Cheap Rides in Boston: STAT139 Final Project

Yuanbiao Wang, Hao Wang, Rain Wu, Dean Huang

1 Introduction

The core question of this project is to identify the pricing model of Uber / Lyft rides, and get an intuition about how to get cheap rides in Boston. The main tools for such analysis is regression.

The dataset used across this project is Uber and Lyft Dataset Boston, MA[1]. The predictors we selected and cared about for the regression analysis includes:

- Distance of the ride
- Weather condition, including a categorical variable for the weather, and other statistics like temperature, humidity, visibility, etc.
- Pick up / Destination location
- Ride type: Uber or Lyft; more specific information about the vehicle, like Lyft XL, UberX, etc.

The interesting question includes: is Uber cheaper than Lyft or vice versa? What's the pricing model between distance and price? Since the data is around seasonal holidays, do time and location affect the price?

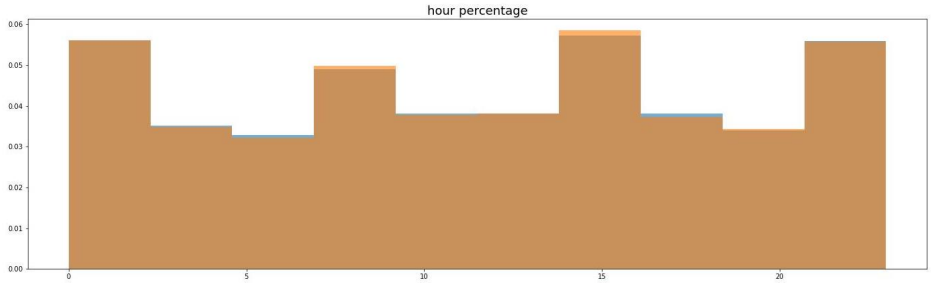
Our code is available at: https://github.com/haowang0402/stat139-final-project

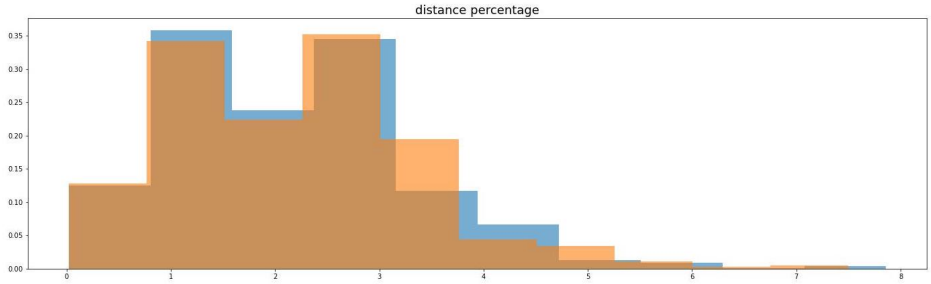
2 Preprocessing

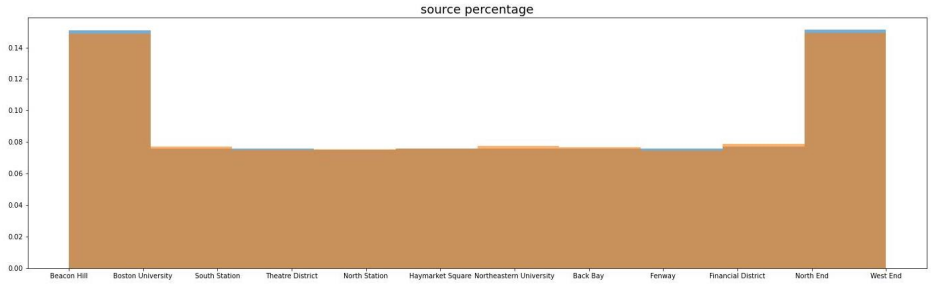
Since the original dataset is too big to process in R, we downsample the dataset to a smaller dataset. However, we'd like to make sure the distribution of the smaller dataset is similar to the original dataset.

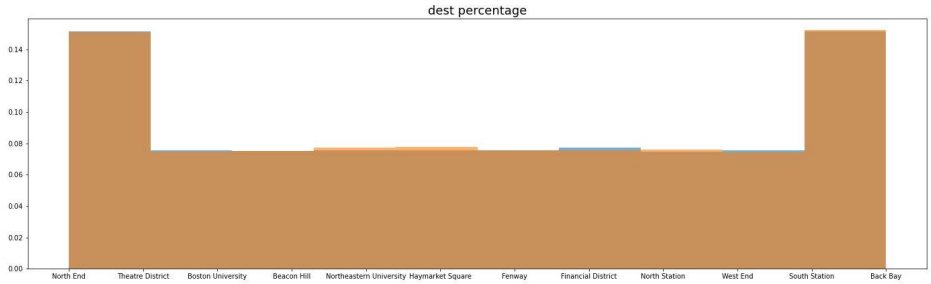
[1] `https://www.kaggle.com/datasets/brllrb/uber-and-lyft-dataset-boston-ma`

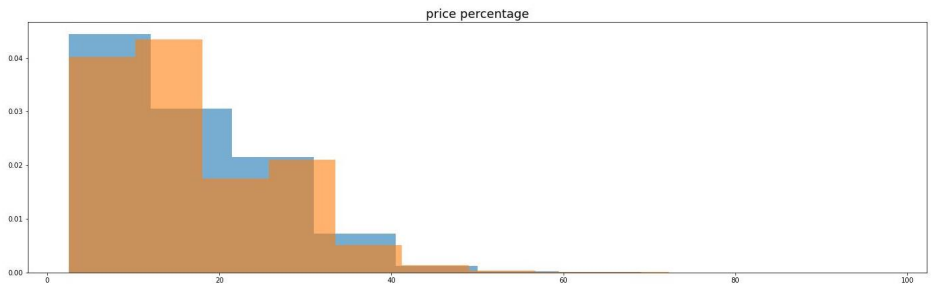
Preprint. Under review.

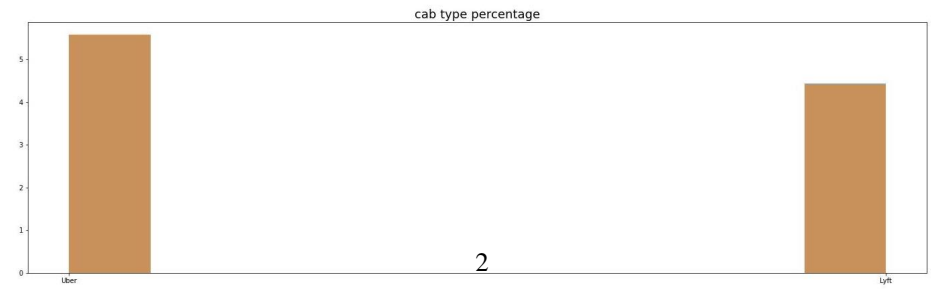

hour percentage
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
5
10
15
20


distance percentage
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
2
3
4
5
6
7
8


source percentage
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Beacon Hill
Boston University
South Station
Theatre District
North Station
Haymarket Square
Northeastern University
Back Bay
Fenway
Financial District
North End
West End


dest percentage
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
North End
Theatre District
Boston University
Beacon Hill
Northeastern University
Haymarket Square
Fenway
Financial District
North Station
West End
South Station
Back Bay


price percentage
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100


cab type percentage
5
4
3
2
1
0
Uber
Lyft

It seems that the downsampling distribution is pretty similar to the original one.

As of the predictors, as is stated before, we did the following things:

- We process the numerical variables `hour` to be categorical (divide by certain time points to yield 4 categories: morning, afternoon, evening, and midnight). We call this `period`.
- We discarded the variable `month` and instead use the day counts starting from Nov. 11 because our data only include months of 11 and 12 in 2018. We call this `daycount`.
- Weather descriptions are condensed into 4 categories: clear, rain, cloudy and foggy. We call this `weather`.

In the end, we used the following variables as predictors: `distance`, `cab_type`, `daycount`, `weather`, `source`, `destination`, `name`, `temperature`, `humidity`, `windSpeed`, `visibility`, `pressure`, `period`. The response variable is `price`. Among the predictors, `cab_type` can be either Lyft or Uber, and `name` describes the specific type of the ride, like Lyft Luxury, or Uber XL, etc.

In the end, we randomly sampled 40000 out of 45984 samples (which is the count after the NA filtering) as the training dataset, and the rest as the test dataset.

3 Modelling and Analysis

3.1 Simple Linear Regression

(Yuanbiao) In this section we will aim to build a very simple model to analyze the relationship between ride distance and ride price.

The code for this part can be found in 4

Simple linear regression From the scatter plot in Figure 1 we can see there is a positive relationship between ride distance and price, although we can see that the variability of the response at the same predictor value is fairly big, and we can expect that the R^2 for a simple linear model would not be very ideal.

We then fit a simple linear regression model and check the assumptions for simple linear regression. We call it `lm.simple`.

From the point estimates we can make the rough interpretation: first, the intercept is 10.33, indicating that a "starting price" would be around 10.33 dollar, even if it's a super short ride; second, the slope for the ‘distance‘ is 2.82, meaning that every 1 mile would be estimated to cost you 2.82 dollar more in general cases. The p-value for the slope is less than 2×10^{-16}, so we can say that the association is very very significant.

R^2 is roughly 0.1188. It's rather distant from 1. It means that nearly 90% of the variability is due the residual errors. We get a pretty high RMSE 8.75. This is not surprising as we expect more factors aside from sheer distance would affect the price as we learned they are considered into the pricing model. Also, we notice that the test RMSE 8.84 is not significantly higher than that of the training, so the model is not overfitting at all (which is also within our expectation, as the model is rather simple and has a very low variance).

The prediction line plot in Figure 1 shows a match with the tendency we observed in the scattered plot but there are still a lot of variability and it is clear that predicting with only the distance is not sufficient.

We also checked the assumptions using residual and Q-Q plot. According to the residual plot, the linearity and constant variance assumption are all poorly conformed. There a lot more positive residuals than negative ones, so `lm.simple` is underestimating the prices. The variability of residuals seem greater when the predicted prices are within range $[18, 23]$. From the Q-Q plot and the histogram we can see that the normality is also not met. The data looks quite right-skew, and slightly bi-modal.

Polynomial Regression Since the linear model has a relatively not so satisfying performance, and the linearity is not well met, we think it might be worth a try to use polynomial model. We started with a quadratic model `lm.quadratic` and ran the ANOVA test. From the p-values of the F-test between `lm.simple` and `lm.quadratic` (0.706), we can see that the added quadratic term is not of much help, so we will not further more consider it.

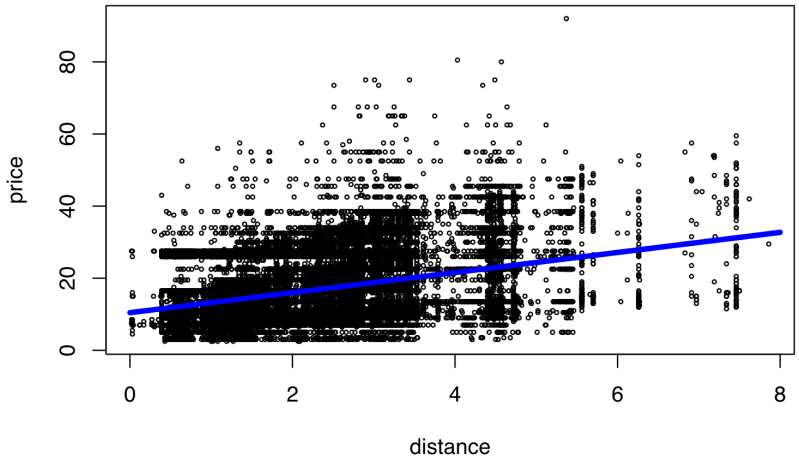


Figure 1: Scatter plot of price against distance, as well as the prediction line of `lm.simple` on training data

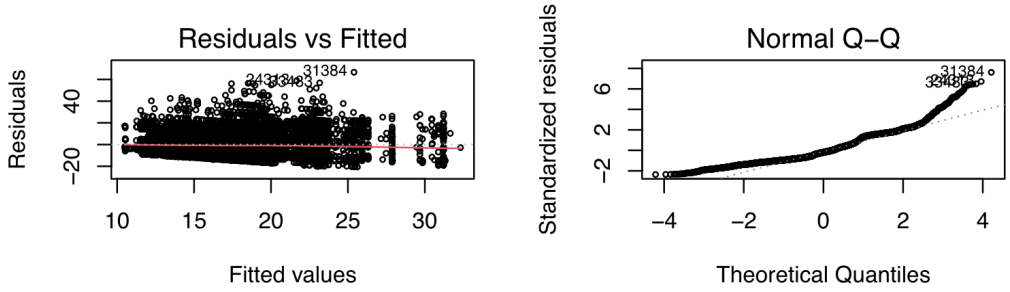


Figure 2: Check the assumptions for simple linear regression

Uber or Lyft? Here we want to research into a very intriguing question: is Uber and Lyft exhibiting different pricing models? To answer this question, we will experiment with another linear model incorporating the type of the ride into the consideration. We call this model `lm.brand`. It considers the `distance` and `cab_type`, as well as their interaction.

From the summary of the model we can derive some useful insights. When it is a Lyft ride, the regression formula is

$$\text{price} = 10.28 + 3.24 \times \text{distance}$$

While if it is an Uber ride, the regression formula is

$$\text{price} = 10.44 + 2.43 \times \text{distance}$$

This suggests that Lyft has a slightly lower starting price but has a significantly higher fare rate w.r.t the distance compared to Uber rides. The pricing model for these two companies are in fact quite different. The t-test result supports this claim: the coefficients for `cab_typeUber` is not significant with a p-value as big as 0.55, while the coefficients for `distance:cab_typeUber` has a p-value less than 2×10^{-16}, indicating extreme significance. So we would have the impression that in Boston during 2018 in the winter, Lyft would be more expensive for long-distance rides and cheaper for short-distance ones.

To explore whether this is an improvement compared to the naive linear regression, we perform an F-test and calculated the RMSEs. The p-value for the F-test is less than 2.2×10^{-16}, suggesting a significant improve. Also both train test RMSEs (8.70 and 8.79) are better compared to `lm.simple`.

We also provided some visualizations to support this intuition. We plotted the scattered points as well as the prediction line, but in different colors to separate these two cab brands: pink for Lyft, and Orange for Uber.

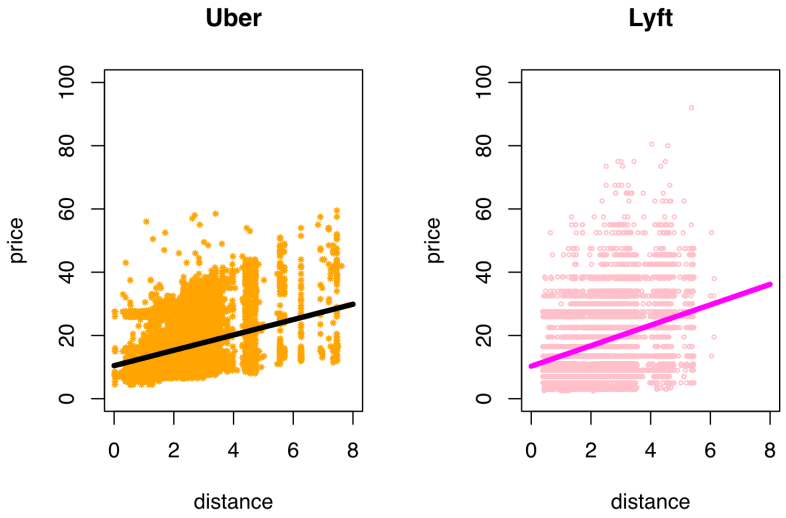


Figure 3: Prediction plot for Uber and Lyft rides using interaction terms, separately. Uber on the left, Lyft on the right.

