Analyzing and Visualizing Uber and Lyft Data Set

2023-04-28

Introduction

Uber and Lyft are two popular ride-hailing services that allow users to request rides from drivers through their apps. Uber is available in more countries and cities than Lyft, but Lyft has a larger market share in the United States. Both services have similar features, but there are some key differences, such as their pricing models.

We have a dataset of ride data from Boston, Massachusetts. We will use this data to analyze the factors that affect dynamic pricing and to compare the pricing models of Uber and Lyft.

Here are some of the factors that we will consider:

- Time of day
- Day of week
- Location
- Weather
- Traffic

We will also consider the influences of surge rate for Lyft

- Peak hour
- Traffic
- Number of available rides in the area

We believe that this analysis will provide valuable insights into the factors that affect dynamic pricing and the differences between Uber and Lyft's pricing models.

About the data set

The data set is taken from Kaggle. It was collected from 11-26-18 to 12-18-18 in Boston, MA at regular intervals of 5 mins. The data was gathered for Uber and Lyft. It has 637,976 rows of data, the split between Uber and Lyft rides was roughly 50/50.

NOTE: The data has been collected from different sources, including real-time data collection using Uber and Lyft API (Application Programming Interface) queries. Link: https://www.kaggle.com/datasets/brllrb/uber-and-lyft-dataset-boston-ma

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 7.2.1; sf_use_s2() is TRUE

## Loading required package: viridisLite

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

##
## Attaching package: 'scales'

## The following object is masked from 'package:viridis':
##
##     viridis_pal

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

rideshare <- read.csv("rideshare_kaggle.csv")
```

Data Cleaning

Removing missing values

```
print("Count of missing values by column wise")

## [1] "Count of missing values by column wise"

sapply(rideshare, function(x) sum(is.na(x)))

##                         id                  timestamp
##                          0                          0
##                       hour                        day
##                          0                          0
##                      month                   datetime
##                          0                          0
##                   timezone                     source
##                          0                          0
##                destination                   cab_type
##                          0                          0
##                 product_id                       name
```

```
##                           0                           0
##                       price                    distance
##                       55095                           0
##            surge_multiplier                    latitude
##                           0                           0
##                   longitude                 temperature
##                           0                           0
##         apparentTemperature               short_summary
##                           0                           0
##                long_summary             precipIntensity
##                           0                           0
##           precipProbability                    humidity
##                           0                           0
##                   windSpeed                    windGust
##                           0                           0
##               windGustTime                  visibility
##                           0                           0
##             temperatureHigh         temperatureHighTime
##                           0                           0
##              temperatureLow          temperatureLowTime
##                           0                           0
##     apparentTemperatureHigh apparentTemperatureHighTime
##                           0                           0
##      apparentTemperatureLow  apparentTemperatureLowTime
##                           0                           0
##                        icon                    dewPoint
##                           0                           0
##                    pressure                 windBearing
##                           0                           0
##                  cloudCover                     uvIndex
##                           0                           0
##               visibility.1                       ozone
##                           0                           0
##                sunriseTime                 sunsetTime
##                           0                           0
##                   moonPhase          precipIntensityMax
##                           0                           0
##                uvIndexTime              temperatureMin
##                           0                           0
##          temperatureMinTime              temperatureMax
##                           0                           0
##          temperatureMaxTime      apparentTemperatureMin
##                           0                           0
##  apparentTemperatureMinTime      apparentTemperatureMax
##                           0                           0
##  apparentTemperatureMaxTime
##                           0
```

“Price” has 55095 missing values. So, we will remove all the NA values from the dataset

```
data <- as_tibble(rideshare)
# removing all NA values from the dataset
data <- data[complete.cases(data), ]
```

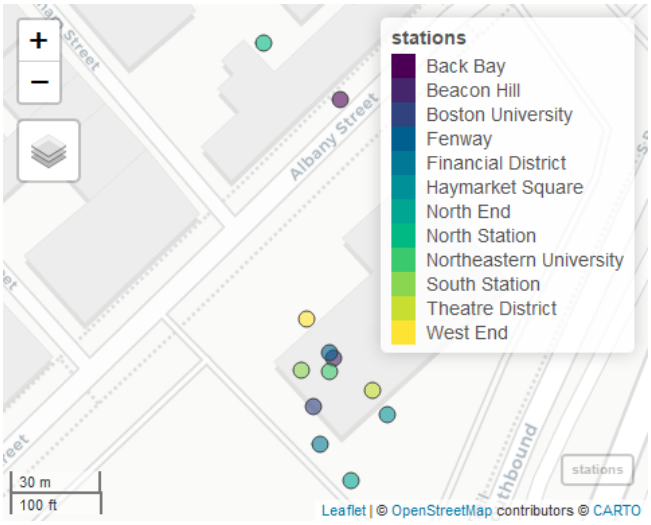
Adding longitude and latitude values for destinations also

Our data set had longitude and latitude value for source column only, so we have added coordinated for destination by taking individual coordinates for each location and merging it with destination column

```
# create a new dataframe with to get name of the place and its coordinates
df_label <- data %>%
  select(source, latitude, longitude) %>%
  group_by(source) %>%
  summarize(latitude = mean(latitude),
            longitude = mean(longitude))

# rename source with label_coord to avoid confusion while merging dataframes
df_label <- df_label %>%
  rename(label_coord = source)
#df_label

df <- left_join(data, df_label, by = c("destination" = "label_coord"))
df1<- left_join(df, df_label, by = c("source" = "label_coord"))

# chunk 4
data <- df1 %>%
  rename(source_lat = latitude,
         source_lon = longitude,
         destination_lat = latitude.y,
         destination_lon = longitude.y)

#data
```

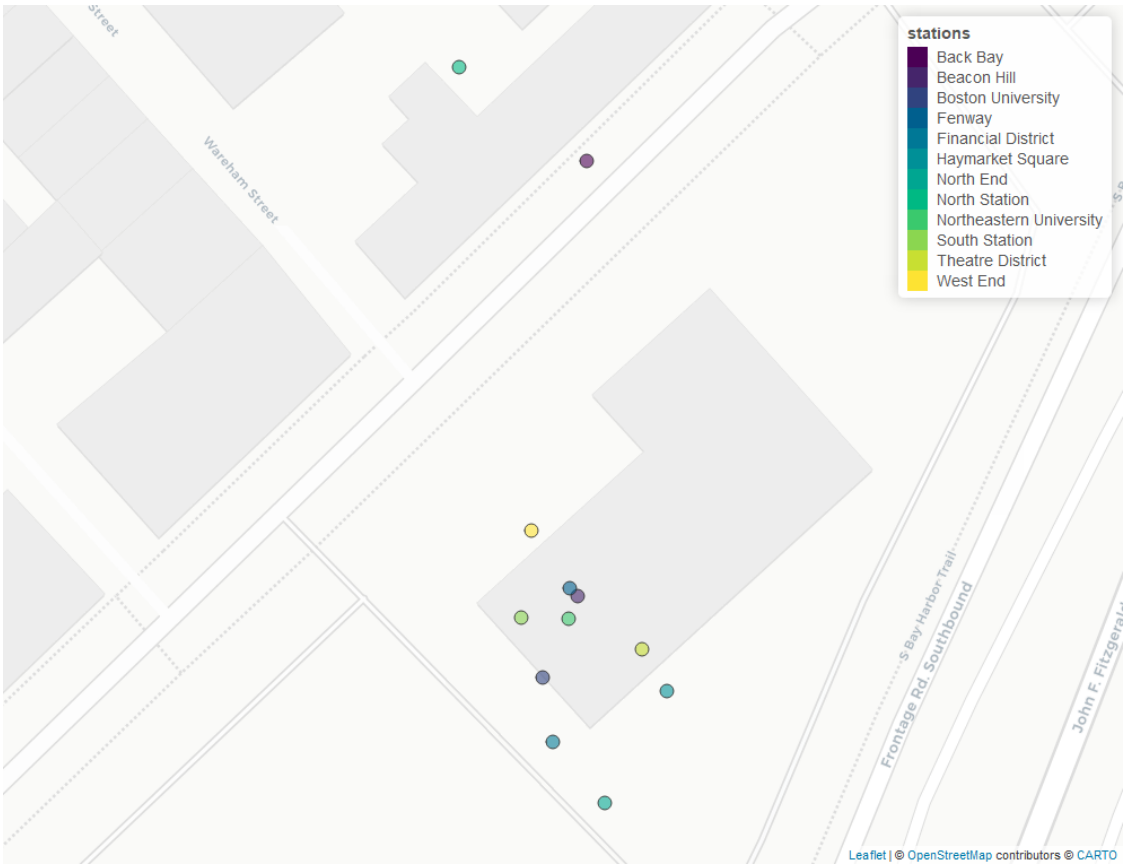
Map of all the stations

```
# Convert the data frame to a spatial points data frame
stations = st_as_sf(df_label,
                    coords = c("longitude", "latitude"), crs = 4326)


map1 <- mapview(stations,label = stations$label_coord, tooltip =
stations$label_coord)
map1
```



Since we cannot display mapview and leaflet maps in word file as they are are designed for interactive use and the mapview object created by the code is not readily knitted , we are saving it in png format and showing the images.

