match(unlist(taxi_trips_2020_final$dropoff_community_area_code),
community_areas$community_area_code )]
```

In the final dataset we have created have some special characters in the values. So, we are removing special characters in pickup and dropoff community code columns to do analysis easier. We do not want any special character that would interfere with analysis later.

```r
# removing special characters in pickup and dropoff community code columns

taxi_trips_2020_final$pickup_community_area_code = as.factor(gsub("[^a-zA-Z0-
9.]","", taxi_trips_2020_final$pickup_community_area_code))
taxi_trips_2020_final$dropoff_community_area_code = as.factor(gsub("[^a-zA-Z0
-9.]","", taxi_trips_2020_final$dropoff_community_area_code))
```

3. Glimpse of Dataset

```
## Rows: 504,000
## Columns: 22
## $ trip_id                  <chr> "889ad3cb1e4047b9236345910357c6d568497
5...
## $ year                     <int> 2020, 2020, 2020, 2020, 2020, 2020, 20
2...
## $ month                    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
...
## $ date                     <int> 24, 23, 29, 16, 7, 29, 27, 10, 7, 7, 1
5...
## $ hour                     <int> 14, 19, 17, 18, 22, 17, 16, 7, 0, 14,
9...
## $ minute                   <int> 30, 15, 30, 15, 45, 30, 30, 0, 0, 0, 1
5...
## $ weekday                  <fct> Friday, Thursday, Wednesday, Thursday,
...
## $ trip_seconds             <int> 849, 2474, 787, 300, 660, 660, 239, 49
0...
## $ trip_miles               <dbl> 4.27, 15.65, 1.50, 0.00, 4.80, 1.40, 0
....
```

```
## $ pickup_community_area_code  <fct> NEARNORTHSIDE, OHARE, NA, LOOP, NEARWE
S...
## $ dropoff_community_area_code <fct> LINCOLNPARK, LAKEVIEW, NA, LOOP, LAKEV
I...
## $ fare                        <dbl> 14.00, 41.50, 9.00, 4.75, 15.00, 8.00,
...
## $ tips                        <dbl> 1.50, 14.10, 0.00, 2.00, 0.00, 2.00, 0
....
## $ tolls                       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
...
## $ extras                      <dbl> 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
,...
## $ trip_total                  <dbl> 16.00, 61.10, 9.00, 6.75, 15.00, 10.00
,...
## $ payment_type                <fct> Credit Card, Credit Card, Cash, Credit
...
## $ company                     <fct> "Chicago Carriage Cab Corp", "Nova Tax
i...
## $ pickup_centroid_latitude    <dbl> 41.89204, 41.97907, NA, 41.88099, 41.8
7...
## $ pickup_centroid_longitude   <dbl> -87.63186, -87.90304, NA, -87.63275, -
8...
## $ dropoff_centroid_latitude   <dbl> 41.92188, 41.94269, NA, 41.88099, 41.9
4...
## $ dropoff_centroid_longitude  <dbl> -87.66408, -87.65177, NA, -87.63275, -
8...
```

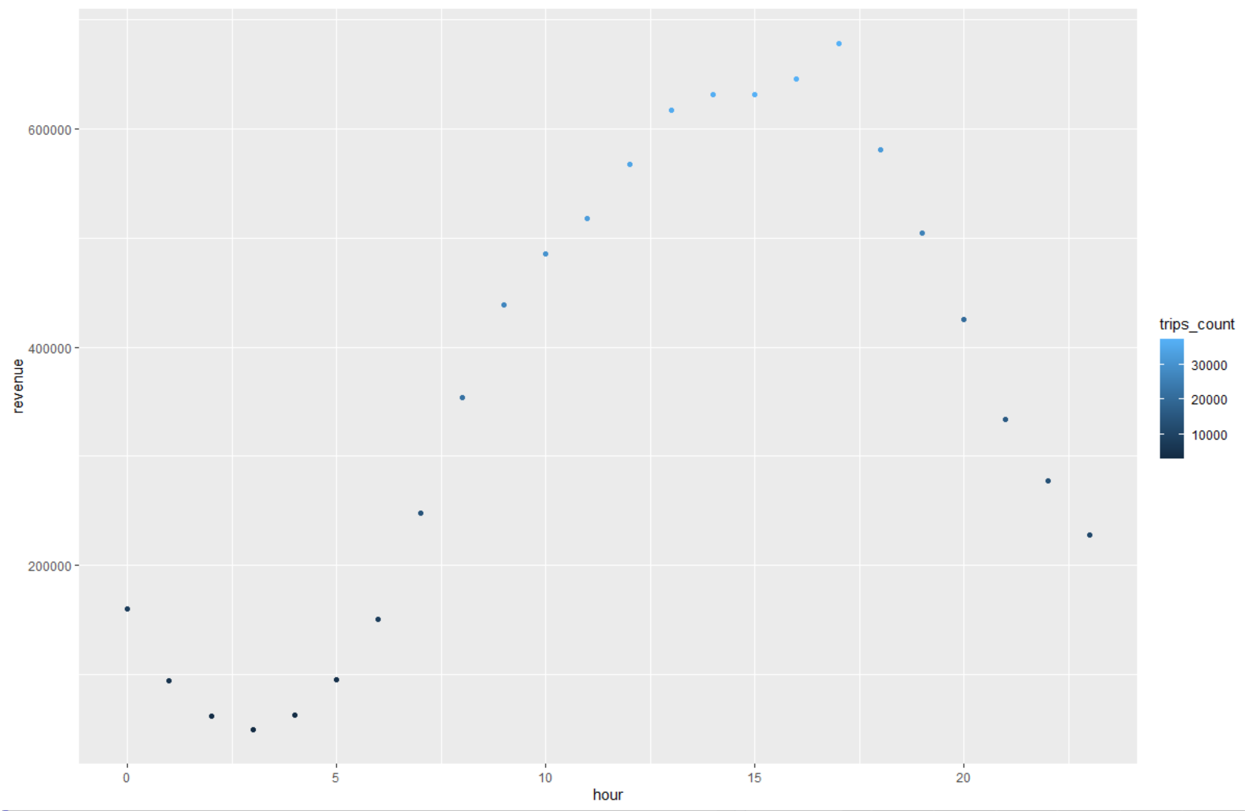
Queries & Visualizations.

We have created visualizations and regular queries for all the variables in the dataset we want to explore for final analysis.

Since we are planning to start a new taxi company in Chicago, we wanted to check the Chicago Taxi Market first before entering so we know what strategies we should make, who are our potential competitors in the market, how much should we charge to stay competitive while also keeping affordable for our potential customers, which community areas in Chicago we should focus more on and less on according to their demand. Next, we will start analysing dataset to get some useful information out of it.

1. Busiest hour of the day for taxi trips with revenue.

The idea behind the question is we wanted to find out what times during the day have the most volumes of trips and find out the trip total which we consider as revenue for each hour for our company to run more taxis at the busy time so we can take advantage of the busy hour and earn more sales, and not deploy more taxis than what we needed when its slow. This strategy will help our company save cost in long term.



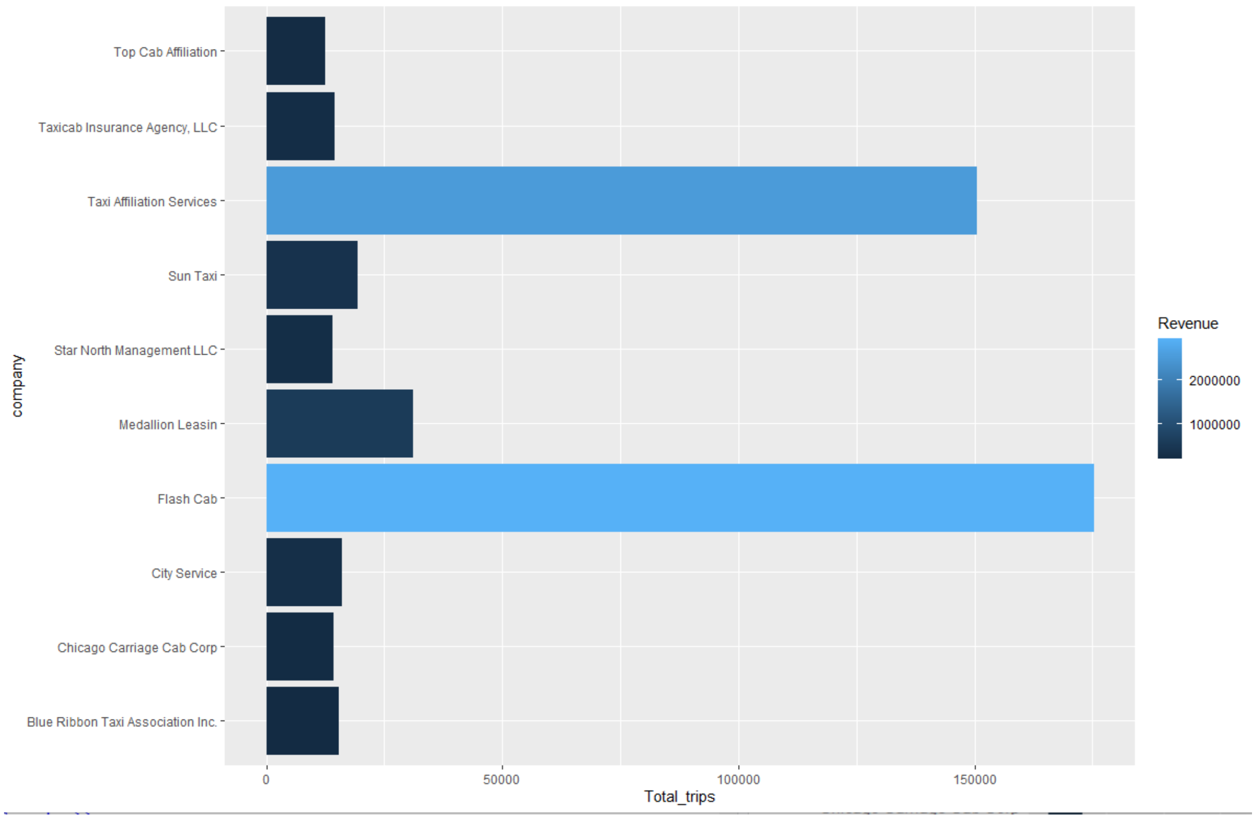
Our hour variables is in 24 hour format/ military time format which is surprising because I don't believe there is much business for taxis past midnight, but some competitors still run past midnight to earn more sales which is interesting strategy to think about.

Based on the output, the busiest hour of the day for taxi trips is between 12PM to 5PM. We found that during the busiest time between 12PM to 5PM, the revenue overall is above $600,000 so our company should deploy more taxis during the busy time to take advantage of it.

Next, let's look at which company we should look out for in the market according to their revenue.

2. Which taxi company made the most revenue out of the top 10 companies?

The idea behind the question is we want to find out who our competitors in Taxi Market are in terms of revenue before we enter the market from the top 10 companies.



For our analysis, we consider trip_total as revenue.

The trip_total is taken as revenue by our group because it is the sum of fare, tips, toll and extras.

Also, for revenue, the numbers are in million, so it shows as e after the number because of space constraints.

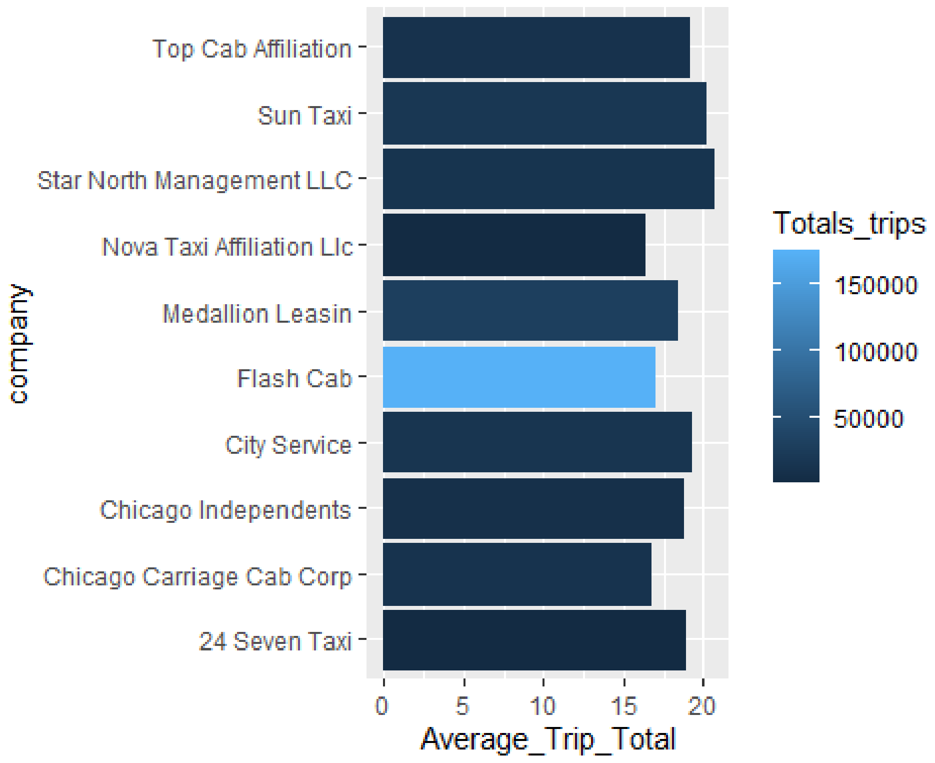
Taxi Affiliation Services & Flash Cab made the most revenue in 2020 even with COVID Pandemic going on. This shows that we have to be prepared for main competition from these 2 taxi companies in order to build our base in the taxi Market in Chicago.

We have not taken Uber and Lyft into account because the dataset does not have any information about those 2 rides sharing companies so it will be difficult for us to make any conclusion on Uber and Lyft since they also operate in other cities in US as well compared to small taxi companies in Chicago that only operates in Chicago.

Next, let's look at which taxi company makes the most number of taxi trips.

3. Which company is making the most number of trips out of the top 10 companies?

The idea behind this question is we want to find out who our competitors in the Taxi Market in terms of total number of trips is out of the top 10 companies. In previous question, we have already asked the same question, but it was in terms of revenue. We believe that it is not necessary that the company that has the most market share makes the most revenue since everyone has a different price model and their own strategy.



Based on the output, we believe 24 Seven Taxi & 312 Medallion Management Corp has captured most trip market shares even though they are not making the most revenue. This will help us make smart business strategy where we can price the trip correctly to get more passengers to use our service instead of our competitors to capture most trip market share and most revenue market share.

Next, we will look at how much is the average tip is paid in Chicago.

4. Average tip that is paid.

The idea behind this question is we want to know, how much tip on average per ride does the passenger pay in Chicago to their taxi drivers. The tips does not make much difference in our price model, but it makes a difference in the employees lives, it keeps them happy, it is a smaller source of their income so we are going to keep tips as optional, and the tip would directly go towards the employee. Since we care about our employees, we want to know on average how much tip does the taxi driver gets paid by the passengers per ride.