From the prediction line in Figure 3 we can see that Lyft has a steeper prediction line, suggesting a significant higher fare rate. However, according to the scatter plots, we can easily find that Uber ride fares have better linearity w.r.t the ride distance, and has a smaller variance in the price.

3.2 Multiple Linear Regression

For this part, we fitted a linear regression model with all the predictors we selected, but without interaction terms.

From the model summary we found that the significant predictors are `cab_type, name, distance, source, destination`. The coefficient for distance is 2.92, slightly higher than that we got in the simple linear regression. This means, holding every other factor constant, every mile would cost you 2.92 dollar on average.

The coefficient for `cab_typeUber` is 14.62, which is drastically different from the result we had when we used simple linear regression, indicating that generally Uber is more expensive compared to Lyft. The difference between the fare of these two brands do not seem to have so much gap, so it's probably due to the multi-colinearities since we had another categorical predictor `name` which describes the specific type of the car. The reference group for this variable is `Black`, which is one of the Lyft's luxury ride type. Based on the coefficients, we created a visual shown in Figure 4 to see the fare ranking of these ride types.

From the ranked barplot we can see that:

1. shared rides have NA estimates, which is due to it is linearly related to other columns (prabably `cab_typeUber` and all other ride types of Uber will determine it).
2. Uber Pool, Uber X and Lyft are the cheapest rides generally speaking.
3. WAV is also very cheap because it is to certain extent a social welfare for the physically disabled citizens.
4. The XL rides are more expensive, and the luxury models (Lyft Black SUV, Uber Lux, Lyft Black XL, etc) are much more expensive.
5. It's surprising that Lyft Black actually seems to be cheaper than Lyft standard. But it may happen sometimes because the pricing model is probably due to machine learning and the over-demand of Lyft standard might cause a rise in the fare rate, while not many people choose Lyft Black and will cause a lower price.
6. Generally for the same level (like Uber Lux compared to Lyft Lux Black, or UberX compared to Lyft standard), the coefficients for Uber rides are significantly lower, probably because

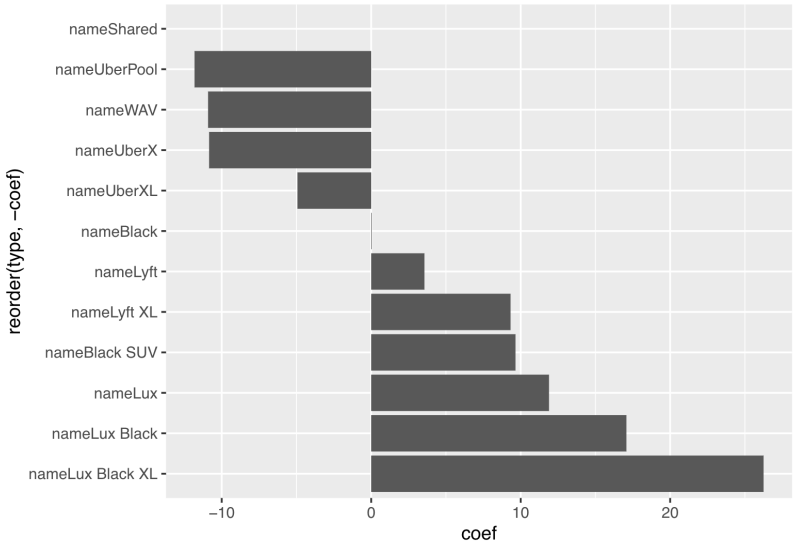
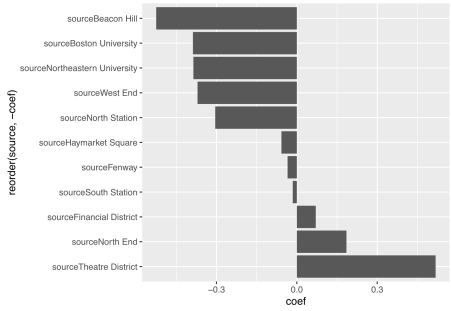
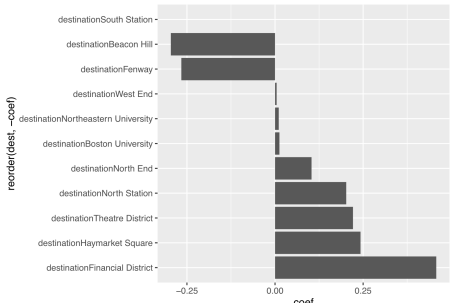


Figure 4: Coefficients for different car types. (The predictor name is `name`. One of the coefficient (Uber Pool) is missing due to singularities, and the coefficient for Lyft Black is zero because it is the reference group.



(a) Coefficients for the source locations



(b) Coefficients for the destination locations

of there is a large compensation coefficient for `cab_typeUber` already. Considering both factors, the Lyft ride still seems a better deal, but with marginal advantage. For example, the coefficient for `nameLyft` is 3.56, while the summation of `cab_typeUber` and `UberX` is 3.78.

We also created two more visuals for source and destination location coefficients analysis.

According to this visual in Figure 5a and 5b, the seemingly most expensive source location is Theatre District near Downtown Crossing, followed by North Station where TD Garden is located. This makes perfect sense because Boston Downtown has a large population density and TD Garden has many games and performances which will induce heavy traffic, so the dynamic pricing system might yield a higher price due to the high demand.

The least expensive source locations are Beacon Hill, NEU and BU. Beacon Hill is located at Park Street, near Downtown, so it's a bit confusing. NEU and BU are located in Malden and Allston, and the riders should be mostly students nearby, and they might tend to choose cheaper rides.

The visual for the destination shows a similar trend. Places near Downtown like Financial District and Theatre District, and places near TD Garden has a higher ride price in general.

We didn't see significances in the factor of weather, which is out of our expectation, because we presumed that during the Holiday seasons, the extreme weather conditions like storm would impact the ride prices positively. The estimated coefficients for `weatherFoggy, weatherRain,`

`weatherCloudy` are negative, contradicting to our theory. But there is an explanation. These categorical predictors have a high correlation with the numerical predictors like `visibility, temperature, windSpeed, humidity`. And the coefficients are in accordance with our assumption: the worse the weather condition is, like lower visibility and higher wind speed, would have a positive impact on the ride price. The only exception is when the temperature is higher, the ride would cost more. Since the data is collected during November and December, it seems a little bit confusing.

The `daycount` has a negative coefficient, meaning the later it is, the lower the price is. This also contradicts our common sense, as we would expect a higher price during holiday season near Christmas, but since the p-value is super high (0.88), so maybe this can be neglegible.

The time period seems not so significant in the prediction, but generally during late night and early morning when there are less passengers, the price would be lower.

The RMSE for this model is much lower (3.05 for train and 3.17 for test), but still not satisfying, the prediction can have up to 3 dollar error off the real price.

3.3 Step-wise Feature Selection

In this section we described how we tried to explore which predictors and interactions are important with forward model selection. We started with a full interaction model that takes into account all the predictors as well as their interaction terms.

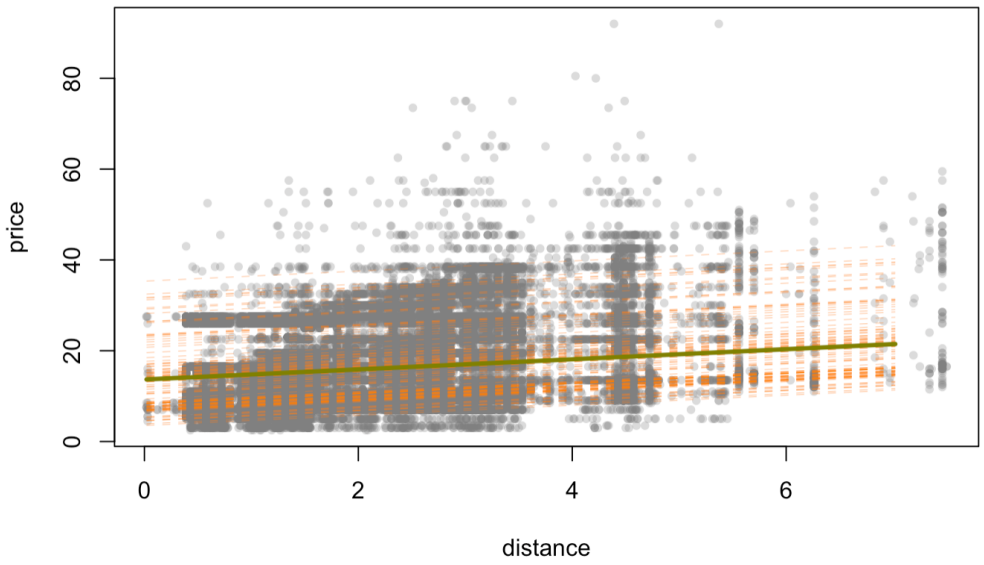
Both train and test RMSEs for this model are decreased (2.55 for train and 2.78 for test), indicating an improvement in the prediction capability. However, for the purpose of analysis, we also used AIC forward selection. For the selected model, The train RMSE increases a little bit compared to the full model (2.57), but test RMSE are decreased (2.77), indicating a slight improvement in the overfitting issue. The formula of this model is as follows:

```
price ~ distance + name + source + destination + distance:name +
name:source + distance:source + source:destination + name:destination +
distance:destination
```

From the formula we can see that the important predictors are just the same as what we observed in `lm.all`, the key factors are `distance`, `name` (which is the specific car type), `destination`, `source` and their pairwise interaction.

3.4 Regularization

We first try to build a well-tuned ridge regression model predict price from the 4 predictors in the training set ("distance", "name", "humidity","temperature") along with their 2-way interaction effects. Based on the difference between train and test RMSE, we see that the model doesn't seem to experienmence much overfitting (2.901905, 3.091595)



We see that the well-tuned model predicts to have a general positive relationship of winning percentage with payroll for all kinds of cars, showing that there's no fundamental error and fits intuition. Yet again, consistent with our result in the multi-linear regression part, price possess a very slight negative relationship with humidity (-0.7435835), while temperature also has a negative influence on the fare, even with the scales taken into account (-0.01605055). However, this result actually makes sense in that humidity is positively associated with temperature (the higher the temperature, the higher the humidity); again, when the weather is warm and not too dry, the fare is predicted to be smaller, while in inclement but dry winter day more charges are applied to the trips.

For another well-tuned lasso model as the ridge counterpart, we see that it possesses a lower RMSE (train: 2.692655, test: 2.861235)and also almost not indication of overfitting (difference = 0.1685796).

3.5 Trees and Forests

In this section, we described how we tried both decision tree and random forest to provide the best test rmse.

1. Decision Tree: We started with a non-tuned decision tree model which takes **name**, **distance**, **source**, and **destination** with $cp = 0$. However, the non-tuned linear model is very over-fitted because the test rmse (8.44) is much larger than the train rmse (5.16).

   Therefore, we tried multiple cp values and chose the cp value based on the test rmse. The final test rmse for the fine-tuned decision tree model is 8.3, which is worse than our best linear model.

2. Random Forest: Since decision tree is not very robust, we also tried random forest models. Since random forest models would automatically pick up the interaction terms, we trained a random forest model based on all of the features with different **mtry**, **maxnode**, **ntree**.

   The best test rmse for the random forest model is 7.76. As we can see, the random forest model is better than the decision tree model, but both of them are still worse than our linear model.

   In figure 6 we can observe that distance is the most decisive predictor with a dominant advantage, and the second to best predictor is the car type, followed by multiple predictors regarding weather and location.

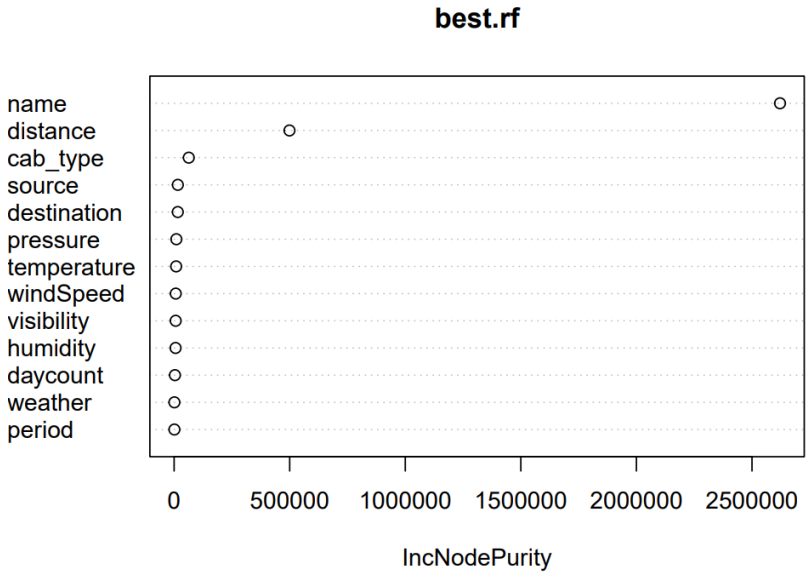


Figure 6: Important predictors deemed by the random forest model.

Model	Train RMSE	Test RMSE
`lm.simple`	8.75	8.84
`lm.brand`	8.70	8.79
`lm.all`	3.05	3.17
`lm.full`	2.55	2.78
`lm.AIC`	2.57	**2.77**
`LASSO`	2.69	2.86
`Ridge`	2.90	3.09
`rf`	2.61	2.78

Table 1: A table summarizing the train and test RMSEs for all the models

4 Conclusion and Takeaways

In conclusion, the linear model has the best prediction power because it has the lowest test RMSE among all the models we tried. The most significant factors for the price prediction are **distance**, **name** (car type), **source**, and **dest**.

The price difference between the two major apps are quite dynamic based on the market condition: the number of available drivers and the number of passengers. In addition, Lyft and Uber have different pricing models, so it's impossible to state which one is cheaper. Lyft has a slightly lower starting price but has a significantly higher fare rate w.r.t the distance compared to Uber rides, which shows that it's a better idea to choose Lyft over Uber for the short-distance trips.

Appendix part I: EDA

Introduction

Brief recap about our proposal

The core question of this project is to identify the pricing model of Uber / Lyft rides, and get an intuition about how to get cheap rides in Boston. The main tools for such analysis is regression.

The dataset used across this project is Uber and Lyft Dataset Boston, MA. The predictors we selected and cared about for the regression analysis includes:

- Distance of the ride
- Weather condition, including a categorical variable for the weather, and other statistics like temperature, humidity, visibility, etc.
- Pick up / Destination location
- Ride type: Uber or Lyft; more specific information about the vehicle, like Lyft XL, UberX, etc.

The interesting question includes: is Uber cheaper than Lyft or vice versa? What's the pricing model between distance and price? Since the data is around seasonal holidays, do time and location affect the price?

A preview of what did in this milestone

Specifically, we did the following things:

- Subsampled the original dataset to create a scale that is processible by R, and compared the distribution of predictor variables to confirm that the distributions won't be off the original ones.
- Created visualizations and analysis without modeling to explore how time, location and weather conditions affect the price, and preprocess them in a proper way for future modeling.
- Built several models to perform regression analysis, including linear, polynomial regression using only the distance, and linear regression with interaction terms between distance and ride type (Uber and Lyft). We used these models to analyze the pricing model and try to answer the first two questions.

