ST635 Final Project: Uber & Lyft Price Analysis

Taoyanran Sun, Cheng Ling, Dana Barclay, Yun-Ting Sun

Introduction

Uber and Lyft's ride prices are not constant like public transport. They are greatly affected by the demand and supply of rides at a given time. So what exactly drives this demand?

Uber and Lyft are both ride sharing companies that have phone applications where customers request a car to a certain location and then are driven to another location by a driver in the driver's own car. Uber and Lyft are safer, cleaner, and more accessible alternatives to taxis and public transportation. Uber and Lyft are direct competitors in the ridesharing industry. Uber and Lyft were both founded in San Francisco, CA in 2009 and 2012, respectively. Uber has 69% of marketshare while Lyft has 29% marketshare. Uber has a lower base fare, and cost per minute while Lyft has a lower minimum ride price and cost per mile, making their pricing, as well as companies in general, extremely similar.

The data we analyzed is called "Uber & Lyft Cab Prices and was found on Kaggle.com. The data was collected from"hot" locations in Boston November 26th to December 18th, 2018. There are 632,403 observations overall and the data includes ride data for Uber and Lyft, which was collected every five minutes, and weather data for Boston, which was collected every 1 hour. There were 18 variables between both data sets.

```
weather <- read.csv("~/Desktop/ST635/UberPrices/dataset/weather.csv")
cab<- read.csv("~/Desktop/ST635/UberPrices/dataset/cab_rides.csv")
```

1.Data Preparation

The first step we took was to merge the weather and cab datasets based on the same time and location into a new data frame. The variables that we ended up using were source, the starting point of the ride; destination, the destination of the ride; distance, between the source and destination; price, the estimated cost for the ride in USD; cab_type, Uber or Lyft service; name, the type of the cab (e.g. UberX, UberXL); temp, the termperature in Fahrenheit; rain, in inches for the previous hour; and wind, measured in MPH.

We noticed that the rain variable had several missing values, which signified that there was no rain, so we changed the missing values to 0's. We also located and removed all other missing values in other columns and made sure there were no duplicate observations. We then decided adding certain columns to our dataset would strengthen our analysis. We created the columns weekday, Mon-Fri; weekend, Sat-Sun; price per mile, price / distance; surge, 1 if no surge, more than 1 if surge; bad weather, more than 0.1 inches of rain or a temperature of less than 32 degrees; service, to rename the values in the "name" variable for clarity; rush hour, 6AM to 10AM and 3PM to 7PM; pickup, coded as nearby or downtown depending on distance from Boston Center; and dropoff, coded as nearby or downtown depending on distance from Boston Center. One anomaly we found was a few unusually large values for price per mile. We believe these values signify cancelled rides that the customer still gets charged for.

```
#Merge two data (base on same time and same location)
sum(is.na(cab$time_stamp))
```

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

```
sum(is.na(weather$time_stamp))
```

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

```
#transfer time_stamp column

full_df=cab

full_df[,"time_stamp"] = as.numeric(unlist((full_df[,"time_stamp"])))/1000
full_df[,"time_stamp"] <- anytime(unlist(full_df[,"time_stamp"]))


full_df$day = str_sub(full_df$time_stamp,1,10)
full_df$hours = str_sub(full_df$time_stamp,12,13)
full_df$merge_time = paste(full_df$day,full_df$hours, full_df$source, sep = "-")
#glimpse(full_df)

w_df <- weather
w_df[,"time_stamp"] = as.numeric(unlist((w_df[,"time_stamp"])))
w_df[,"time_stamp"] <- anytime(unlist(w_df[,"time_stamp"]))


w_df$day = str_sub(w_df$time_stamp,1,10)
w_df$hours = str_sub(w_df$time_stamp,12,13)
w_df$merge_time = paste(w_df$day,w_df$hours, w_df$location, sep = "-")
#glimpse(w_df)
```

```
#clean w_df

sum(is.na(w_df$rain))
```

```
## [1] 5382
```

```
w_df$rain[is.na(w_df$rain)] <- 0

length(unique(w_df$merge_time))
```

```
## [1] 3960
```

```
w_df = w_df[!duplicated(w_df$merge_time),]
```

```
#merge two data set by merge time (inner join)
df1 <- merge(x = full_df, y = w_df, by = "merge_time")


#check missing values
sapply(df1, function(x) sum(is.na(x)))
```

```
##       merge_time         distance         cab_type     time_stamp.x
##                0                0                0                0
##      destination           source            price surge_multiplier
##                0                0            54865                0
##               id       product_id             name            day.x
##                0                0                0                0
##          hours.x             temp         location           clouds
##                0                0                0                0
##         pressure             rain     time_stamp.y         humidity
##                0                0                0                0
##             wind            day.y          hours.y
##                0                0                0
```

```
#add some columns & change data type
df1$weekday <- weekdays(as.Date(df1$day.x))
df1$price_per_mile <- df1$price / df1$distance
df1$surge = ifelse(df1$surge_multiplier==1,0,1)
df1$bad_weather <- ifelse((df1$rain >=0.1)|(df1$temp<=32),1,0)

df1$day.x <- as.Date(df1$day.x)
df1$hours.x <- as.factor(df1$hours.x)
df1$weekday <- as.factor(df1$weekday)
df1$surge <- as.factor(df1$surge)
df1$bad_weather <- as.factor(df1$bad_weather)

df1$Isweekend <- ifelse((df1$weekday == 'Saturday')|(df1$weekday == 'Sunday'),1,0)
df1$rush_hour <- ifelse(df1$hours.x %in% c(06,07,08,09,15,16,17,18),1,0)
df1$rush_hour <- as.factor(df1$rush_hour)

df1$service <- car::recode(df1$name,"c('Black', 'Black SUV')='Uber Premium'; c('UberX
L', 'UberX','UberPool','WAV')='Uber Economy';c('Lux','Lux Black XL', 'Lux Black')='Ly
ft Premium';c('Lyft', 'Lyft XL','Shared')='Lyft Economy'")

df1$Isweekend <- as.factor(df1$Isweekend)
df1$rush_hour <- as.factor(df1$rush_hour)
```

```
df1$pickup <- car::recode(df1$source,"c('Back Bay', 'Boston University', 'Fenway', 'N
ortheastern University')='Nearby'; c('Beacon Hill', 'Financial District','Haymarket S
quare','North End','North Station','South Station','Theatre District','West End')='Do
wntown'")

df1$dropoff <- car::recode(df1$destination,"c('Back Bay', 'Boston University', 'Fenwa
y', 'Northeastern University')='Nearby'; c('Beacon Hill', 'Financial District','Hayma
rket Square','North End','North Station','South Station','Theatre District','West End
')='Downtown'")

df1$pickup <- as.factor(df1$pickup)
df1$dropoff <- as.factor(df1$dropoff)
```

```
keeps <- c("distance", "pickup", 'dropoff', 'price','surge_multiplier','price_per_mil
e','surge','bad_weather','Isweekend','rush_hour','service','destination','source','we
ekday')
df1 <- df1[keeps]

summary(df1)
```

```
##    distance          pickup           dropoff           price
##  Min.   :0.020   Downtown:460044   Downtown:460047   Min.   : 2.50
##  1st Qu.:1.280   Nearby  :230063   Nearby  :230060   1st Qu.: 9.00
##  Median :2.160                                       Median :13.50
##  Mean   :2.189                                       Mean   :16.54
##  3rd Qu.:2.920                                       3rd Qu.:22.50
##  Max.   :7.860                                       Max.   :97.50
##                                                      NA's   :54865
##  surge_multiplier price_per_mile    surge      bad_weather Isweekend
##  Min.   :1.000    Min.   :   0.56   0:669223   0:578893    0:505444
##  1st Qu.:1.000    1st Qu.:   4.66   1: 20884   1:111214    1:184663
##  Median :1.000    Median :   7.50
##  Mean   :1.014    Mean   :   9.69
##  3rd Qu.:1.000    3rd Qu.:  11.54
##  Max.   :3.000    Max.   :1375.00
##                   NA's   :54865
##  rush_hour          service                      destination
##  0:574226   Lyft Economy:153028   Financial District: 58600
##  1:115881   Lyft Premium:153074   Haymarket Square  : 57574
##             Taxi        : 54865   Fenway            : 57535
##             Uber Economy:219439   Boston University : 57520
##             Uber Premium:109701   North End         : 57518
##                                   Back Bay          : 57507
##                                   (Other)           :343853
##                 source            weekday
##  Financial District: 58589   Friday   : 90324
##  Back Bay          : 57567   Monday   :137631
##  Theatre District  : 57564   Saturday : 89856
##  Haymarket Square  : 57518   Sunday   : 94807
##  Boston University : 57512   Thursday : 89352
##  North End         : 57507   Tuesday  : 90473
##  (Other)           :343850   Wednesday: 97664
```

To begin our analysis, we split our data set into two based on whether the cab type was Uber or Lyft. We then created a training set with half the observations from the full data set, and created a test set using the other half of the observations.

```
# Split dataset
# uber
uber <- df1[df1$service %in% c('Uber Economy','Uber Premium'),]

set.seed(123)
train_num_u <- sample(1:nrow(uber), size= as.integer(nrow(uber)*0.5))
train_u <- uber[train_num_u,]
test_u <- uber[-train_num_u,]

# lyft
lyft <- df1[df1$service %in% c('Lyft Economy','Lyft Premium'),]

set.seed(123)
train_num_l <- sample(1:nrow(lyft), size= as.integer(nrow(lyft)*0.5))
train_l <- lyft[train_num_l,]
test_l <- lyft[-train_num_l,]
```

2.Price Per Mile Analysis

Linear Regression

We decided to begin with a linear regression model to find which predictors most accurately predict price per mile for Uber and then for Lyft. For each cab type, we started with a basic multiple linear regression model. This model had a poor R^2, so we continued to try to fit a better model. Next, we found that the best transformation for the distance variable is to raise it to the 0.4, so we applied that to the previous model and still had a low R^2, so we found that the best transformation for the response variable was to take it's log. Our best linear model for predicting price per mile in Ubers used the transformed distance variable and service as interaction terms, bad weather, weekend, rush hour, pick up, and drop off to predict price per mile. Using the Anova test, we see that a change in the variables distance, service, pick up and drop off will affect the price per mile. The R^2 was 0.8054, which is high, and had almost all useful predictors. Additionally, the p-value was small, so this model is a good fit for the data. Additionally, the MSE was 0.25, so the model explains 26% of variance in testing data set. We then checked that the model assumptions held, which is explained below.

```
#uber

m4 <- lm(price_per_mile ~ distance * service + bad_weather + Isweekend + rush_hour +
pickup + dropoff , data = train_u)
summary(m4)
```

```
##
## Call:
## lm(formula = price_per_mile ~ distance * service + bad_weather +
##     Isweekend + rush_hour + pickup + dropoff, data = train_u)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -10.91   -2.61   -1.20    1.11 1344.24
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  12.067796   0.111238 108.486  < 2e-16 ***
## distance                     -2.347918   0.044501 -52.761  < 2e-16 ***
## serviceUber Premium          18.823369   0.179566 104.827  < 2e-16 ***
## bad_weather1                  0.009401   0.111981   0.084    0.933
## Isweekend1                    0.006787   0.092471   0.073    0.941
## rush_hour1                   -0.016741   0.107711  -0.155    0.876
## pickupNearby                 -0.419238   0.091730  -4.570 4.87e-06 ***
## dropoffNearby                -0.593326   0.090142  -6.582 4.65e-11 ***
## distance:serviceUber Premium -4.324342   0.072005 -60.056  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.28 on 164561 degrees of freedom
## Multiple R-squared:  0.1489, Adjusted R-squared:  0.1488
## F-statistic:  3598 on 8 and 164561 DF,  p-value: < 2.2e-16
```

```
#transform predictors(0.4)
summary(powerTransform(distance~1, data = train_u, family = 'bcPower'))
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.4003         0.4       0.3936        0.407
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                              LRT df       pval
## LR test, lambda = (0) 15493.18  1 < 2.22e-16
##
## Likelihood ratio test that no transformation is needed
##                              LRT df       pval
## LR test, lambda = (1) 28010.37  1 < 2.22e-16
```

```r
m5 <- lm(price_per_mile ~ I(distance^0.4) * service + bad_weather + Isweekend + rush_
hour + pickup + dropoff , data = train_u)
summary(m5)
```

```
##
## Call:
## lm(formula = price_per_mile ~ I(distance^0.4) * service + bad_weather +
##     Isweekend + rush_hour + pickup + dropoff, data = train_u)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -14.50   -2.45   -0.70    1.50 1319.60
##
## Coefficients:
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             23.574375   0.221900 106.239   <2e-16 ***
## I(distance^0.4)                        -13.454106   0.172517 -77.987   <2e-16 ***
## serviceUber Premium                     39.393914   0.372102 105.869   <2e-16 ***
## bad_weather1                             0.025920   0.107915   0.240    0.810
## Isweekend1                               0.002604   0.089113   0.029    0.977
## rush_hour1                               0.003486   0.103800   0.034    0.973
## pickupNearby                             1.299612   0.089726  14.484   <2e-16 ***
## dropoffNearby                            0.973013   0.088074  11.048   <2e-16 ***
## I(distance^0.4):serviceUber Premium -22.753806   0.274811 -82.798   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.69 on 164561 degrees of freedom
## Multiple R-squared:  0.2096, Adjusted R-squared:  0.2095
## F-statistic:  5453 on 8 and 164561 DF,  p-value: < 2.2e-16
```

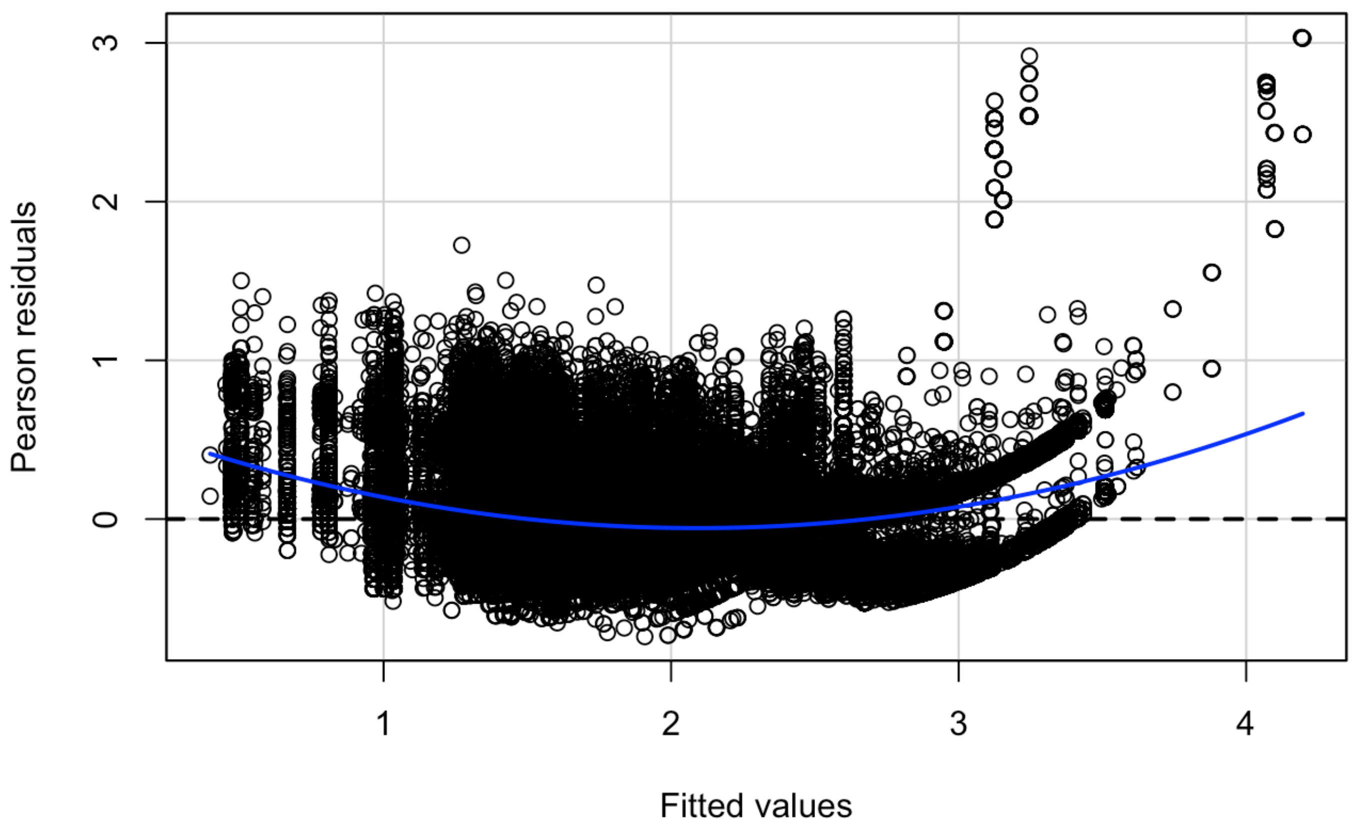
```r
# transform response(use log for better explaination)
summary(powerTransform(price_per_mile ~ I(distance^0.4) * service + bad_weather + Isw
eekend + rush_hour+ pickup + dropoff , data = train_u,family = 'bcPower'))
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1   -0.2755       -0.28      -0.2801       -0.271
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                             LRT df       pval
## LR test, lambda = (0) 16970.85  1 < 2.22e-16
##
## Likelihood ratio test that no transformation is needed
##                             LRT df       pval
## LR test, lambda = (1) 674841.8  1 < 2.22e-16
```