```
summarise(taxi_trips_2020_final, average_tip = mean(tips, na.rm = TRUE))
```

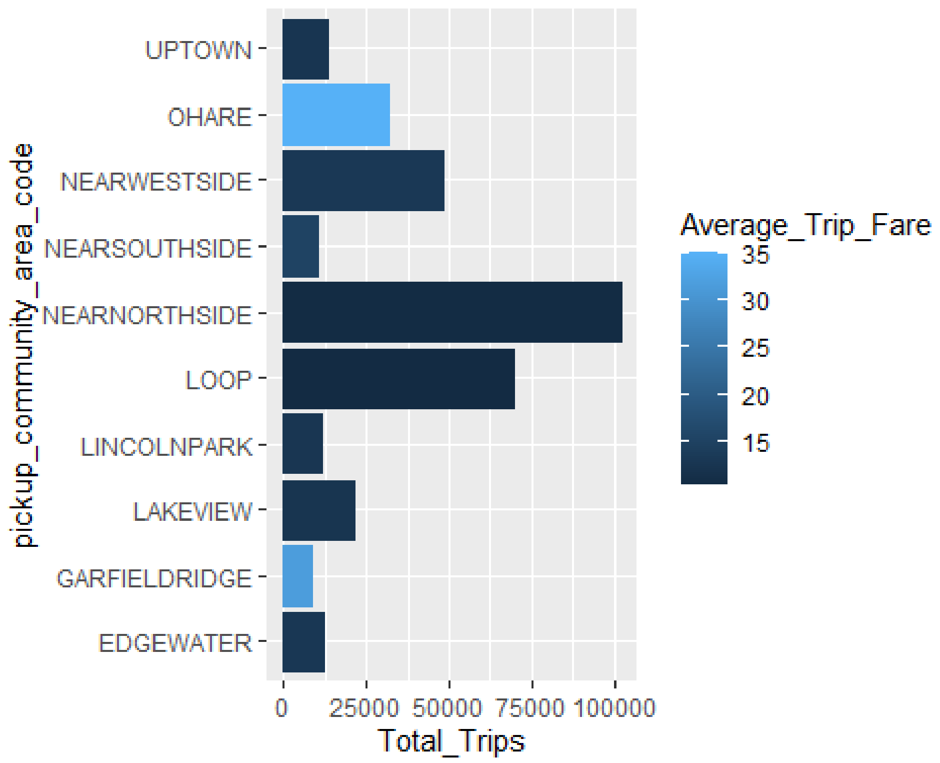
```
## # A tibble: 1 x 1
##   average_tip
##         <dbl>
## 1       0.999
```

Based on the output, the average tip is $1 per ride is not much in Chicago since some passengers do not like to pay any tips to taxi. But we cannot make conclusion on tips since there could be other factors involved in why customer/rider does not provide tip to the taxi driver.

Next, we will look at which community areas in Chicago has the most frequent pickups.

5. Which area has the most frequent pickups out of top 10 community areas?

The idea behind this question is since Chicago has lots of different community areas about 77 to be exact, we want to get better idea of what area we could deploy more of our taxi's to from the top 10 community area so we would deploy more of our resources in this area rather than other areas that brings in less revenue.

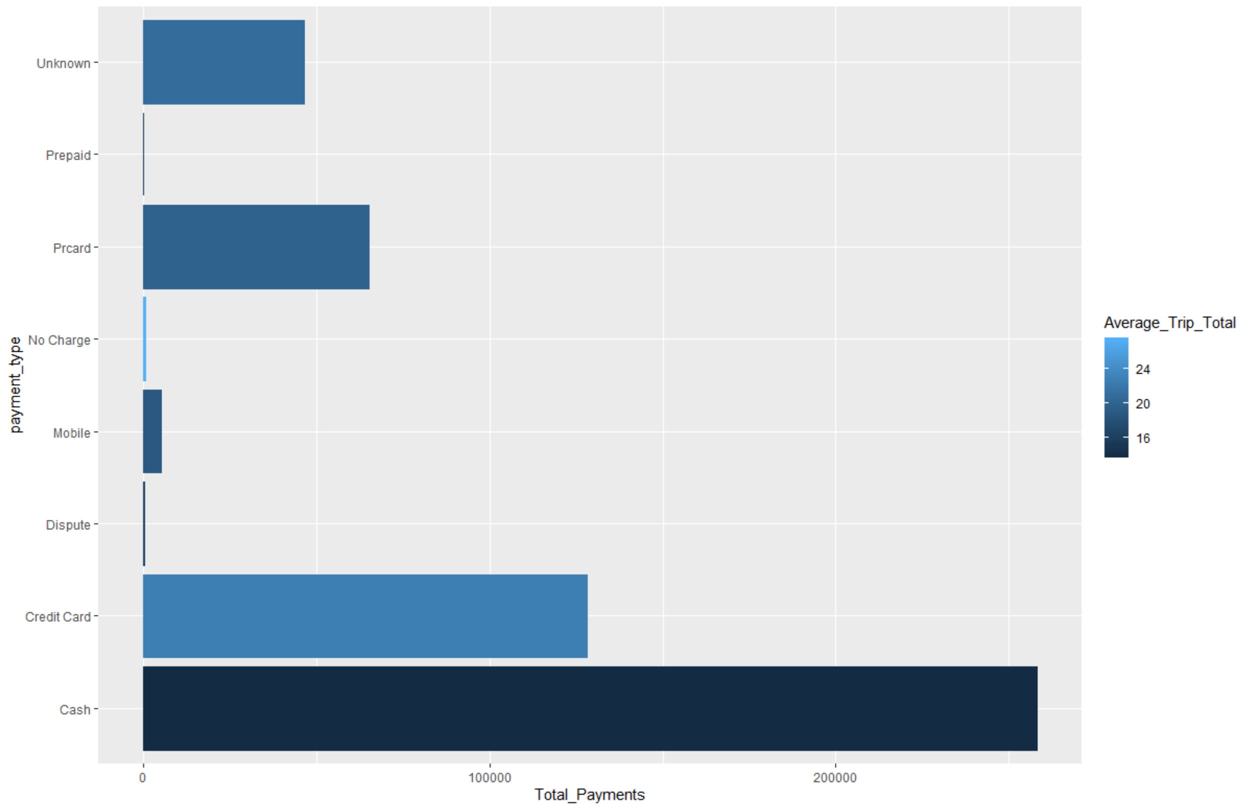


Based on the output, NearNorthSide & Loop are the most frequent pickup locations that brings in more of our revenue than other areas. We will run our taxis in all the community area, but we will run more taxis in these 2 locations. Our plan is that we will reduce my taxis in community areas that brings in less revenue and deploy them in community areas that brings in more revenue like NearNorthSide & Loop, this way we wouldn't have to purchase more vehicles and saving cost.

Next, we will look at payment method used the most so we know whether to go cashless by accepting only Creditcard and mobile payment or we also need to keep working with cash and accept credit card and mobile payments.

6. Which payment method was used the most.

The idea behind this question is to gather data on the various payment methods used by the rider while riding a taxi. At times, the driver is forced to give a free ride or cut a trip short due to a possible dispute. At instances, where cash is used as a payment method and the trip ends up in a dispute, the company might have to bear losses for those trips.

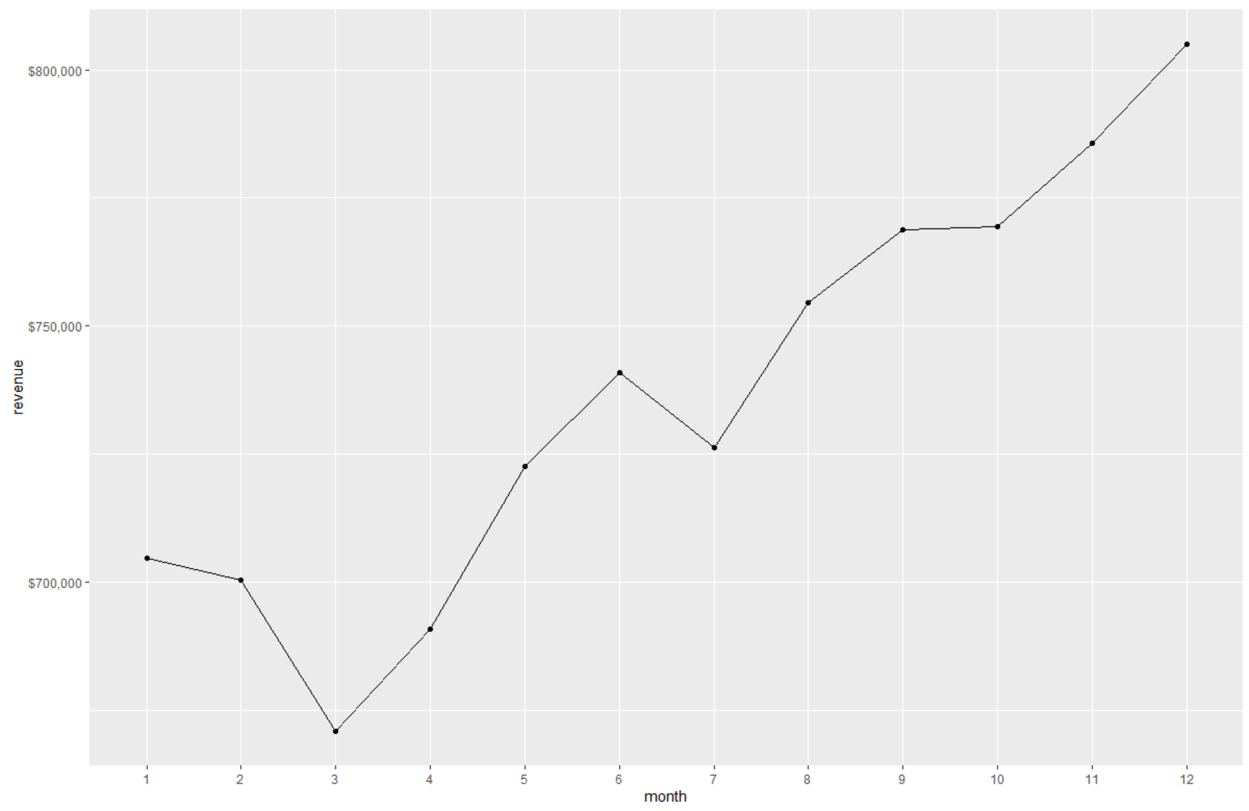


Based on the output, we see that cash and credit card are the two topmost used payment methods, with cash topping the charts. Hence, we can say that, if at all a trip ends up in a dispute or cut short, there is a high possibility that the company might have to suffer a loss. Our company wanted to check whether our customers are ready to go cashless by directly paying from phone like Apple Pay, Android Pay or Samsung Pay but this chart says otherwise, so our company will have to continue to support cash transactions.

Next, we will look at how much is the revenue of each month with numbers of trips.

7. Revenue of each month with the number of trips in each month being constant

The idea behind this question is to find out the total revenue made during each month over the year. We can then compare the monthly revenue and decide if the company needs to deploy more number of taxis during the busier months in order to complete more trips which in turn will yield higher revenue.

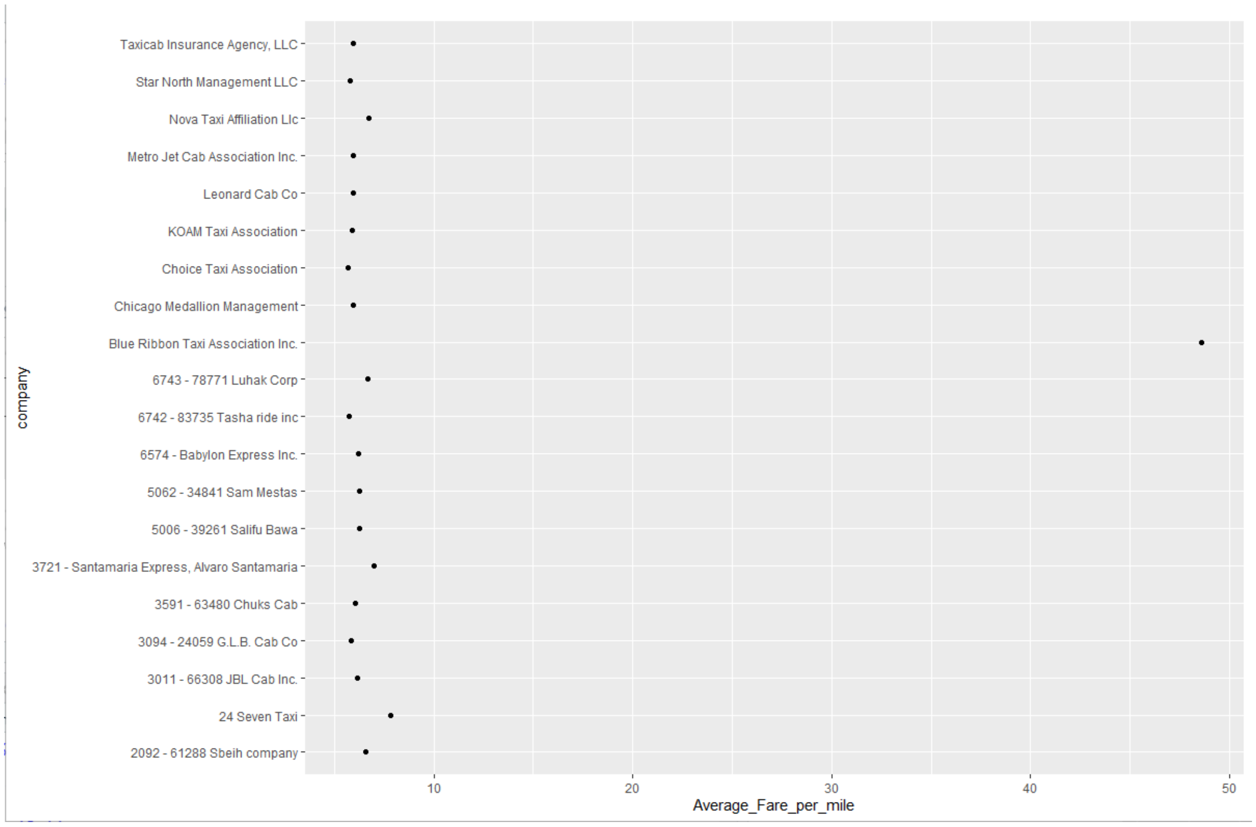


Based on the output, we see that the company starts off the year making decent revenue around $700,000 for January & February which is the winter season and people are more likely to book a taxi to avoid walking. Then in March 2020, the revenue took a hit since COVID Pandemic made the nation go into lockdown and revenue went below $650,000. Then in April until June, revenue started to pickup even though lockdown was still on, we believe the revenue started picking up since essential workers like doctors, nurse, factory workers and other essential workers. Then in July, lockdown was lifted in Illinois, but revenue went down a bit to $725,000 for unknown reason even though everyone was back to work and those needed taxi to get to work would rent a taxi. Then from August until December, revenue kept rising and revenue went up to $800,000 which is not bad considering that year 2020 was not a normal year. We notice that there is a positive trend of revenue throughout the year across the holiday season, during which we need to have the maximum number of taxis deployed in the city.