```
# save map in png format to include in README
mapview::mapshot(map1, file = "./images/map_stations.png")
```



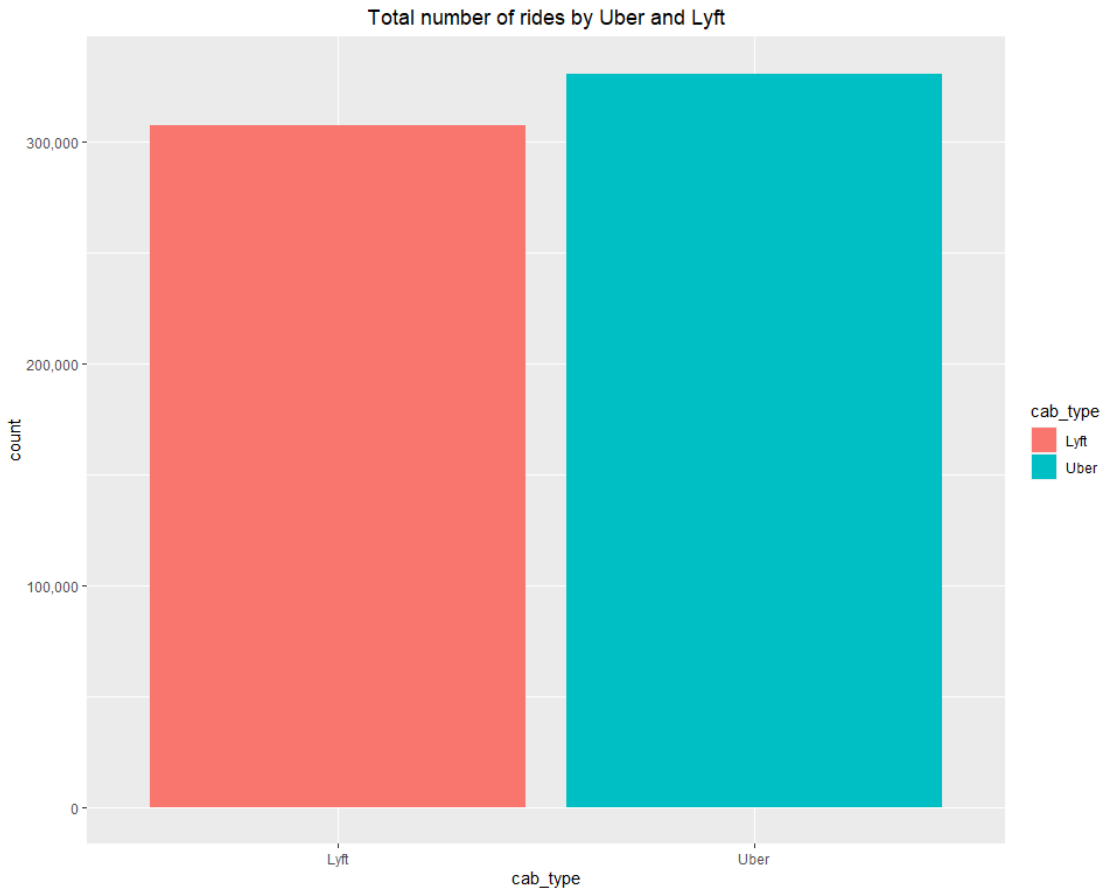
map-stations

There are total 12 locations in our data set

```
# creating separate data sets for each cab_type

data_uber <- data %>%
  filter(cab_type == "Uber")

data_lyft <- data %>%
  filter(cab_type == "Lyft")
```

Which company has more rides?

```
ggplot(data, aes(cab_type)) +
  geom_bar(aes(fill = cab_type)) +
  ggtitle("Total number of rides by Uber and Lyft") +
  theme(plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5)) +
  scale_y_continuous(labels=comma)
```



Uber has more rides than Lyft by a small margin. So, our data is pretty balanced.

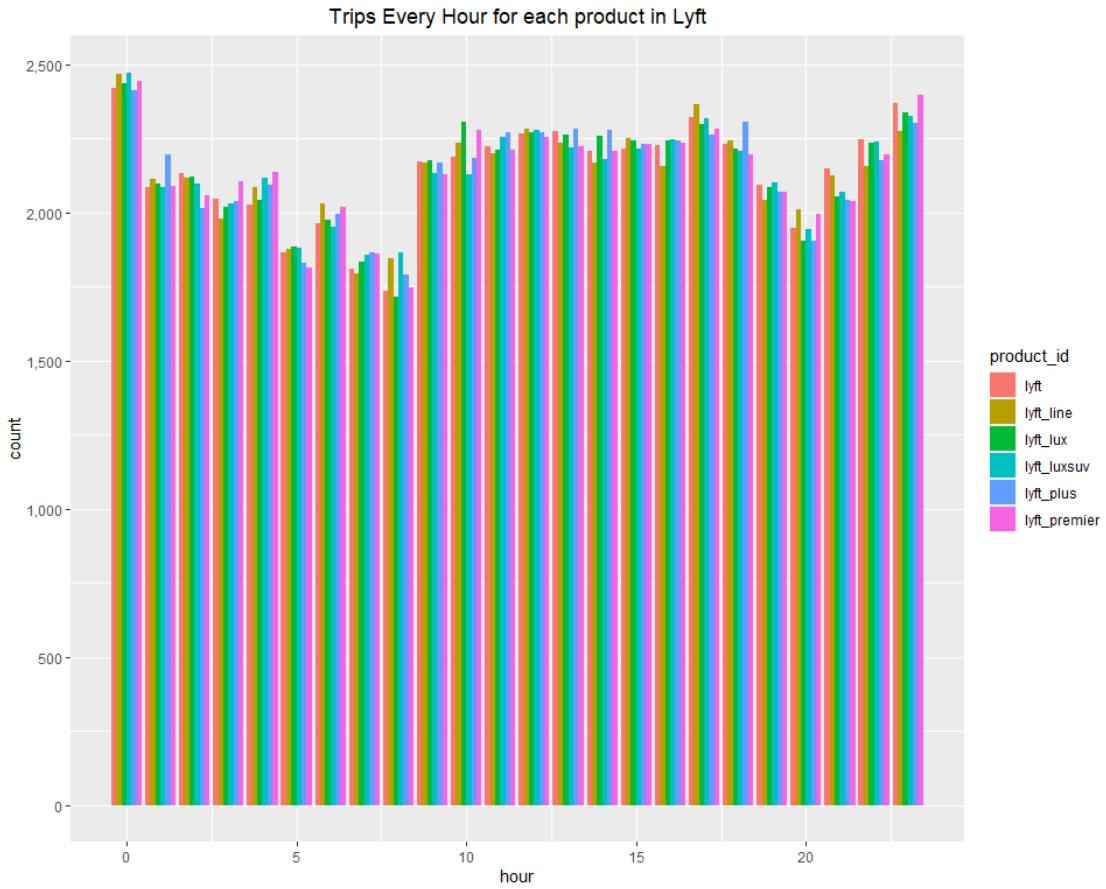
```
#
ggplot(data) +
  geom_bar(aes(x = hour, fill=cab_type), position= 'dodge') +
```

```
  ggtitle("Trips Every Hour for Uber and Lyft") +
theme(plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5)) +
scale_y_continuous(labels=comma)
```

Uber has more rides than Lyft periodically.

Which Lyft product is most popular in terms of number of rides?