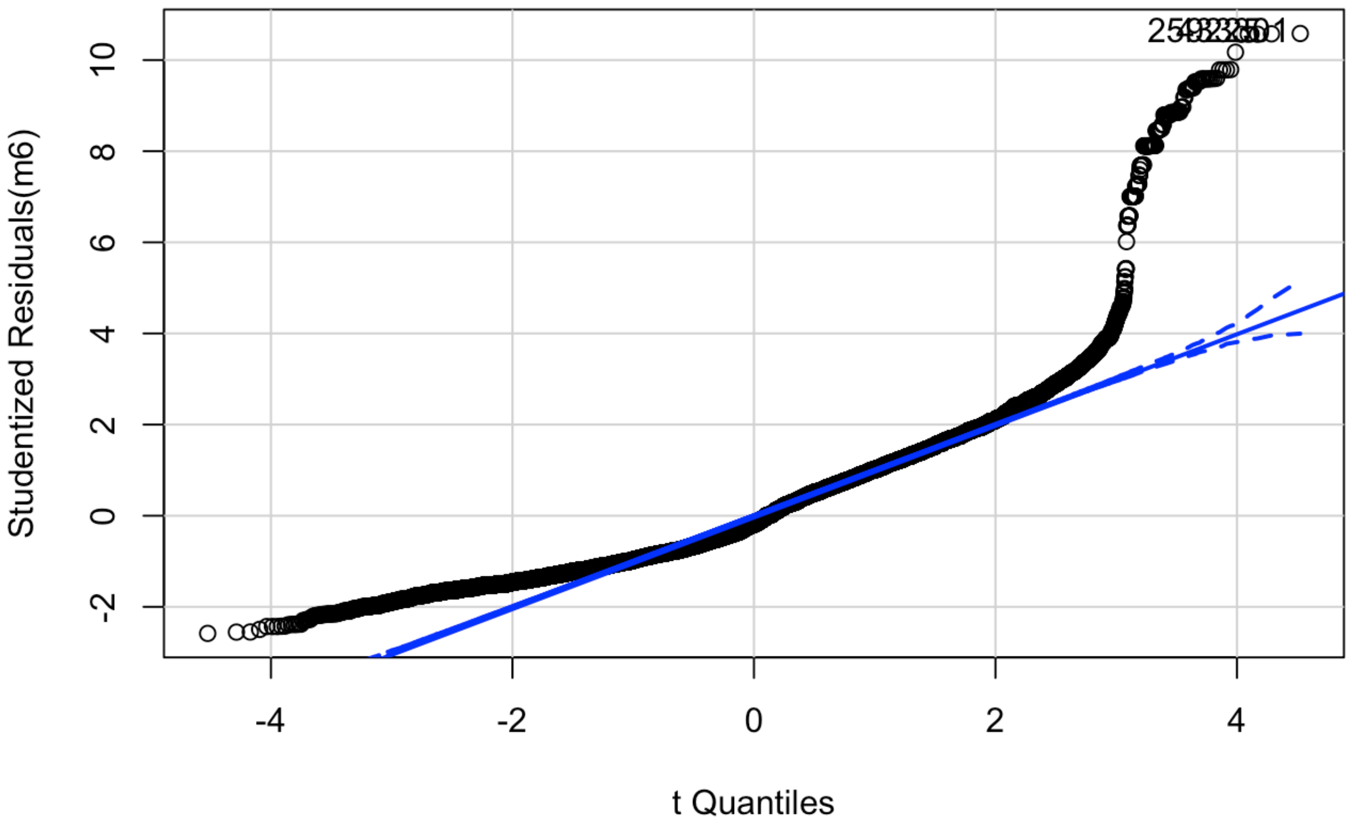
```
m6 <- lm(log(price_per_mile) ~ I(distance^0.4) * service + bad_weather + Isweekend +
rush_hour+ pickup + dropoff , data = train_u)
summary(m6)
```

```
##
## Call:
## lm(formula = log(price_per_mile) ~ I(distance^0.4) * service +
##     bad_weather + Isweekend + rush_hour + pickup + dropoff, data = train_u)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.74082 -0.22443 -0.05162  0.19394  3.03420
##
## Coefficients:
##                                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                      3.530e+00  4.058e-03  869.875   <2e-16 ***
## I(distance^0.4)                 -1.355e+00  3.155e-03 -429.393   <2e-16 ***
## serviceUber Premium              9.680e-01  6.804e-03  142.270   <2e-16 ***
## bad_weather1                    -2.709e-03  1.973e-03   -1.373    0.170
## Isweekend1                       3.766e-05  1.629e-03    0.023    0.982
## rush_hour1                      -1.987e-03  1.898e-03   -1.047    0.295
## pickupNearby                    -4.261e-02  1.641e-03  -25.974   <2e-16 ***
## dropoffNearby                   -2.886e-02  1.610e-03  -17.919   <2e-16 ***
## I(distance^0.4):serviceUber Premium -8.419e-02  5.025e-03  -16.754   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2869 on 164561 degrees of freedom
## Multiple R-squared:  0.8059, Adjusted R-squared:  0.8059
## F-statistic: 8.539e+04 on 8 and 164561 DF,  p-value: < 2.2e-16
```

```
#check assumptions
residualPlot(m6)
```

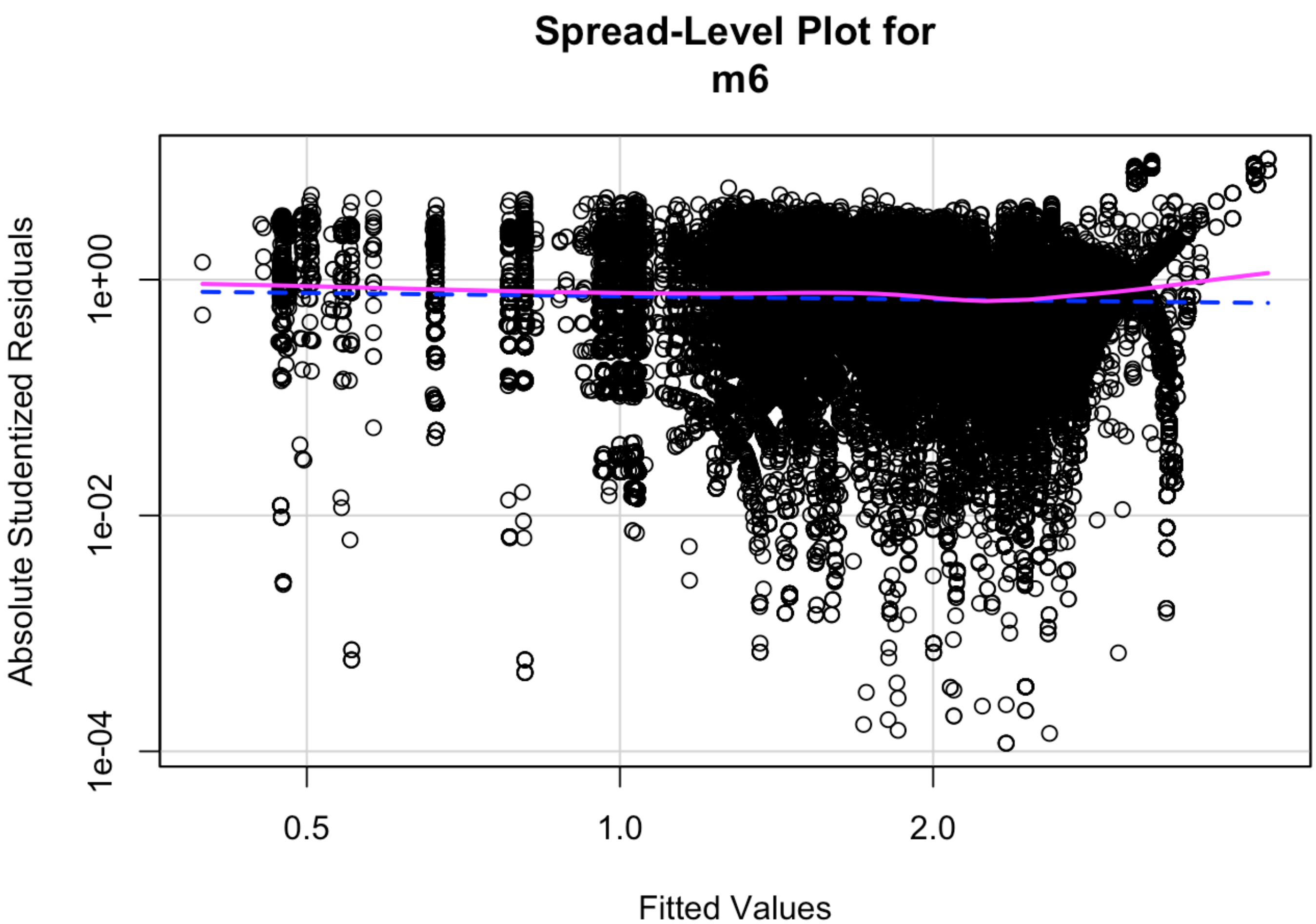



```
qqPlot(m6)
```




```
## 433301 259225
##  80565 138791
```

```
spreadLevelPlot(m6)
```




```
##
## Suggested power transformation:  1.091977
```

```
#compute test rate
pred_u <- predict(m6,newdata = test_u,se.fit = TRUE)
rss_u <- sum((test_u$price_per_mile - exp(pred_u$fit))^2)
(rsquare_u <- 1- rss_u/sum((test_u$price_per_mile - mean(test_u$price_per_mile))^2))
```

```
## [1] 0.2495121
```

First, the residuals-fitted plot shows a line, that is not terribly curved, around 0, which means we have no violation in linear relationship assumptions. The qq-plot shows a skew on the top right and means the residuals may not be normally distributed, which might caused by those presumed cancelled orders we mentioned previously. For spread level plot, It is fine that the plot shows a flat line around 0.

We then moved on to fitting a linear regression model to predict price per mile for Lyft rides. We found that surge data was only available for Lyft, so we included that as a regressor for our linear model. After running our first model, we found that the R^2 was around 0.5. While this is not bad, we thought we could do better, so again we found the best transofrmation for the predictors. We again found that distance should be transformed to distance^0.4, and fit another model to include that. The R^2 in this model increased only marginally, so we found the best transformation for the response variable, which ended up being a log transformation again. This increased the R^2 considerably to 0.7209. This model uses distance, service, bad weather, weekend, rush hour, pick up, drop off, and surge multiplier to predict the price per mile of a Lyft. Using the Anova test, we see that a change in the variables distance, service, pick up, drop off, and surge multiplier will affect the price per mile. Since the p-value for the model summary is so small, this model is a good fit for the training data. The test rate for this model is 0.66, so the model explains 66% of the variance in in the testing data set.

```
#lyft

m7 <- lm(price_per_mile ~ distance + service + bad_weather + Isweekend + rush_hour +
pickup + dropoff +surge_multiplier , data = train_l)
summary(m7)
```

```
##
## Call:
## lm(formula = price_per_mile ~ distance + service + bad_weather +
##     Isweekend + rush_hour + pickup + dropoff + surge_multiplier,
##     data = train_l)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.059  -3.094  -0.530   1.818  85.633
##
## Coefficients:
##                     Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          3.56772    0.10487   34.019   <2e-16 ***
## distance            -3.00560    0.01370 -219.345   <2e-16 ***
## serviceLyft Premium  7.92121    0.02657  298.106   <2e-16 ***
## bad_weather1         0.00494    0.03699    0.134   0.8938
## Isweekend1           0.03091    0.03058    1.011   0.3122
## rush_hour1          -0.06695    0.03565   -1.878   0.0604 .
## pickupNearby        -1.29644    0.03010  -43.074   <2e-16 ***
## dropoffNearby       -0.96410    0.03083  -31.268   <2e-16 ***
## surge_multiplier     9.19537    0.09754   94.270   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.193 on 153042 degrees of freedom
## Multiple R-squared:  0.5301, Adjusted R-squared:  0.5301
## F-statistic: 2.158e+04 on 8 and 153042 DF,  p-value: < 2.2e-16
```

```
#transform predictors(0.4)
summary(powerTransform(distance~1, data = train_l, family = 'bcPower'))
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.4394        0.44       0.4307        0.448
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                            LRT df       pval
## LR test, lambda = (0) 10287.44  1 < 2.22e-16
##
## Likelihood ratio test that no transformation is needed
##                            LRT df       pval
## LR test, lambda = (1) 15405.83  1 < 2.22e-16
```

```
m8 <- lm(price_per_mile ~ I(distance^0.4) + service + bad_weather + Isweekend + rush_
hour + pickup + dropoff +surge_multiplier , data = train_l)
summary(m8)
```

```
##
## Call:
## lm(formula = price_per_mile ~ I(distance^0.4) + service + bad_weather +
##     Isweekend + rush_hour + pickup + dropoff + surge_multiplier,
##     data = train_l)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.693  -3.074  -0.377   2.014  83.118
##
## Coefficients:
##                       Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          14.567232   0.113667  128.157   <2e-16 ***
## I(distance^0.4)     -13.609147   0.050893 -267.405   <2e-16 ***
## serviceLyft Premium   7.924410   0.025150  315.091   <2e-16 ***
## bad_weather1          0.004565   0.035010    0.130    0.896
## Isweekend1            0.027894   0.028947    0.964    0.335
## rush_hour1           -0.052642   0.033738   -1.560    0.119
## pickupNearby         -0.693387   0.028837  -24.045   <2e-16 ***
## dropoffNearby        -0.400106   0.029422  -13.599   <2e-16 ***
## surge_multiplier      9.248361   0.092323  100.174   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.915 on 153042 degrees of freedom
## Multiple R-squared:  0.579,  Adjusted R-squared:  0.579
## F-statistic: 2.631e+04 on 8 and 153042 DF,  p-value: < 2.2e-16
```