Next, we will investigate fare per mile for each company.

8. Fare per mile for each company

The idea behind this question is to compute the average fare per mile charged by different companies with respect to their services. By comparing these rates, we can get an idea of what an average fare per mile rate looks like, based on which we can set our fare price. So we are picking top 20 companies.

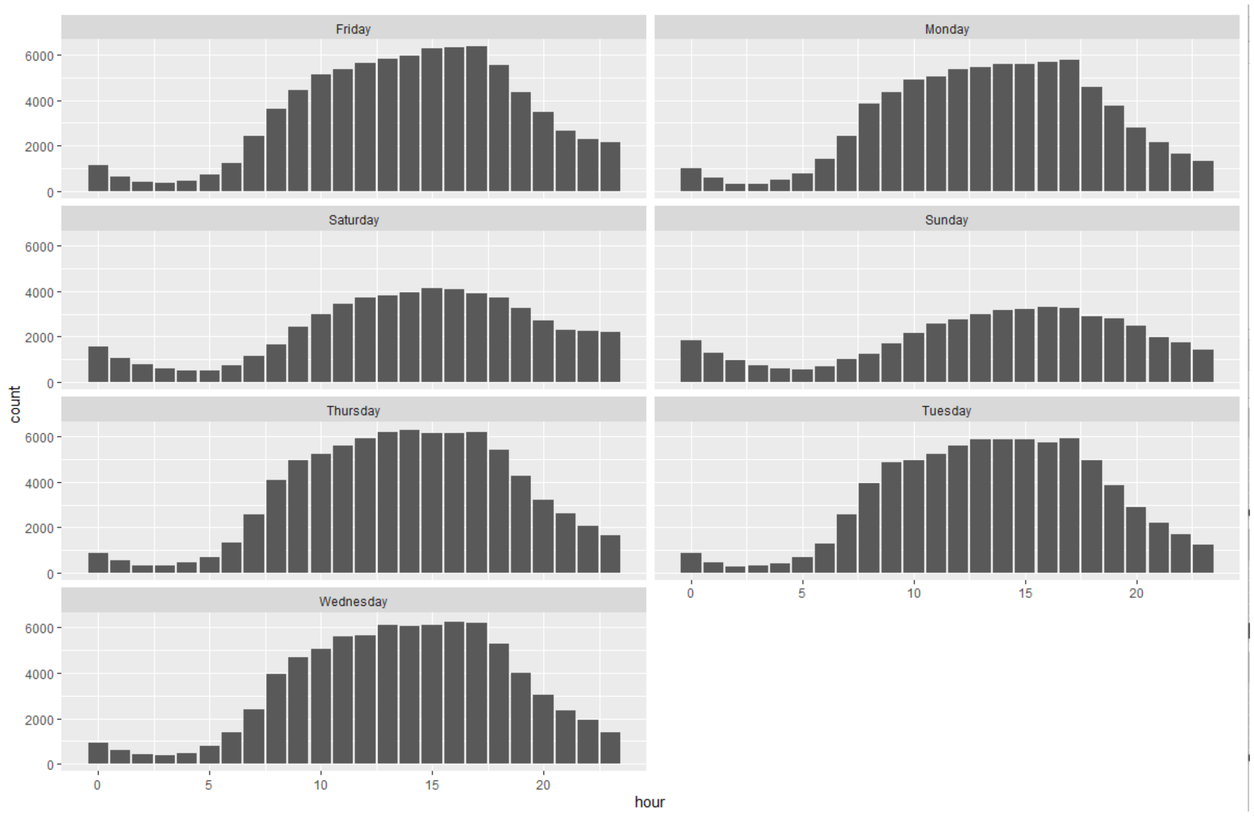


Based on the output, we can set a comparatively less than average rate for fare per mile while offering the same services as some of the top taxi companies, at least during the initial days the company enters the market to draw attention from as many people as possible. We can possibly set fare rate between 0 to 10 dollars per mile to be competitive with other companies.

Next, we will look at number of trips for each weekday.

9. Number of trips for each weekday

The idea behind this question is to identify the busiest days during a week so that we as a company can have more number of taxis deployed on a particular day to meet the expected demand.



Based on the output, we can say that the demand is fairly high during all the weekdays and not as much on the weekends. Days starting from Monday until Friday are equally busy from morning until evening which may suggest that most people need a ride to get to work and then on their way back home, whereas on the weekend, which is Saturday and Sunday it only starts getting busy after noon and extends until later in the evening. This could be a good time to deploy the maximum number of taxis during the weekday.

Project Part 2

In project part 1, our group made changes according to the professor's feedback, and we have written a simple explanation in form of comment each line of code that needed explanation. The changes are as listed below:

- We corrected spelling errors and made a slight change to story in introduction.

- Added comments to explain why we ran the code we ran under tidying dataset section.

- Our dataset had to add community areas columns from community area csv file to make our dataset complete and usable to perform better analysis since the community area csv file had pickup and dropoff community area names with the codes.

- We ran code to separate one column into 4 to 5 different column for example we separated date & time column which was in "MM/DD/YYYY HH:MM" format into separate columns like date, month, year, hour, minutes and weekday.

- We added comments to explain why we had to change datatypes for certain columns.

- Question 1: We added extra variable revenue to perform better analysis like you suggested.

- Question 2: We added extra variable total trips to perform better analysis like you suggested. We have explained why we consider trip total as a revenue. Also, to fix the scaling error, we have added extra "Scales" package to the library to properly display the scale. We also wrote explanation why we didn't included Uber & Lyft into our analysis.

- Question 3: We removed count and added variable average trip total and total trips variable to perform better analysis.

- Question 5: We removed count and added total trips and average trip fare to perform better analysis.

- Question 6: We removed multiple color to represent all the different form of payment methods and added color to represent average trip total variable which we also added for part 2 and added total payments.

- Question 7: We have fixed the scaling issue that we had earlier in the chart by adding extra "Scales" package to the library to properly display the scale.

- Question 8: We added average fare per mile variable to perform better analysis.

- Question 9: We removed color from visualization based on your feedback.

K-Mean Clustering

K-Means clustering is done to generate the models to analyze how month, weekday and community_area_code variables are divided into different clusters based on variables: trip_miles, trip_fare and trip_total. This helps us to make business decisions for complete group of observations in the cluster as they are similar.

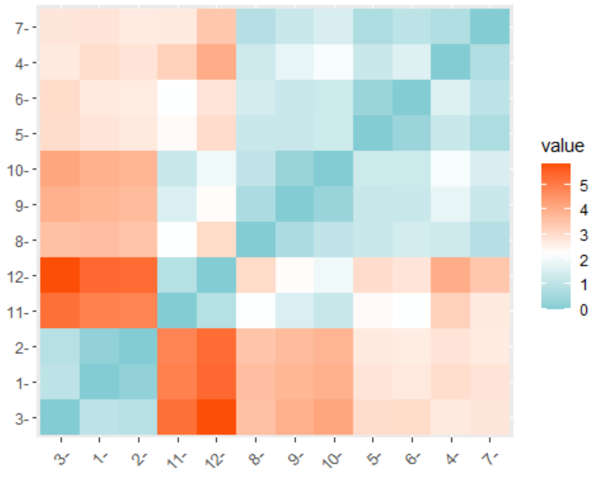
Clustering Model: month~trip_miles+trip_fare+trip_total