What we will do next

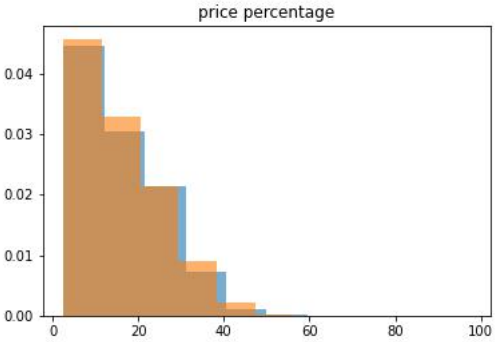
We will be focusing on building a predictive model that utilizes all of the predictor variables we cared about. After that, we will try to interpret them with visuals and formal tests, and we will also use cross validation and RMSE metric to properly evaluate the predictive capability of such models.

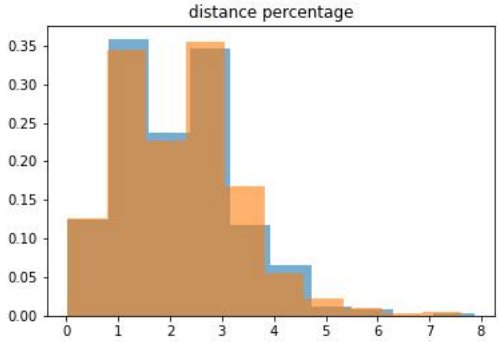
In the end, we will attempt to answer all the questions we proposed and provide the Bostonians a practical ride suggestion supported by data.

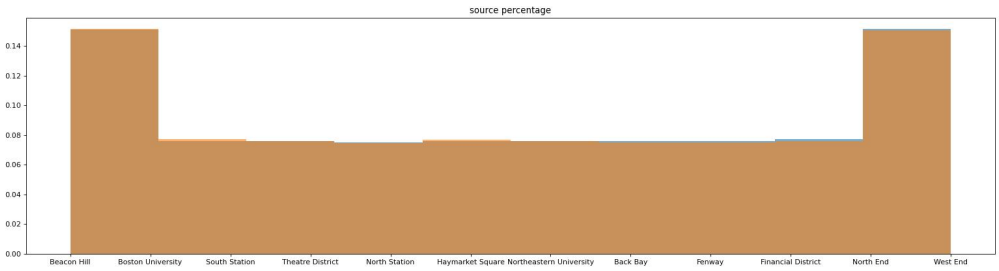
Data Explorations

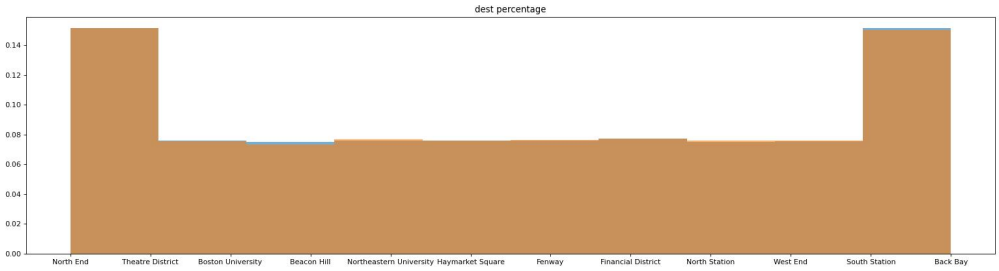
Downsample Distribution

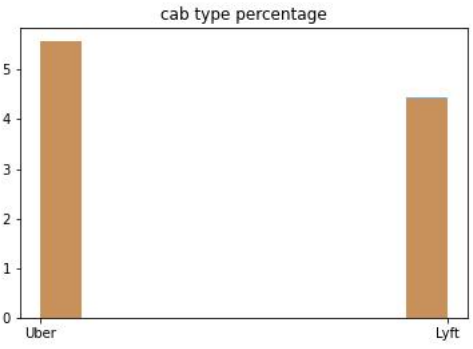
Since the original dataset is too big to process in R, we downsample the dataset to a smaller dataset. However, we'd like to make sure the distribution of the smaller dataaset is similar to the original dataset.

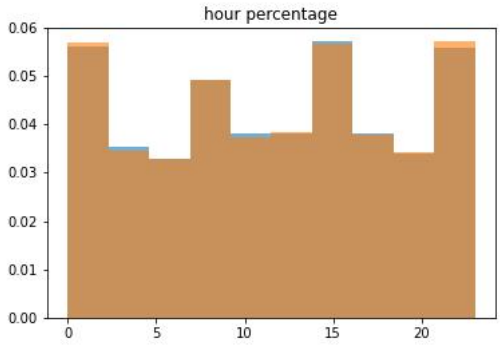












It seems that the downsampling distribution is pretty similar to the original one besides the price distribution, which is a little bit off. We might want to adjust this in the future.

Evaluation of number of orders within the same range

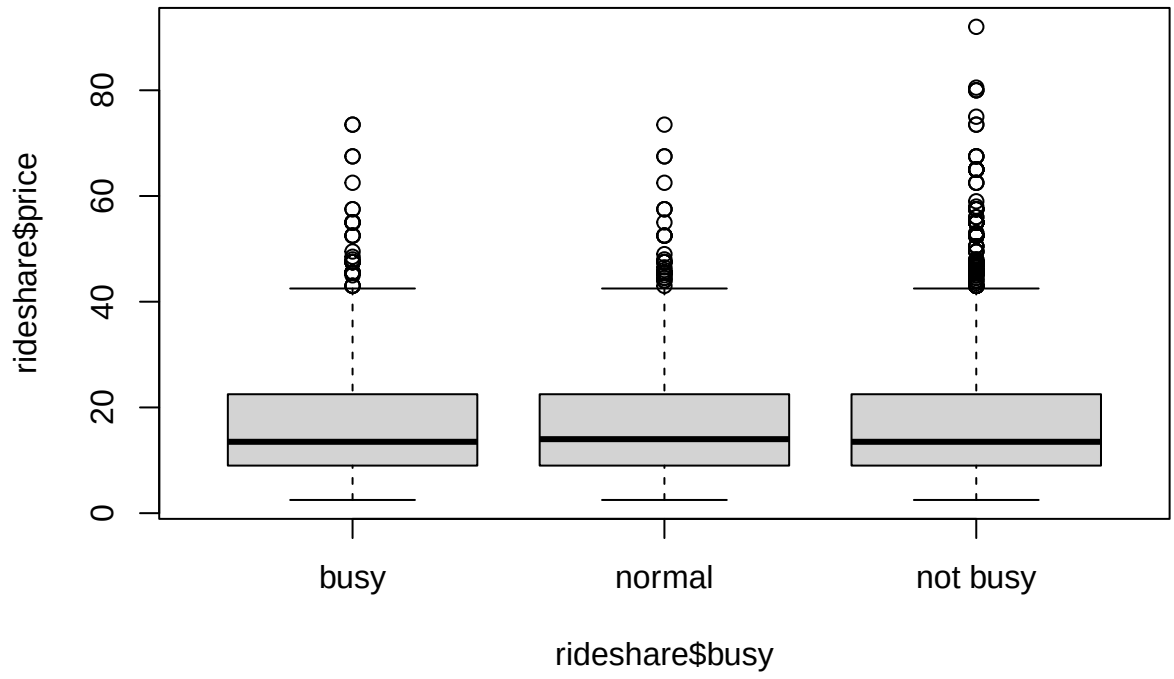
It's intuitive for Uber/Lyft to use supply and demand to price the rides. Therefore, the price of the rides might be much higher if there are a lot of demands. We did some data preprocessing in python by calculating the number of rides within certain time intervals (15 mins here). If there are a lot of rides in the same time range, we expect the price of the rides will be higher.

```
# read the data
rideshare = read.csv("data/sampled_rideshare.csv")
slide_window_df = read.csv("./data/rides_slide_window.csv")
hist(slide_window_df$number_of_orders_within_15_mins)
```

Based on the histogram, it seems that the number of rides is not very continuous. However, we could certainly see that there are certain time ranges (peak hours) with a lot of more rides. Since the value is not very continuous, it might be helpful to convert them into categorical variables based on some time ranges or use a decision tree model in the future.

```
rideshare$busy = as.factor(
    ifelse(rideshare$number_of_orders_within_15_mins < 500,
            "not busy",
            ifelse(rideshare$number_of_orders_within_15_mins < 1000,
            "normal",
            "busy")))
```

```
boxplot(rideshare$price ~ rideshare$busy)
```

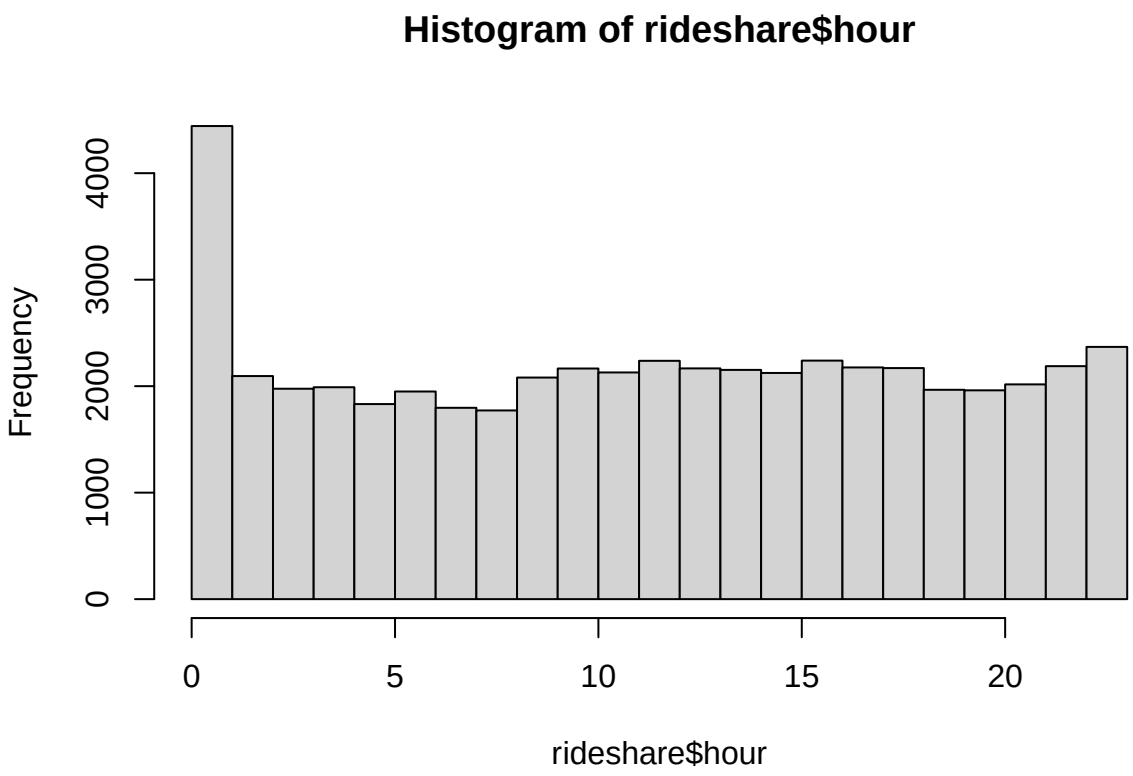



```
summary(aov(rideshare$price ~ rideshare$busy))
```

```
##                   Df  Sum Sq Mean Sq F value Pr(>F)
## rideshare$busy     2      82   41.14   0.478   0.62
## Residuals      45977 3957617   86.08
## 4020 observations deleted due to missingness
```

Based on the boxplot and the anova result, there is no significant difference between the price and the number of orders in the same time range. This is counter-intuitive. It might due to the data generation process. The data provider might presample some data to make each hour contains similar amount of orders. Since it does not represent the true distribution, then our calculation might be misleaded.

```
hist(rideshare$hour)
```



Preprocessing of data: Categorical or Numerical

1 MonthDayHour

Since the data comes solely from the span of two months, it makes sense to simply use the first day of Nov. as the baseline and the days as numerical in the range [1,61] and simply ignore the month column.

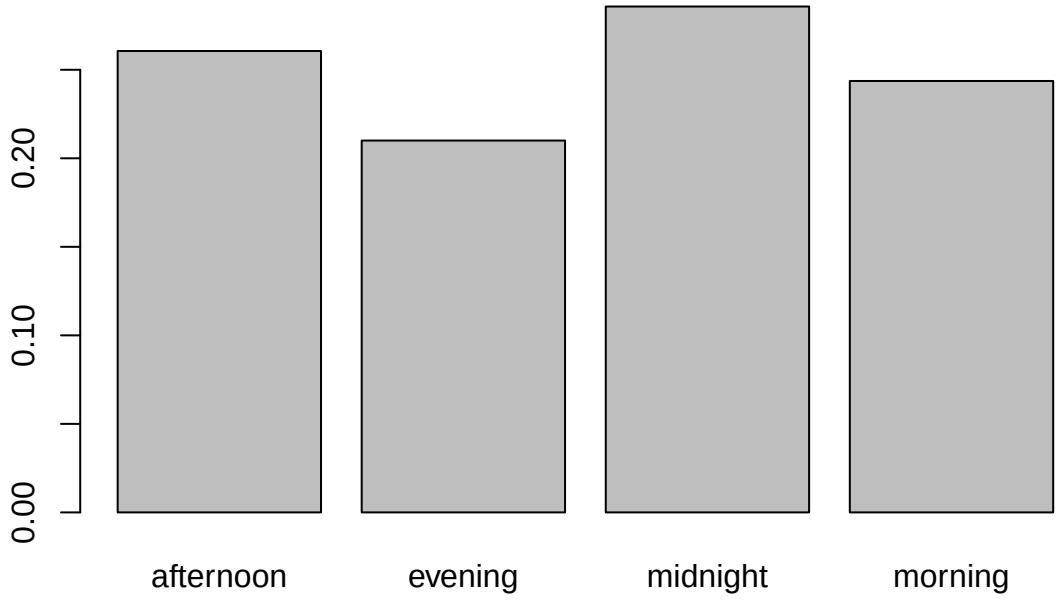
```
n_sample = dim(rideshare)
for(i in 1:50000){
  if(rideshare[i,"month"] == "12"){
    rideshare[i,"day"] = rideshare[i,"day"] + 30
  }
}
```

For the hours columns, it makes sense to create another categorical variable that divides the hours into different periods of the day

```
tmp = rep(0,50000)
for(i in 1:50000){
  if((rideshare[i,"hour"] > 6) && (rideshare[i,"hour"] <= 12)){
    tmp[i]="morning"
  }
  if((rideshare[i,"hour"] > 12) && (rideshare[i,"hour"] <= 18)){
    tmp[i]="afternoon"
```