```
ggplot(data %>% filter(cab_type == "Lyft")) +
  geom_bar(aes(x = hour, fill=product_id), position= 'dodge') +
  ggtitle("Trips Every Hour for each product in Lyft") +
theme(plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5)) +
scale_y_continuous(labels=comma)
```




```
# +  scale_fill_manual(values=c("red", "blue",
"green","yellow","purple","brown"))
```

There is no Lyft product that is more popular than the others.

What influences the surge rate for each company? Is it peak hour, traffic, or less number of available rides in the area

To answer our question, lets see correlation between surge_multiplier, price, hours, and distance

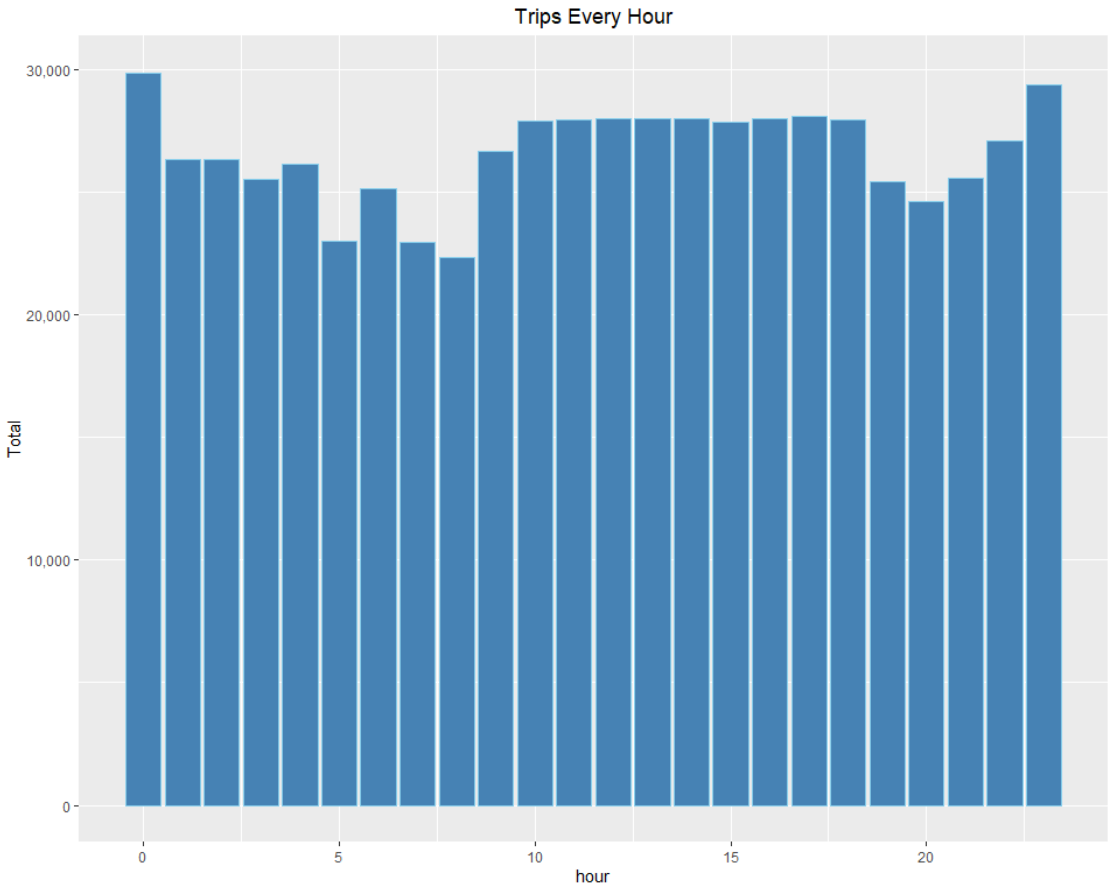
```
data_cor <- data_lyft %>%
  select(surge_multiplier,price,distance,hour, day,month,temperature)

ggcorr(data_cor, method = c("everything", "pearson"), label = TRUE) +
  ggtitle("Correlation between Surge Multplier, Price, Hour, and Distance") +
  theme(plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5))
```

From this graph, surge_multiplier and price, and price and distance are weakly correlated. But it does not answer our question clearly. Trying same correlation but for peak hours

Let's find out the peak hours to fully understand the data

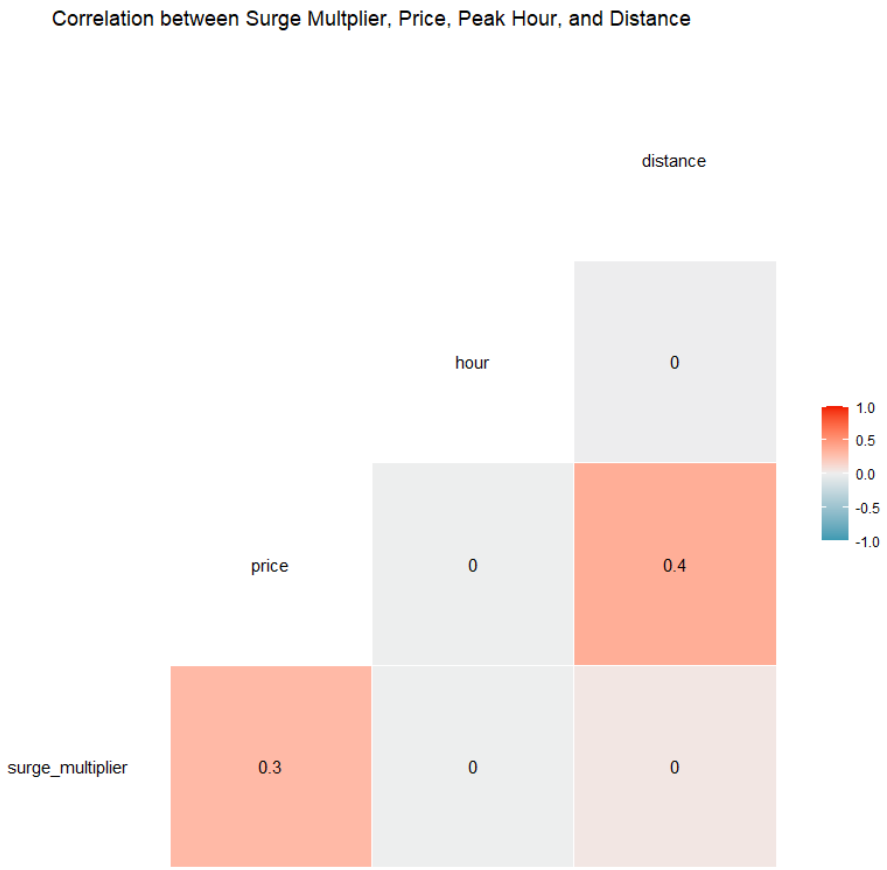
What are the peak hours for Lyft cab?