```
#clustering for months in R

taxi_trips_cluster_month <- taxi_trips_2020_final %>%
  na.omit() %>%
  group_by(month) %>%
  summarise(mean_trip_miles = mean(trip_miles, na.rm=T),
            mean_trip_fare = mean(fare, na.rm=T),
            mean_trip_total = mean(trip_total, na.rm=T),
            n = n())
```

For creating the distance matrix for month variable, we are extracting the month variable observation into rownames.

```
##   mean_trip_miles mean_trip_fare mean_trip_total     n
## 1        3.123338       12.82365        15.37896 37930
## 2        3.081541       12.84686        15.40292 37803
## 3        2.987314       12.48871        14.83303 37508
## 4        3.612567       14.32201        15.04670 36347
## 5        3.726548       14.86351        15.81414 36785
## 6        3.737657       14.82675        16.08225 36937
```

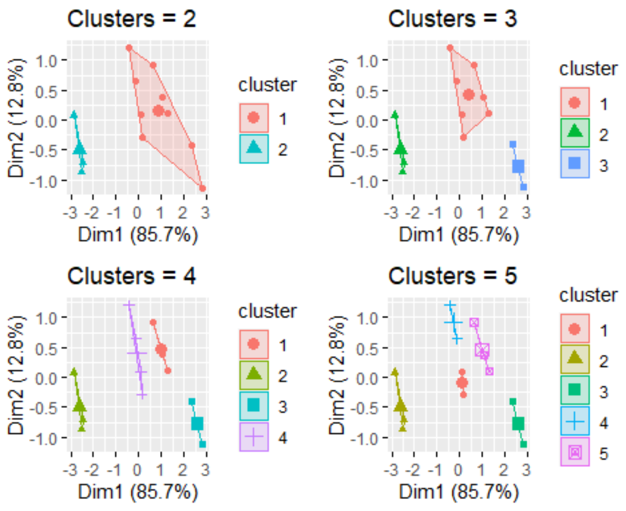
The data for K-Means clustering is being scaled to have same units and distance matrix is created using get_dist function followed by visualizing the distance and (dis)similarity of the observations with each other.



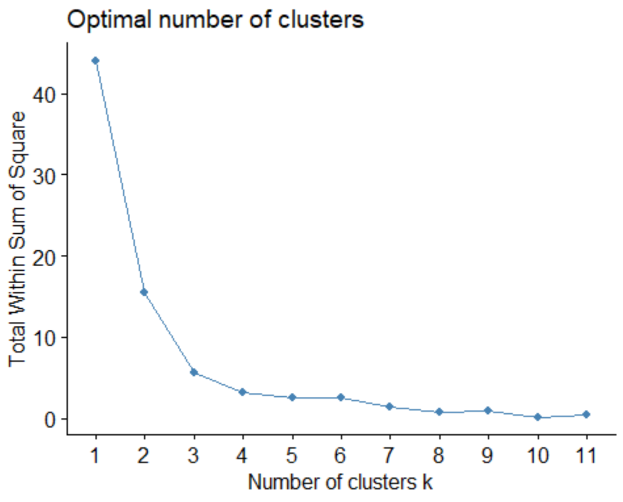
From the above chart, we can summarize that months 1,2&3 are similar to each other, 4 to 10 are similar to each other and 11,12 are mostly dissimilar with other months. With the given information, we can know that, we can divide all the months into three different clusters. For better clarity on optimum clusters, we will be performing other methods in the further steps.

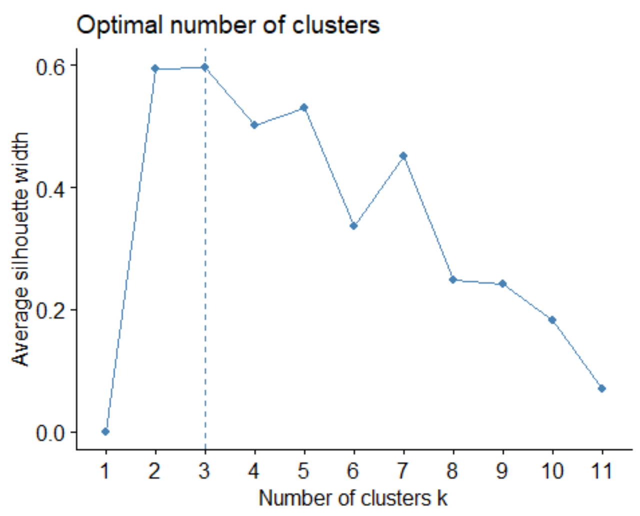
Performing K-means clustering starting with 2,3,4&5 as clusters.

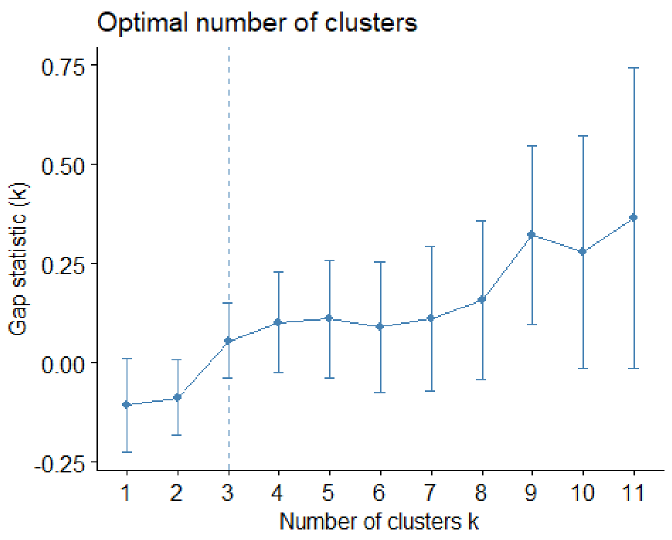
Visualizing all the models with different clusters.



Performing methods for optimum number of clusters:





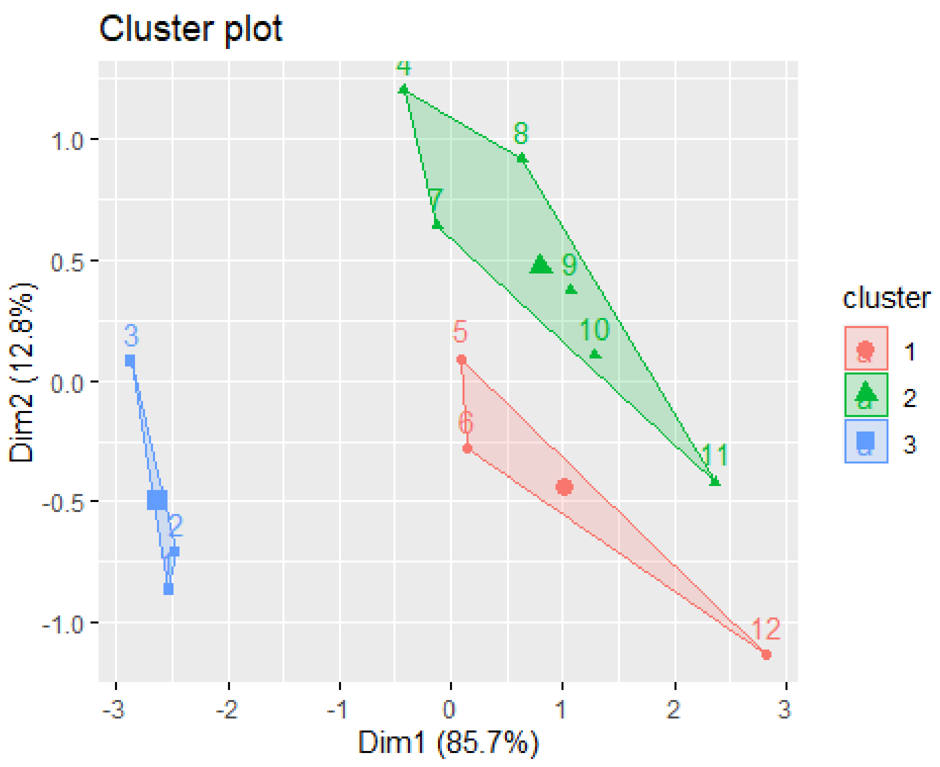


From the above charts, Elbow graph and gap statistics methods indicate that 3 is the optimum clusters we can for this set of observations. So, we are moving forward to compute and visualize the results when clusters are 3.

Extracting the results for optimum number of clusters.