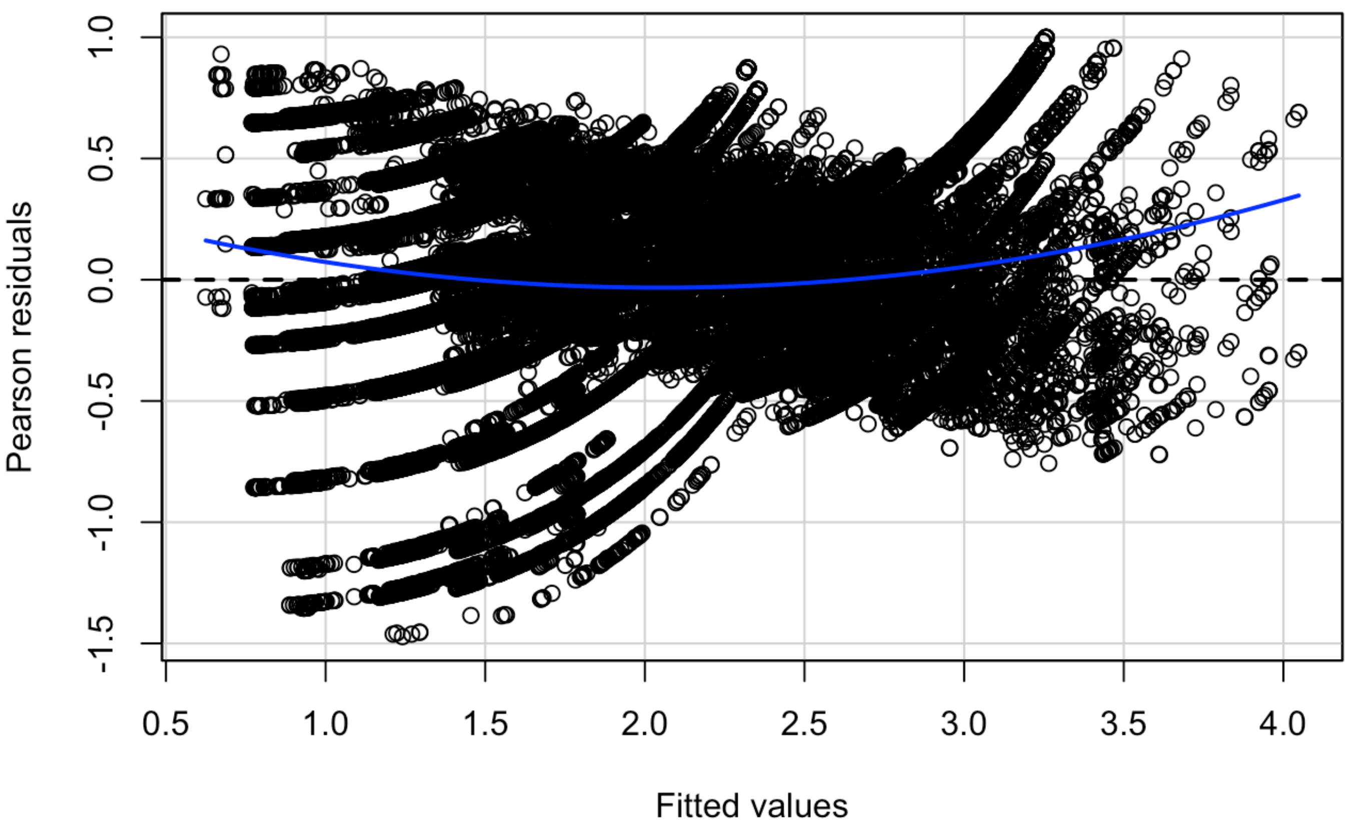
```
# transform response(log)
summary(powerTransform(price_per_mile ~ I(distance^0.4) + service + bad_weather + Isw
eekend + rush_hour + pickup + dropoff+surge_multiplier, data = train_l,family = 'bcPo
wer'))
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.0929        0.09       0.0889       0.0969
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                             LRT df       pval
## LR test, lambda = (0) 2021.444  1 < 2.22e-16
##
## Likelihood ratio test that no transformation is needed
##                             LRT df       pval
## LR test, lambda = (1) 175462.7  1 < 2.22e-16
```

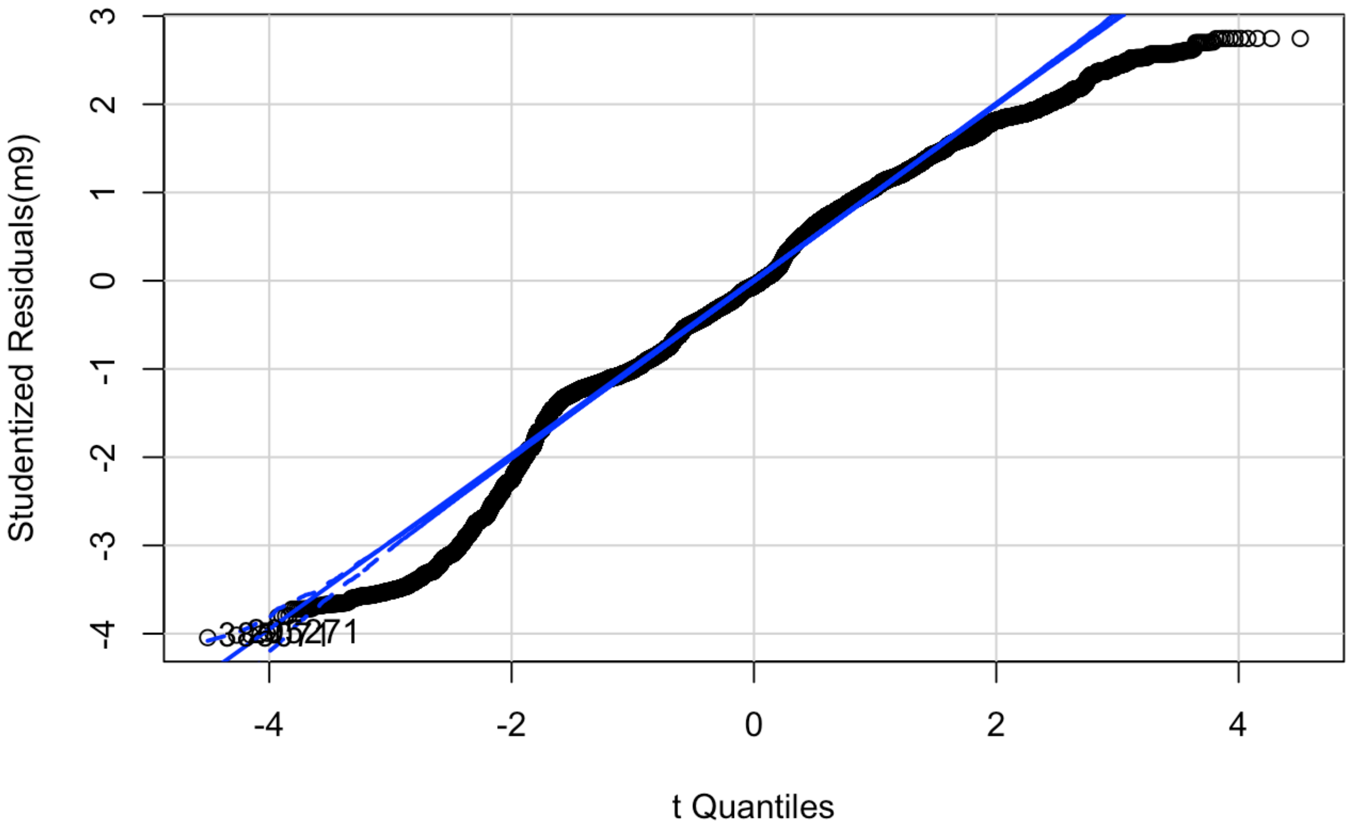
```
m9 <- lm(log(price_per_mile) ~ I(distance^0.4) + service + bad_weather + Isweekend +
rush_hour + pickup + dropoff+surge_multiplier , data = train_l)
summary(m9)
```

```
##
## Call:
## lm(formula = log(price_per_mile) ~ I(distance^0.4) + service +
##     bad_weather + Isweekend + rush_hour + pickup + dropoff +
##     surge_multiplier, data = train_l)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47190 -0.25624 -0.02085  0.28655  0.99921
##
## Coefficients:
##                        Estimate Std. Error  t value Pr(>|t|)
## (Intercept)           2.3369465  0.0084171  277.642   <2e-16 ***
## I(distance^0.4)      -1.2075774  0.0037687 -320.423   <2e-16 ***
## serviceLyft Premium  0.9010964  0.0018623  483.850   <2e-16 ***
## bad_weather1          0.0001785  0.0025926    0.069    0.945
## Isweekend1            0.0008278  0.0021435    0.386    0.699
## rush_hour1           -0.0020629  0.0024983   -0.826    0.409
## pickupNearby         -0.0478533  0.0021354  -22.409   <2e-16 ***
## dropoffNearby        -0.0332124  0.0021788  -15.244   <2e-16 ***
## surge_multiplier      0.8471319  0.0068366  123.912   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3639 on 153042 degrees of freedom
## Multiple R-squared:  0.7203, Adjusted R-squared:  0.7203
## F-statistic: 4.926e+04 on 8 and 153042 DF,  p-value: < 2.2e-16
```

```
#check assumptions
residualPlot(m9)
```

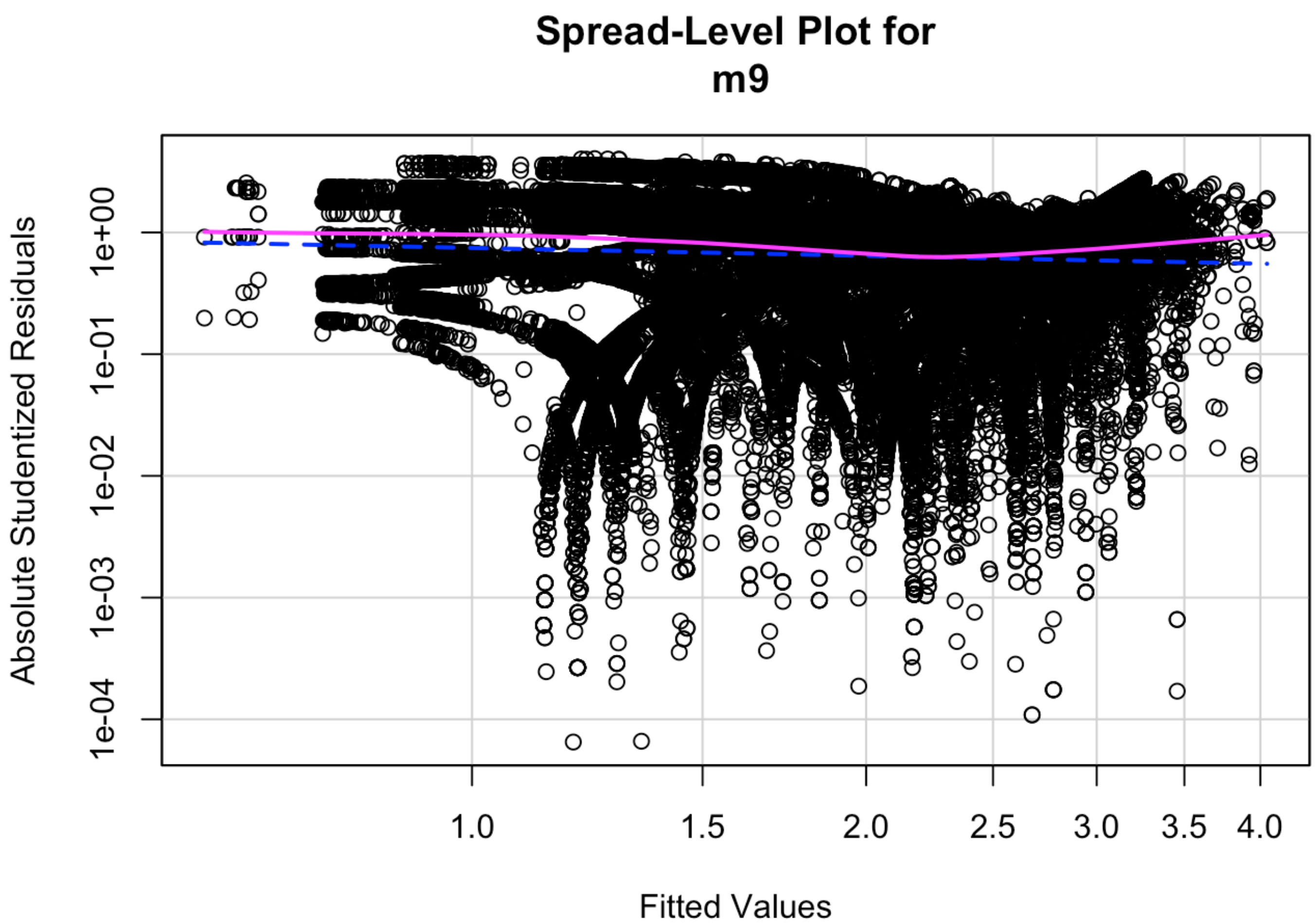



```
qqPlot(m9)
```




```
## 385071 295271
##  34627 118540
```

```
spreadLevelPlot(m9)
```




```
##
## Suggested power transformation:  1.214356
```

```
#test rate 0.66
pred_l <- predict(m9,newdata = test_l,se.fit = TRUE)
rss_l <- sum((test_l$price_per_mile - exp(pred_l$fit))^2)
(rsquare_l <- 1- rss_l/sum((test_l$price_per_mile - mean(test_l$price_per_mile))^2))
```

```
## [1] 0.6661905
```

The residuals-fitted plot shows a flat line around 0, which means we have no violation in linear relationship assumptions. The qq-plot stays close to the line, meaning the residuals are likely normally distributed. For spread level plot, It is fine that the plot shows a flat line around 0.

Regression Tree

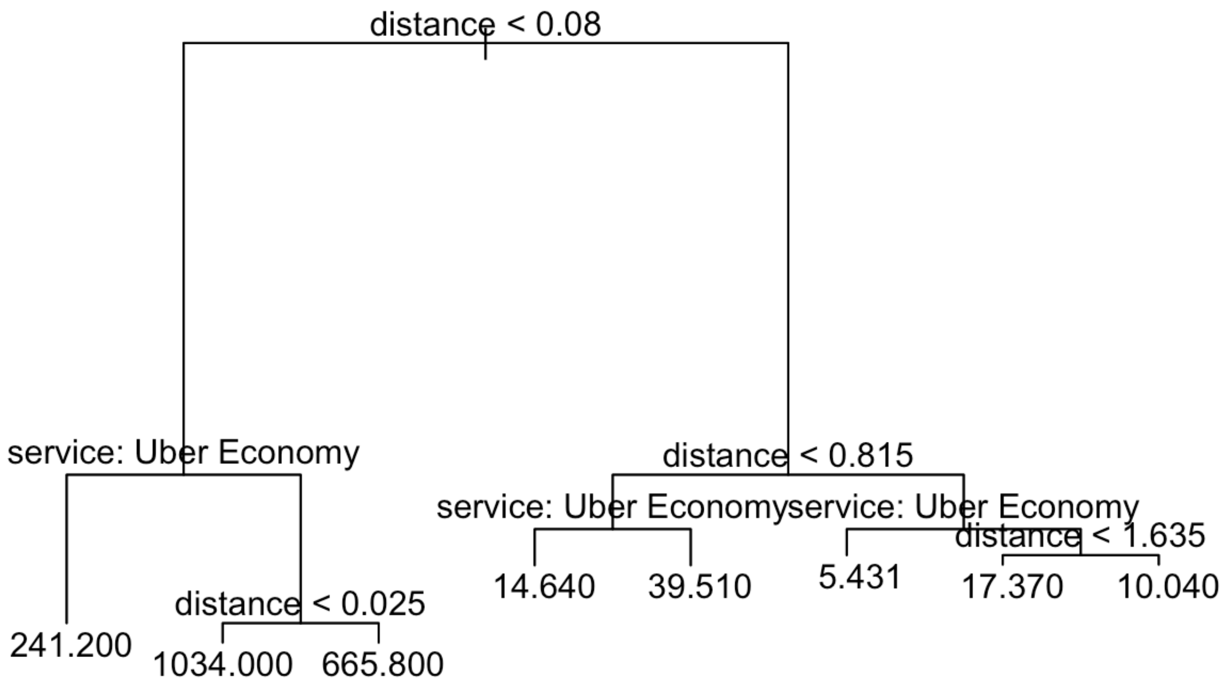
We then fit a regression tree for the best linear models for both Uber and Lyft.