```
# Find hourly data for entire dataset
hourly_data <- data %>%
  group_by(hour) %>%
  dplyr::summarize(Total = n())

#hourly_data

# Plotting the trips by hours in a day
ggplot(hourly_data, aes(hour, Total)) +
geom_bar(stat="identity",
         fill="steelblue",
         color="skyblue") +
ggtitle("Trips Every Hour") +
theme(legend.position = "none",
      plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5)) +
scale_y_continuous(labels=comma)
```



12am, 11pm, and 5 pm are the top 3 peak hours for Lyft.

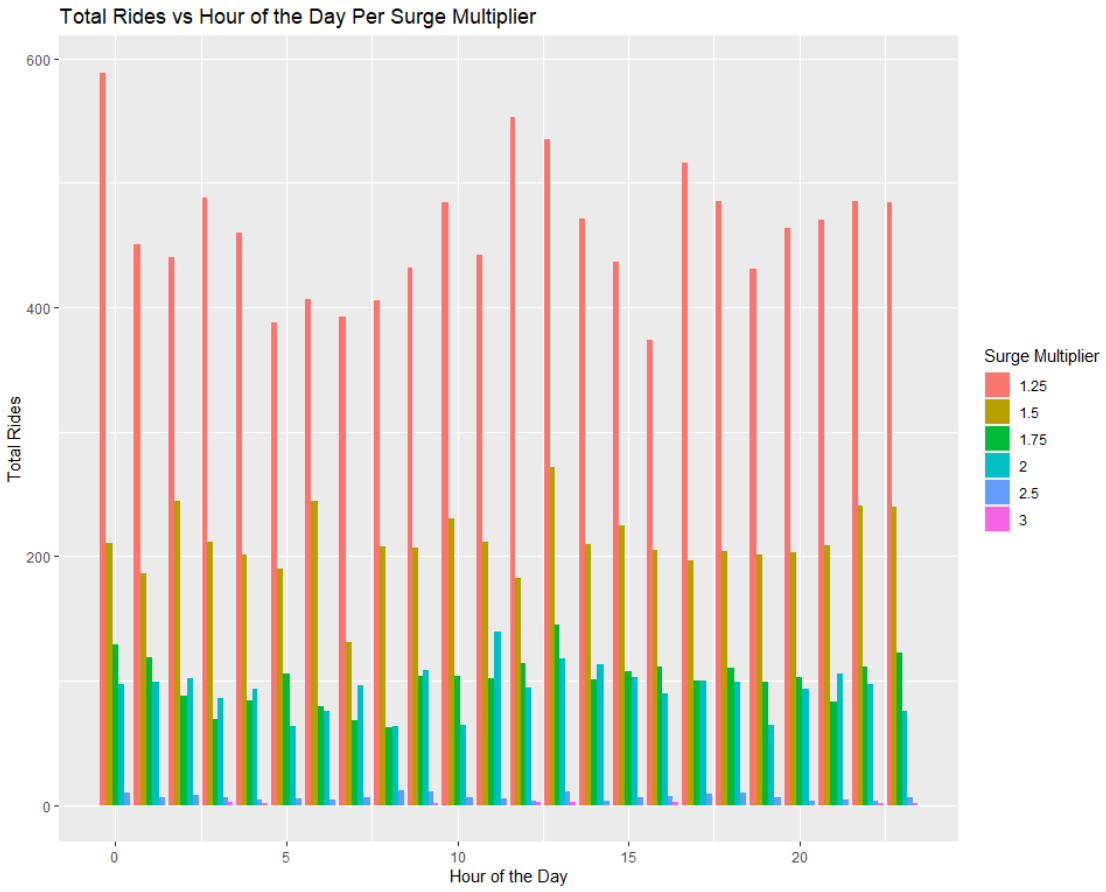
```
peakHours <- data_lyft %>%
  filter(hour == 0 | hour ==23 | hour == 17)

#peakHours

data_cor_peak <- peakHours %>%
  select(surge_multiplier, price, hour, distance)

ggcorr(data_cor_peak, method = c("everything", "pearson"), label = TRUE) +
   ggtitle("Correlation between Surge Multplier, Price, Peak Hour, and
Distance") +
  theme(plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5))
```



Changing to only peak hours does not make any difference. Let's look at other ways to find the answer.

Relationship between surge price and hours

```
data_lyft_surge_hour <- data_lyft %>%
        filter(surge_multiplier > 1.00) %>%
        group_by(hour, surge_multiplier) %>%
        dplyr::summarize(total_rides=n())

## `summarise()` has grouped output by 'hour'. You can override using the
## `.groups` argument.

data_lyft_surge_hour_plot <- ggplot(data = data_lyft_surge_hour, aes(x =
hour, y = total_rides, fill = factor(surge_multiplier))) +
        geom_bar(stat = "identity", position = "dodge") +
        xlab("Hour of the Day") + ylab("Total Rides") + ggtitle("Total Rides
vs Hour of the Day Per Surge Multiplier") +
        labs(fill="Surge Multiplier")

data_lyft_surge_hour_plot
```



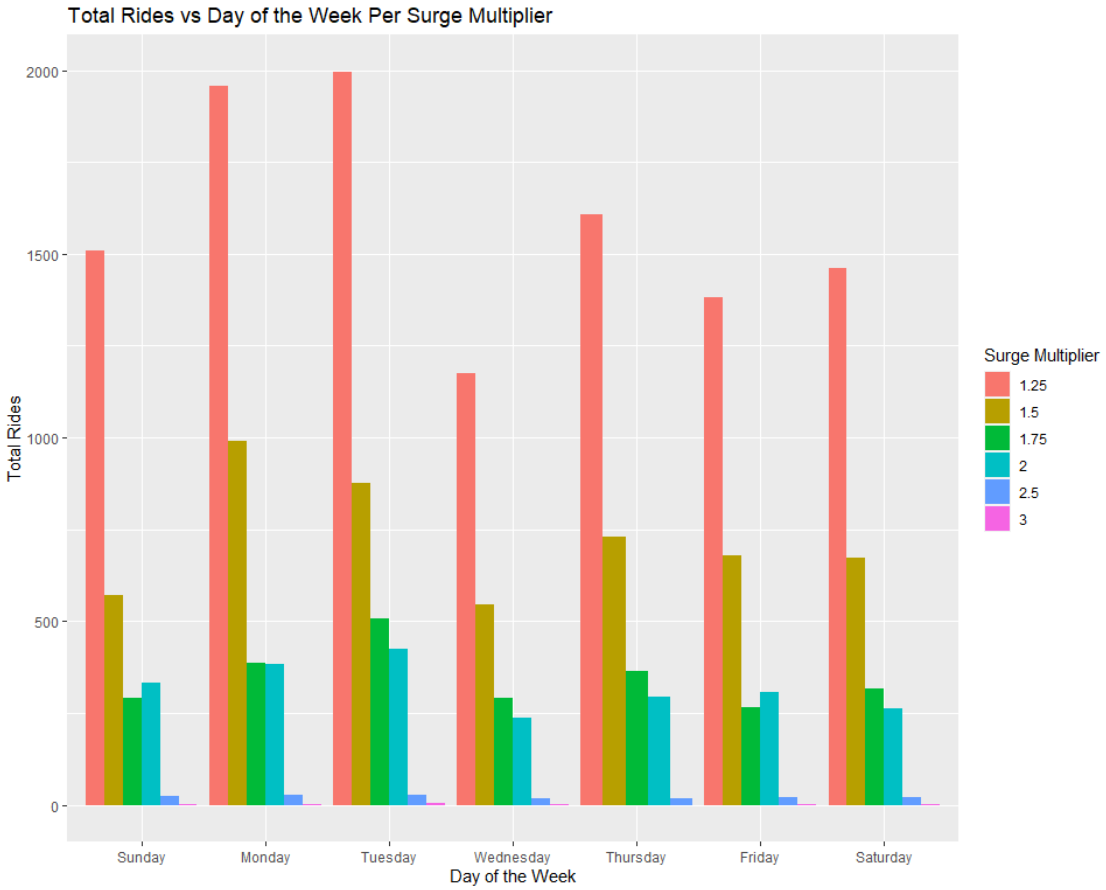
As we can see, the 1.25x surge is highest at 12am, which is the peak hour. However, 1.5x surge is highest at 1 pm, and 1.75x surge is highest at 1pm and 12 am. And 2x surge is highest at 11 am. The highest amount of 2.5x surge is at 8 am, which coincidentally has the least number of rides during the day.

This could mean that there are a smaller number of drivers available and that's why there is surge price. 3x surge price is only at 3am, 4am, 9am, 12pm, 1pm, 4pm, 10pm, and 11pm. This shows that max surge price lasts for 2 hours.

Relationship between surge price and week

```
#  Find Character Day of Week (Using Abbreviated Labels)
data_lyft$week <- wday(data_lyft$datetime, label = TRUE, abbr = FALSE)

data_lyft_surge_week <- data_lyft %>%
        filter(surge_multiplier > 1.00) %>%
        group_by(week, surge_multiplier) %>%
        dplyr::summarize(total_rides=n())

## `summarise()` has grouped output by 'week'. You can override using the
## `.groups` argument.

data_lyft_surge_week_plot <- ggplot(data = data_lyft_surge_week, aes(x =
week, y = total_rides, fill = factor(surge_multiplier))) +
```