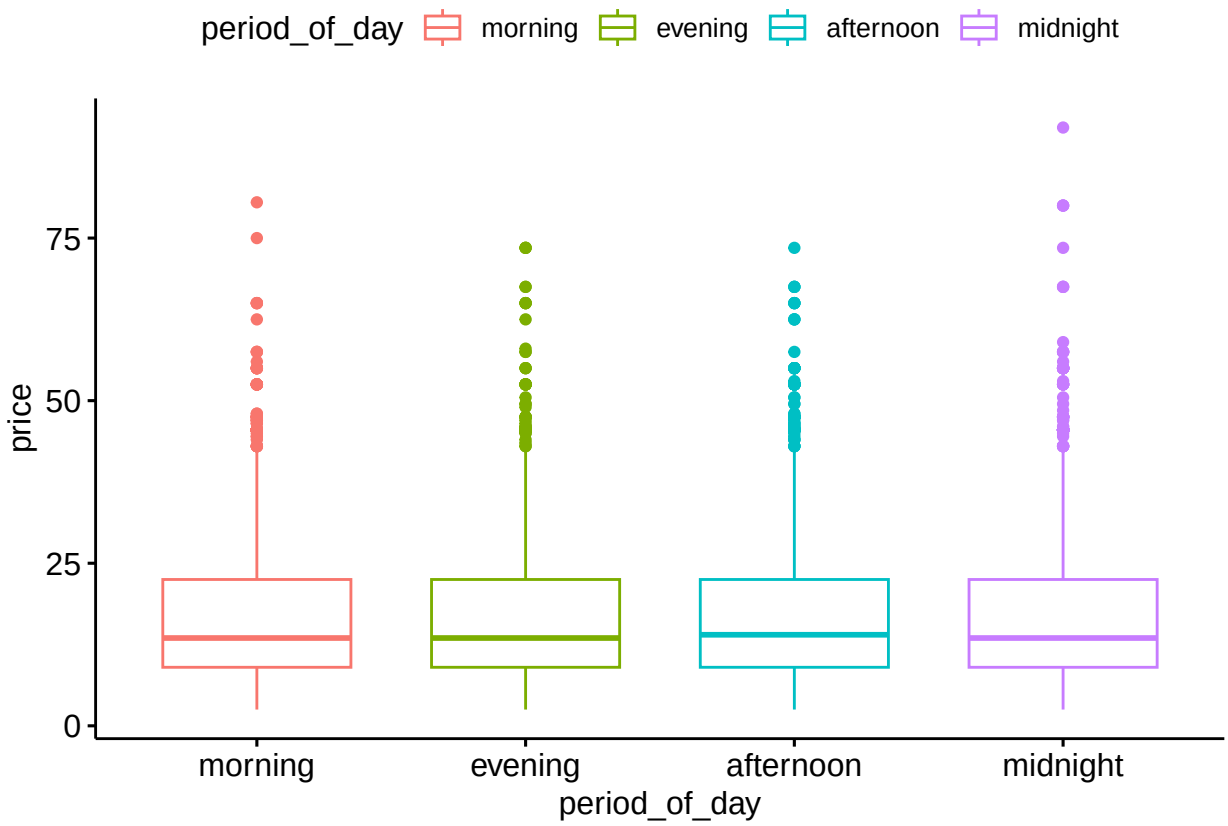
```
  }
  if((rideshare[i,"hour"] > 18) && (rideshare[i,"hour"] <= 23)){
    tmp[i]="evening"
  }
  if((rideshare[i,"hour"] >= 0) && (rideshare[i,"hour"] <= 6)){
    tmp[i]="midnight"
  }
}
rideshare$period_of_day = tmp
barplot(prop.table(table(rideshare$period_of_day)))
library("ggpubr")
```

```
## Loading required package: ggplot2
```




```
ggboxplot(rideshare, x = "period_of_day", y = "price",
          color = "period_of_day",
        ylab = "price", xlab = "period_of_day")
```

```
## Warning: Removed 4020 rows containing non-finite values (‘stat_boxplot()‘).
```



And we see from the plot that the proportion is close to being even among the categories, which also justifies our categorization choice.

```
morning_data = na.omit(rideshare$price[(rideshare$period_of_day == "morning")])
afternoon_data = na.omit(rideshare$price[(rideshare$period_of_day == "afternoon")])
evening_data = na.omit(rideshare$price[(rideshare$period_of_day == "evening")])
midnight_data = na.omit(rideshare$price[(rideshare$period_of_day == "midnight")])
mean_morning = mean(na.omit(rideshare$price[(rideshare$period_of_day == "morning")]))
mean_afternoon = mean(na.omit(rideshare$price[(rideshare$period_of_day == "afternoon")]))
mean_evening = mean(na.omit(rideshare$price[(rideshare$period_of_day == "evening")]))
mean_midnight = mean(na.omit(rideshare$price[(rideshare$period_of_day == "midnight")]))

mean_df = data.frame(morning = mean_morning,
                     afternoon = mean_afternoon,
                     evening = mean_evening,
                     midnight = mean_midnight)

# Pairwise t-test for the means
t.test(morning_data, afternoon_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  morning_data and afternoon_data
## t = -0.9929, df = 23102, p-value = 0.3208
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  -0.3615679  0.1184229
## sample estimates:
## mean of x mean of y
##  16.46247  16.58404
```

```
t.test(morning_data, evening_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  morning_data and evening_data
## t = -0.64309, df = 20401, p-value = 0.5202
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3349041  0.1694342
## sample estimates:
## mean of x mean of y
##  16.46247  16.54520
```

```
t.test(morning_data, midnight_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  morning_data and midnight_data
## t = 0.42894, df = 23702, p-value = 0.668
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1818680  0.2837664
## sample estimates:
## mean of x mean of y
##  16.46247  16.41152
```

```
t.test(afternoon_data, evening_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  afternoon_data and evening_data
## t = 0.30481, df = 20759, p-value = 0.7605
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2109053  0.2885804
## sample estimates:
## mean of x mean of y
##  16.58404  16.54520
```

```
t.test(afternoon_data, midnight_data)
```

```
##
```

```
##  Welch Two Sample t-test
##
## data:  afternoon_data and midnight_data
## t = 1.469, df = 24811, p-value = 0.1418
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.05766511  0.40270851
## sample estimates:
## mean of x mean of y
##  16.58404  16.41152
```

```
t.test(evening_data, midnight_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  evening_data and midnight_data
## t = 1.079, df = 20735, p-value = 0.2806
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1091690  0.3765373
## sample estimates:
## mean of x mean of y
##  16.54520  16.41152
```

```
res.ftest <- aov(price ~ period_of_day, data = rideshare)
```

```
res.ftest
```

```
## Call:
##    aov(formula = price ~ period_of_day, data = rideshare)
##
## Terms:
##                 period_of_day Residuals
## Sum of Squares            223   3957477
## Deg. of Freedom             3     45976
##
## Residual standard error: 9.27777
## Estimated effects may be unbalanced
## 4020 observations deleted due to missingness
```

Appendix Part II: Data Processing

Introduction

In this section, we will try to process the data for further work in predictive modeling.

Data processing

First we load the downsampled data as is mentioned in Appendix I.

```
# drop rows with price being NA
rideshare = read.csv("data/sampled_rideshare.csv")
rideshare = rideshare[is.na(rideshare$price) == F & is.na(rideshare$cab_type) == F, ]

# is there any NA values after we dropped the NAs in price and cab_type?
sum(is.na(rideshare))
```

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

```
# sample the data to take a peek at the predictors
head(rideshare)
```

```
##        X                                    id  timestamp hour day month
## 1  94613 abb1cae0-8ce3-41ed-8ee3-d83fa972eccb 1543647484    6   1    12
## 2 665025 49ddacaf-a32d-44c2-9376-1bbfa5b6427c 1543444688   22  28    11
## 3  99257 6e2b6b90-bfff-42a3-9ec6-51aaebe4adc7 1543486083   10  29    11
## 4 340794 c50c80d8-9e61-4278-9fe5-9db9b125dce8 1544989213   19  16    12
## 5 552051 8997949b-4983-470f-8937-de58c07131ba 1543684681   17   1    12
## 6 490295 05cc2fab-3b8f-4acc-9311-60e9a307d075 1543425068   17  28    11
##              datetime         timezone             source        destination
## 1 2018-12-01 06:58:03 America/New_York  Boston University Financial District
## 2 2018-11-28 22:38:07 America/New_York      South Station   Theatre District
## 3 2018-11-29 10:08:03 America/New_York   Theatre District   Haymarket Square
## 4 2018-12-16 19:40:13 America/New_York      North Station      South Station
## 5 2018-12-01 17:18:00 America/New_York      North Station   Haymarket Square
## 6 2018-11-28 17:11:07 America/New_York Financial District             Fenway
##   cab_type                            product_id         name price distance
## 1     Lyft                         lyft_luxsuv Lux Black XL  45.5     4.36
## 2     Uber 997acbb5-e102-41e1-b155-9df7de0a73f2     UberPool   8.0     1.30
## 3     Lyft                           lyft_line       Shared   9.0     1.72
## 4     Lyft                            lyft_lux    Lux Black  19.5     1.77
## 5     Lyft                        lyft_premier          Lux  10.5     0.77
## 6     Lyft                         lyft_luxsuv Lux Black XL  38.0     3.68
##   surge_multiplier latitude longitude temperature apparentTemperature
```

```
## 1               1  42.3647  -71.0542       34.59              32.16
## 2               1  42.3647  -71.0542       40.43              34.59
## 3               1  42.3519  -71.0643       37.92              32.00
## 4               1  42.3644  -71.0661       43.06              38.26
## 5               1  42.3505  -71.1054       41.89              41.89
## 6               1  42.3644  -71.0661       40.77              35.14
##     short_summary                              long_summary precipIntensity
## 1       Overcast   Light rain in the morning and overnight.               0
## 2       Overcast          Mostly cloudy throughout the day.               0
## 3  Partly Cloudy          Partly cloudy throughout the day.               0
## 4       Overcast                   Rain throughout the day.               0
## 5  Partly Cloudy   Light rain in the morning and overnight.               0
## 6       Overcast          Mostly cloudy throughout the day.               0
##   precipProbability humidity windSpeed windGust windGustTime visibility
## 1                 0     0.78      3.07     3.07   1543672800      9.945
## 2                 0     0.64      9.08    12.72   1543431600     10.000
## 3                 0     0.67      8.11    12.38   1543514400      9.995
## 4                 0     0.71      8.12    13.44   1545015600     10.000
## 5                 0     0.57      2.51     4.03   1543672800      9.953
## 6                 0     0.63      8.76    14.90   1543431600     10.000
##   temperatureHigh temperatureHighTime temperatureLow temperatureLowTime
## 1           44.66          1543690800          35.04         1543712400
## 2           42.61          1543438800          37.60         1543489200
## 3           44.80          1543510800          28.70         1543579200
## 4           43.74          1544990400          34.07         1545044400
## 5           44.54          1543690800          34.74         1543712400
## 6           42.57          1543438800          37.37         1543489200
##   apparentTemperatureHigh apparentTemperatureHighTime apparentTemperatureLow
## 1                   43.99                  1543690800                  35.69
## 2                   36.57                  1543438800                  32.12
## 3                   38.51                  1543510800                  26.30
## 4                   38.36                  1544986800                  28.17
## 5                   43.87                  1543690800                  35.39
## 6                   36.55                  1543438800                  31.91
##   apparentTemperatureLowTime                 icon dewPoint pressure
## 1                 1543712400               cloudy     28.47  1019.21
## 2                 1543478400               cloudy     29.27   994.99
## 3                 1543575600  partly-cloudy-night     27.86  1003.62
## 4                 1545044400               cloudy     34.25  1015.00
## 5                 1543712400    partly-cloudy-day     27.66  1022.54
## 6                 1543478400               cloudy     29.17   991.33
##   windBearing cloudCover uvIndex visibility.1 ozone sunriseTime sunsetTime
## 1         296       0.94       0        9.945 287.3  1543665331 1543698851
## 2         295       1.00       0       10.000 354.8  1543405936 1543439716
## 3         306       0.22       0        9.995 341.1  1543492402 1543526097
## 4          71       1.00       0       10.000 322.7  1544962122 1544994842
## 5         325       0.34       2        9.953 275.8  1543665341 1543698866
## 6         303       1.00       1       10.000 352.4  1543405938 1543439719
##   moonPhase precipIntensityMax uvIndexTime temperatureMin temperatureMinTime
## 1      0.82             0.0000  1543683600          31.71         1543658400
## 2      0.72             0.0000  1543420800          33.85         1543399200
## 3      0.75             0.0000  1543510800          35.02         1543550400
## 4      0.30             0.1246  1544979600          38.88         1544954400
## 5      0.82             0.0000  1543683600          31.31         1543662000
```

```
## 6      0.72             0.0000  1543420800          33.70         1543399200
##   temperatureMax temperatureMaxTime apparentTemperatureMin
## 1          44.66         1543690800                  28.06
## 2          42.61         1543438800                  30.03
## 3          44.80         1543510800                  30.81
## 4          43.74         1544990400                  33.68
## 5          44.54         1543690800                  28.10
## 6          42.57         1543438800                  29.94
##   apparentTemperatureMinTime apparentTemperatureMax apparentTemperatureMaxTime
## 1                 1543658400                  43.99                 1543690800
## 2                 1543399200                  36.57                 1543438800
## 3                 1543550400                  38.51                 1543510800
## 4                 1545019200                  38.36                 1544986800
## 5                 1543662000                  43.87                 1543690800
## 6                 1543399200                  36.55                 1543438800
```

```
# number of records
nrow(rideshare)
```

```
## [1] 45984
```

Next we process some of the special predictors

```
nsamples = nrow(rideshare)

# day
rideshare$daycount = rideshare$day
for(i in 1 : nsamples) {
  if(rideshare[i, "month"] == "12"){
    rideshare[i, "daycount"] = rideshare[i, "daycount"] + 30
  } else {
    rideshare[i, "daycount"] = rideshare[i, "day"]
  }
}

# hour -> time periods
rideshare$period = rep(NA, nsamples)
for (i in 1 : nsamples){
  if ((rideshare$hour[i] > 6) && (rideshare$hour[i] <= 12)) {
    rideshare$period[i] = "morning"
  }
  if ((rideshare$hour[i] > 12) && (rideshare$hour[i] <= 18)) {
    rideshare$period[i] = "afternoon"
  }
  if ((rideshare$hour[i] > 18) && (rideshare$hour[i] <= 23)) {
    rideshare$period[i] = "evening"
  }
  if ((rideshare$hour[i] > 23) || (rideshare$hour[i] <= 6)) {
    rideshare$period[i] = "midnight"
  }
}

# check again for NA values
sum(is.na(rideshare))
```

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

```
# combine some of the descriptions
rideshare$weather = rideshare$short_summary
for(i in 1 : nsamples) {
  if(rideshare$short_summary[i] %in% c(" Overcast ", " Partly Cloudy ", " Mostly Cloudy ")){
    rideshare$weather[i] = " Cloudy "
  } else if (rideshare$short_summary[i] %in% c(" Rain ", " Light Rain ", " Possible Drizzle ", " Drizzl
    rideshare$weather[i] = " Rain "
  } else {
    rideshare$weather[i] = rideshare$short_summary[i]
  }
}
```

Then select needed predictors, perform train / test split, and save the processed data to result

```
# select columns
rideshare = rideshare[c(
  "distance", "cab_type",
  "daycount", "weather",
  "source", "destination", "name",
  "temperature", "humidity", "windSpeed", "visibility", "pressure",
  "period",
  "price"
)]

# take a peek
head(rideshare, 5)
```

```
##   distance cab_type daycount  weather            source        destination
## 1     4.36     Lyft       31  Cloudy  Boston University Financial District
## 2     1.30     Uber       28  Cloudy       South Station   Theatre District
## 3     1.72     Lyft       29  Cloudy   Theatre District   Haymarket Square
## 4     1.77     Lyft       46  Cloudy       North Station      South Station
## 5     0.77     Lyft       31  Cloudy       North Station   Haymarket Square
##           name temperature humidity windSpeed visibility pressure    period
## 1 Lux Black XL       34.59     0.78      3.07      9.945  1019.21  midnight
## 2     UberPool       40.43     0.64      9.08     10.000   994.99   evening
## 3       Shared       37.92     0.67      8.11      9.995  1003.62   morning
## 4    Lux Black       43.06     0.71      8.12     10.000  1015.00   evening
## 5          Lux       41.89     0.57      2.51      9.953  1022.54 afternoon
##   price
## 1  45.5
## 2   8.0
## 3   9.0
## 4  19.5
## 5  10.5
```

```
# train / test split
shuffled.rideshare = rideshare[sample(1:nrow(rideshare)), ]
rideshare.train = shuffled.rideshare[1:40000, ]
rideshare.test = shuffled.rideshare[40001:45984, ]
```

```r
# save to csv
write.csv(rideshare.train, "data/rideshare_train.csv", row.names=F)
write.csv(rideshare.test, "data/rideshare_test.csv", row.names=F)
```

Appendix Part III: Advanced Linear Modeling

Introduction

In this section, we will use the processed data to build predictive models and in the meantime try to analyze the coefficients. Forward feature selection will also be used to reduce the multi-colinearity of the predictors.