```
## K-means clustering with 3 clusters of sizes 3, 6, 3
##
## Cluster means:
##   mean_trip_miles mean_trip_fare mean_trip_total        n
## 1        3.960460       15.40166        16.52748 36774.67
## 2        3.840422       14.94121        16.16927 36285.17
## 3        3.064064       12.71974        15.20497 37747.00
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12
##  3  3  3  2  1  1  2  2  2  2  2  1
##
```

```
## Within cluster sum of squares by cluster:
## [1]  56276.89 107707.88  93746.30
##  (between_SS / total_SS =  94.3 %)
```



```
## # A tibble: 3 x 5
##   cluster mean_trip_miles mean_trip_fare mean_trip_total      n
##     <int>           <dbl>          <dbl>           <dbl>  <dbl>
## 1       1            3.96           15.4            16.5 36775.
## 2       2            3.84           14.9            16.2 36285.
## 3       3            3.06           12.7            15.2 37747
```

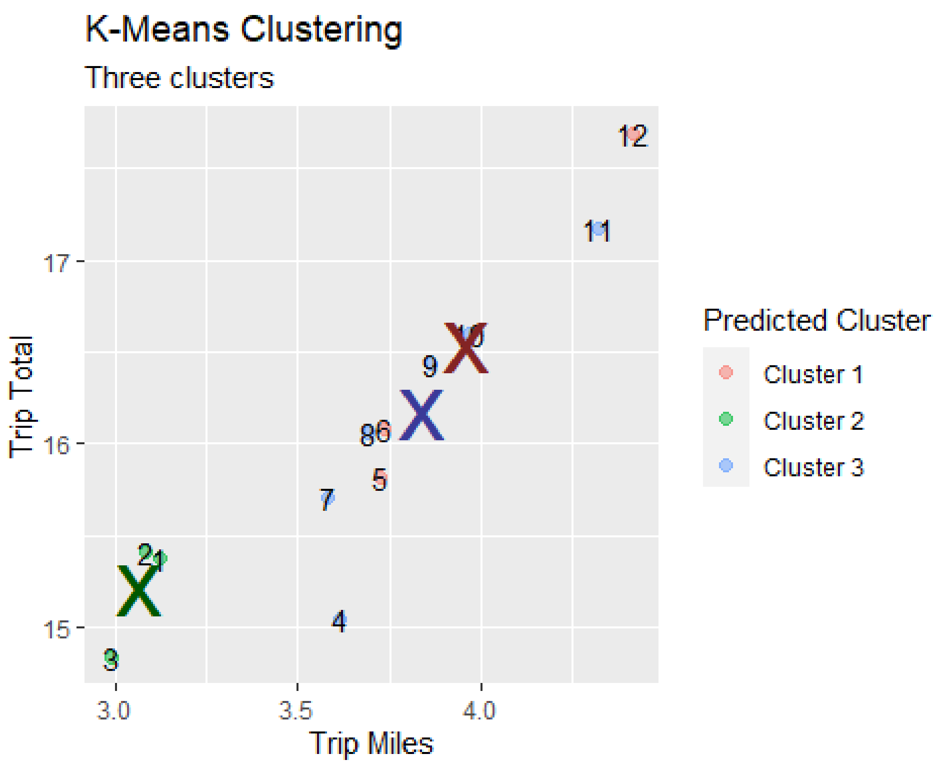
From the resultant chart, months 5,6&12 are grouped into 1st cluster because of the similarities between each other and compared to other two clusters, cluster 1 has highest mean total as months 5,6 are prime summertime and month 12 has lot of events so people prefer travelling more which in result generate high revenues. Whereas in 3rd cluster, months 1,2,3 are grouped as they are similar because of the weather in Chicago, and they have lowest means compared to other 2 clusters.

As we have already found 3 as the optimum clusters, we are analysing K-Means clustering using Spark. To analyze in spark, Spark connection has been created and loaded object "taxi_trips_2020_final" into spark environment for further use.

Genrating K-Means model with centers as 3 and visualizing the results.

```
## K-means clustering with 3 clusters
##
```

```
## Cluster centers:
##   mean_trip_miles mean_trip_fare mean_trip_total        n
## 1        3.960460       15.40166        16.52748 36774.67
## 2        3.064064       12.71974        15.20497 37747.00
## 3        3.840422       14.94121        16.16927 36285.17
##
## Within Set Sum of Squared Errors =  257731.1

## Joining, by = "month"

##
##          0     1     2
##   1      0 42000     0
##   2      0 42000     0
##   3      0 42000     0
##   4      0     0 42000
##   5  42000     0     0
##   6  42000     0     0
##   7      0     0 42000
##   8      0     0 42000
##   9      0     0 42000
##   10     0     0 42000
##   11     0     0 42000
##   12 42000     0     0
```



As we saw the charts above, we can see that month 5, 6 in Cluster 1; month 7, 8, 9, 10 in Cluster 3 brought in trip total between $15.50 & $17 per trip for trip miles between 3.5 to

4.0 miles. The trip total between $15.50 & $17 is for trip miles between 3.5 to 4.0 miles is good in months 5, 6, & 7 since those are summer months so everyone would have destination, they would like to attend quickly so its better to get a taxi than taking other modes of transportation which takes times. Month 11 in Cluster 3 and month 12 in Cluster 1 brings in trip total above $17 per trip for trip miles above 4.0 since these months have events coming up so people would go out and celebrate so they will be more likely to rent a taxi to get to their destination. Whereas month 1, 2, 3 in Cluster 2 brought in trip total between $14.5 & $15.5 for trip miles between 3.0 & 3.5 & month 4 in Cluster 3 brought in trip total about $15.0 for trip miles between 3.5 & 4.0 since all these months are winter months and these months do not have many events coming up so not many people would go out to places and travel much.

Given the page limits constrains we are only reporting and interpreting the results on one clustering model. However, the code for the other two models is available from lines 613 - 884 of the RMarkdown file.

Conclusion:

After creating three different clustering models for variables: month, weekday and community_area_code, we can draw some inferences that months 5,6,12 are more profitable compared to other months, weekdays Saturday and Sunday has more revenue generated days and community areas like ArcherHeights, Ashburn, AvalonPark, etc., are areas where there are high chances of making more trips and generate more revenue. Based on the inferences, we can make business decisions based on the models to survive and become a better company.

Linear Regression:

We will now be doing a linear regression model in order to try to see how our trip total is affected by other variables in our dataset. We can use this data in order to try to figure what variable most affects the trip total so we can better optimize that variable in order to give the best price to our customers.

When starting our regression model, the first thing we wanted to do was to create a sample of our dataset. The reason we decided to do a sample was because our dataset is large and to make the computations faster, we wanted to use a smaller dataset. We also wanted to remove any unnecessary columns which are not needed for the regression analysis. The variables that we will use for our regression analysis are trip_total, which is the "y" variable that we are looking to analyze, as well as trip_seconds, trip_miles, tolls, tips, and extras. These variables can be used as our predictor variables. All of these variables are either directly or indirectly affecting our trip_total. The one other variable that has an effect on the trip_total is the fare. However, we felt that this has a very obvious and direct impact on the total, so much so that we felt that we should remove it on order to get a better understanding of how much the other variables can explain the variance of our trip total.

```r
#sampling dataset
taxi_sample <- taxi %>%
  sample_n(100000) %>%
  select(2, 9, 10, 14, 15, 16, 17)
```

Once we created the sample dataset, we needed to make sure we remove all of the NA values for that variables we will be using from the dataset. This will allow the regression model to run smoothly and avoid any possible errors while running the code.

```r
#removing all NA values from the sample dataset
taxi_sample2 <- taxi_sample %>%
  filter(!(is.na(trip_total))) %>%
  filter(!(is.na(trip_seconds))) %>%
  filter(!(is.na(trip_miles))) %>%
  filter(!(is.na(tolls))) %>%
  filter(!(is.na(tips))) %>%
  filter(!(is.na(extras)))
```

After the data is cleaned, we wanted to create another column in our dataset called trip_minutes. With this new column, we will have a better scale for our regression output because trip_seconds give us very large numbers. Trip_minutes will give us number more comparable to the other variables. This should give us a better scaling for our coefficients.