```
        geom_bar(stat = "identity", position = "dodge") +
        xlab("Day of the Week") + ylab("Total Rides") + ggtitle("Total Rides
vs Day of the Week Per Surge Multiplier") +
        labs(fill="Surge Multiplier")

data_lyft_surge_week_plot
```

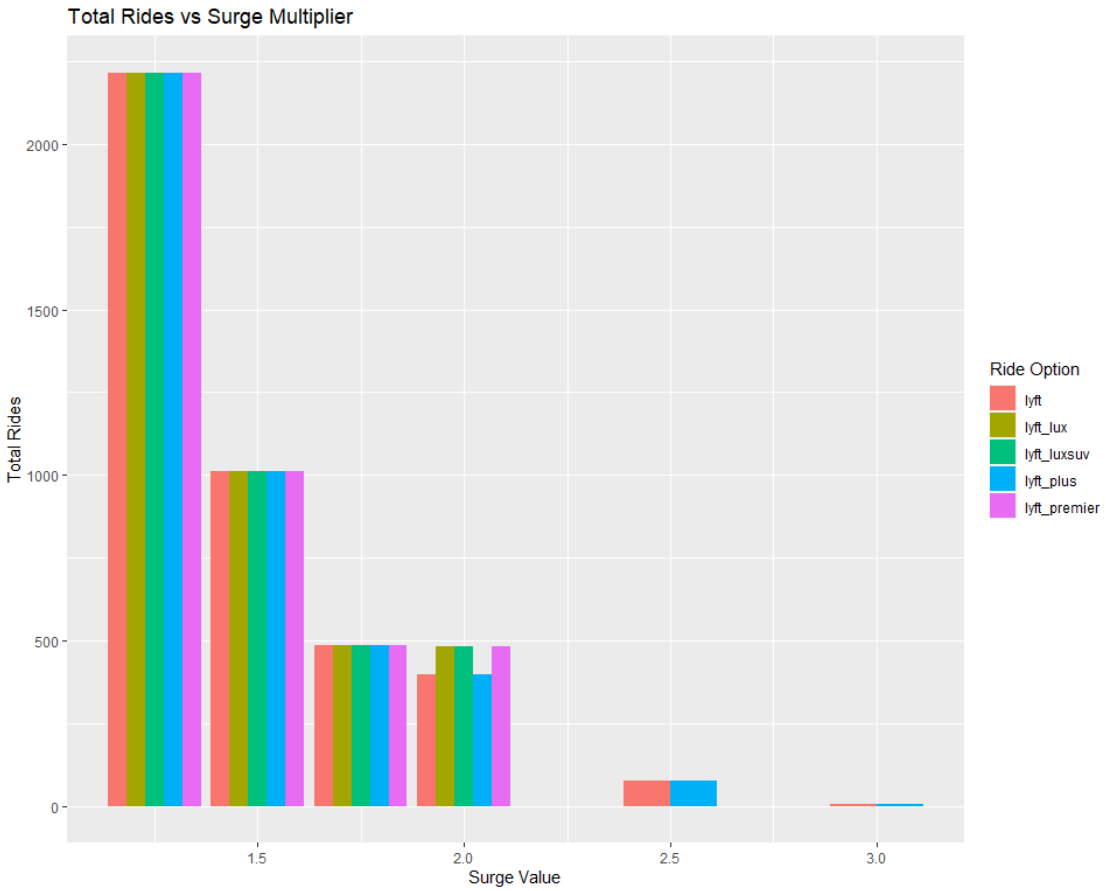


From the graph, it is clear that Tuesdays and Mondays have the highest surged price. However, Wednesdays have the least surged price.

Relationship between available rides and surged price

```
data_lyft_surge_rides <- data_lyft %>%
        filter(surge_multiplier > 1.00) %>%
        group_by(surge_multiplier, product_id) %>%
        dplyr::summarize(total_rides=n())

## `summarise()` has grouped output by 'surge_multiplier'. You can override
using
## the `.groups` argument.

data_lyft_surge_rides_plot <- ggplot(data = data_lyft_surge_rides, aes(x =
surge_multiplier, y = total_rides, fill = factor(product_id))) +
        geom_bar(stat = "identity", position = "dodge") +
        xlab("Surge Value") + ylab("Total Rides") + ggtitle("Total Rides vs
```

```
Surge Multiplier") +
        labs(fill="Ride Option")

data_lyft_surge_rides_plot
```



Lyft and Lyft_plus are the most basic ride options. It offers rides in cars that seat up to 4 and 6 people respectively.

Lyft_lux, Lyft_luxsuv, and Lyft_premier are the luxury ride options. It offers rides in cars that are high-end and comfortable.

From the graph we can see that only Lyft and Lyft_plus ride options have surge prices till 3.0. Other luxury ride options have maximum surge price till 2.0.

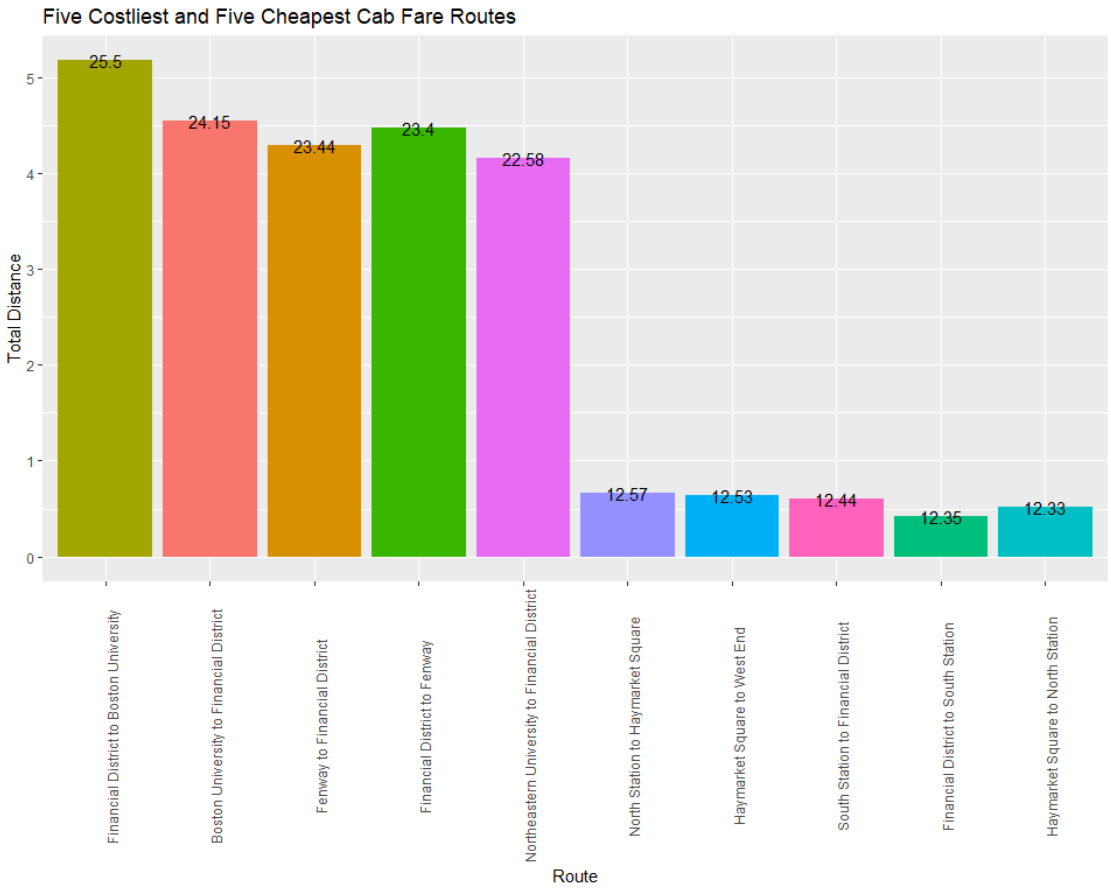
From all the graph, we can say that peak hours, day of the week, ride options, and number of available rides influence the surge price.

Which are the most popular pickup points and drop-off locations?

Popular pick-up and drop-off locations by price and number of rides

Creating a new data frame that combines source and destination routes