This appendix will correspond to section 3.2 to 3.3 in the final report.

Simple Linear regression

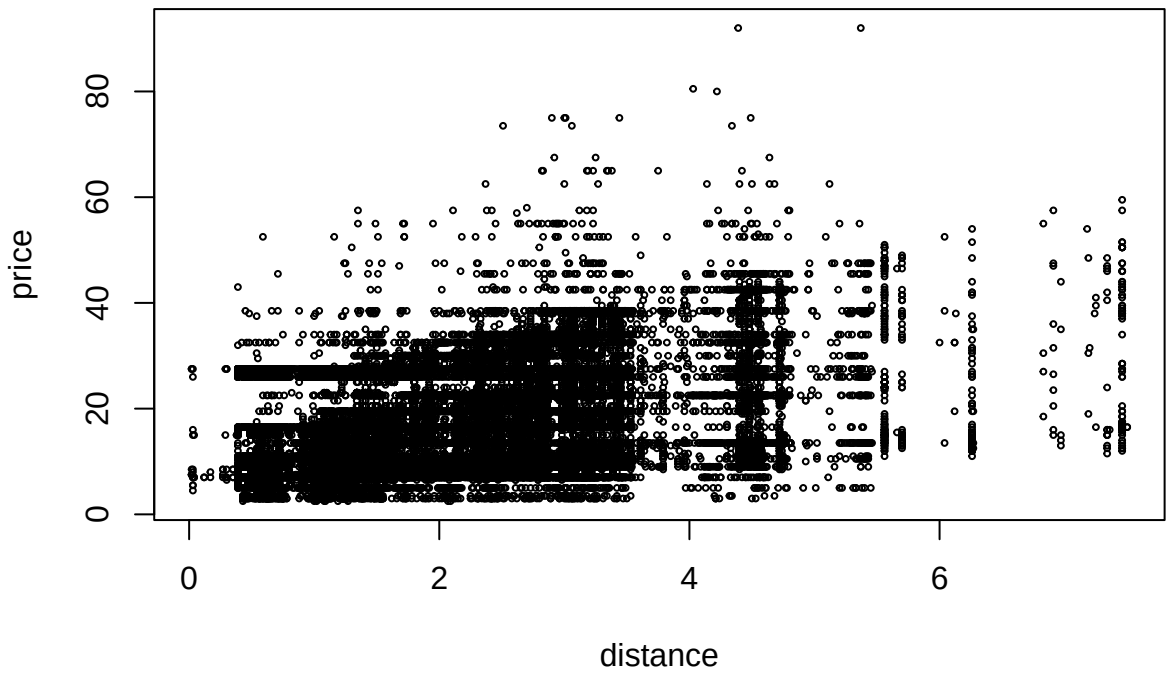
Visualization

In this section we will aim to build a very simple model to analyze the relationship between ride distance and ride price. As is stated earlier, we didn't use the entire dataset because it is too slow to process them all in R. So we randomly sampled 10% of the observations and use the downsampled version in the subsequent model building and analysis. We will start with some simple visualizations to explore the relationship between ride price and ride distance.

```
# helper function for calculating RMSE
RMSE = function(y, yhat) {
  return (sqrt(mean((y - yhat) ^ 2)))
}

# load data from the pre-processed csv
rideshare.train = read.csv("data/rideshare_train.csv")
rideshare.test = read.csv("data/rideshare_test.csv")

# scatter plot for ride price vs ride distance
plot(price ~ distance, data=rideshare.train, cex=0.4)
```



From the scatter plot we can see there is a positive relationship between ride distance and price, although we can see that the variability of the response at the same predictor value is fairly big, and we can expect that the R^2 for a simple linear model would not be very ideal.

Simple Linear model

The next step is to actually fit a simple linear regression model and check the assumptions for simple linear regression. We call it `lm.simple`.

```
lm.simple = lm(price ~ distance, data=rideshare.train)

# model summary
summary(lm.simple)
```

```
##
## Call:
## lm(formula = price ~ distance, data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.713  -6.955  -1.697   4.952  69.278
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.33454    0.09491  108.88   <2e-16 ***
```

```
## distance     2.82165    0.03842   73.45   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.747 on 39998 degrees of freedom
## Multiple R-squared:  0.1188, Adjusted R-squared:  0.1188
## F-statistic:  5395 on 1 and 39998 DF,  p-value: < 2.2e-16
```

```
# RMSE, which can also be calculated via residual standard error and dF
y.pred.train = predict(lm.simple)
y.pred.test = predict(lm.simple, newdata=rideshare.test)
lm.simple.train.rmse = RMSE(rideshare.train$price, y.pred.train)
lm.simple.test.rmse = RMSE(rideshare.test$price, y.pred.test)

# print out the RMSE result
data.frame(trainRMSE=lm.simple.train.rmse, testRMSE=lm.simple.test.rmse)
```

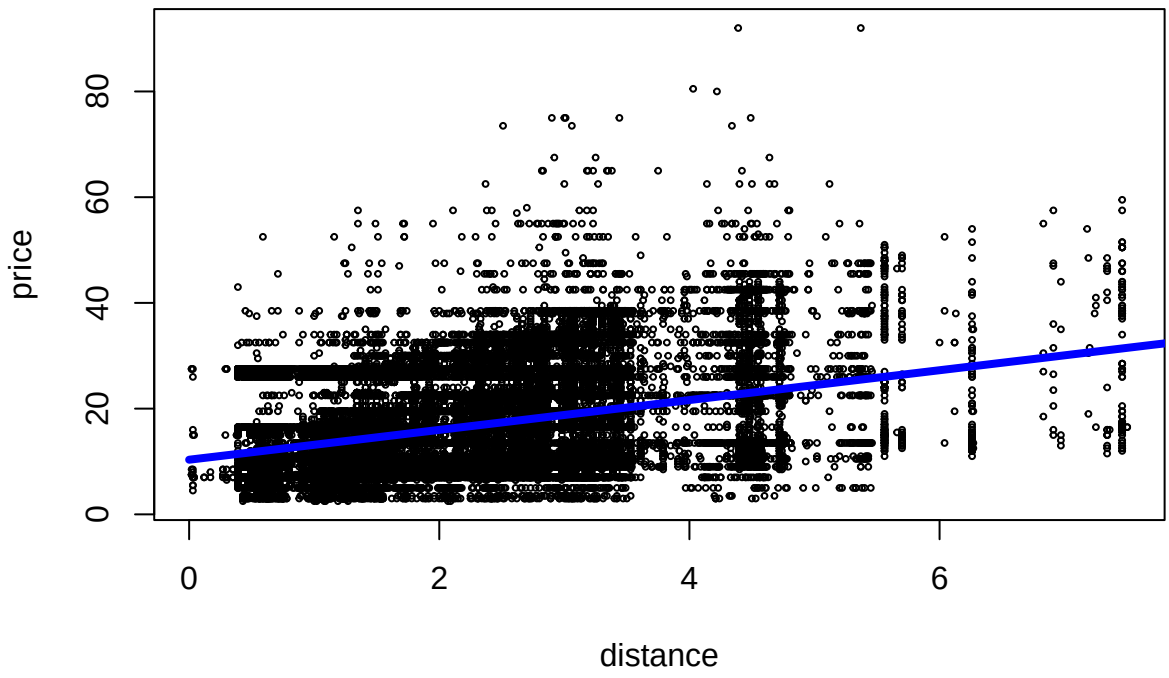
```
##   trainRMSE testRMSE
## 1  8.746425 8.838331
```

From the point estimates we can make the rough interpretation: first, the intercept is 10.33, indicating that a "starting price" would be aroun 10.33 dollar, even if it's a super short ride; second, the slope for the `distance` $is 2.82$ meaning that every 1 mile would be estimated to cost you 2.82 dollar more in general cases. The p-value for the slope is less than 2×10^{-16}, so we can say that the association is very very significant.

R^2 is roughly 0.1188. It's rather distant from 1. It means that nearly 90% of the variability is due the residual errors. We get a pretty high RMSE. This is not surprising as we expect more factors aside from sheer distance would affect the price as we learned they are considered into the pricing model. Also, we notice that the test RMSE is not significantly higher than that of the training, so the model is not overfitting at all (which is also within our expectation, as the model is rather simple and has a very low variance).

Further more we plot the prediction line to see if it matches the scatter plot we created before.

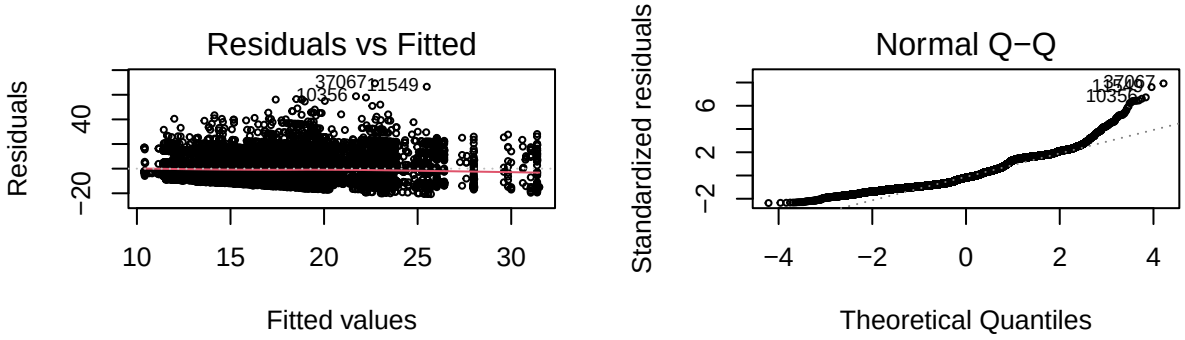
```
x.plot = seq(0, 8, 0.1)
y.pred.plot = predict(lm.simple, newdata=data.frame(distance=x.plot))
plot(price ~ distance, data=rideshare.train, cex=0.4)
lines(x.plot, y.pred.plot, col="blue", lwd=4)
```

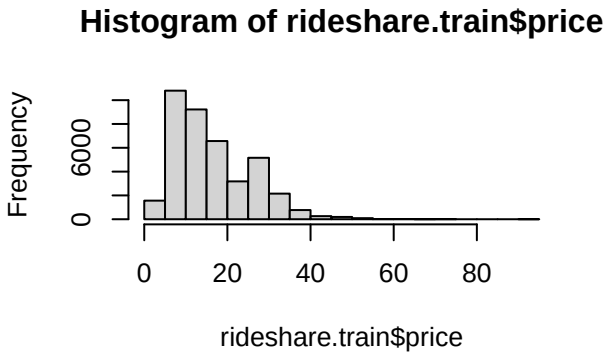


Next we check the assumptions for this linear model and see if we would need any transformations to relieve the skewness issue.

```
# residual and QQ plot
par(mfrow=c(2, 2))
plot(lm.simple, which=1:2, cex=0.5)

# y distribution plot
hist(rideshare.train$price)
```





According to the residual plot, the linearity and constant variance assumption are all poorly conformed. There a lot more positive residuals than negative ones, so `lm.simple` is underestimating the prices. The variability of residuals seem greater when the predicted prices are within range $[18, 23]$. From the QQ plot and the histogram we can see that the normality is also not met. The data looks quite right-skew, and slightly bi-modal.

Polynomial Regression

Since the linear model has a relatively not so satisfying performance, and the linearity is not well met, we think it might be worth a try to use polynomial model. We want to start with a quadratic model

```
lm.quadratic = lm(price ~ poly(distance, 2, raw=T), data=rideshare.train)

# quadratic model summary
summary(lm.quadratic)
```

```
##
## Call:
## lm(formula = price ~ poly(distance, 2, raw = T), data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.774  -6.950  -1.693   4.939  69.264
##
## Coefficients:
```

```
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  10.388273   0.152747  68.010   <2e-16 ***
## poly(distance, 2, raw = T)1  2.770116   0.121035  22.887   <2e-16 ***
## poly(distance, 2, raw = T)2  0.009709   0.021625   0.449    0.653
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.747 on 39997 degrees of freedom
## Multiple R-squared:  0.1189, Adjusted R-squared:  0.1188
## F-statistic:  2697 on 2 and 39997 DF,  p-value: < 2.2e-16
```

```
# F-test
anova(lm.simple, lm.quadratic)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ distance
## Model 2: price ~ poly(distance, 2, raw = T)
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1  39998 3059998
## 2  39997 3059983  1    15.422 0.2016 0.6535
```

From the p-values from both the t-test of the quadratic term and the F-test between `lm.simple` and `lm.quadratic` we can see that the added quadratic term is not of much help, so we will not further more consider it.

Uber or Lyft?

Here we want to research into a very intriguing question: is Uber and Lyft exhibiting different pricing models? To answer this question, we will experiment with another linear model incorporating the type of the ride into the consideration. We call this model `lm.brand`. It considers the `distance` and `cab_type`, as well as their interaction

```
lm.brand = lm(price ~ (distance + cab_type) ^ 2, data=rideshare.train)
summary(lm.brand)
```

```
##
## Call:
## lm(formula = price ~ (distance + cab_type)^2, data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.918  -6.613  -1.650   4.857  67.526
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            10.21331    0.14047  72.710   <2e-16 ***
## distance                3.24850    0.05728  56.711   <2e-16 ***
## cab_typeUber            0.11376    0.18978   0.599    0.549
## distance:cab_typeUber -0.77203    0.07689 -10.041   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 8.7 on 39996 degrees of freedom
## Multiple R-squared:  0.1282, Adjusted R-squared:  0.1282
## F-statistic:  1961 on 3 and 39996 DF,  p-value: < 2.2e-16
```

```
# compare this to the simple model
anova(lm.simple, lm.brand)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ distance
## Model 2: price ~ (distance + cab_type)^2
##   Res.Df     RSS Df Sum of Sq   F    Pr(>F)
## 1  39998 3059998
## 2  39996 3027449  2     32549 215 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# RMSEs
y.pred.brand.train = predict(lm.brand)
y.pred.brand.test = predict(lm.brand, new=rideshare.test)
lm.brand.train.rmse = RMSE(rideshare.train$price, y.pred.brand.train)
lm.brand.test.rmse = RMSE(rideshare.test$price, y.pred.brand.test)
data.frame(trainRMSE=lm.brand.train.rmse, testRMSE=lm.brand.test.rmse)
```

```
##   trainRMSE testRMSE
## 1  8.699784 8.793993
```

From the summary of the model we can derive some useful insights. When `cab_type` is `Lyft`, the regression formula is

$$\text{price} = 10.28 + 3.24 \times \text{distance}$$

; While if `cab_type` is `Uber`, the regression formula is

$$\text{price} = 10.44 + 2.43 \times \text{distance}$$

. This suggests that `Lyft` have a slightly lower starting price but would have a significantly higher fare rate w.r.t the distance compared to Uber rides. The pricing model for these two companies are in fact quite different. The t-test result supports this claim: the coefficients for `cab_typeUber` is not significant with a p-value as big as 0.55, while the coefficients for `distance:cab_typeUber` has a p-value less than 2×10^{-16}, indicating extreme significance. So we would have the impression that in Boston during 2018 in the winter, Lyft would be more expensive for long-distance rides and cheaper for short-distance ones.