The original regression tree for Uber had 8 terminal nodes. We then pruned the tree to ensure there was no overfitting. We used 5 fold cross validation and found that the best size for the tree was still 8 terminal nodes. The most important factor in predicting price per mile is distance. If the distance is less than 0.08 miles, the price per mile of an Uber is always greater than if the distance is greater than 0.08 miles. The service being Uber Economy also affects the price per mile for trips of all distances. The training set for MSE for the Uber regression tree is 35.46 and the training set MSE for the Uber linear model is 198.26. Based on this, we see that the regression tree does a better job predicting price per mile, since the MSE value is much lower.

```
# uber
# fit a tree
tree_u <- tree(price_per_mile ~ distance + service +pickup + dropoff+ bad_weather + I
sweekend + rush_hour , data = train_u)
summary(tree_u)
```

```
##
## Regression tree:
## tree(formula = price_per_mile ~ distance + service + pickup +
##     dropoff + bad_weather + Isweekend + rush_hour, data = train_u)
## Variables actually used in tree construction:
## [1] "distance" "service"
## Number of terminal nodes:  8
## Residual mean deviance:  37.33 = 6142000 / 164600
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -290.8000   -1.9830   -0.4315    0.0000    1.4800  340.9000
```

```
plot(tree_u)
text(tree_u,pretty=0)
```




```
## 5 fold cross validation
set.seed(567)
(cv.u <- cv.tree(tree_u, FUN = prune.tree, K = 5))
```

```
## $size
## [1] 8 7 6 5 4 3 2 1
##
## $dev
## [1]  6516271  7662647  8208130  9755004 12008492 15383975 24712395 51241336
##
## $k
## [1]       -Inf   585468.7  1204342.3  1617995.9  2251736.9  3378482.8  9224282.5
## [8] 26836203.9
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
## best size equals to original size
(bestsize <- cv.u$size[which.min(cv.u$dev)])
```

```
## [1] 8
```

```
# MSE from uber regression tree
pred1 <- predict(tree_u,test_u)
head(pred1)
```

```
##       10       14       16       18       19       50
## 10.03841  5.43145  5.43145  5.43145  5.43145 39.50913
```

```
MSE1 <- mean((test_u$price_per_mile-pred1)^2)
MSE1
```

```
## [1] 36.59903
```

```
#MSE from uber linear regression
(MSE_u <- mean((test_u$price_per_mile - exp(pred_u$fit))^2))
```

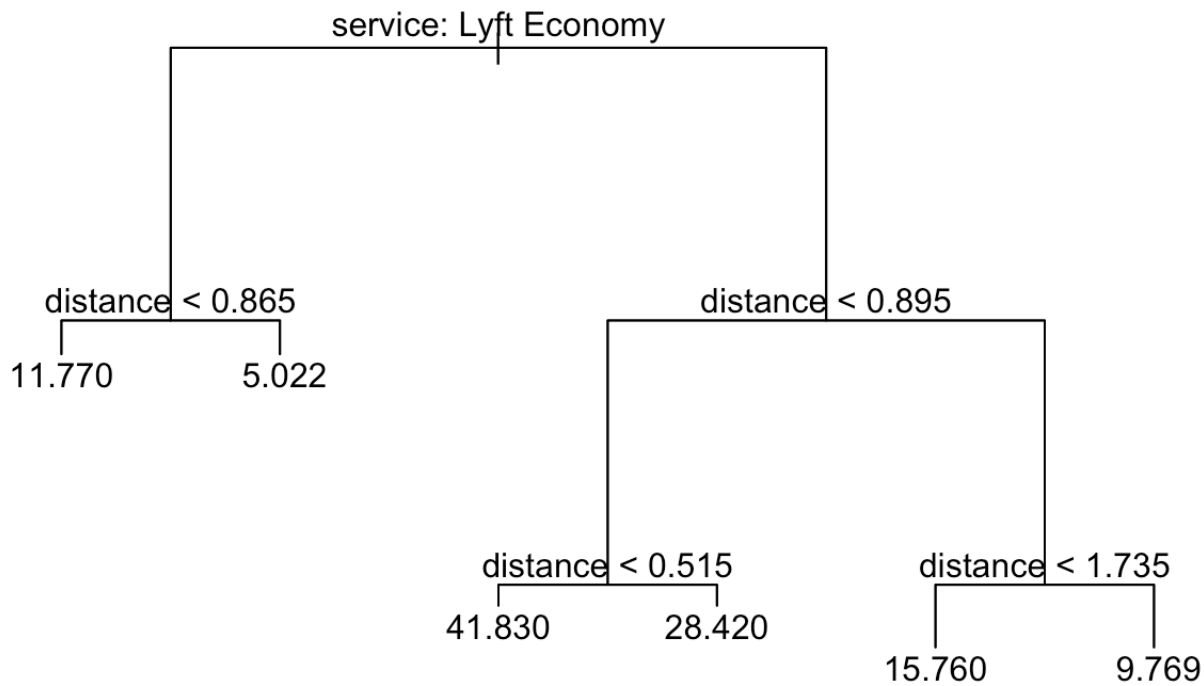
```
## [1] 231.3615
```

The original regression tree for Lyft had 6 terminal nodes. After we pruned the tree using 5 fold cross validation, we found that the best size for the tree was still 6 terminal nodes. The most important factor in predicting price per mile is the service being Lyft Economy, while distance also plays a role in predicting price per mile. If the service is Lyft Economy and the distance is less than 0.505 miles, the price per mile will be the largest. If the service is Lyft Economy and the distance is between 0.895 and 1.735 miles, the price per mile will be the lowest. The training set MSE for the Lyft regression tree is 18.66, while the training set MSE for the Lyft linear model is 19.07. Again, the regression tree for Lyft has a smaller MSE than that of the Lyft linear regression, so the regression tree does a better job of predicting price per mile.

```
# lyft
# fit a tree
tree_l <- tree(price_per_mile ~ distance + service + dropoff + pickup + bad_weather +
Isweekend + rush_hour , data = train_l)
summary(tree_l)
```

```
##
## Regression tree:
## tree(formula = price_per_mile ~ distance + service + dropoff +
##     pickup + bad_weather + Isweekend + rush_hour, data = train_l)
## Variables actually used in tree construction:
## [1] "service"  "distance"
## Number of terminal nodes:  6
## Residual mean deviance:  18.73 = 2867000 / 153000
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -21.2400  -2.3670  -0.5478   0.0000   1.7680  72.3000
```

```
plot(tree_l)
text(tree_l,pretty = 0)
```




```
## 5 fold cross validation
set.seed(567)
(cv.l <- cv.tree(tree_l, FUN = prune.tree, K = 5))
```

```
## $size
## [1] 6 5 4 3 2 1
##
## $dev
## [1] 2874302 3058724 3362735 3928103 8781336 8781336
##
## $k
## [1]      -Inf  188096.2  304643.0  565153.2 2390111.9 2465885.7
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
## best size equals to original size
(bestsize <- cv.l$size[which.min(cv.l$dev)])
```

```
## [1] 6
```

```
# MSE from lyft regression tree
pred2 <- predict(tree_l,test_l)
head(pred2)
```

```
##        2        4        6        7        9       17
## 9.769159 5.021727 5.021727 5.021727 5.021727 9.769159
```

```
MSE2 <- mean((test_l$price_per_mile-pred2)^2)
MSE2
```

```
## [1] 18.58343
```

```
#MSE from lyft linear regression
(MSE_l <- mean((test_l$price_per_mile - exp(pred_l$fit))^2))
```

```
## [1] 19.15212
```

3.Price difference: Uber VS Lyft

Next, we wanted to analyze whether there is a price difference between Uber and Lyft when given the same exact condtitions.

The two comparisons we did were: 1. Uber Eco and Lyft Eco 2. Uber Premium and Lyft Premium

We merged the Uber and Lyft data sets based on the same distance, if there is bad weather, if it’s a weekend, if it’s rush hour, and if it has the same pick up and drop off locations and then split the new merged data set in half, with one half being a training set and the other a test set. Our first analysis was with UberEconomy vs LyftEconomy.

3-1 Uber Economy VS Lyft Economy

```
## Uber Economy VS Lyft Economy
# merge dataset based on same situation:
# distance/bad_weather/isweekend/rushhour/pickip/dropoff
head(uber)
```

```
##    distance   pickup  dropoff price surge_multiplier price_per_mile surge
## 8      6.26   Nearby Downtown  34.0                1       5.431310     0
## 10     2.19 Downtown Downtown  17.5                1       7.990868     0
## 13     1.35 Downtown Downtown  14.0                1      10.370370     0
## 14     2.19 Downtown Downtown   8.0                1       3.652968     0
## 16     2.19 Downtown Downtown  13.0                1       5.936073     0
## 18     3.39   Nearby Downtown  18.0                1       5.309735     0
##    bad_weather Isweekend rush_hour      service   destination            source
## 8            0         1         0 Uber Premium South Station          Back Bay
## 10           0         1         0 Uber Premium     North End       Beacon Hill
## 13           0         1         0 Uber Economy     North End       Beacon Hill
## 14           0         1         0 Uber Economy     North End       Beacon Hill
## 16           0         1         0 Uber Economy     North End       Beacon Hill
## 18           0         1         0 Uber Economy North Station Boston University
##    weekday
## 8   Sunday
## 10  Sunday
## 13  Sunday
## 14  Sunday
## 16  Sunday
## 18  Sunday
```

```
samesub_e <- merge.data.frame(uber[uber$service == "Uber Economy",],lyft[lyft$service
=="Lyft Economy",],by = intersect(c('distance','bad_weather','Isweekend','rush_hour',
'destination','source',"weekday","pickup","dropoff"),c('distance','bad_weather','Iswe
ekend','rush_hour','destination','source',"weekday","pickup","dropoff")))
# calculate uber,lyft price difference
samesub_e$pricediff <- samesub_e$price.x - samesub_e$price.y
```

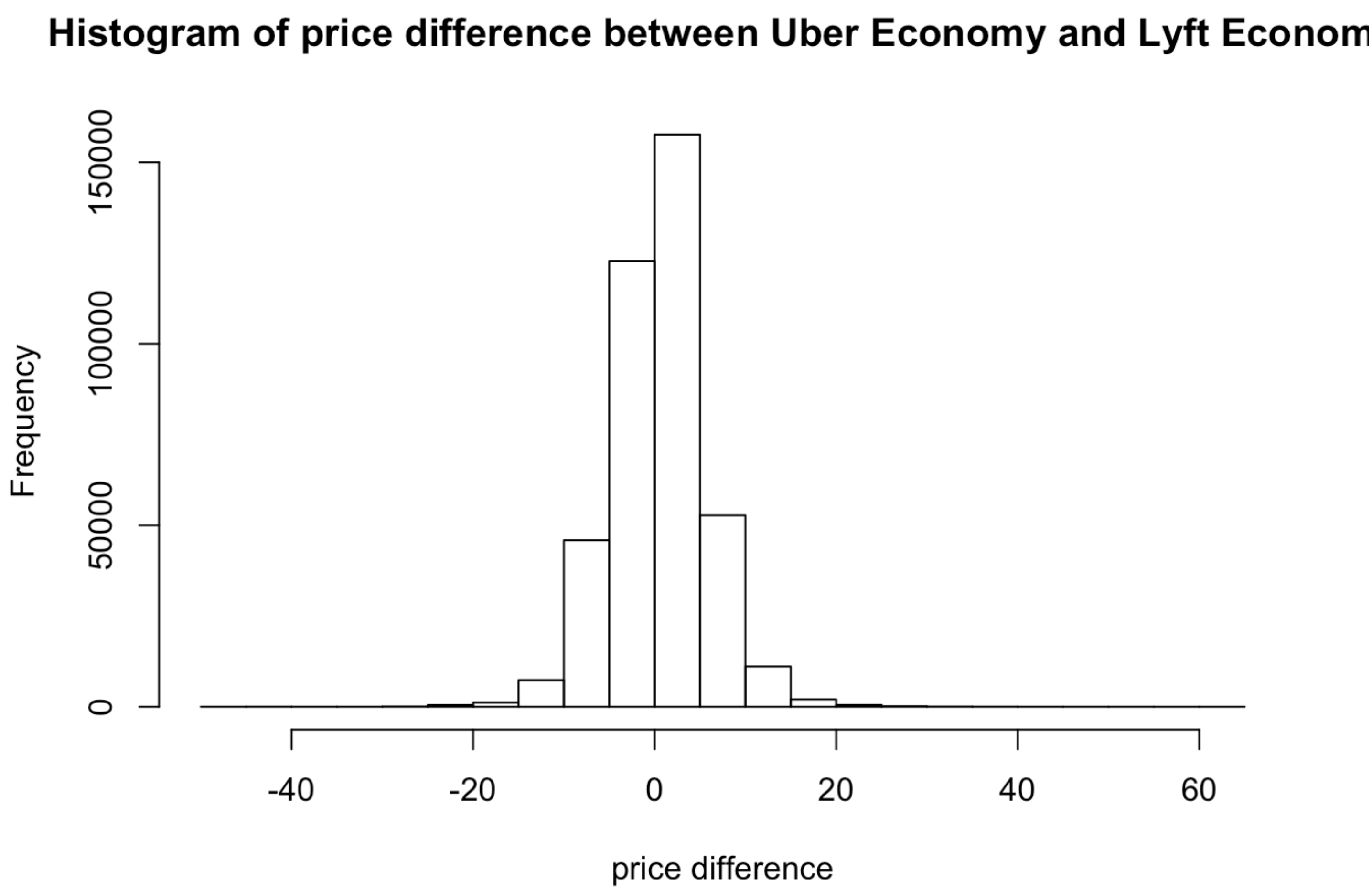
```
# split dataset
set.seed(123)
index <- sample(1:nrow(samesub_e),size= as.integer(nrow(samesub_e)*0.5))
samesub_e.train <- samesub_e[index,]
samesub_e.test <- samesub_e[-index,]
```

We fit a linear regression model to predict the price difference and used distance, drop off location, pick up location, if there's bad weather, if it's a weekend, if it's rush hour, and surge multiplier. Distance, drop off, if there's bad weather and surge_multiplier all contribute to a price difference between Uber Economy and Lyft

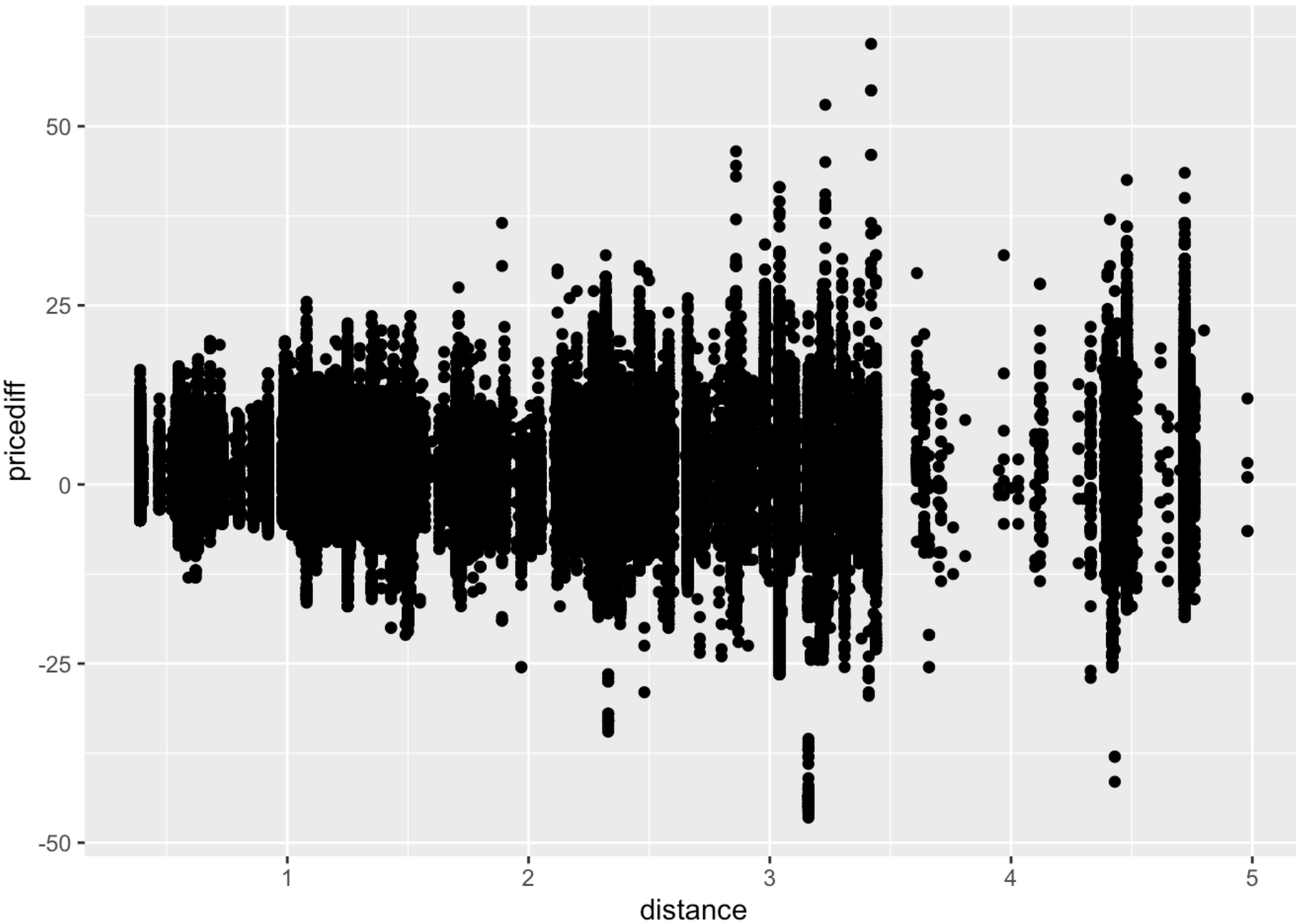
Economy, proven by their small p-values. The residual plot, normality plot, and variance plot for this model all pass the model assumptions, so we can assume this model is a good fit to the data.