```r
#creating new column called trip_minutes
taxi_reg <- taxi_sample2 %>%
  mutate(trip_minutes = trip_seconds/60)
```

Once we have all the variables we want to analyze, it is now time to start creating our train and est sample. Our train sample is the sample data that we will use to create our regression model. The test sample is the data we will use to test our model on once the model is created. We split the dataset into 60% train and 40% test. This gives us a majority of our dataset to be used to create the model. This will hopefully make the model more accurate. We will set the seed number to 12345 in order the replicate the same samples every time we run the regression again.

```r
set.seed(12345)
sample <- sample(c(TRUE, FALSE), nrow(taxi_reg), replace = T, prob = c(0.6,0.
4))
train <- taxi_reg[sample, ]
test <- taxi_reg[!sample, ]
```

Now that we have created our train dataset, we can use it in order to create the model. We first wanted to create a multiple regression analysis. The difference between multiple regression and simple regression is that with multiple regression, we are taking more predictor variables into account when running the model. This will hopefully allow more of the variance of the trip_total to be explained. The variables that we are planning to add are trip_miles, trip_minutes, tolls, tips, and extras. All of these variables seem to make logical sense to add because we feel like they all have a positive correlation with trip total. We can see exactly what the correlation is by using the cor function.

```
taxi_reg %>%
  select(!(trip_id)) %>%
  cor()

##              trip_seconds  trip_miles       tips       tolls     extras
## trip_seconds  1.000000000 0.266598656 0.05963987 0.006994085 0.04747509
## trip_miles    0.266598656 1.000000000 0.24831116 0.007305131 0.19198636
## tips          0.059639865 0.248311161 1.00000000 0.004298200 0.25265629
## tolls         0.006994085 0.007305131 0.00429820 1.000000000 0.08596324
## extras        0.047475092 0.191986362 0.25265629 0.085963242 1.00000000
## trip_total    0.309298977 0.666687336 0.40244800 0.069596761 0.43470026
## trip_minutes  1.000000000 0.266598656 0.05963987 0.006994085 0.04747509
##              trip_total trip_minutes
## trip_seconds 0.30929898  1.000000000
## trip_miles   0.66668734  0.266598656
## tips         0.40244800  0.059639865
## tolls        0.06959676  0.006994085
## extras       0.43470026  0.047475092
## trip_total   1.00000000  0.309298977
## trip_minutes 0.30929898  1.000000000
```

You can see based on the correlation output that if you look under the trip_total column, all of the variables seem to have a positive relationship with trip_total with trip_miles being the most correlated and tolls being the least correlated. The tolls correlation seems to very low however, so we have decided to not include the variable because there is not as big of a positive correlation as we initially thought. The remaining variables that we will add are trip_minutes, tips, and extras.

We will now use these variables to run our multiple regression model.

```
model2 <- lm(trip_total ~ trip_minutes + trip_miles + tips + extras, data = t
rain)
```

After the model is run, we once again look at the summary of the model as well as the confidence interval to determine how significant our model is as well as to help make our predictions.

```
summary(model2)

##
## Call:
## lm(formula = trip_total ~ trip_minutes + trip_miles + tips +
##     extras, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -954.29   -3.60   -2.20    1.52  918.46
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.540587   0.060777  124.07   <2e-16 ***
## trip_minutes 0.082854   0.001511   54.84   <2e-16 ***
## trip_miles   1.528794   0.008139  187.83   <2e-16 ***
## tips         1.428504   0.019031   75.06   <2e-16 ***
## extras       1.223089   0.011795  103.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.91 on 59893 degrees of freedom
## Multiple R-squared:  0.5956, Adjusted R-squared:  0.5955
## F-statistic: 2.205e+04 on 4 and 59893 DF,  p-value: < 2.2e-16
```

```
tidy(model2)
```

```
## # A tibble: 5 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    7.54      0.0608     124.         0
## 2 trip_minutes   0.0829    0.00151     54.8        0
## 3 trip_miles     1.53      0.00814    188.         0
## 4 tips           1.43      0.0190      75.1        0
## 5 extras         1.22      0.0118     104.         0
```

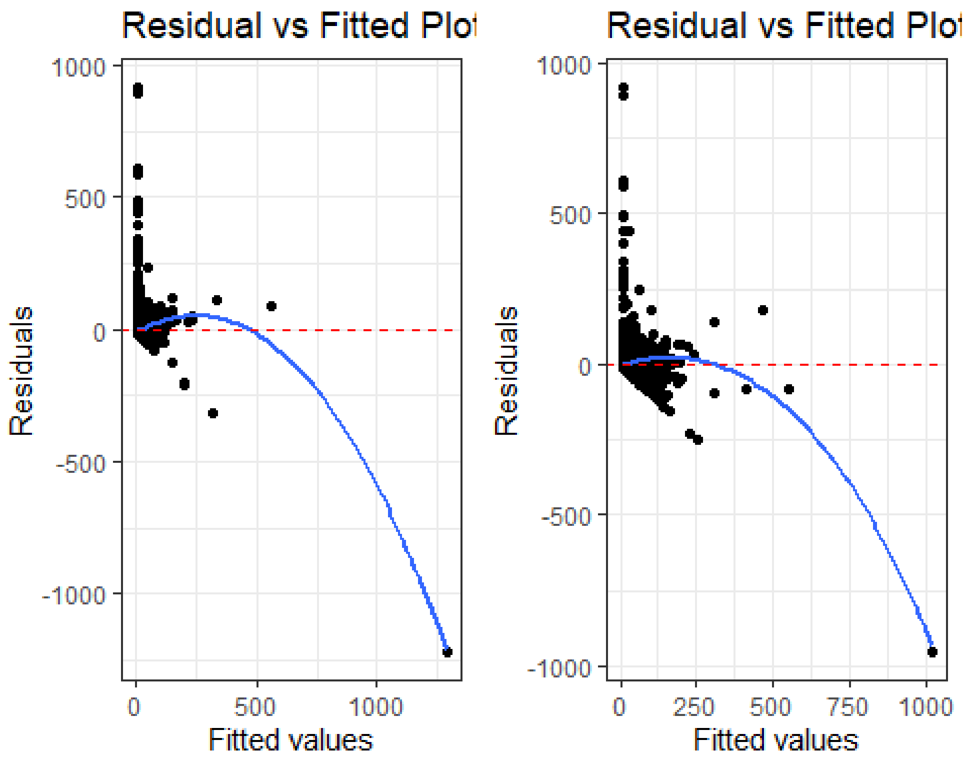
```
confint(model2)
```

```
##                   2.5 %     97.5 %
## (Intercept)  7.42146318 7.65971031
## trip_minutes 0.07989294 0.08581573
## trip_miles   1.51284122 1.54474762
## tips         1.39120346 1.46580544
## extras       1.19996967 1.24620768
```

By looking at the summary, we can see that all the predictors as well as the model as a whole has a much smaller p-value than our alpha of 0.05. This makes the predictors and our model significant. We can also see that the r-squared is 0.5956 which means that about 59.56% of the variance in trip_total can be explained by our model. This is about a 16% increase from the simple linear regression of just trip_miles (simple linear regression can be found in the rmd file from line 932- 972). While it did increase, it did not increase as high as we had thought it would. This shows us that a lot of variance is explained by the “fare”. Which makes perfect sense to us because the fare is ultimately a majority of the value of the trip total.

We next wanted to visualize the model to see how much the linearity and homoscedasticity improved from our previous model.

```
model2_results <- augment(model2, train) %>%
  mutate(Model = "Model")

p1 <- ggplot(model_results, aes(.fitted, .resid))+
  geom_point()+
  geom_smooth(method="loess", se = FALSE)+
  geom_hline(yintercept=0, col="red", linetype="dashed")+
  xlab("Fitted values")+
  ylab("Residuals")+
  ggtitle("Residual vs Fitted Plot (simple)")+
  theme_bw()

p2 <- ggplot(model2_results, aes(.fitted, .resid))+
  geom_point()+
  geom_smooth(method="loess", se = FALSE)+
  geom_hline(yintercept=0, col="red", linetype="dashed")+
  xlab("Fitted values")+
  ylab("Residuals")+
  ggtitle("Residual vs Fitted Plot (multiple)")+
  theme_bw()

gridExtra::grid.arrange(p1, p2, nrow = 1)

## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



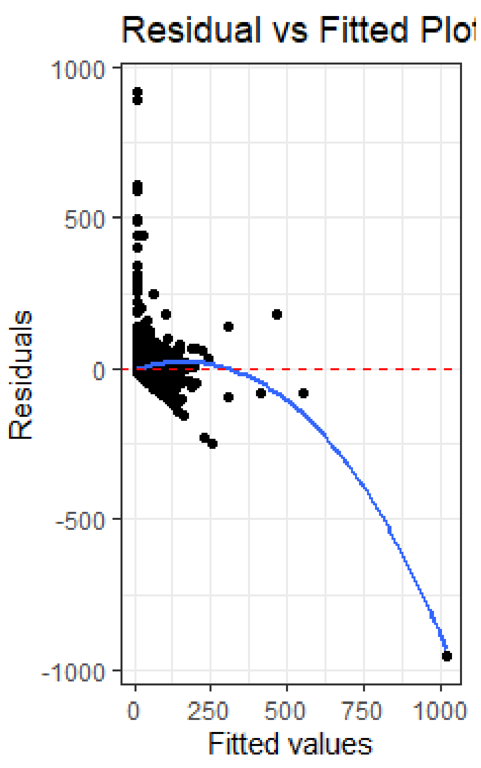
We can see by comparing the 2 graphs that there is definitely an improvement on linearity, however, not by too much. The left graph is our simple regression model and the right is the multiple regression model. The line still seems to slope downward even though it stay linear a little bit longer in the beginning. Homoscedasticity is much harder to see if there is improvement on. There does not seem to be much improvement.