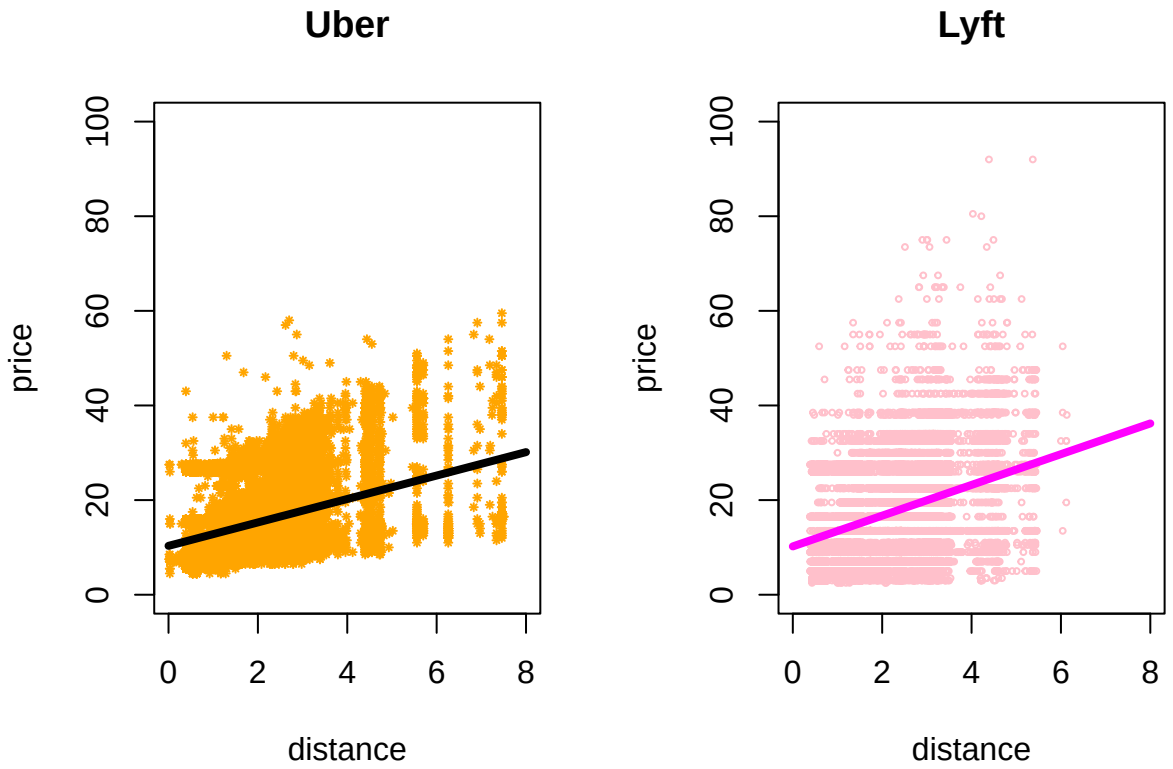
To explore whether this is an improvement compared to the naive linear regression, we perform an F-test and calculated the RMSEs. The p-value for the F-test is less than 2.2×10^{-16}, suggesting a significant improve. Also both train & test RMSEs are better compared to `lm.simple`.

Next, we will provide some visualizations to support this intuition. We will plot the scattered points as well as the prediction line, but in different colors to separate these two cab brands: pink for Lyft, and Orange for Uber.

```
par(mfrow=c(1,2))
rideshare.train.uber = rideshare.train[rideshare.train$cab_type=="Uber", ]
rideshare.train.lyft = rideshare.train[rideshare.train$cab_type=="Lyft", ]
```

```r
# Uber plot
plot(price ~ distance, data=rideshare.train.uber, col="orange",
     cex=0.4, pch=8,  xlim=c(0, 8), ylim=c(0, 100),
     main="Uber")
y.pred.plot.uber = predict(lm.brand,
                           newdata=data.frame(distance=x.plot, cab_type=rep("Uber",
                          length(x.plot))))
lines(x.plot, y.pred.plot.uber, col="black", lwd=4)

# Lyft plot
plot(price ~ distance, data=rideshare.train.lyft, col="pink",
     cex=0.4,
     xlim=c(0, 8),
     ylim=c(0, 100), main="Lyft")
y.pred.plot.lyft = predict(lm.brand,
                           newdata=data.frame(distance=x.plot, cab_type=rep("Lyft", length(x.plot))))
lines(x.plot, y.pred.plot.lyft, col="Magenta", lwd=4)
```



From the prediction line we can see that Lyft has a steeper prediction line, suggesting a significant higher fare rate. However, according to the scatter plots, we can easily find that Uber ride fares have better linearity w.r.t the ride distance, and has a smaller variance in the price.

Linear model using all the predictors

```
# a linear regression model using all predictors, without any interaction terms
lm.all = lm(price ~ ., data=rideshare.train)
summary(lm.all)
```

```
##
## Call:
## lm(formula = price ~ ., data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.459  -1.638  -0.319   1.185  53.201
##
## Coefficients: (2 not defined because of singularities)
##                                  Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                    -6.368e-02  2.006e+00   -0.032 0.974677
## distance                        2.922e+00  1.747e-02  167.267  < 2e-16 ***
## cab_typeUber                    1.462e+01  7.555e-02  193.539  < 2e-16 ***
## daycount                        4.969e-04  2.483e-03    0.200 0.841382
## weather Cloudy                 -6.094e-02  5.105e-02   -1.194 0.232557
## weather Foggy                  -3.274e-01  1.626e-01   -2.013 0.044076 *
## weather Rain                   -1.527e-01  9.547e-02   -1.600 0.109636
## sourceBeacon Hill              -4.660e-01  7.516e-02   -6.200 5.70e-10 ***
## sourceBoston University        -3.375e-01  1.045e-01   -3.230 0.001241 **
## sourceFenway                   -5.038e-02  1.040e-01   -0.484 0.628072
## sourceFinancial District        1.230e-01  7.528e-02    1.634 0.102285
## sourceHaymarket Square         -5.546e-03  1.025e-01   -0.054 0.956855
## sourceNorth End                 2.368e-01  1.017e-01    2.329 0.019861 *
## sourceNorth Station            -2.930e-01  7.521e-02   -3.896 9.78e-05 ***
## sourceNortheastern University  -3.129e-01  1.041e-01   -3.005 0.002654 **
## sourceSouth Station             3.628e-02  1.014e-01    0.358 0.720611
## sourceTheatre District          5.224e-01  7.556e-02    6.914 4.79e-12 ***
## sourceWest End                 -3.370e-01  7.567e-02   -4.453 8.48e-06 ***
## destinationBeacon Hill         -2.857e-01  7.511e-02   -3.804 0.000142 ***
## destinationBoston University    2.772e-02  7.916e-02    0.350 0.726235
## destinationFenway              -3.073e-01  7.818e-02   -3.930 8.49e-05 ***
## destinationFinancial District   4.773e-01  7.537e-02    6.333 2.44e-10 ***
## destinationHaymarket Square     2.487e-01  7.502e-02    3.315 0.000918 ***
## destinationNorth End            8.443e-02  7.495e-02    1.126 0.259980
## destinationNorth Station        1.838e-01  7.482e-02    2.457 0.014021 *
## destinationNortheastern University  3.059e-02  7.711e-02    0.397 0.691603
## destinationSouth Station               NA         NA       NA       NA
## destinationTheatre District     2.328e-01  7.479e-02    3.113 0.001854 **
## destinationWest End            -5.452e-02  7.490e-02   -0.728 0.466664
## nameBlack SUV                   9.643e+00  7.414e-02  130.062  < 2e-16 ***
## nameLux                         1.187e+01  7.655e-02  155.115  < 2e-16 ***
## nameLux Black                   1.704e+01  7.598e-02  224.294  < 2e-16 ***
## nameLux Black XL                2.623e+01  7.660e-02  342.458  < 2e-16 ***
## nameLyft                        3.536e+00  7.615e-02   46.434  < 2e-16 ***
## nameLyft XL                     9.265e+00  7.640e-02  121.262  < 2e-16 ***
## nameShared                             NA         NA       NA       NA
```

```
## nameUberPool                       -1.179e+01  7.403e-02 -159.315  < 2e-16 ***
## nameUberX                          -1.083e+01  7.384e-02 -146.655  < 2e-16 ***
## nameUberXL                         -4.951e+00  7.432e-02  -66.613  < 2e-16 ***
## nameWAV                            -1.091e+01  7.351e-02 -148.439  < 2e-16 ***
## temperature                         7.947e-04  2.960e-03    0.269 0.788313
## humidity                           -1.311e-01  1.849e-01   -0.709 0.478175
## windSpeed                           7.025e-04  7.789e-03    0.090 0.928131
## visibility                         -2.168e-02  1.228e-02   -1.766 0.077418 .
## pressure                           -4.020e-06  1.961e-03   -0.002 0.998364
## periodevening                       1.997e-02  4.632e-02    0.431 0.666445
## periodmidnight                     -1.335e-02  4.328e-02   -0.308 0.757734
## periodmorning                      -2.356e-02  4.828e-02   -0.488 0.625602
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.055 on 39954 degrees of freedom
## Multiple R-squared:  0.8926, Adjusted R-squared:  0.8925
## F-statistic:  7382 on 45 and 39954 DF,  p-value: < 2.2e-16
```

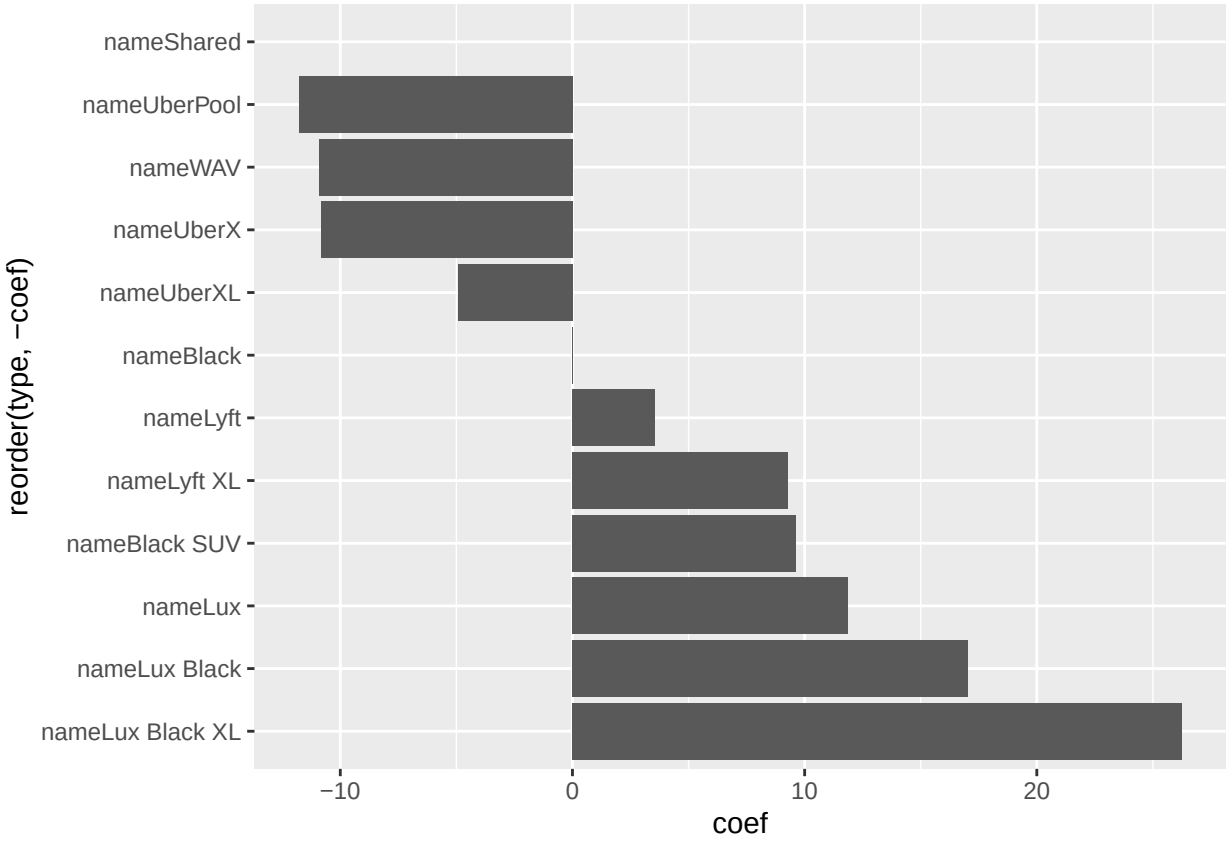
From the summary we can see that the significant predictors are `cab_type, name, distance, source, destination`. The coefficient for distance is 2.92, slightly higher than that we got in the simple linear regression. This means, holding every other factor constant, every mile would cost you 2.92 dollar on average.