```r
data_routes <- data %>% unite(Routes, source:destination, sep = " to ",
remove = FALSE)
#data_routes

#Finding total number of rides and average price for each route
data_routes_rides <- data_routes %>%
  group_by(Routes) %>%
  summarize(rides = n(), avg_price = mean(price), distance = mean(distance))


data_routes_rides[c('Source_Station', 'Destination_Station')] <-
str_split_fixed(data_routes_rides$Routes, ' to ', 2)

# getting top 5 and bottom 5 costly routes
data_top_5_routes <- data_routes_rides %>%
# Top N highest values by group
  arrange(desc(avg_price)) %>%
  slice(1:5)
#data_top_5_routes

data_bottom_5_routes <- data_routes_rides %>%
# Bottom N highest values by group
  arrange(avg_price) %>%
  slice(1:5)
#data_bottom_5_routes

data_N_routes_by_price <- rbind(data_top_5_routes, data_bottom_5_routes)
#data_N_routes_by_price

# Five Costliest and Five Cheapest Cab Routes (arranged by cab fare)
ggplot(data = data_N_routes_by_price, aes(forcats::fct_reorder(Routes,
desc(avg_price)), distance)) +
  geom_bar(stat = "identity", position = "dodge", aes(fill=Routes)) +
  xlab("Route") +
  ylab("Total Distance") +
  theme(axis.text.x = element_text(angle=90), legend.position = "none") +
  ggtitle("Five Costliest and Five Cheapest Cab Fare Routes") +
  geom_text(aes(label=round(avg_price, 2)))
```

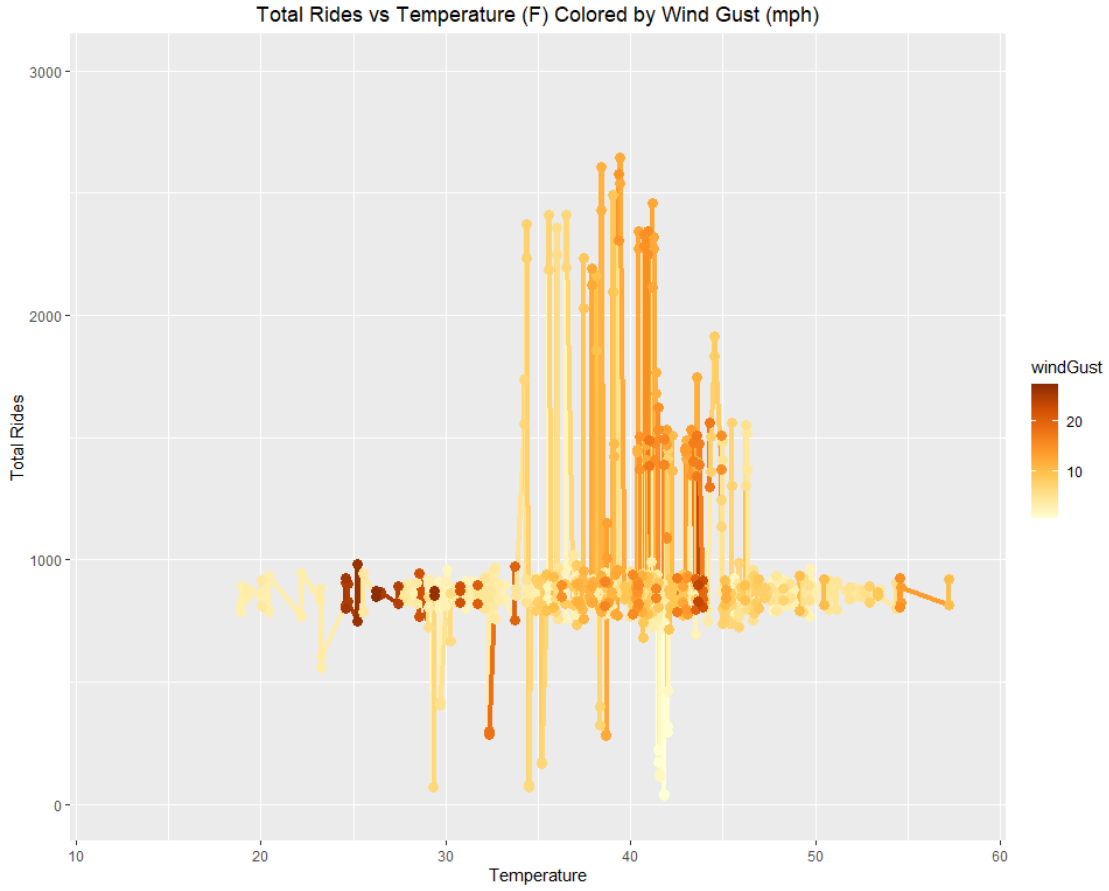


Financial District to Boston University is the costliest route with an average cost of 25.5 USD per ride. And, Haymarket Square to North Station is the cheapest route with an average of 12.33 USD per ride

Are people more likely to use a cab during cold temperature?

```
data_wind_gust <- data %>%
        dplyr::group_by(windGust, temperature, cab_type) %>%
        dplyr::summarize(total_rides = n())

## `summarise()` has grouped output by 'windGust', 'temperature'. You can
override
## using the `.groups` argument.

#data_wind_gust

data_wind_gust_plot <- ggplot(data_wind_gust, aes(temperature, total_rides,
color= windGust)) +
  geom_line(linewidth=1.5) +
  geom_point(size=3) +
  scale_color_distiller(palette = "YlOrBr", direction = 1,)+
  #scale_color_grey(start = 0.2, end = 0.8)+
  #guides(color = guide_colorbar(reverse = TRUE)) +
  theme(legend.position="right") +
```

```
  ggtitle("Total Rides vs Temperature (F) Colored by Wind Gust (mph)") +
  ylab("Total Rides") + xlab("Temperature") +
  theme(plot.title = element_text(hjust=0.5)) + xlim(12,58) + ylim(0,3000)

data_wind_gust_plot
```



The graph shows that people are more likely to ride cab when temperature is around 37–39-degree Fahrenheit. Also, wind gust has more impact on total rides. People take more cab rides when it is windy. The dark color on the graph means higher wind speed.

What are the most popular Source Destination combo by number of rides?

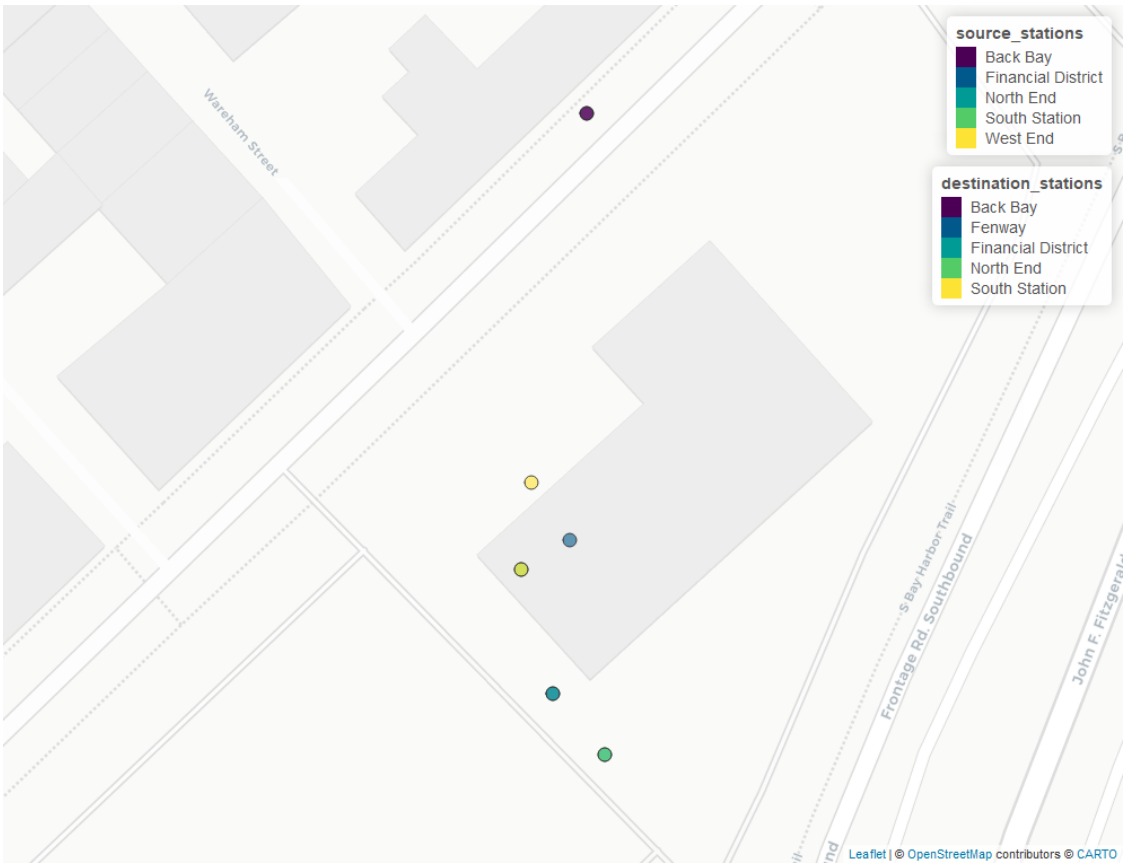
Creating a new data frame with the geo data for unique values of source and destination of the stations that make the top 5 routes