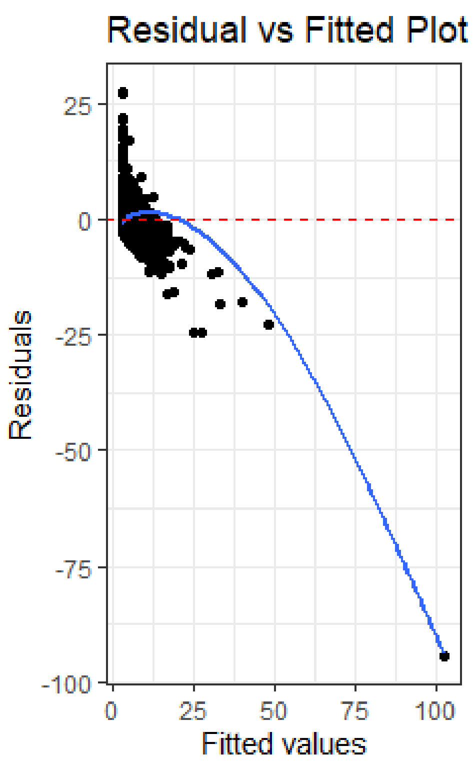
The next thing that we can try to improve linearity and homoscedasticity is by applying a transformation. We can try to apply a square root transformation to the Y variable in order to change the scaling of the residuals and see if there is possibly a more even variance in the residuals.

```
model_sqrt <- lm(sqrt(trip_total) ~ trip_minutes + trip_miles + tips + extras
, data = train)

summary(model_sqrt)

##
## Call:
## lm(formula = sqrt(trip_total) ~ trip_minutes + trip_miles + tips +
##     extras, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -94.428  -0.521  -0.130   0.443  27.549
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.8807976  0.0051872  555.37   <2e-16 ***
## trip_minutes 0.0082358  0.0001290   63.87   <2e-16 ***
## trip_miles   0.1505088  0.0006947  216.66   <2e-16 ***
## tips         0.1417422  0.0016243   87.27   <2e-16 ***
## extras       0.0816663  0.0010067   81.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.016 on 59893 degrees of freedom
## Multiple R-squared:  0.6344, Adjusted R-squared:  0.6344
## F-statistic: 2.598e+04 on 4 and 59893 DF,  p-value: < 2.2e-16

tidy(model_sqrt)

## # A tibble: 5 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   2.88     0.00519      555.        0
## 2 trip_minutes  0.00824  0.000129      63.9       0
## 3 trip_miles    0.151    0.000695     217.        0
## 4 tips          0.142    0.00162       87.3       0
## 5 extras        0.0817   0.00101       81.1       0

confint(model_sqrt)

##                    2.5 %      97.5 %
## (Intercept)  2.870630654 2.890964511
## trip_minutes 0.007983067 0.008488564
## trip_miles   0.149147208 0.151870348
## tips          0.138558694 0.144925805
## extras        0.079693150 0.083639461
```

Looking at the summary, we can see that the r-squared is improved by about 4% from the previous normal model. This is a good sign. We will now look at the fitted values vs residuals plot to see how much that has improved.

```
model_results_sqrt <- augment(model_sqrt, train) %>%
  mutate(Model = "Model")

p3 <- ggplot(model2_results, aes(.fitted, .resid))+
  geom_point()+
  geom_smooth(method="loess", se = FALSE)+
  geom_hline(yintercept=0, col="red", linetype="dashed")+
  xlab("Fitted values")+
  ylab("Residuals")+
  ggtitle("Residual vs Fitted Plot (normal)")+
  theme_bw()


p4 <- ggplot(model_results_sqrt, aes(.fitted, .resid))+
  geom_point()+
  geom_smooth(method="loess", se = FALSE)+
  geom_hline(yintercept=0, col="red", linetype="dashed")+
  xlab("Fitted values")+
  ylab("Residuals")+
  ggtitle("Residual vs Fitted Plot (sqrt)")+
  theme_bw()

gridExtra::grid.arrange(p3, p4, nrow = 1)

## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```

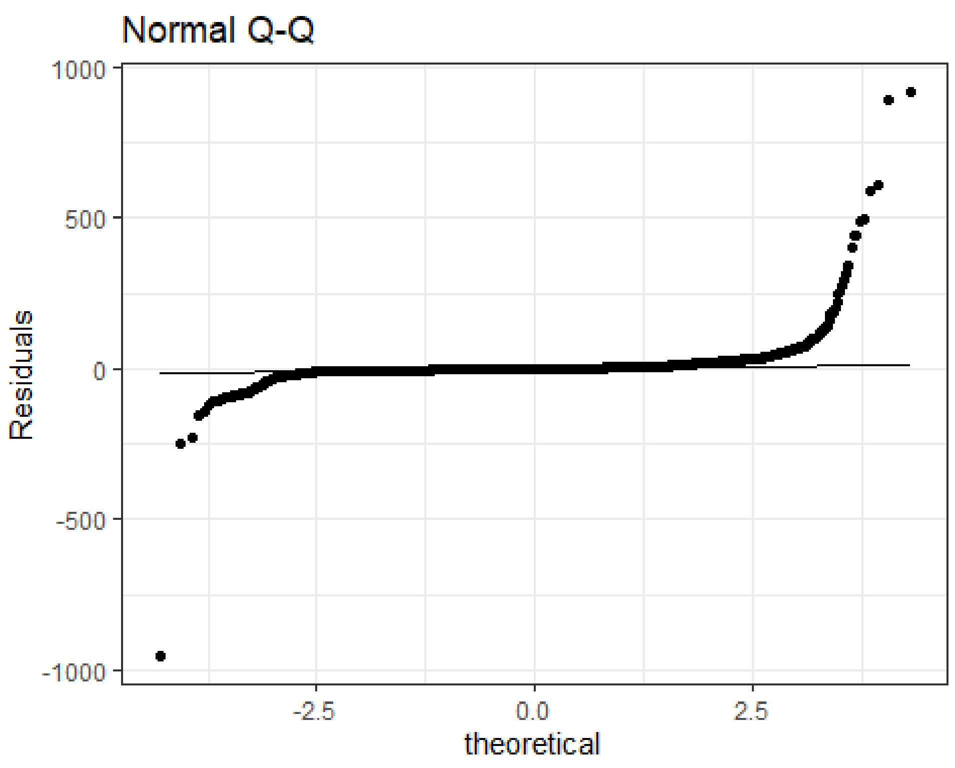




Comparing the 2 graphs together, we noticed that both the linearity and the homoscedasticity seem to get worse. This does not justify the 4% increase in the r-squared. Because of this, we felt that the best model for our regression would be the normal multiple regression model.

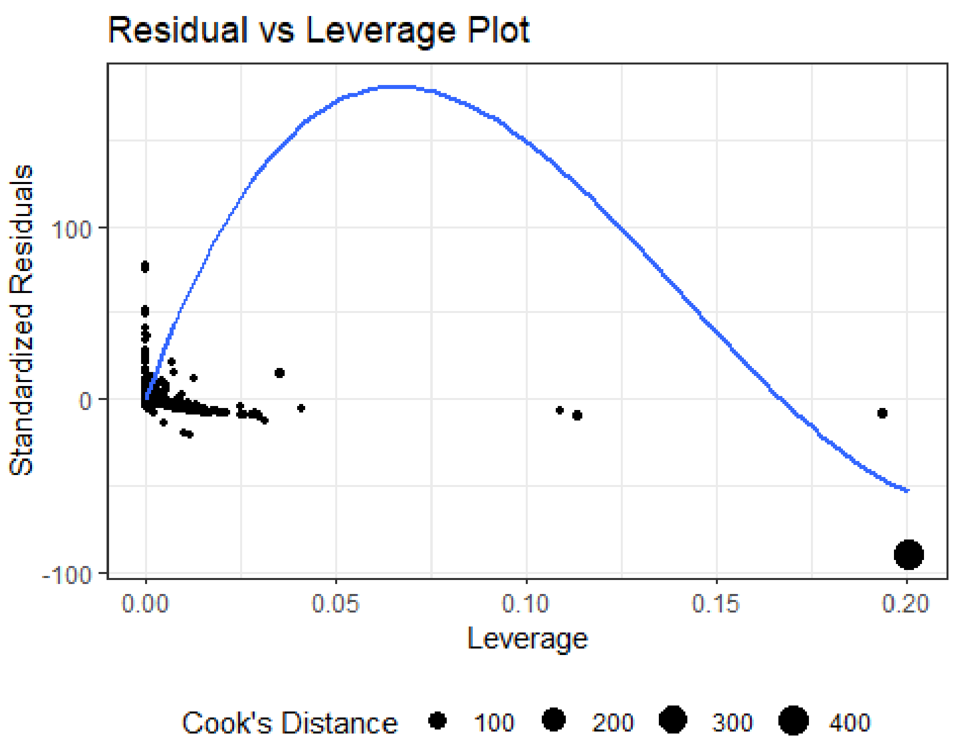
We can try to analyze the residuals in order to see exactly why the linearity and homoscedasticity seems to be so bad. The first plot we will look at is the Q-Q plot which looks at normality of the residuals. With this plot we can see if our residuals are in a normal distribution.

```
ggplot(model2_results, aes(sample = .resid)) +
  stat_qq() +
  stat_qq_line() +
  ylab("Residuals") +
  ggtitle("Normal Q-Q") +
  theme_bw()
```



Looking at the plot, we see that there is sort of a snaking pattern going on with the Q-Q plot. with snaking pattern lets us conclude that the residuals are not in a normal distribution. One reason for this could be cause there are some big influential points that is skewing our residuals in a certain direction. We can identify influential points in our residuals by looking at the cooks distance of the residuals. One way to quantify cooks distance is by looking at the following residual vs leverage plot.

```
ggplot(model2_results, aes(.hat, .std.resid))+
  geom_point(aes(size=.cooksd), na.rm=TRUE)+
  stat_smooth(method="loess", na.rm=TRUE, se = FALSE)+
  xlab("Leverage")+
  ylab("Standardized Residuals")+
  ggtitle("Residual vs Leverage Plot")+
  scale_size_continuous("Cook's Distance", range=c(1,5))+
  theme_bw()+
  theme(legend.position="bottom")

## `geom_smooth()` using formula 'y ~ x'
```



By looking at the plot, we can see the residuals plotted with the size of the point being the value of the cooks distance. We can see right away that there is 1 point that is much larger than the other. This point is the point of concern. We can pull up a table of the top 5 cooks distances to see the value of this point.

```
model2_results %>%
  top_n(5, wt = .cooksd) %>%
  select(.cooksd) %>%
  arrange(desc(.cooksd))
```

```
## # A tibble: 5 x 1
##   .cooksd
##     <dbl>
## 1  403.
## 2    2.58
## 3    1.81
## 4    1.80
## 5    1.20
```

From this table, we can see the values of this influential point. By removing this point, we should be able to better normalize the distribution of our residuals which will in turn, help improve the linearity and the homoscedasticity of our model.

The main takeaway from running the regression models is that we need to find the best way to optimize the pricing algorithm that calculates trip fare. That is ultimately what will drive the price for trip total, Trip mileage seems to have a bigger effect on the trip total than trip minutes, so trip miles should be the main pricing variable when calculating the fare price.

Support Vector Machine (SVM):

We will now be doing a support vector machine in order to try to predict the frequency of tips that is received. With a support vector machine, If the data we are analyzing has a variable with classes, then the SVM will be able to separate those classes on a graph using a line. Then whenever future data comes in, the SVM can try to predict which class that future data will fall into. The variable we will be analyzing is tip_occurence which will tell us if a ride had lead to a tip being received. We will try to see if we can also predict, with new data, if a tip will occur in the future.

In order to start with the support vector machine, we first had to take a sample of our data. We needed to do this due to the slow runtime issues of of a larger dataset. We decided to take a sample of 100k observations.