The coefficient for `cab_typeUber` is 14.62, which is drastically different from the result we had when we used simple linear regression, indicating that generally Uber is more expensive compared to Lyft. The difference between the fare of these two brands do not seem to have so much gap, so it's probably due to the multicolinearities since we had another categorical predictor `name` which describes the specific type of the car. The reference group for this variable is `Black`, which is one of the Lyft's luxury ride type. Based on the coefficients, we can create a visual to see the fare ranking of these ride types.

```
lm.all.ridetype = lm.all$coefficients[30:40]
lm.all.ridetype['nameBlack'] = 0
lm.all.ridetype = data.frame(lm.all.ridetype)
lm.all.ridetype['type'] = rownames(lm.all.ridetype)
colnames(lm.all.ridetype) = c('coef', 'type')
rownames(lm.all.ridetype) = NULL

library(ggplot2)
ggplot(data=lm.all.ridetype, aes(x=coef, y=reorder(type, -coef))) +
  geom_bar(stat="identity")
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```



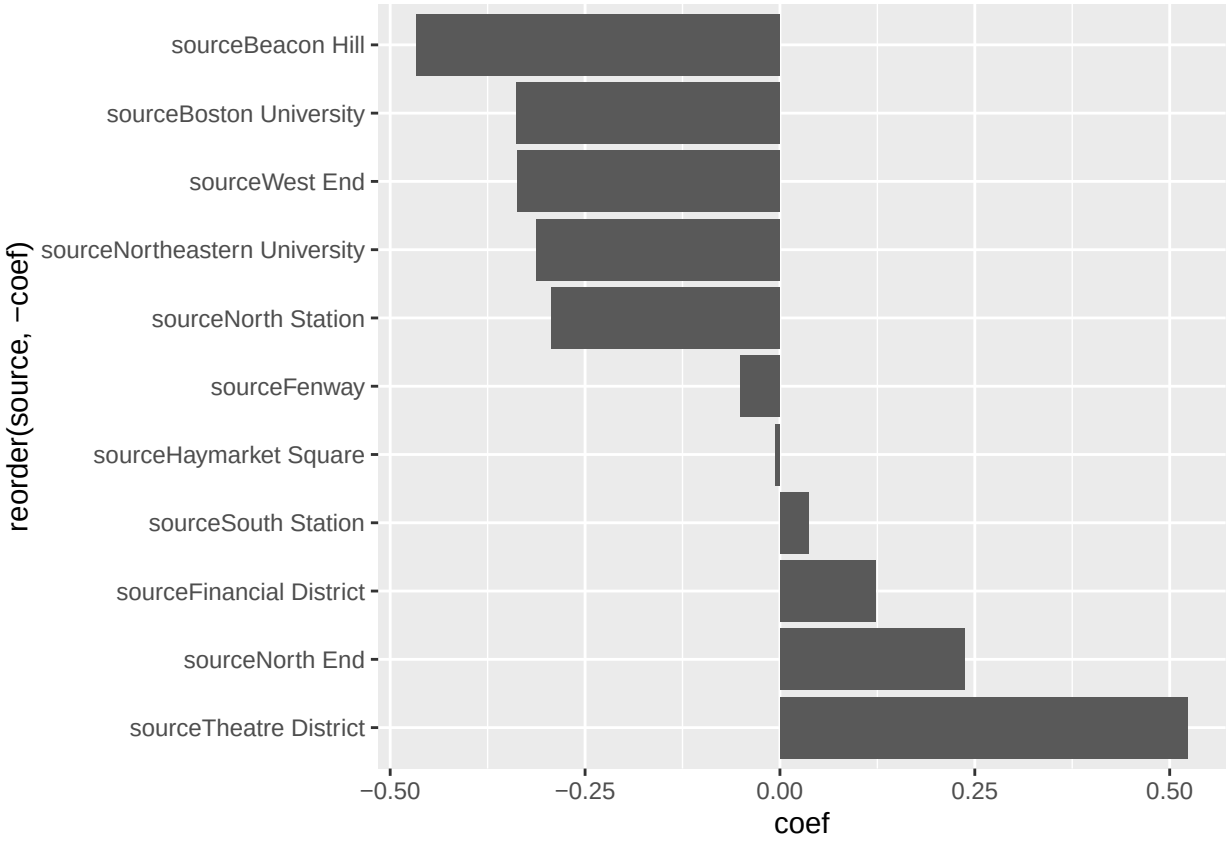
From the ranked barplot we can see that:

- shared rides have NA estimates, which is due to it is linearly related to other columns (prabably `cab_typeUber` and all other ride types of Uber will determine it).
- Uber Pool, Uber X and Lyft are the cheapest rides generally speaking.
- WAV is also very cheap because it is to certain extent a social welfare for the physically disabled citizens.
- The XL rides are more expensive, and the luxury models (Lyft Black SUV, Uber Lux, Lyft Black XL, etc) are much more expensive.
- It's surprising that Lyft Black actually seems to be cheaper than Lyft standard. But it may happen sometimes because the pricing model is probably due to machine learning and the over-demand of Lyft standard might cause a rise in the fare rate, while not many people choose Lyft Black and will cause a lower price.
- Generally for the same level (like Uber Lux compared to Lyft Lux Black, or UberX compared to Lyft standard), the coefficients for Uber rides are significantly lower, probably because of there is a large compensation coefficient for `cab_typeUber` already. Considering both factors, the Lyft ride still seems a better deal, but with marginal advantage. For example, the coefficient for `nameLyft` is 3.56, while the summation of `cab_typeUber` and `UberX` is 3.78.

Let's create two more visuals for the source and destination coefficients.

```
lm.all.source = lm.all$coefficients[8:18]
lm.all.source = data.frame(lm.all.source)
lm.all.source['source'] = rownames(lm.all.source)
colnames(lm.all.source) = c('coef', 'source')
rownames(lm.all.source) = NULL
```

```
ggplot(data=lm.all.source, aes(x=coef, y=reorder(source, -coef))) +
  geom_bar(stat="identity")
```

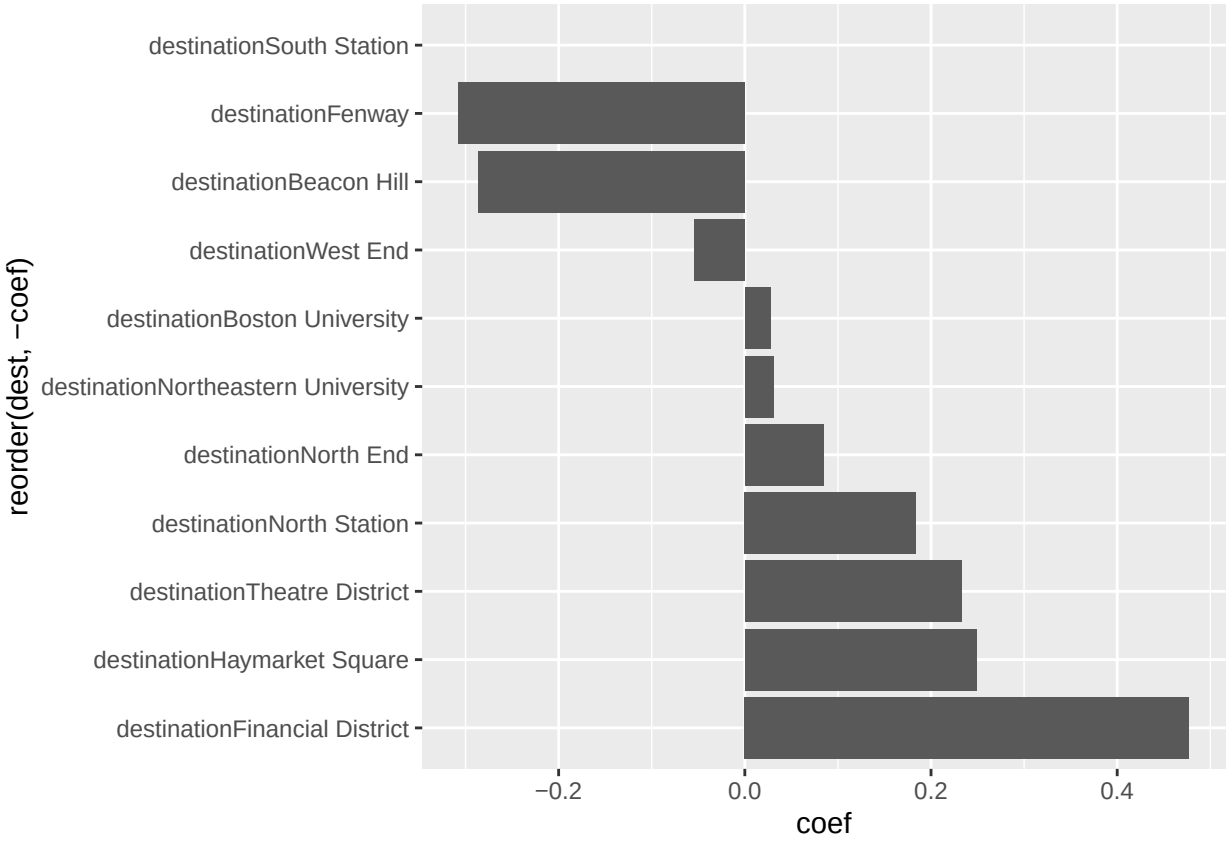


According to this visual, the seemingly most expensive source location is Theatre District near Downtown Crossing, followed by North Station where TD Garden is located. This makes perfect sense because Boston Downtown has a large population density and TD Garden has many games and performances which will induce heavy traffic, so the dynamic pricing system might yield a higher price due to the high demand.

The least expensive source locations are Beacon Hill, NEU and BU. Beacon Hill is located at Park Street, near Downtown, so it's a bit confusing. NEU and BU are located in Malden and Allston, and the riders should be mostly students nearby, and they might tend to choose cheaper rides.

```
lm.all.dest = lm.all$coefficients[19:29]
lm.all.dest = data.frame(lm.all.dest)
lm.all.dest['dest'] = rownames(lm.all.dest)
colnames(lm.all.dest) = c('coef', 'dest')
rownames(lm.all.dest) = NULL
ggplot(data=lm.all.dest, aes(x=coef, y=reorder(dest, -coef))) +
  geom_bar(stat="identity")
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```



The visual for the destination shows a similar trend. Places near Downtown like Financial District and Theatre District, and places near TD Garden has a higher ride price in general.

We didn't see significances in the factor of weather, which is out of our expectation, because we presumed that during the Holiday seasons, the extreme weather conditions like storm would impact the ride prices positively. The estimated coefficients for `weatherFoggy, weatherRain, weatherCloudy` are negative, contradicting to our theory. But there is an explanation. These categorical predictors have a high correlation with the numerical predictors like `visibility`, `temperature`, `windSpeed`, `humidity`. And the coefficients are in accordance with our assumption: the worse the weather condition is, like lower visibility and higher wind speed, would have a positive impact on the ride price. The only exception is when the temperature is higher, the ride would cost more. Since the data is collected during November and December, it seems a little bit confusing.

The `daycount` has a negative coefficient, meaning the later it is, the lower the price is. This also contradicts our common sense, as we would expect a higher price during holiday season near Christmas, but since the p-value is super high (0.88), so maybe this can be neglegible.

The time period seems not so significant in the prediction, but generally during late night and early morning when there are less passengers, the price would be lower.

Let's calculate the RMSE as a comparison to other models

```
lm.all.rmse.train = RMSE(
  rideshare.train$price,
  predict(lm.all)
)

lm.all.rmse.test = RMSE(
```

```
  rideshare.test$price,
  predict(lm.all, new=rideshare.test)
)
```

```
## Warning in predict.lm(lm.all, new = rideshare.test): prediction from a rank-
## deficient fit may be misleading
```

```
lm.all.rmse = data.frame(trainRMSE=lm.all.rmse.train, testRMSE=lm.all.rmse.test)
lm.all.rmse
```

```
##   trainRMSE testRMSE
## 1   3.05299 3.169056
```

The RMSE is much lower (3.05 for train and 3.17 for test), but still not satisfying, the prediction can have up to 3 dollar error off the real price.

Forward Model Selection

In this section we will try to explore which predictors and interactions are important with forward model selection. Let's start with a full interaction model that takes into account all the predictors as well as their interaction terms.

```
lm.full = lm(price ~ .^2, data=rideshare.train)
lm.full.rmse.train = RMSE(
  rideshare.train$price,
  predict(lm.full)
)

lm.full.rmse.test = RMSE(
  rideshare.test$price,
  predict(lm.full, new=rideshare.test)
)

lm.full.rmse = data.frame(trainRMSE=lm.full.rmse.train, testRMSE=lm.full.rmse.test)
lm.full.rmse
```

```
##   trainRMSE testRMSE
## 1  2.554777 2.781018
```

Both train and test RMSEs are decreased (2.55 for train and 2.78 for test), indicating an improvement in the prediction capability. However, for the purpose of analysis, we also used AIC forward selection:

```
lm.aic = step(
  lm.simple,
  scope=list(upper=lm.full, lower=~1),
  direction="forward",
  trace=F
)

lm.aic.rmse.train = RMSE(
```

```
  rideshare.train$price,
  predict(lm.aic)
)

lm.aic.rmse.test = RMSE(
  rideshare.test$price,
  predict(lm.aic, new=rideshare.test)
)
```

```
## Warning in predict.lm(lm.aic, new = rideshare.test): prediction from a rank-
## deficient fit may be misleading
```

```
lm.aic.rmse = data.frame(trainRMSE=lm.aic.rmse.train, testRMSE=lm.aic.rmse.test)

# RMSE result for AIC forward selection model
lm.aic.rmse
```

```
##   trainRMSE testRMSE
## 1  2.570056 2.770926
```

The train RMSE increases a little bit compared to the full model (2.57), but test RMSE are decreased (2.77), indicating a slight improvement in the overfitting issue. We can see the formula of this forward selection model:

```
formula(lm.aic)
```

```
## price ~ distance + name + source + destination + distance:name +
##     name:source + distance:source + source:destination + name:destination +
##     distance:destination
```

From the formula we can see that the important predictors are just the same as what we observed in `lm.all`, the key factors are `distance`, `name` (which is the specific car type), `destination`, `source` and their pairwise interaction.

Appendix Part IV: Tree and Random Forest

Introduction

In this section, we will use the processed data to explore decision trees and random forests to build predictive models.

This appendix will correspond to section 4.5 in the final report.

Decision Trees

```
#loading libraries
library(rpart)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice
```

```
# helper function for calculating RMSE
RMSE = function(y, yhat) {
  return (mean((y - yhat) ^ 2))
}

# load data from the pre-processed csv
rideshare.train = read.csv("data/rideshare_train.csv")
rideshare.test = read.csv("data/rideshare_test.csv")
```

Based on our previous data analysis for the linear models, we know that **distance**, **name**, **destination**, and **source** are the significant factors in terms of predictions for the price.

Therefore, we'd like to tree the same features but with decision tree.

```
tree1 = rpart(price ~ distance + name + destination + source, data = rideshare.train,
              control=list(maxdepth = 50, minbucket = 2, cp = 0))
price.tree1.train = predict(tree1)
price.tree1.test = predict(tree1, new=rideshare.test)
rmse.tree1.train = RMSE(rideshare.train$price,price.tree1.train)
rmse.tree1.test = RMSE(rideshare.test$price,price.tree1.test)
c(rmse.tree1.train, rmse.tree1.test)
```

```
## [1] 5.165786 8.440624
```

As we can see, the testing RMSE is larger than our training RMSE, which shows that the decision tree model is overfitting. It might be overfitting because of our low cp value.

We want to fine tune the cp value to get the best RMSE score we can achieve with the decision tree model.

```
best.rmse.test = sd(rideshare.test$price)
cps = seq(0,0.02,0.0002)
for(i in 1:length(cps)){
  cp = cps[i]
  tree2 = rpart(price ~ distance + name + destination + source, data = rideshare.train,
              control=list(maxdepth = 50, minbucket = 2, cp = cp))
  price.tree2.train = predict(tree2)
  price.tree2.test = predict(tree2, new=rideshare.test)
  rmse.tree2.train = RMSE(rideshare.train$price,price.tree2.train)
  rmse.tree2.test = RMSE(rideshare.test$price,price.tree2.test)
  if(rmse.tree2.test < best.rmse.test){
    best.rmse.test = rmse.tree2.test
    best.rmse.train = rmse.tree2.train
    best.cp = cp
    best.tree2 = tree2
  }
}
c(best.rmse.train, best.rmse.test)
```

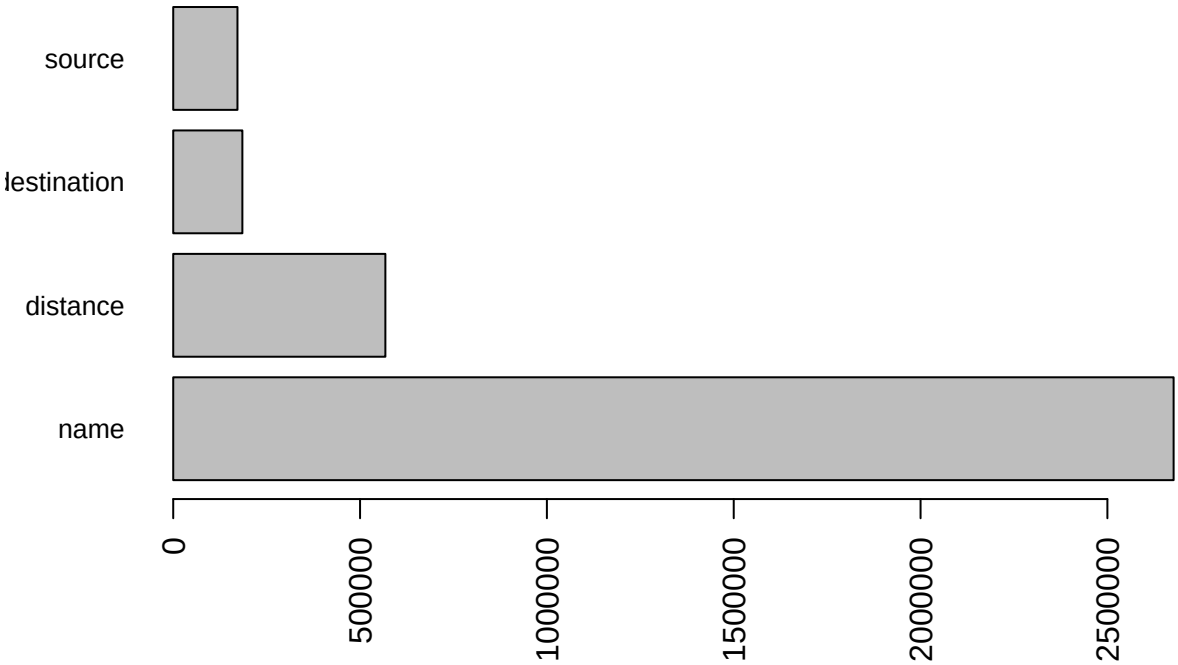
```
## [1] 7.060691 8.304238
```

Even though the test rmse for the fune-tuning tree is lower than that of **tree1**, the test rmse is still higher than the test rmse of the best linear model we had.