First, let’s take a look at our price diff distribution

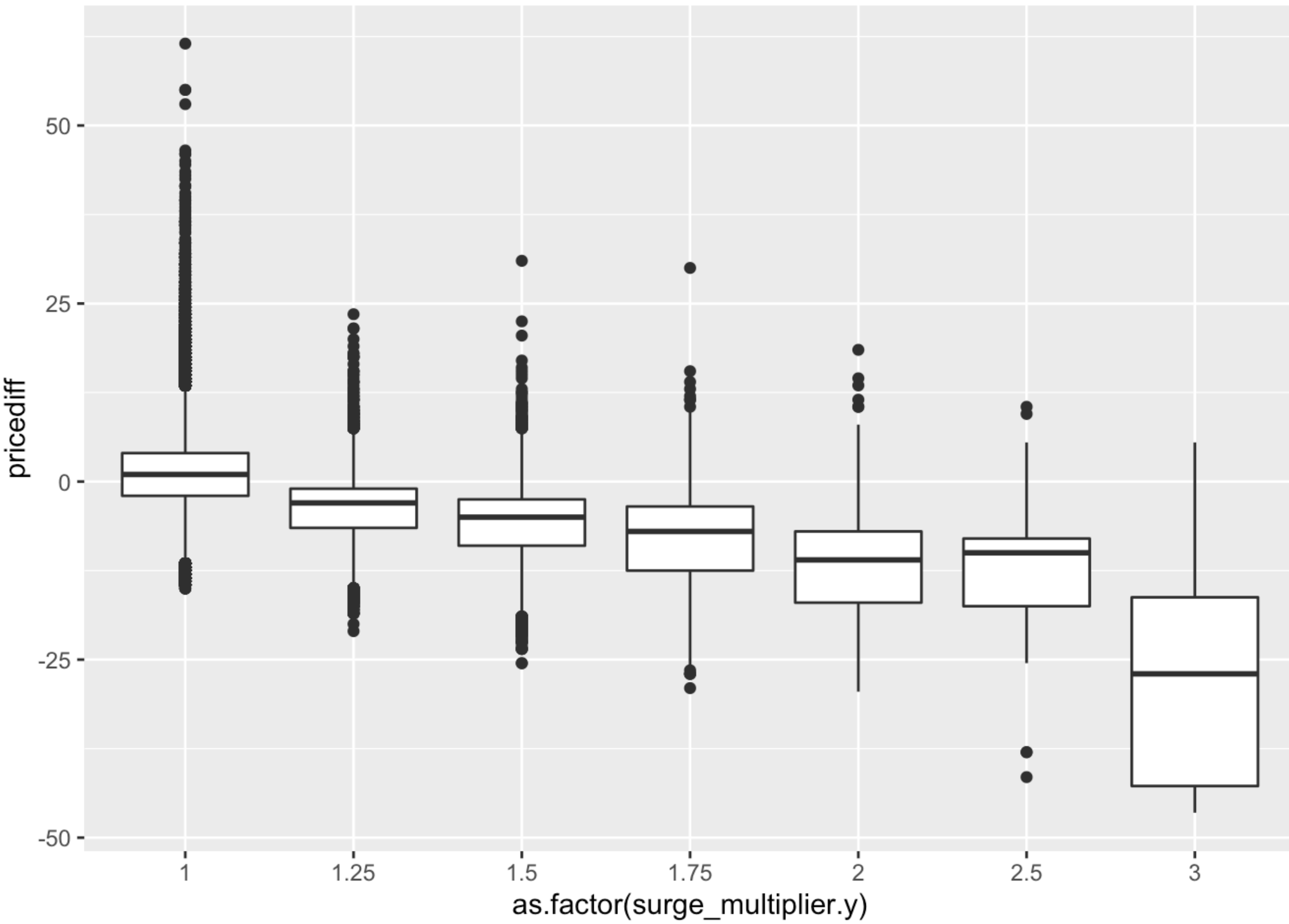
```
hist(samesub_e.train$pricediff,main = "Histogram of price difference between Uber Eco
nomy and Lyft Economy",xlab = "price difference")
```




```
# Overall, Uber Economy's price tend to be higher than lyft
ggplot(samesub_e.train,aes(x=distance,y=pricediff))+geom_point()
```



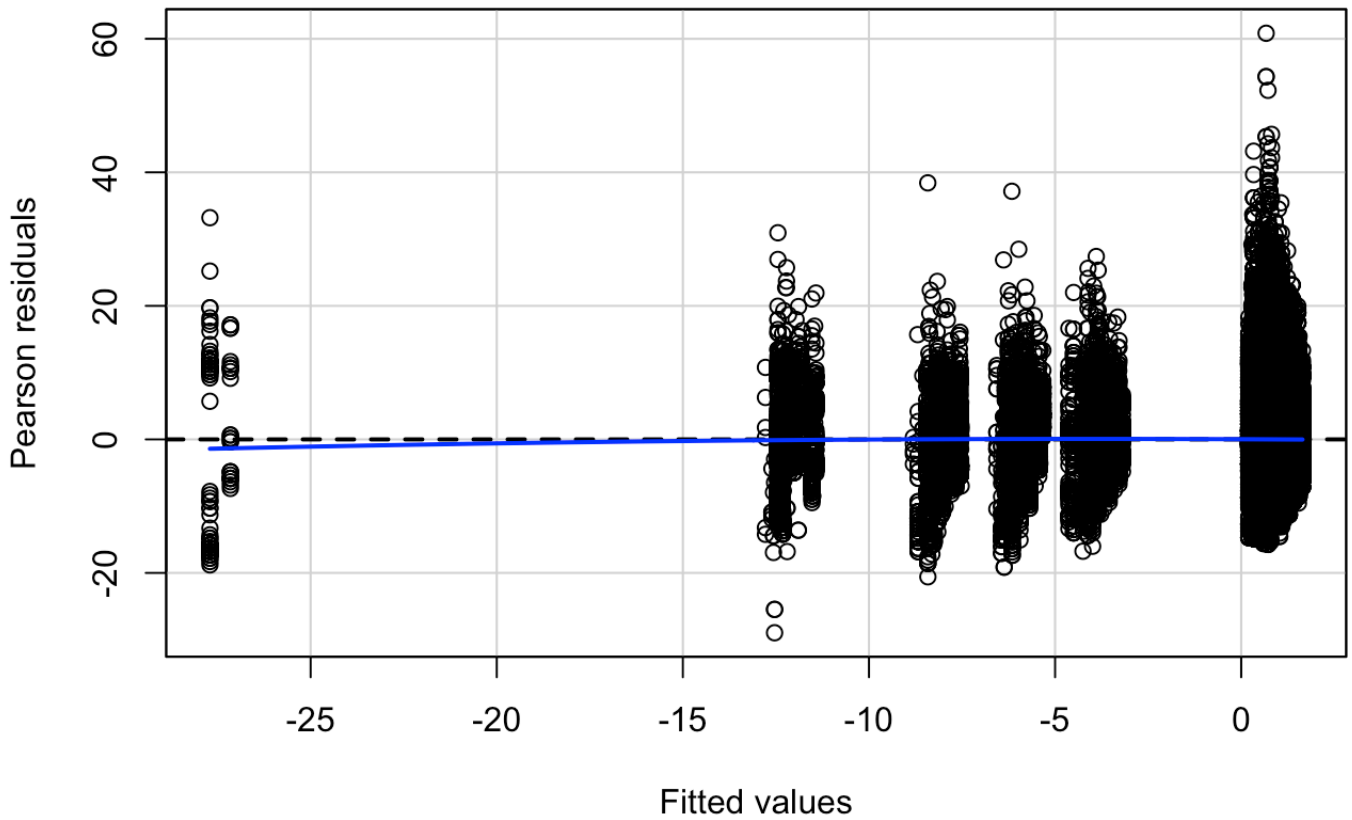

```
ggplot(samesub_e.train,aes(x=as.factor(surge_multiplier.y),y=pricediff)) + geom_boxpl
ot()
```



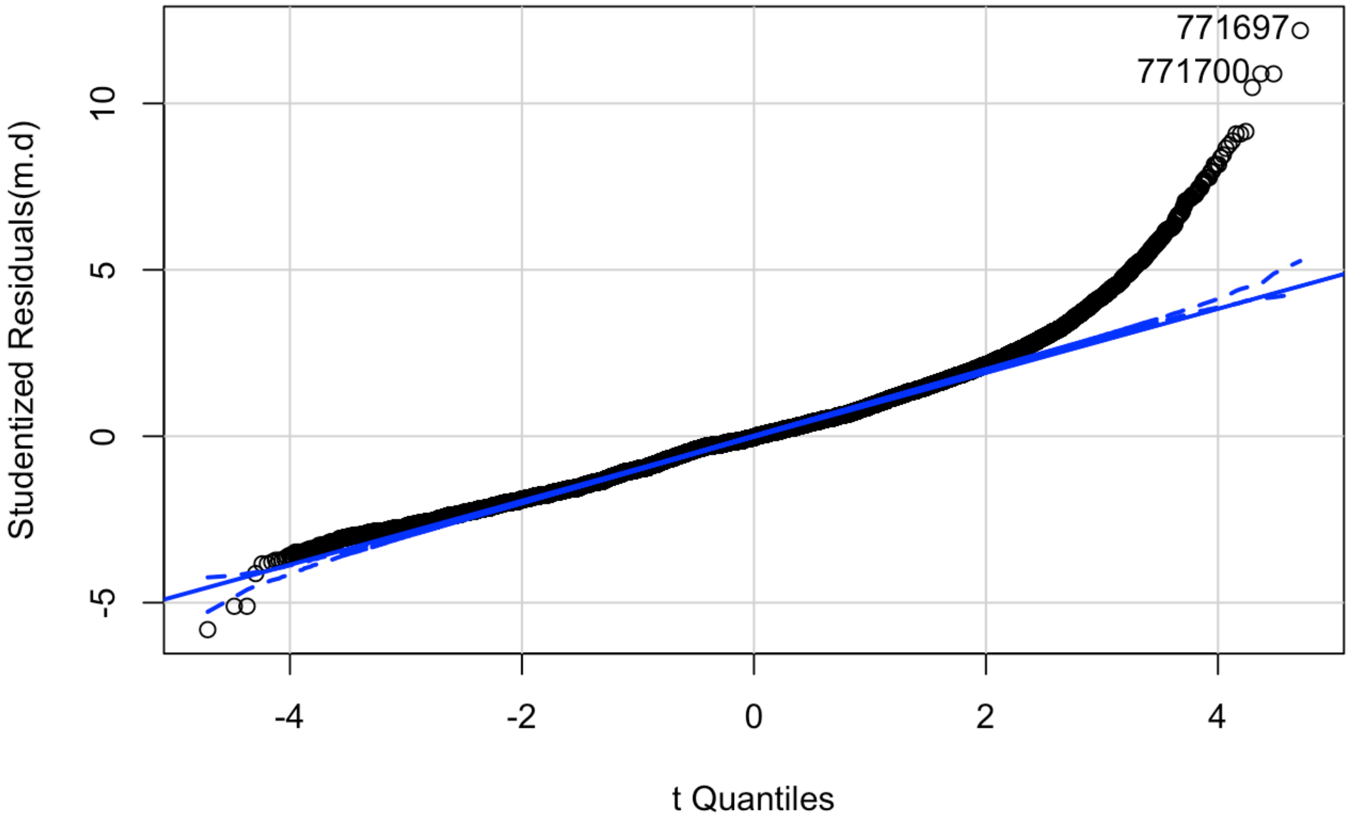

```
m.d <- lm(pricediff ~ distance + dropoff + pickup + bad_weather + Isweekend + rush_ho
ur + as.factor(surge_multiplier.y), data = samesub_e.train)
summary(m.d)
```

```
##
## Call:
## lm(formula = pricediff ~ distance + dropoff + pickup + bad_weather +
##     Isweekend + rush_hour + as.factor(surge_multiplier.y), data = samesub_e.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.964  -3.054  -0.168   2.739  60.830
##
## Coefficients:
##                                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                     1.519614   0.017981   84.512  < 2e-16 ***
## distance                       -0.258942   0.009154  -28.287  < 2e-16 ***
## dropoffNearby                   0.244105   0.019950   12.236  < 2e-16 ***
## pickupNearby                    0.035933   0.019949    1.801  0.07166 .
## bad_weather1                   -0.127838   0.033050   -3.868  0.00011 ***
## Isweekend1                      0.126795   0.017601    7.204 5.86e-13 ***
## rush_hour1                     -0.022586   0.040004   -0.565  0.57236
## as.factor(surge_multiplier.y)1.25  -4.835781   0.051438  -94.012  < 2e-16 ***
## as.factor(surge_multiplier.y)1.5   -6.975854   0.063351 -110.115  < 2e-16 ***
## as.factor(surge_multiplier.y)1.75  -9.090181   0.113432  -80.138  < 2e-16 ***
## as.factor(surge_multiplier.y)2    -13.215164   0.118804 -111.235  < 2e-16 ***
## as.factor(surge_multiplier.y)2.5  -12.944310   0.228995  -56.527  < 2e-16 ***
## as.factor(surge_multiplier.y)3    -28.442867   0.482556  -58.942  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.99 on 402060 degrees of freedom
## Multiple R-squared:  0.1028, Adjusted R-squared:  0.1028
## F-statistic:  3840 on 12 and 402060 DF,  p-value: < 2.2e-16
```

```
# residual
residualPlot(m.d)
```