```
source_geo_data <- unique(data_routes[, c('source','source_lon',
"source_lat")])
#source_geo_data

destination_geo_data <- unique(data_routes[,
c('destination','destination_lon', "destination_lat")])
#destination_geo_data
```

```
df3 <- right_join(data_routes_rides, destination_geo_data,
by=c('Destination_Station'='destination'))
df3 <- unique(df3)

df4 <- right_join(df3, source_geo_data, by=c('Source_Station'='source'))
df4 <-unique(df4)

df5 = df4%>%
  arrange(rides)%>%
  tail(5)

#data_routes_rides
#df5
```

Map of the Stations that make the top 5 Routes by number of Rides

```
# Create example data of points
lng = df5$source_lon
lat =  df5$source_lat
names = df5$Source_Station

# Create a data frame with the point data
points_df = data.frame(lng, lat, names)

# Convert the data frame to a spatial points data frame
source_stations = st_as_sf(points_df,
                  coords = c("lng", "lat"), crs = 4326)

lng = df5$destination_lon
lat =  df5$destination_lat
names = df5$Destination_Station

# Create a data frame with the point data
points_df = data.frame(lng, lat, names)

# Convert the data frame to a spatial points data frame
destination_stations = st_as_sf(points_df,
                  coords = c("lng", "lat"), crs = 4326)

# Plot the points on a map
#mapview(source_stations, label = points_sdf$names)
map2 <- mapview(source_stations,label = source_stations$names, tooltip =
source_stations$names) + mapview(destination_stations,label =
destination_stations$names,tooltip = destination_stations$names)

# save map in png format to include in README
mapview::mapshot(map2, file = "./images/map_routes.png")
```

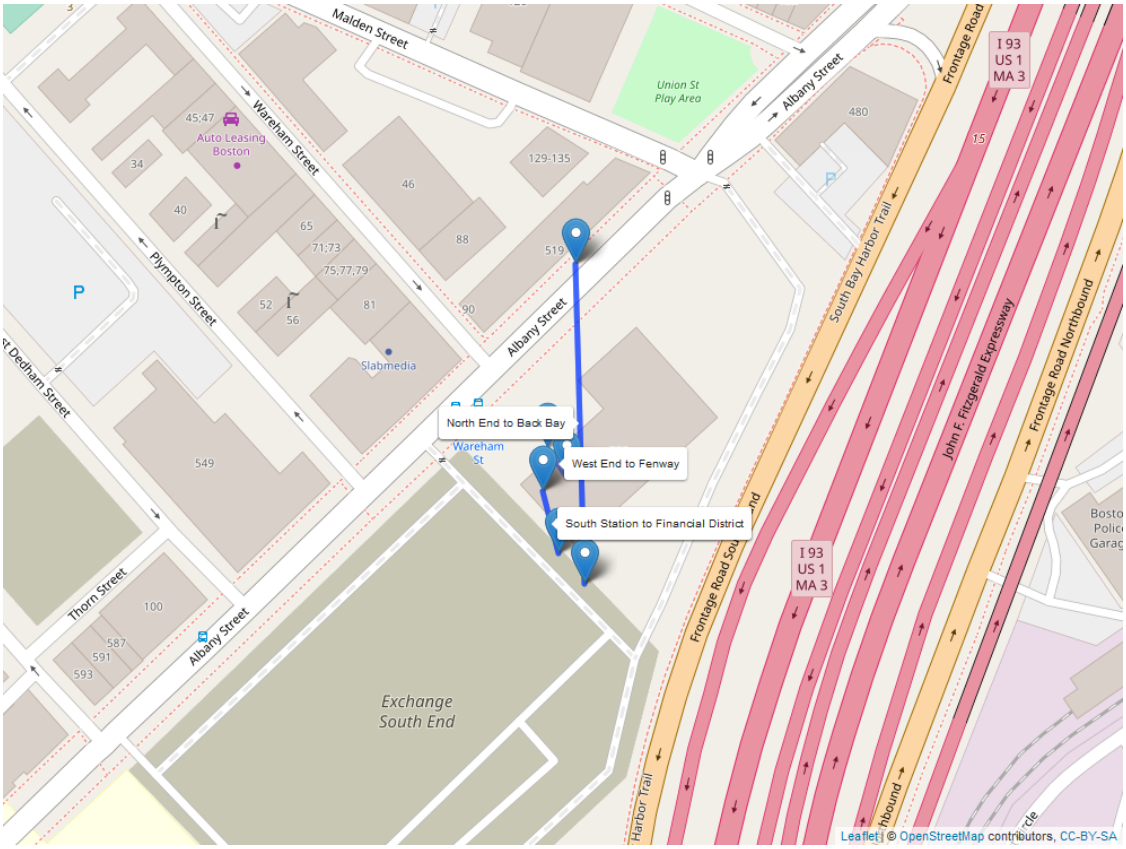


map-routes

Map of the top 5 Routes by number of Rides

```
map_stations <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addMarkers(lng = df5$source_lon,
             lat = df5$source_lat,
             popup = df5$Source_Station)

map_stations<-map_stations%>% addMarkers(lng = df5$destination_lon,
             lat = df5$destination_lat,
             popup = df5$Destination_Station)

for (i in 1:nrow(df5))

map_stations <- map_stations%>%

addPolylines(lat=c(df5[i,]$source_lat,df5[i,]$destination_lat),lng=c(df5[i,]$
source_lon,df5[i,]$destination_lon),label=df5[i,]$Routes ,
    labelOptions = labelOptions(noHide = T))
map_stations
```