Since **tree1** is already overfitting, we should not consider adding more features or try the full model

It's quite interesting to see that the **name** variable has the highest variable importance among all the variables.

```
barplot(tree1$variable.importance,horiz=T, las=2,cex.names=0.8)
```



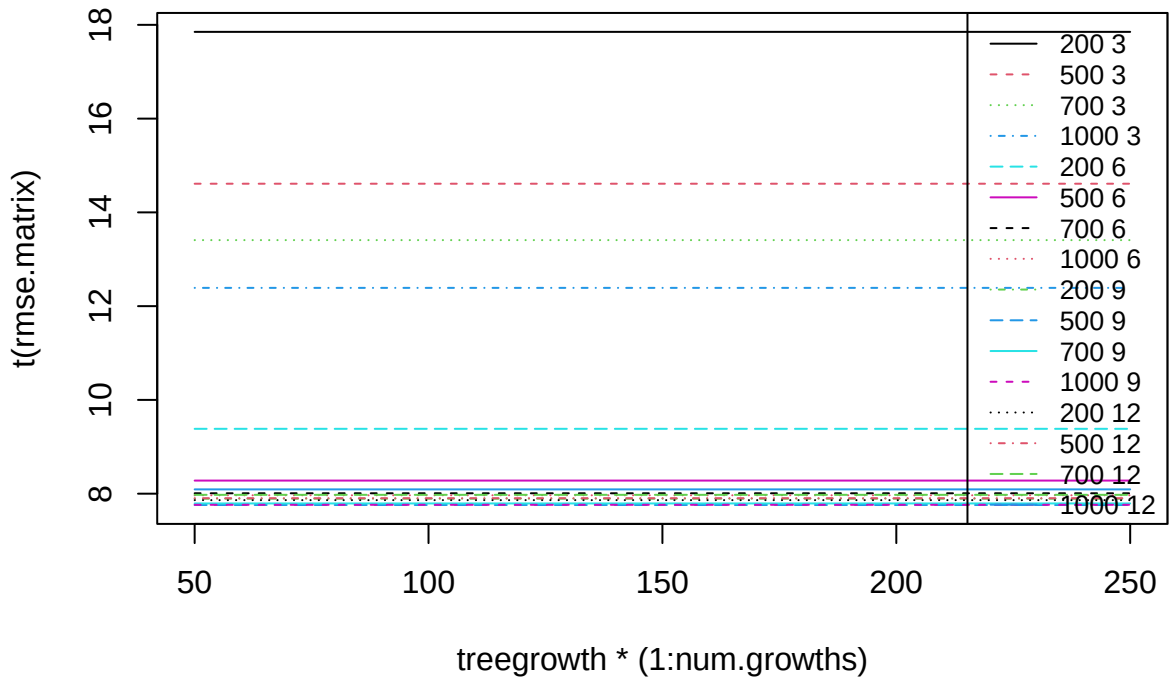
Decision trees are not very robust, which might contribute to the overfitting, so we'd like to train a random forest.

Random Forest

For the random forest model, we tried a random forest model with all the features. Since random forest models should pick up the interaction relationship automatically, there is no need for us to speicify the interaction terms.

```
set.seed(139)
maxnodes.vec=c(200,500, 700, 1000)
mtry.vec = c(3, 6, 9, 12)
par.grid=expand.grid(maxnodes=maxnodes.vec,mtry=mtry.vec)
treegrowth = 50
num.growths = 5
rf.list = vector("list",nrow(par.grid))
yhats.list = vector("list",nrow(par.grid))
for(i in 1:nrow(par.grid)){
  yhats.list[[i]] = matrix(NA,nrow=length(rideshare.test$price),ncol=num.growths)
  model = randomForest(price~. ,data=rideshare.train, maxnodes=par.grid$maxnodes[i],
  mtry=par.grid$mtry[i],ntree=treegrowth)
  rf.list[[i]] = model
  yhats = predict(model,new=rideshare.test)
  yhats.list[[i]][,1] <- yhats
}
```

```
for(j in 2:num.growths){
  for(i in 1:nrow(par.grid)){
    model = grow(rf.list[[i]],treegrowth)
    rf.list[[i]] = model
    yhats = predict(model,new=rideshare.test)
    yhats.list[[i]][,j] <- yhats
  }
}
# calculate RMSEs
rmse.matrix = matrix(NA,nrow=nrow(par.grid),ncol=num.growths)
for(i in 1:nrow(par.grid)){
yhats = yhats.list[[i]]
rmse.matrix[i,] = RMSE(yhats, rideshare.test$price)
}
matplot(treegrowth*(1:num.growths),t(rmse.matrix),type="l")
legend("topright",legend=paste((par.grid)[,1],(par.grid)[,2]),col=1:6, lty=1:5, cex=0.8)
```

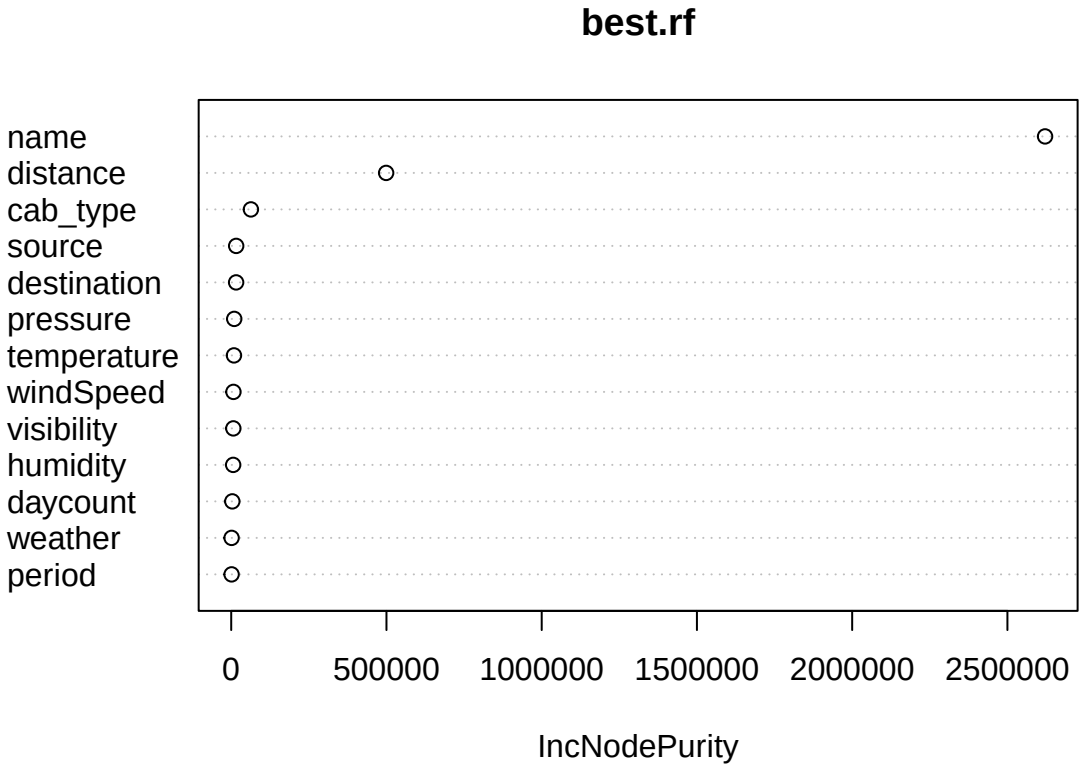



```
best.rf = rf.list[which.min(rmse.matrix)][[1]]
price.train = predict(best.rf)
c(RMSE(price.train, rideshare.train$price), rmse.matrix[which.min(rmse.matrix)])
```

```
## [1] 6.836275 7.758767
```

Based on the result, we can see that the number of trees does not influence the test rmse that much. However,the number of factors at each node and the max number of nodes matter a lot.

```
varImpPlot(best.rf)
```



The best rmse of the random forest is better than that of decision trees. However, it's still not as good as the selected linear model. Maybe the pricing model is closer to a linear model and the ols model generates the unbiased estimator, so it achieves the best test rmse.

Appendix part V: Regularization

```
# Easy regularization
# Ridge model
rideshare = read.csv("data/rideshare_train.csv")
rideshare_test = read.csv("data/rideshare_test.csv")
RMSE = function(y,yhat){
  SSE = sum((y-yhat)^2)
  return(sqrt(SSE/length(y)))
}
rideshare = read.csv("data/rideshare_train.csv")
rideshare_test = read.csv("data/rideshare_test.csv")
    library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
    set.seed(139)
    colnames = c("price", "distance", "name", "humidity","temperature")
    rideshare_test = rideshare_test[, colnames]
    all_model = lm(price ~ (distance + name + humidity + temperature)^2, data=rideshare)
    X = model.matrix(formula(all_model), data= rideshare)[, -1]
    X_test = model.matrix(formula(all_model), data= rideshare_test)[, -1]
    ridges = cv.glmnet(X, rideshare$price,alpha=0, lambda = 2^(seq(1, 15, 0.1)), nfolds=10)
    ridges$lambda.min
```

```
## [1] 2
```

```
    ridge1 = glmnet(X, rideshare$price, alpha =0, lambda=ridges$lambda.min)
    result = predict(ridge1, newx = X)
    result_test = predict(ridge1, newx = X_test)
    rmse =  RMSE(result,rideshare$price)
    rmse_test =  RMSE(result_test,rideshare_test$price)
    print(rmse)
```

```
## [1] 2.901905
```

```
    print(rmse_test)
```

```
## [1] 3.091595
```

```
    sampled = sample(nrow(rideshare),100)
    dummy_df = X[sampled,]
    plot(price~ distance,data=rideshare,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
    dummyx = seq(min(rideshare$distance),max(rideshare$distance), 0.5)
    yhats.ridge=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
    for(i in 1:nrow(dummy_df)){
      rep_df = matrix(0, nrow= length(dummyx), ncol= ncol(dummy_df))
      for(j in 1:length(dummyx)){
        rep_df[j, ] = dummy_df[i, ]
      }
      colnames(rep_df) <- colnames(dummy_df)
      rep_df = data.frame(rep_df)
      rep_df$distance=dummyx
      yhat.ridge = predict(ridge1,newx=data.matrix(rep_df))
      lines(yhat.ridge~dummyx,col=rgb(1,0.5,0,0.5),lwd=0.5,lty=2:3)
      yhats.ridge[i,]=yhat.ridge
    }
    coef(ridge1)["distance", ]
```

```
## [1] 1.11271
```

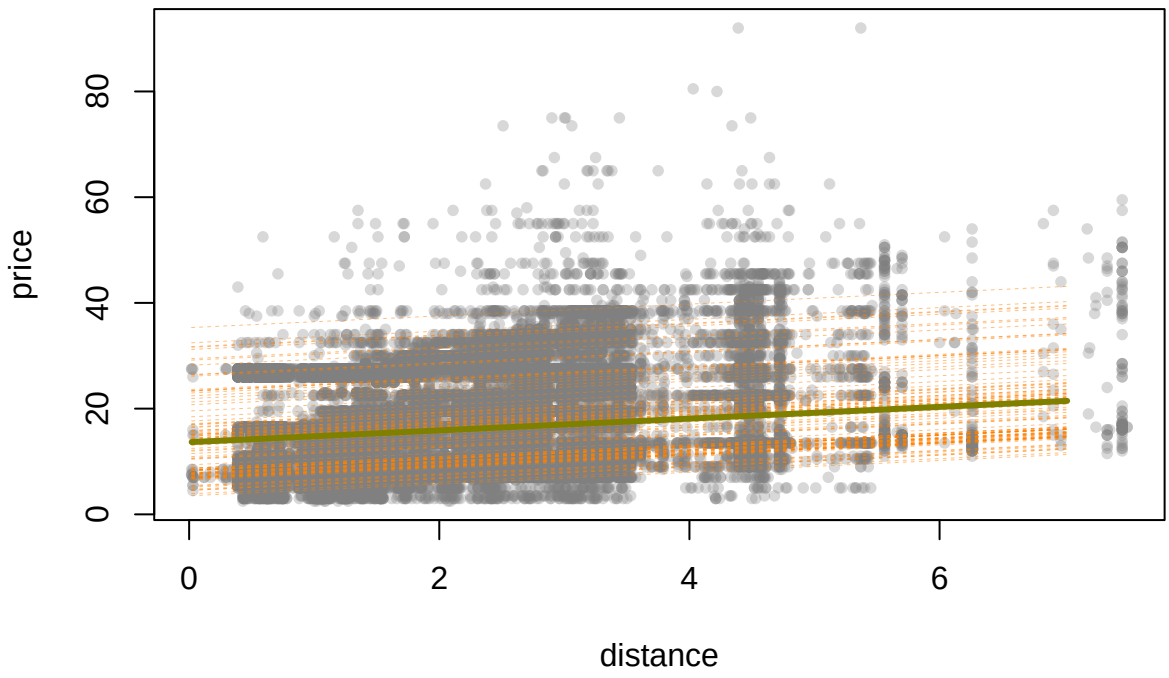
```
    coef(ridge1)["humidity", ]
```

```
## [1] -0.7435835
```

```
    coef(ridge1)["temperature", ]
```

```
## [1] -0.01605055
```

```
    mean_yhat = apply(yhats.ridge,2,mean)
    lines(mean_yhat~dummyx,col=rgb(0.5,0.5,0,1),lwd=3)
```




```
# Lasso Model
library(glmnet)
set.seed(139)
    names = c("price", "distance", "name", "humidity","temperature")
    rideshare_test = rideshare_test[, names]
    all_model = lm(price ~ (distance + name + humidity + temperature) ^ 2, data=rideshare)
    X = model.matrix(formula(all_model), data= rideshare)[, -1]
    lassos = cv.glmnet(X, rideshare$price, alpha=1,
                       lambda = 2^(seq(-15, 15, 0.1)), nfolds=10)
    lassos$lambda.min
```

```
## [1] 0.001202289
```

```
    lassos = glmnet(X, rideshare$price, alpha =1, lambda=lassos$lambda.min)
    yhat.train.lassos = predict(lassos, newx= model.matrix(all_model, data=rideshare)[, -1])
    yhat.test.lassos = predict(lassos, newx= model.matrix(all_model, data=rideshare_test)[, -1])
    RMSE.lassos.train = RMSE(rideshare$price, yhat.train.lassos)
    RMSE.lassos.test = RMSE(rideshare_test$price, yhat.test.lassos)

    RMSE.lassos.train
```

```
## [1] 2.692655
```

```
    RMSE.lassos.test
```

```
## [1] 2.861235
```

```
    RMSE.lassos.test - RMSE.lassos.train
```

```
## [1] 0.1685796
```