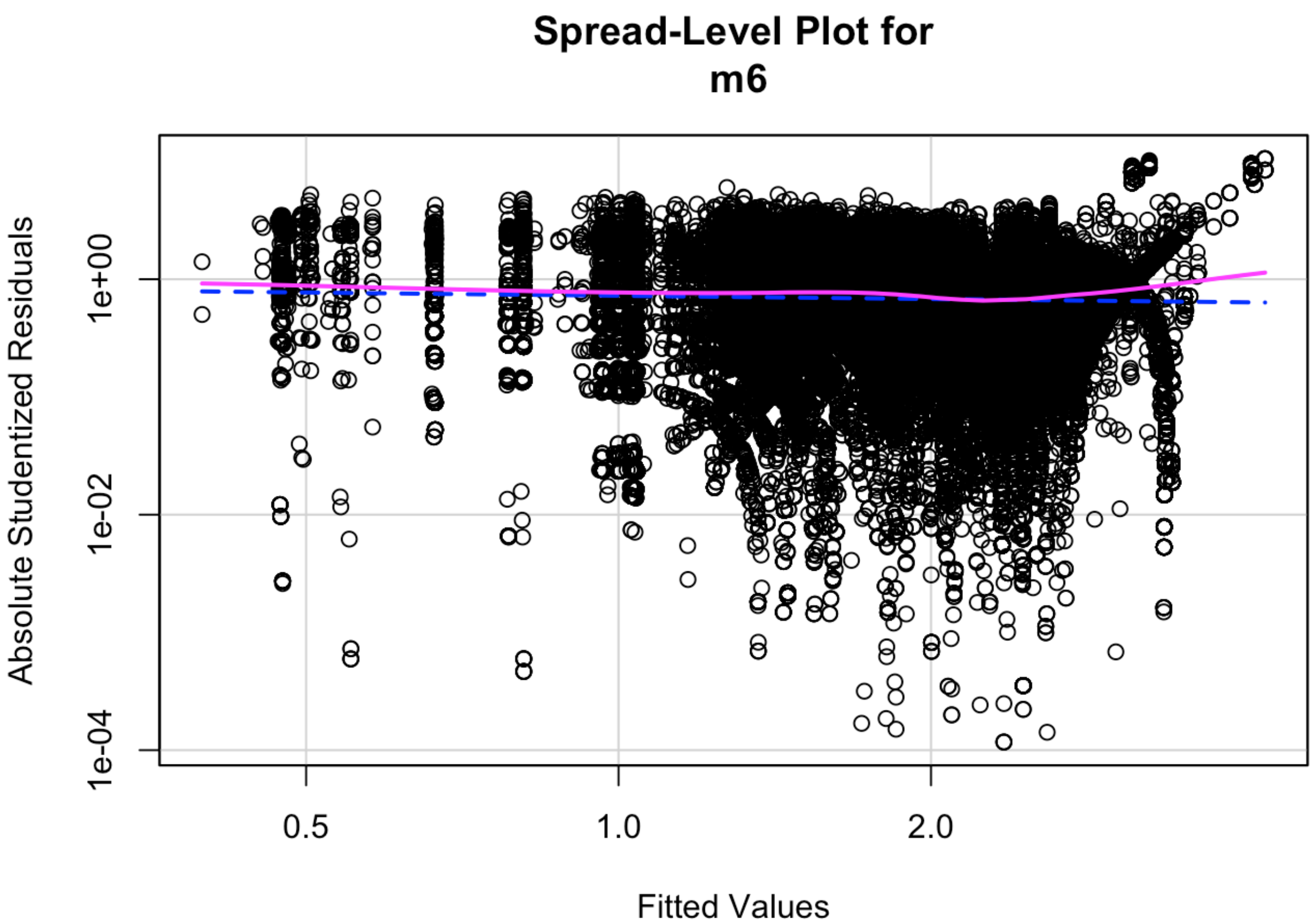



```
#normality
qqPlot(m.d)
```




```
## 771697 771700
##  34783 265561
```

```
#variance
spreadLevelPlot(m6)
```




```
##
## Suggested power transformation:  1.091977
```

```
# multicolinearity
car::vif(m.d)
```

```
##                                  GVIF Df GVIF^(1/(2*Df))
## distance                     1.512008  1        1.229637
## dropoff                      1.139359  1        1.067408
## pickup                       1.448146  1        1.203390
## bad_weather                  1.029635  1        1.014709
## Isweekend                    1.031450  1        1.015603
## rush_hour                    1.003185  1        1.001591
## as.factor(surge_multiplier.y) 1.026253  6        1.002162
```

```
# MSE
mean((m.d$fitted.values - samesub_e.train$pricediff)^2)
```

```
## [1] 24.90098
```

```
mean((predict(m.d,samesub_e.test)-samesub_e.test$pricediff)^2)
```

```
## [1] 24.85959
```

We fit a regression tree to the price difference model and found that the most important variable in predicting a price difference between UberEconomy and LyftEconomy is if there is a surge. This makes sense because Uber data does not include a surge so this would definitely be the number one cause in a difference in price between the two.

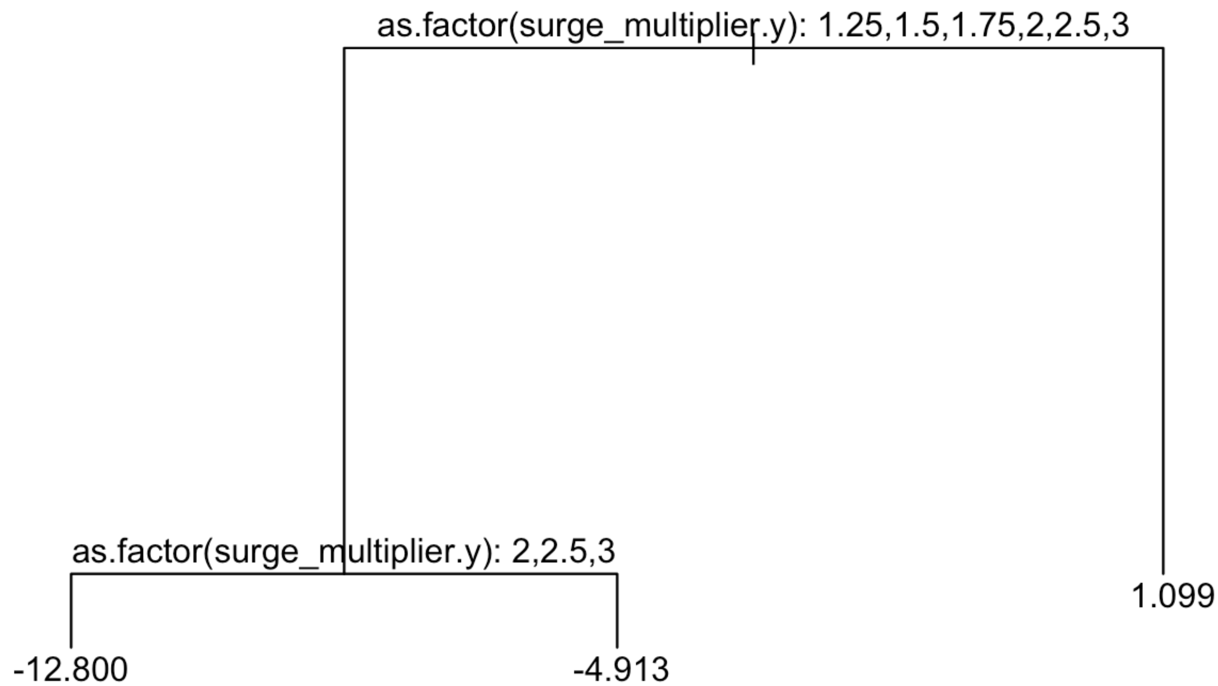
Moreover, we get some insights from the regression tree: 1. If there is no surge which means surge multiplier equals to 1, taking ride with Uber will pay $1 dollar more than Lyft. 2. If surge multiplier is lower than 2, taking ride with lyft will pay $4.9 more than uber. 3. If surge multiplier is even higer, the difference will go up to $12.8 per ride.

The MSE for the Economy linear model is 24.87 and the MSE for the Economy regression tree is 25.11. So, again, we would use the regression tree as a more accurate predictor of price differences between Uber Economy and Lyft Economy.

```
## tree
tree.dif <- tree(pricediff ~ distance + dropoff + pickup + bad_weather + Isweekend +
rush_hour + as.factor(surge_multiplier.y),data = samesub_e.train,split = "deviance")
summary(tree.dif)
```

```
##
## Regression tree:
## tree(formula = pricediff ~ distance + dropoff + pickup + bad_weather +
##     Isweekend + rush_hour + as.factor(surge_multiplier.y), data = samesub_e.train,
##     split = "deviance")
## Variables actually used in tree construction:
## [1] "as.factor(surge_multiplier.y)"
## Number of terminal nodes:  3
## Residual mean deviance:  25.13 = 10100000 / 402100
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -33.7000  -3.0990  -0.0987   0.0000   2.9010  60.4000
```

```
plot(tree.dif)
text(tree.dif,pretty = 0)
```



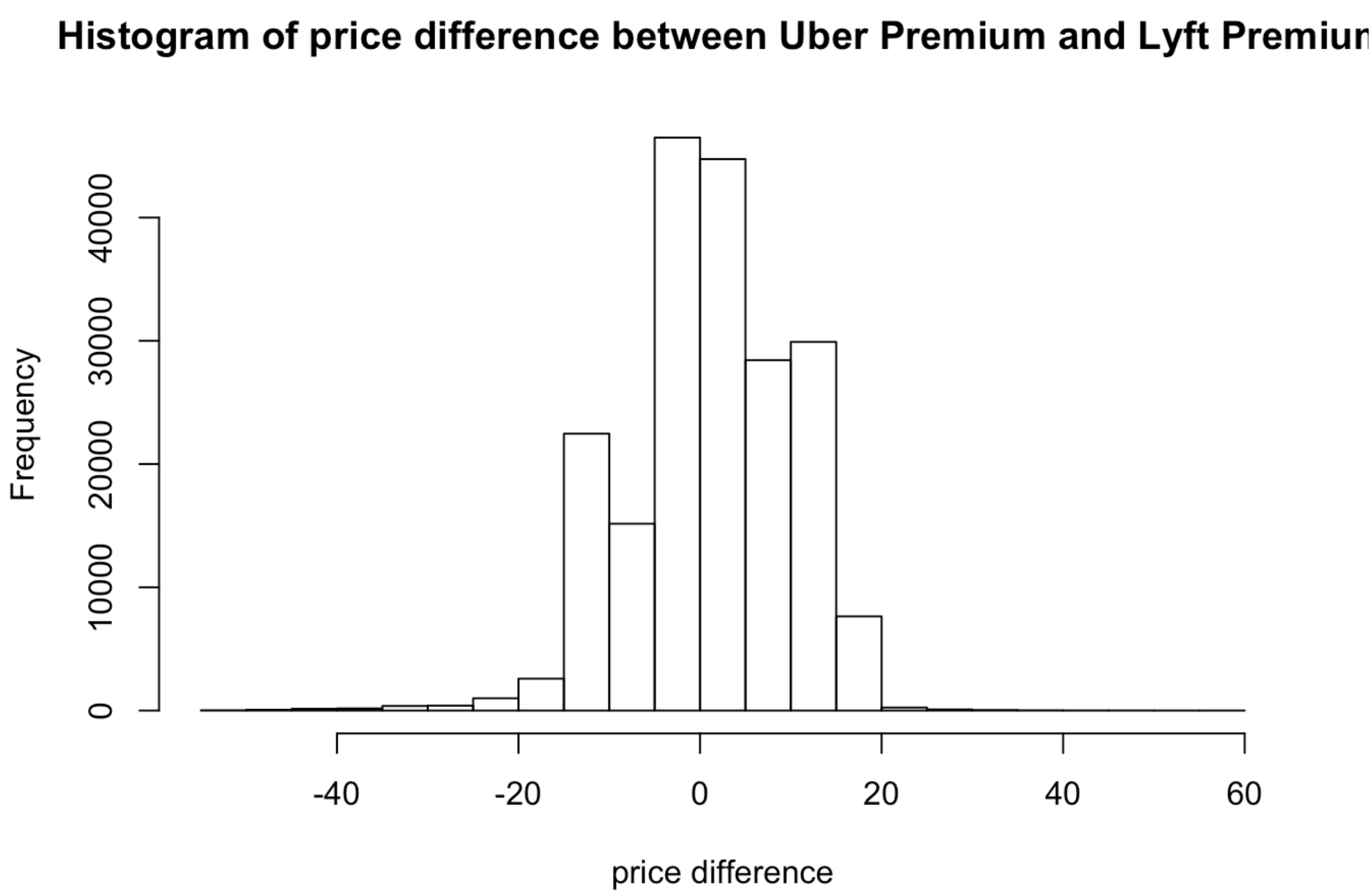

```
# MSE-test
pred.t <- predict(tree.dif,samesub_e.test)
MSE2 <- mean((samesub_e.test$pricediff-pred.t)^2)
MSE2
```

```
## [1] 25.07468
```

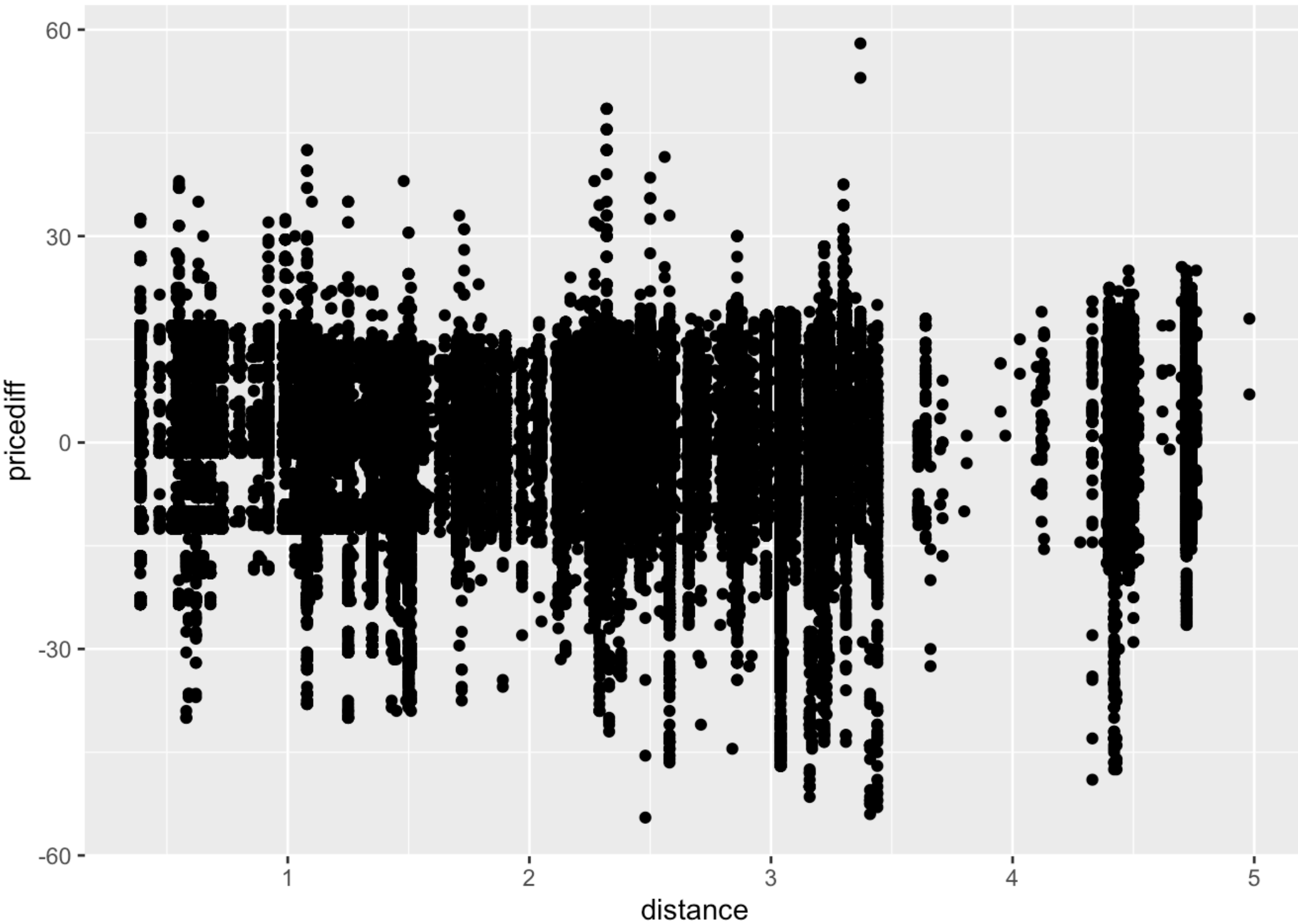
3-2 Uber Premium VS Lyft Premium

We used the same predictors to find whether there is a price difference between Uber Premium and Lyft Premium. Based on the summary, we see that distance, pick up, bad weather, if it's a weekend, if it's a rush hour and surge_multiplier all contribute to a price difference between Uber Premium and Lyft Premium, proven by their small p-values. Some calculations show that if pick up points at downtown, no bad weather, in weekdays and not in rush hours, we would ride with lyft with less expensive unless the ride miles more than 7.8 miles.((It is also interesting that different factors lead to price differences in different types of cars (eg. Uber Economy vs Uber Premium). Further, the model assumptions for this model hold, so we believe this model is a good fit for our data.)