```
sample_svm <- sample_n(taxi, 100000)
```

After our sample was created, we needed to make sure we got rid of all NA values, as having these would interfere with our analysis.

```
sample_svm2 <- na.omit(sample_svm)
```

After we have our sample dataset, we needed to create the variable we are analyzing. Because support vector machines require a variable with classes, we had to create one called tip occurrence. With this variable, if the taxi ride included a tip, then we put a “yes” for that observation and if there was no tip then we out a “no”.

```
tips_svm <- sample_svm2 %>%
  mutate(tip_occurance = ifelse(tips > 0, "yes", "no"))
```

After we created our class variables, we wanted to filter our dataset to only include the variables that we would need for the support vector machine, because we wanted to see if there would be a trend between tip occurrence vs how long the trip was or how much the trip was, we decided to include trip_miles and trip_total along with trip_occurence.

```
tips_svm2 <- tips_svm %>%
  select(trip_miles, trip_total, tip_occurance)
```

We then once again made sure to remove all NA values, just in case there were any remaining during our manipulations.

```
tip_final <- tips_svm2 %>%
  filter(!(is.na(trip_miles))) %>%
  filter(!(is.na(trip_total)))
```

Next, we needed to make sure that the new class variable we had created is in a factor data type. This is because class is a categorical variable and we need to make sure that R knows that it is a class rather than just a character string.

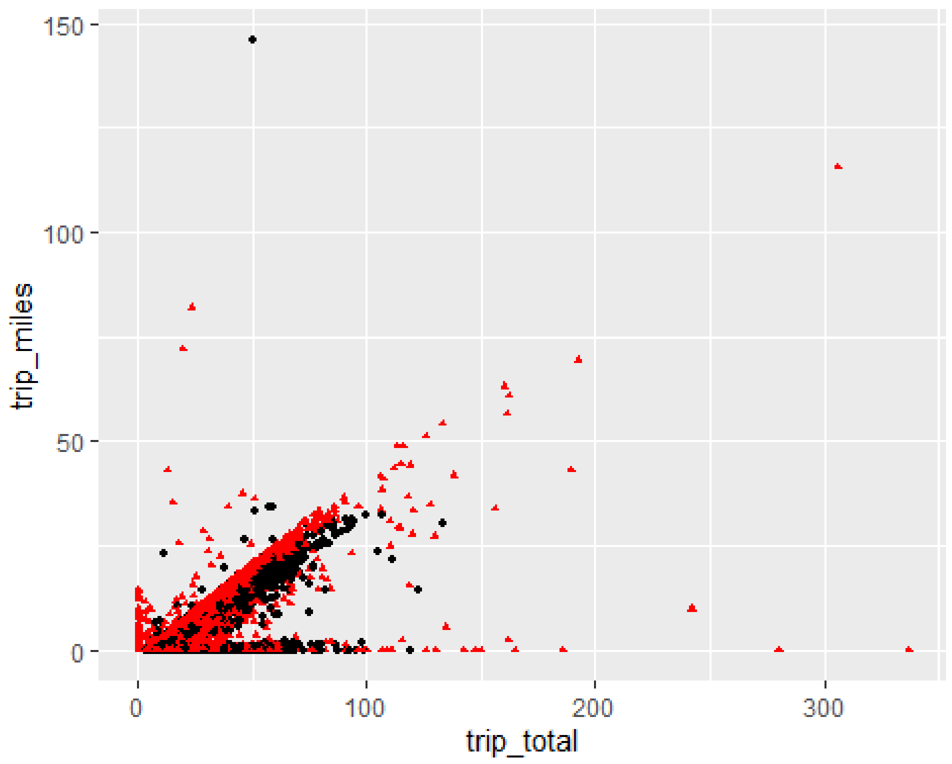
```
tips_svm2$tip_occurance <- factor(tips_svm2$tip_occurance, levels = c("yes","
no"))
```

Now, that our data is completely manipulated the way we need it to be, we can gp ahead with creating our train and test data set. Because we will build this model and then use it to predict future data, we will use the train (60%) in order to build the model and then use the test (40%) as a the new data that the model will use to make predictions.

```
set.seed(12345)
sample <- sample(c(TRUE, FALSE), nrow(tips_svm2), replace = T, prob = c(0.6,0
.4))
train_svm <- tips_svm2[sample, ]
test_svm <- tips_svm2[!sample, ]
```

Now that the data set is split, we can go ahead and plot the classes from the train dataset on the graph so we can visually look to see how the classes are separated.

```
ggplot(data = train_svm, aes(x = trip_total, y = trip_miles, color = tip_occu
rance, shape = tip_occurance)) +
  geom_point(size = 1) +
  scale_color_manual(values=c("#000000", "#FF0000")) +
  theme(legend.position = "none")
```



By looking at the plot, we have the black plots as the “yes” category and the red plots as the “no”. As you can see from the graph, the points seems to be mixed in with each other and it is very tough to notice any trends by just visually looking at them. However, we still want to see of the support vector machine is able to split these classes.

When first building the SVM, we first have to do a tuning of our data in order to figure out the ideal cost and gamma that we should select. This will tell us how strict our model should be when it is creating the lines. Because of runtime issues, we only used a small sample of our train dataset to do the tuning. Only 1000 observations. When we do the tuning function. The kernel we want to use is radial. This is because with radial, the model will draw circles around the classes instead of line. This is good when we have classes mixed in with each other like we do now.

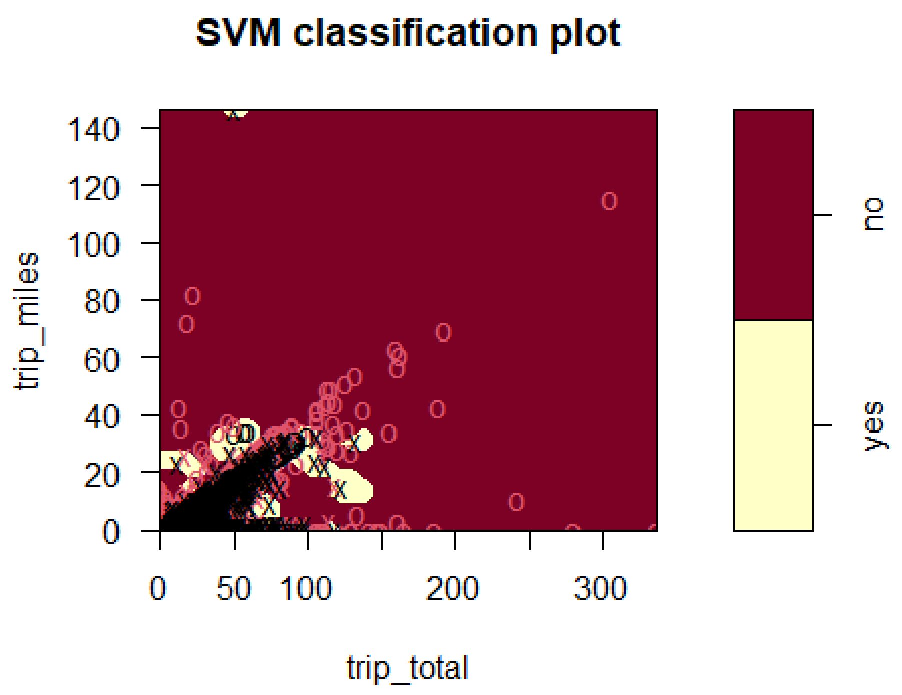
```
tuning_sample <- train_svm %>%
  sample_n(1000)

tune.out <- tune(svm, tip_occurance~., data = tuning_sample, kernel = "radial
",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
                               gamma = c(0.5, 1, 2, 3, 4)))
tune.out$best.model

##
## Call:
## best.tune(method = svm, train.x = tip_occurance ~ ., data = tuning_sample,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100), gamma = c(0.5
,
##         1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  311
```

Based on the output of the tuning function, we can see that the optimal cost for us is 100. It does not tell us the optimal gamma, but we will be going with 0.5, since that is the lowest gamma, and we want to be on the safe side.

Now that we have our optimal settings, we can go ahead and beginning running the SVM model. We ran the model and then plotted the results on a graph.

```
svmfit <- svm(tip_occurance~., data = train_svm, kernel = "radial", cost = 10
0, gamma = 0.5)
```

```
plot(svmfit, train_svm)
```



As you can see from the graph, we have a lot of red spots which indicates just how spread the “no” class is. The model is basically telling us, that in most cases, the predicted tip occurrence will be “no” based on the trip total and trip miles. We do see some area where we have the yellow color for “yes”, however, we still do not see any clear trends.

We can now use this model in order to predict with new data, which category the new data will fall into. So we are essentially trying to see if we can predict the tip occurrence with new data coming in. For the new data, we used the test sample that we created earlier and then ran the prediction to see how many times, the model can place the observation in the correct class.

```
tip_pred <- predict(svmfit, newdata = test_svm)

table(test_svm$tip_occurance, tip_pred) %>%
  prop.table()

##      tip_pred
##              yes         no
##   yes 0.16790095 0.08566609
##   no  0.03963057 0.70680239
```

We can see from the results that the test model had a “yes” to “no” ration of about 20:80. Based on the prediction, about 80% of the “yes” observation was placed in the correct class and about 87% of the “no observations were place in the correct class. This gives us roughly 87% percent accuracy. Based on how random the points were originally plotted, this is a good number. However, the biggest concern for us is a lack of trend. Because of this, we still feel like it will be hard to predict in the real world when exactly these tips will occur. We ultimately will have to focus on teaching the taxi drivers good customer service, because that will be the best way to increase the number of tips.

Lessons Learnt:

Our takeaway from this project is mostly related to how well you understand the data and take the inferences from the output of the data. From the basic manipulations we learnt how to clean the data, wrangle the data, and perform basic operations that helps us to understand the trend or pattern the data follows.

By performing this advanced model like regression, clustering, and Support Vector Machine we learnt the ability to use the data to build models that can be used to predict any other data that is similar in future. Visualization’s techniques that we learned and used throughout this project helped us understand the story the graphics speak more than numbers or words and helped us to convey the information reach across better due to visualizations.

Conclusion:

From our project, we have answered our research question that is analyzing Chicago taxi market in the year 2020 with changes and use the results to help make better business strategies to set up our taxi company “Welcome Taxi” in the year 2021 and compete with our competitors and optimize our revenue. The results that we required from this project and the inferences that we concluded will help us achieve a competitive advantage over other players in market.