```
# save map in png format to include in README
mapview::mapshot(map_stations, file = "./images/map_routes_rides_2.png")
```



map-routes-rides

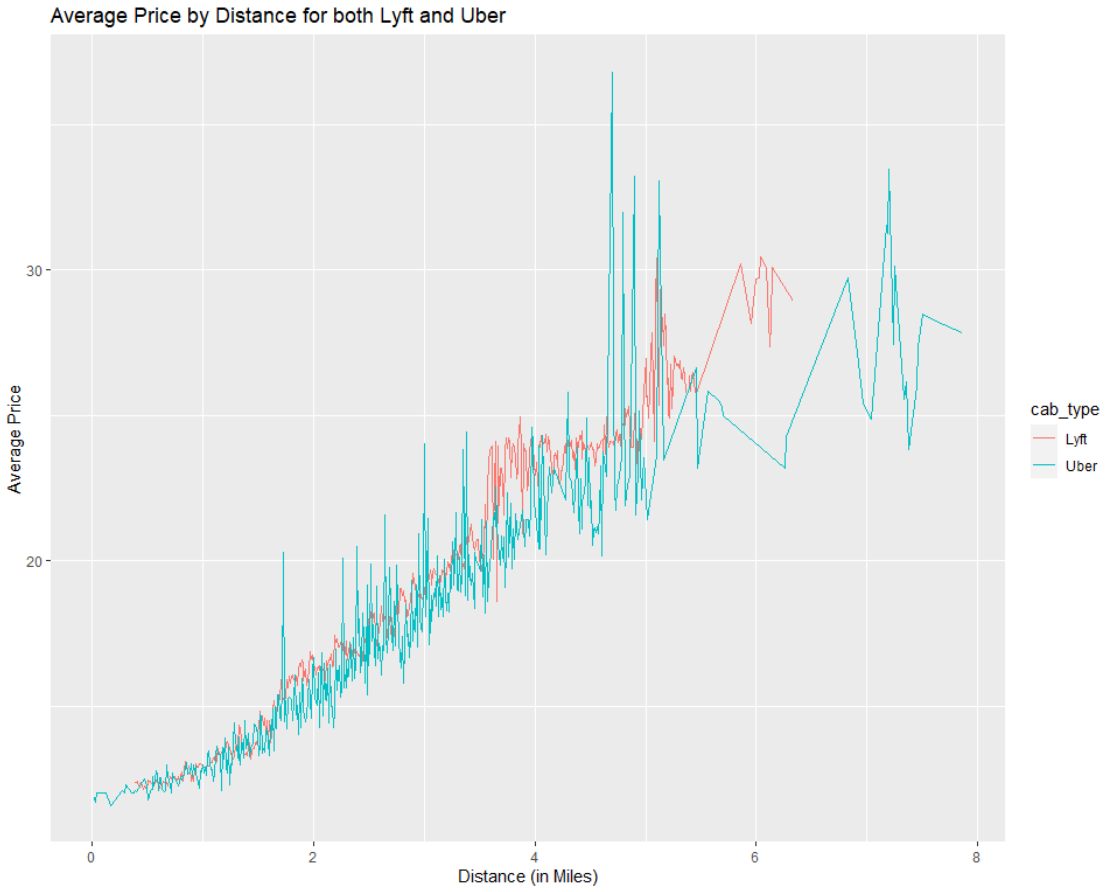
The most popular routes by number of rides are:

Financial District to South Station with 9534 rides South Station to Financial District with 9534 rides. North End to Back Bay with 9414 rides. Back Bay to North End with 9414 rides. West End to Fenway with 9360 rides.

What is the average price by distance for both cabs?

```
# Average price by distance
data_avg_dist_price <- data %>%
  filter(surge_multiplier == 1) %>%
  group_by(distance, cab_type) %>%
  summarize_at(vars(price), list(name = mean))
#data_avg_dist_price

data_avg_dist_price_plot <- ggplot(data_avg_dist_price, aes(distance, name,
color=cab_type)) +
  geom_line() +
  ggtitle("Average Price by Distance for both Lyft and Uber") +
  xlab("Distance (in Miles)") +
  ylab("Average Price")
```

```
data_avg_dist_price_plot
```



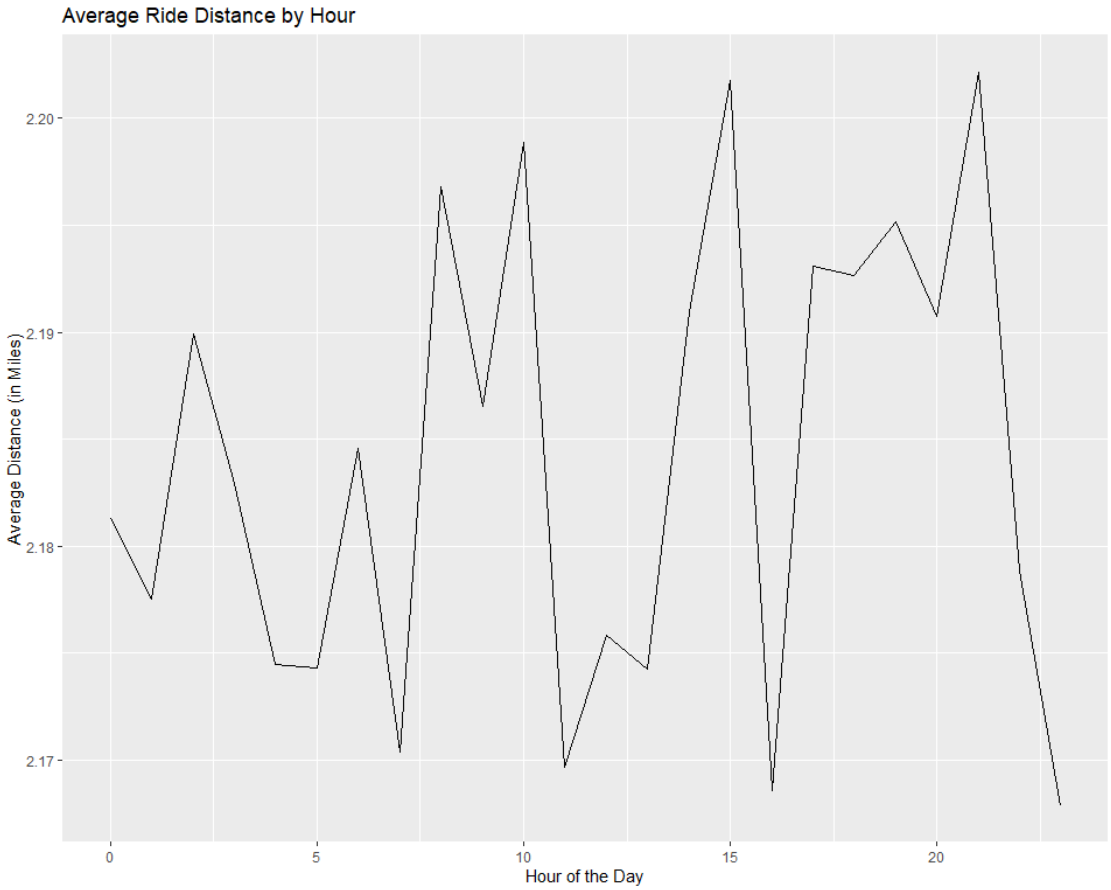
For both cabs, as expected, price increases linearly with distance. Here, we have removed surge pricing as Uber does not have surge price data.

We notice that Uber ride will travel farther up to 8 miles, while Lyft will travel for distance ~ 6 miles. Also, Uber charges the most when distance is ~4.5 miles, while Lyft charges most at ~5 miles.

What is the average distance travelled by hour?

```
# Average price by distance
data_avg_dist_hour <- data %>%
  filter(surge_multiplier == 1) %>%
  group_by(hour) %>%
  summarize_at(vars(distance), list(name= mean))
#ata_avg_dist_hour

data_avg_dist_hour_plot <- ggplot(data_avg_dist_hour, aes(hour, name)) +
  geom_line() +
  ggtitle("Average Ride Distance by Hour") +
  xlab("Hour of the Day") +
  ylab("Average Distance (in Miles)")
```

```
data_avg_dist_hour_plot
```



This graph shows that on average, the longest ride distance took place at roughly 3PM and 9PM. The next longest ride distance intervals were at 7:30AM and 10AM.

This shows that most people use cabs for their office commute as the data coincides with peak office hours.

Conclusion

In this article, we analyzed a dataset of ride-hailing data from Boston, Massachusetts. We found that Uber has more rides than Lyft by a small margin. Uber can be the first choice for long distances. We also found that there is no Lyft product that is more popular than the others.

We then looked at the factors that influence surge pricing. We found that surge pricing is highest during peak hours, on Tuesdays and Mondays, and for Lyft and Lyft_plus ride options. We also found that surge pricing is lower on Wednesdays and for luxury ride options.

Finally, we looked at the most popular pickup points and drop-off locations. We found that the most popular route is Financial District to Boston University. We also found that people are more likely to use a cab during cold temperatures and when it is windy.

We believe that this analysis provides valuable insights into the factors that affect ride-hailing prices. We hope that this information will be helpful to researchers and analysts who are interested in understanding the ride-hailing industry.

Recommendations

Based on our findings, we recommend the following:

Ride-hailing companies should consider factors such as peak hours, day of the week, and ride options when setting surge prices.

Ride-hailing companies should work to increase the number of available rides during peak hours to reduce surge pricing.

Ride-hailing companies should offer discounts or promotions during off-peak hours to encourage riders to use their services at those times.

Ride-hailing companies should collect more data on ridership patterns to better understand how to set surge prices and increase ridership.