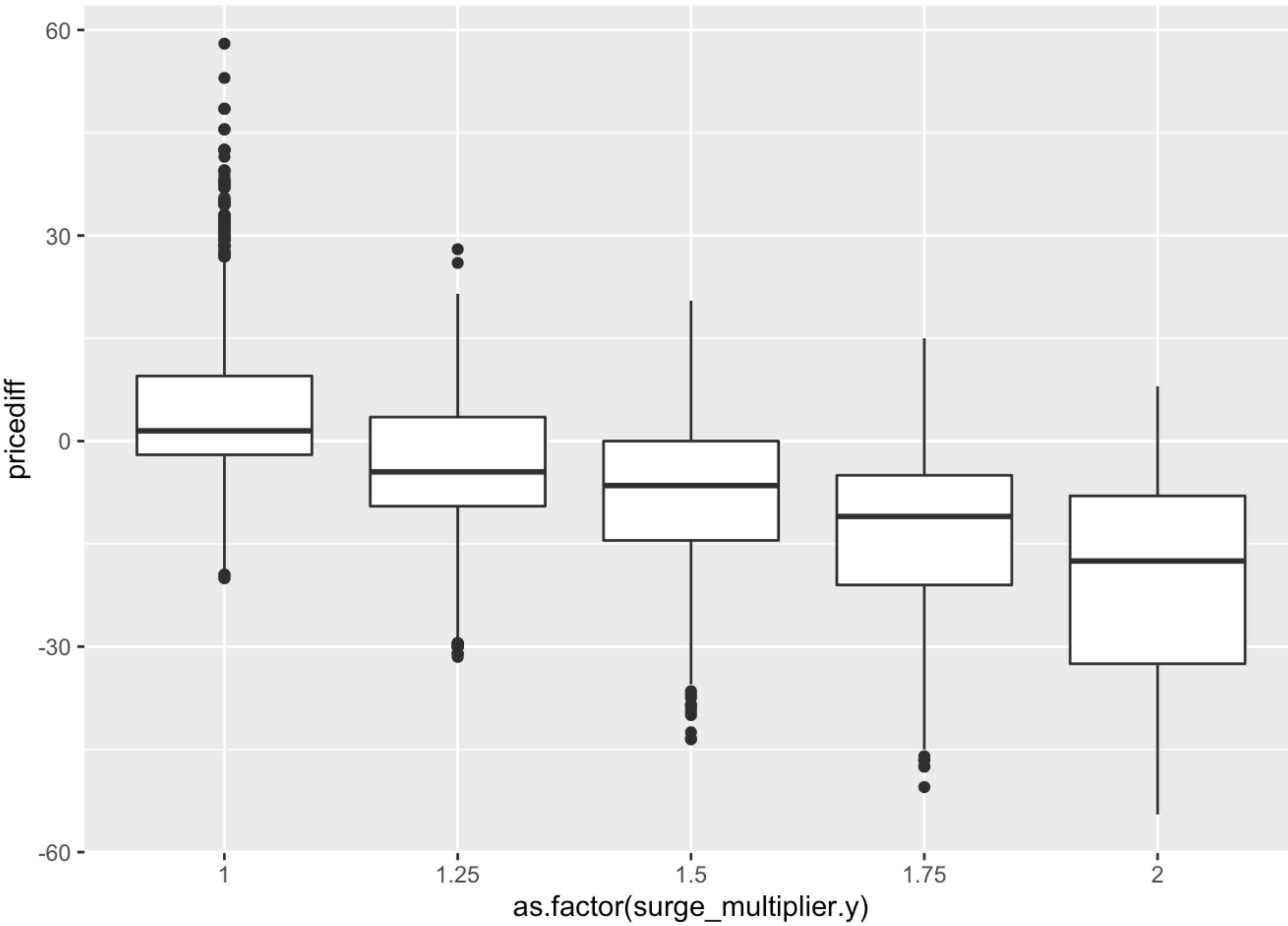
```
## Uber Premium VS Lyft Premium
# merge data
samesub_p <- merge(uber[uber$service == "Uber Premium",],lyft[lyft$service=="Lyft Pre
mium",],by=intersect(c('distance','bad_weather','Isweekend','rush_hour','destination'
,'source',"weekday","pickup","dropoff"),c('distance','bad_weather','Isweekend','rush_
hour','destination','source',"weekday","pickup","dropoff")))
samesub_p$pricediff <- samesub_p$price.x - samesub_p$price.y

# split data
set.seed(123)
index <- sample(1:nrow(samesub_p),as.integer(nrow(samesub_p)*0.5))
samesub_p.train <- samesub_p[index,]
samesub_p.test <- samesub_p[-index,]

#First, let's take a look at our price diff distribution
hist(samesub_p.train$pricediff,main = "Histogram of price difference between Uber Pre
mium and Lyft Premium",xlab = "price difference")
```



```
#The price difference tend to be normal distribution which means overall they tend to
be identical.
ggplot(samesub_p.train,aes(x=distance,y=pricediff))+geom_point()
```



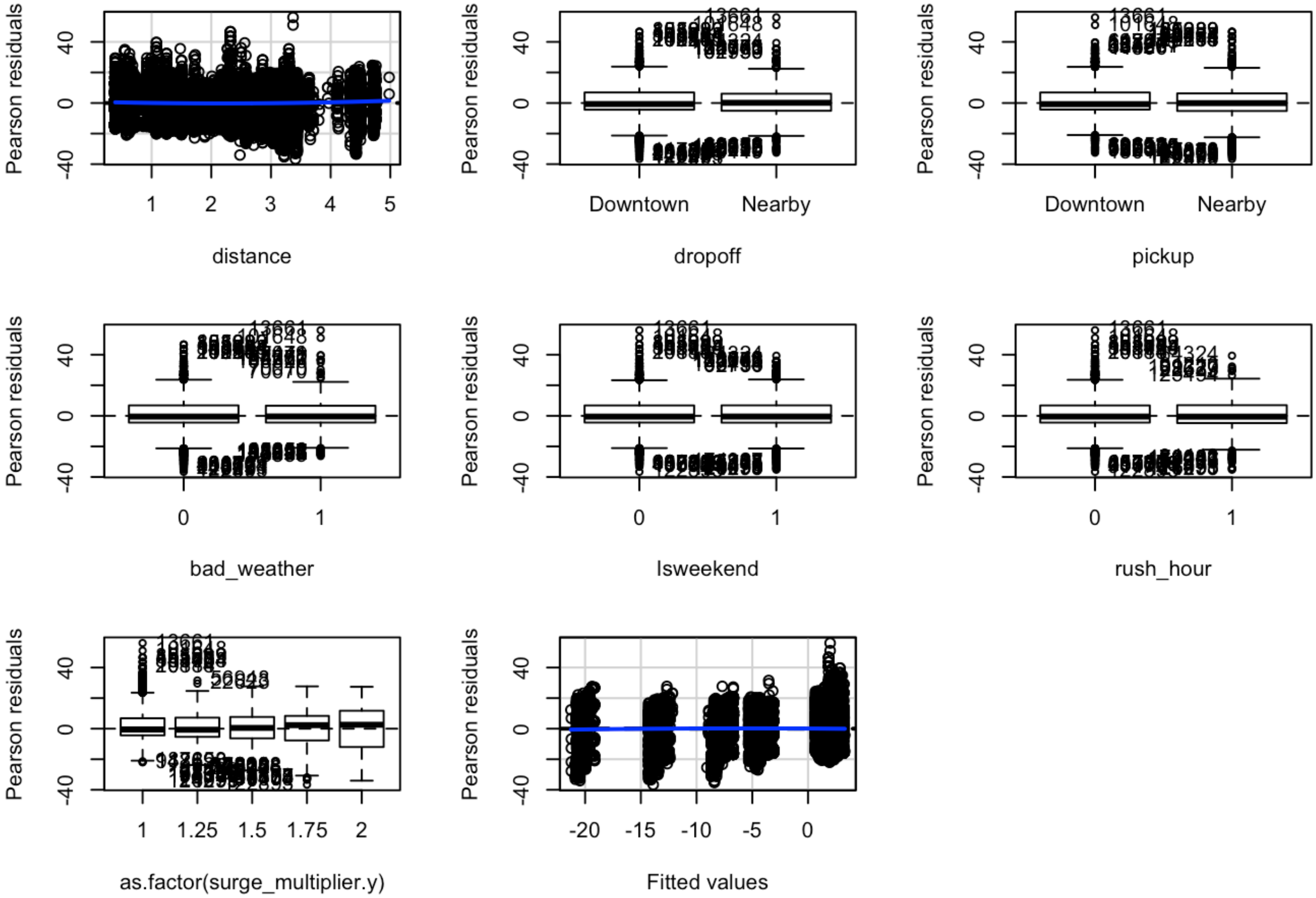

```
ggplot(samesub_p.train,aes(x=as.factor(surge_multiplier.y),y=pricediff)) + geom_boxpl
ot()
```




```
## linear model
m.p <- lm(pricediff~ distance + dropoff + pickup + bad_weather + Isweekend + rush_hou
r + as.factor(surge_multiplier.y),data = samesub_p.train)
summary(m.p)
```

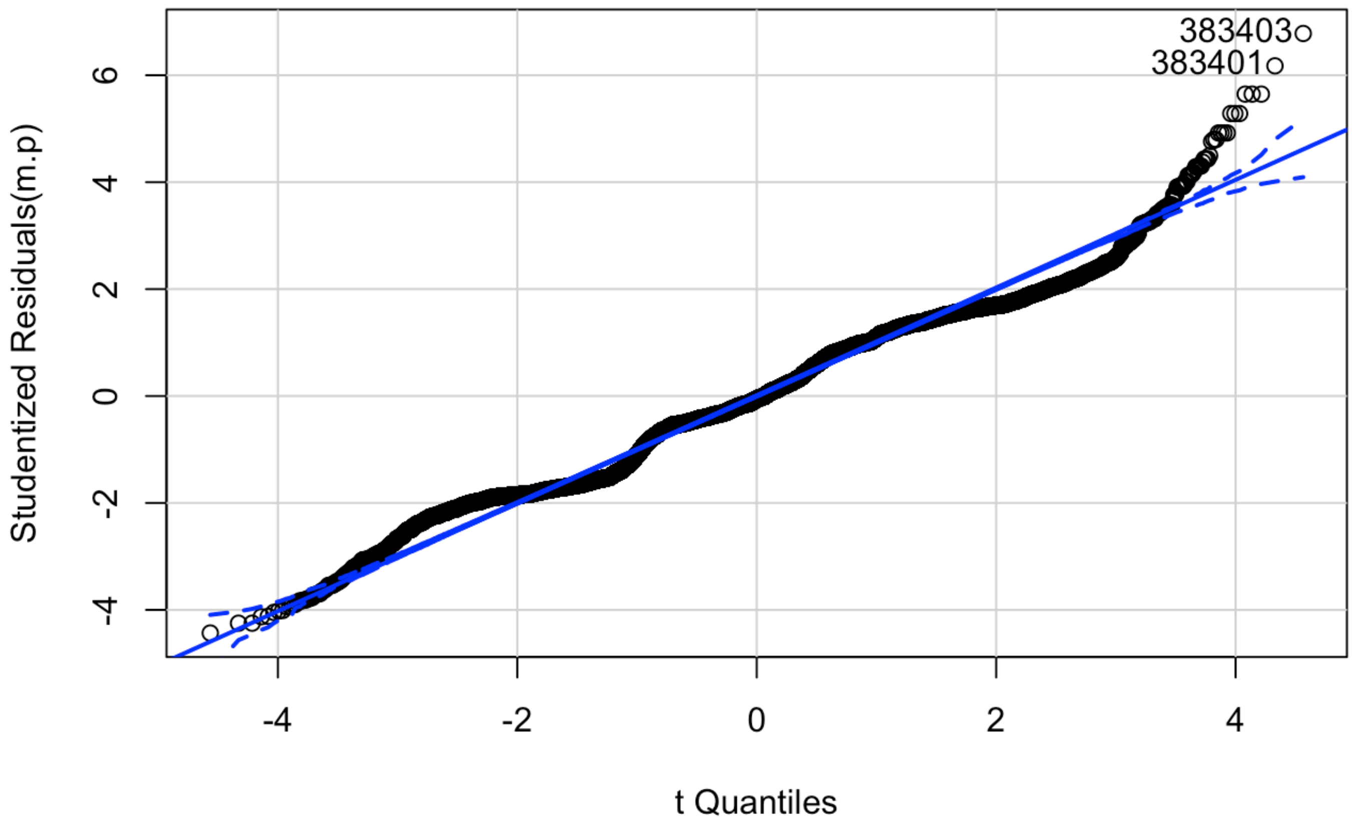
```
##
## Call:
## lm(formula = pricediff ~ distance + dropoff + pickup + bad_weather +
##     Isweekend + rush_hour + as.factor(surge_multiplier.y), data = samesub_p.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.610  -4.414  -0.466   6.812  56.018
##
## Coefficients:
##                                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                     3.13532    0.04232   74.086  < 2e-16 ***
## distance                       -0.40205    0.02129  -18.885  < 2e-16 ***
## dropoffNearby                   0.04033    0.04674    0.863   0.3883
## pickupNearby                   -0.35631    0.04665   -7.638 2.21e-14 ***
## bad_weather1                    0.16131    0.07574    2.130   0.0332 *
## Isweekend1                      0.22321    0.04179    5.341 9.24e-08 ***
## rush_hour1                     -0.20069    0.09132   -2.198   0.0280 *
## as.factor(surge_multiplier.y)1.25  -6.19383    0.09954  -62.228  < 2e-16 ***
## as.factor(surge_multiplier.y)1.5   -9.91040    0.12638  -78.415  < 2e-16 ***
## as.factor(surge_multiplier.y)1.75 -15.29820    0.20515  -74.571  < 2e-16 ***
## as.factor(surge_multiplier.y)2    -22.27591    0.21574 -103.255  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.262 on 199953 degrees of freedom
## Multiple R-squared:  0.1167, Adjusted R-squared:  0.1166
## F-statistic:  2641 on 10 and 199953 DF,  p-value: < 2.2e-16
```

```
# residual
residualPlots(m.p)
```




```
##                                Test stat Pr(>|Test stat|)
## distance                          15.648        < 2.2e-16 ***
## dropoff
## pickup
## bad_weather
## Isweekend
## rush_hour
## as.factor(surge_multiplier.y)
## Tukey test                       -26.850        < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
qqPlot(m.p)
```




```
## 383403 383401
##  13661 101648
```

```
# multicolinearity
car::vif(m.p)
```

```
##                                    GVIF Df GVIF^(1/(2*Df))
## distance                       1.500955  1        1.225135
## dropoff                        1.136440  1        1.066039
## pickup                         1.445832  1        1.202427
## bad_weather                    1.028640  1        1.014219
## Isweekend                      1.028202  1        1.014003
## rush_hour                      1.002900  1        1.001449
## as.factor(surge_multiplier.y) 1.017183  4        1.002132
```

```
# MSE
mean((m.p$fitted.values - samesub_p.train$pricediff)^2)
```

```
## [1] 68.25938
```

```
mean((predict(m.p,samesub_p.test)-samesub_p.test$pricediff)^2)
```

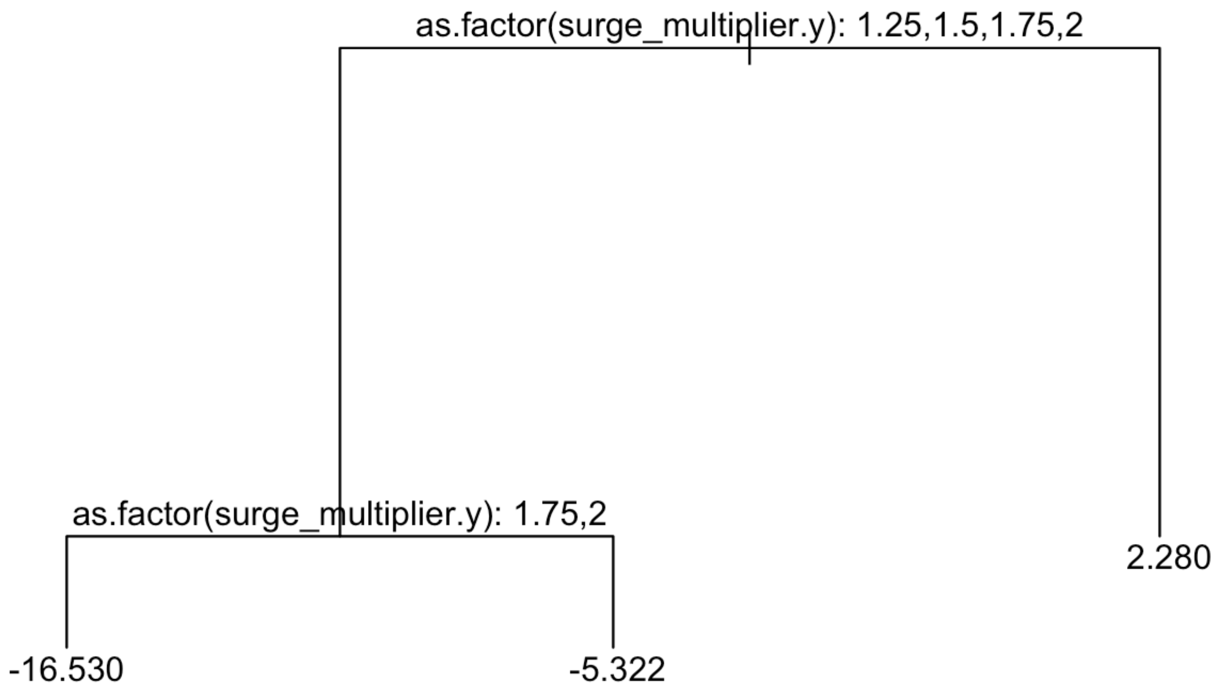
```
## [1] 67.91445
```

Similarly to the regression tree for the Economy style cars, the regression tree shows that the most important predictor in price difference is the surge, which again makes sense since Lyft has a surge and Uber does not. Compared to the regression tree for Economy, the cutoff point is different. As for no surge, uber will be $2.28 higher than lyft per ride. If surge multiplier is lower than 1.75, taking lyft will pay 5.3 dollars more than uber. If the surge multiplier is even higher, taking lyft will pay 16.53 dollars more than uber. The MSE for the Premium linear model is 67.91 and the MSE for the Premium regression tree is 68.49. While the two errors are extremely close, the tree MSE is slightly smaller, so we would use this to predict whether there will be a difference in price between Uber and Lyft.

```
## tree
tree.dp <- tree(pricediff ~ distance + dropoff + pickup + bad_weather + Isweekend + r
ush_hour + as.factor(surge_multiplier.y),data = samesub_p.train,split = "deviance")
summary(tree.dp)
```

```
##
## Regression tree:
## tree(formula = pricediff ~ distance + dropoff + pickup + bad_weather +
##     Isweekend + rush_hour + as.factor(surge_multiplier.y), data = samesub_p.train,
##     split = "deviance")
## Variables actually used in tree construction:
## [1] "as.factor(surge_multiplier.y)"
## Number of terminal nodes:  3
## Residual mean deviance:  68.93 = 13780000 / 2e+05
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -38.1800  -4.2800  -0.7802   0.0000   7.2200  55.7200
```

```
plot(tree.dp)
text(tree.dp,pretty = 0)
```




```
# MSE-test
pred.t <- predict(tree.dp,samesub_p.test)
MSE2 <- mean((samesub_p.test$pricediff-pred.t)^2)
MSE2
```

```
## [1] 68.49443
```

4.Logistic Regression - Surge

Next, we fit a logistic regression model to predict the odds that there will be a surge multiplier. Since Uber does not have a surge multiplier attribute in the given data set, this will only be the log odds that there is a surge multiplier for a Lyft ride.

First, we want to target useful predictors.

```
g1 <- qplot(x=surge,
      y=price_per_mile,
      data=train_l,
      geom="boxplot",
      xlab="Surge",
      main = 'Price per mile',
      ylim = c(0,50))

g2 <- qplot(x=surge,
      y=distance,
      data=train_l,
      geom="boxplot",
      xlab="Surge",
      main = 'Distance',
      ylim = c(0,8))


g3 <- ggplot(train_l, aes(x= surge, group = pickup)) +
    geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
    geom_text(aes( label = scales::percent(..prop..),
                   y= ..prop.. ), stat= "count", vjust = -.5) +
    labs(y = "Percent", fill="surge") +
    facet_grid(~pickup) +
    scale_y_continuous(labels = scales::percent) + ggtitle('Pick up')

chisq.test(train_l$pickup, train_l$surge)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  train_l$pickup and train_l$surge
## X-squared = 1412.8, df = 1, p-value < 2.2e-16
```

```
g4 <- ggplot(train_l, aes(x= surge, group = dropoff)) +
    geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
    geom_text(aes( label = scales::percent(..prop..),
                   y= ..prop.. ), stat= "count", vjust = -.5) +
    labs(y = "Percent", fill="surge") +
    facet_grid(~dropoff) +
    scale_y_continuous(labels = scales::percent) +ggtitle('Drop off')

chisq.test(train_l$dropoff, train_l$surge)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  train_l$dropoff and train_l$surge
## X-squared = 0.40608, df = 1, p-value = 0.524
```

```
g5 <- ggplot(train_l, aes(x= surge, group = bad_weather)) +
    geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
    geom_text(aes( label = scales::percent(..prop..),
                   y= ..prop.. ), stat= "count", vjust = -.5) +
    labs(y = "Percent", fill="surge") +
    facet_grid(~bad_weather) +
    scale_y_continuous(labels = scales::percent) +ggtitle('Bad weather')

chisq.test(train_l$bad_weather, train_l$surge)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  train_l$bad_weather and train_l$surge
## X-squared = 0.20245, df = 1, p-value = 0.6527
```

```
g6 <- ggplot(train_l, aes(x= surge, group = Isweekend)) +
    geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
    geom_text(aes( label = scales::percent(..prop..),
                   y= ..prop.. ), stat= "count", vjust = -.5) +
    labs(y = "Percent", fill="surge") +
    facet_grid(~Isweekend) +
    scale_y_continuous(labels = scales::percent) + ggtitle('Weekend')

chisq.test(train_l$Isweekend, train_l$surge)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  train_l$Isweekend and train_l$surge
## X-squared = 1.2312, df = 1, p-value = 0.2672
```

```
g7 <- ggplot(train_l, aes(x= surge, group = rush_hour)) +
    geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
    geom_text(aes( label = scales::percent(..prop..),
                  y= ..prop.. ), stat= "count", vjust = -.5) +
    labs(y = "Percent", fill="surge") +
    facet_grid(~rush_hour) +
    scale_y_continuous(labels = scales::percent) +ggtitle('Rush hour')

chisq.test(train_l$rush_hour, train_l$surge)
```

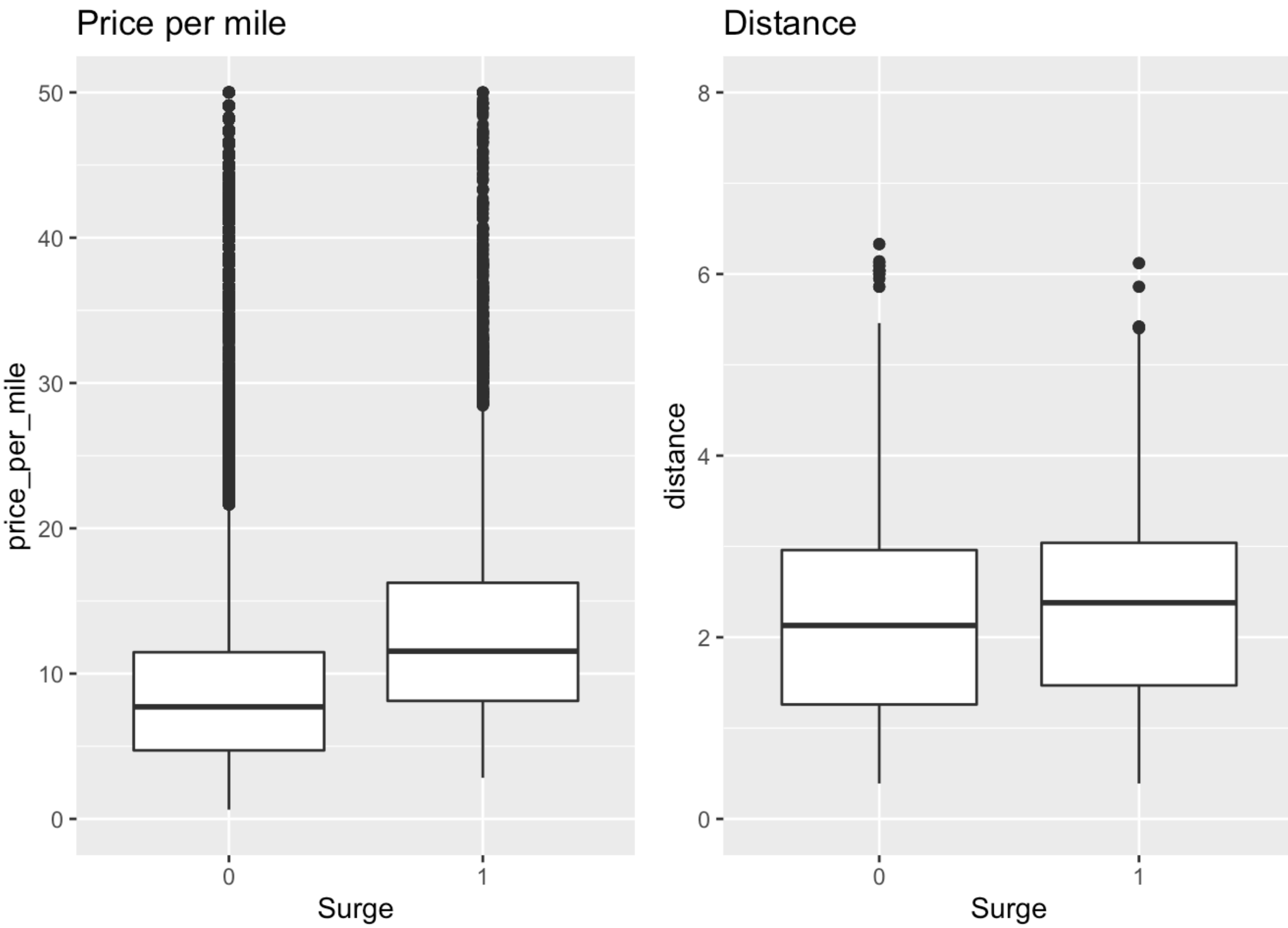
```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  train_l$rush_hour and train_l$surge
## X-squared = 2.2807, df = 1, p-value = 0.131
```

```
g8 <- ggplot(train_l, aes(x= surge, group = service)) +
    geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
    geom_text(aes( label = scales::percent(..prop..),
                  y= ..prop.. ), stat= "count", vjust = -.5) +
    labs(y = "Percent", fill="surge") +
    facet_grid(~service) +
    scale_y_continuous(labels = scales::percent)+ggtitle('Service')

chisq.test(train_l$service, train_l$surge)
```

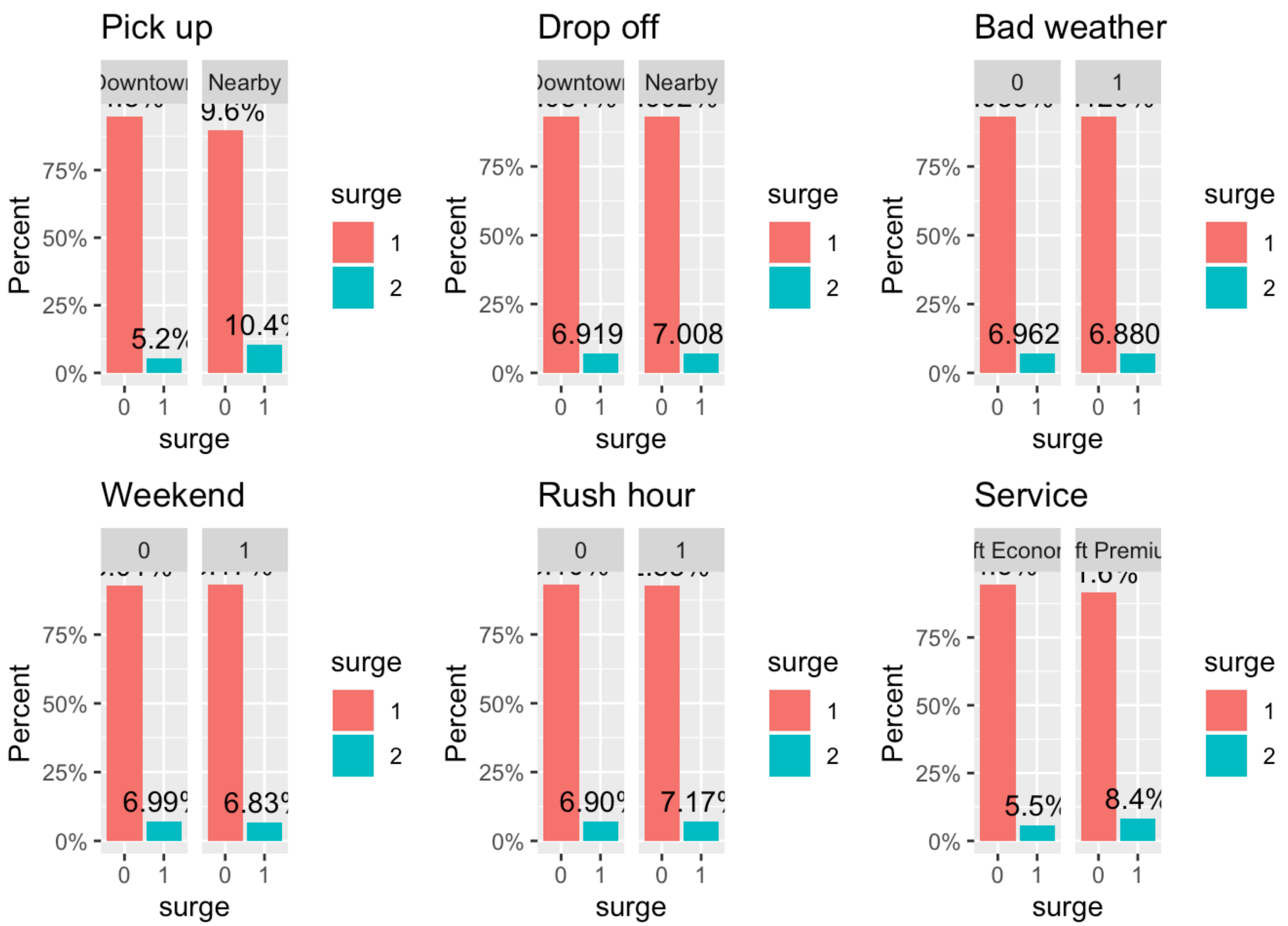
```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  train_l$service and train_l$surge
## X-squared = 466.58, df = 1, p-value < 2.2e-16
```

```
grid.arrange(g1,g2, ncol=2, nrow = 1)
```

```
## Warning: Removed 595 rows containing non-finite values (stat_boxplot).
```




```
grid.arrange(g3,g4,g5,g6,g7,g8, ncol=3, nrow = 2)
```



From the plots and chi-square test, we think price per mile, distance, pickup ,and service would be good predictors.

After building the logistic model, all variables we put are significant, meaning they are useful in predicting the log odds that there will a surge. We than use confusion matrix to compute the classification rate, which are up to 93%.

```
#surge data in training and testing data set
sum(train_l$surge == 1)/ nrow(train_l)
```

```
## [1] 0.06948664
```

```
sum(test_l$surge == 1)/ nrow(test_l)
```

```
## [1] 0.06696461
```

```
#build surge glm
ml1 <- glm(as.factor(surge) ~ price_per_mile + service + pickup + distance, family =
'binomial', data=train_l)
summary(ml1)
```

```
##
## Call:
## glm(formula = as.factor(surge) ~ price_per_mile + service + pickup +
##     distance, family = "binomial", data = train_l)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9839  -0.4148  -0.3030  -0.2464   2.7132
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -4.824178   0.038508 -125.28   <2e-16 ***
## price_per_mile       0.097485   0.001436   67.91   <2e-16 ***
## serviceLyft Premium -0.402702   0.025141  -16.02   <2e-16 ***
## pickupNearby         0.907744   0.022148   40.99   <2e-16 ***
## distance             0.436651   0.011049   39.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 77232  on 153050  degrees of freedom
## Residual deviance: 70804  on 153046  degrees of freedom
## AIC: 70814
##
## Number of Fisher Scoring iterations: 6
```

```
#no collinearity issue
car::vif(ml1)
```

```
## price_per_mile        service         pickup       distance
##       2.213775       1.428173       1.136641       1.655424
```

```
# # confusion matrix
glm.pred  <- rep(0,nrow(test_l))
glm.probs <- predict.glm(ml1,newdata = test_l,type = "response")
glm.pred[glm.probs>.5] <- 1
table(glm.pred, test_l$surge)
```

```
##
## glm.pred      0      1
##        0 142344  10102
##        1    458    147
```

```
mean(glm.pred==test_l$surge)
```

```
## [1] 0.9310034
```

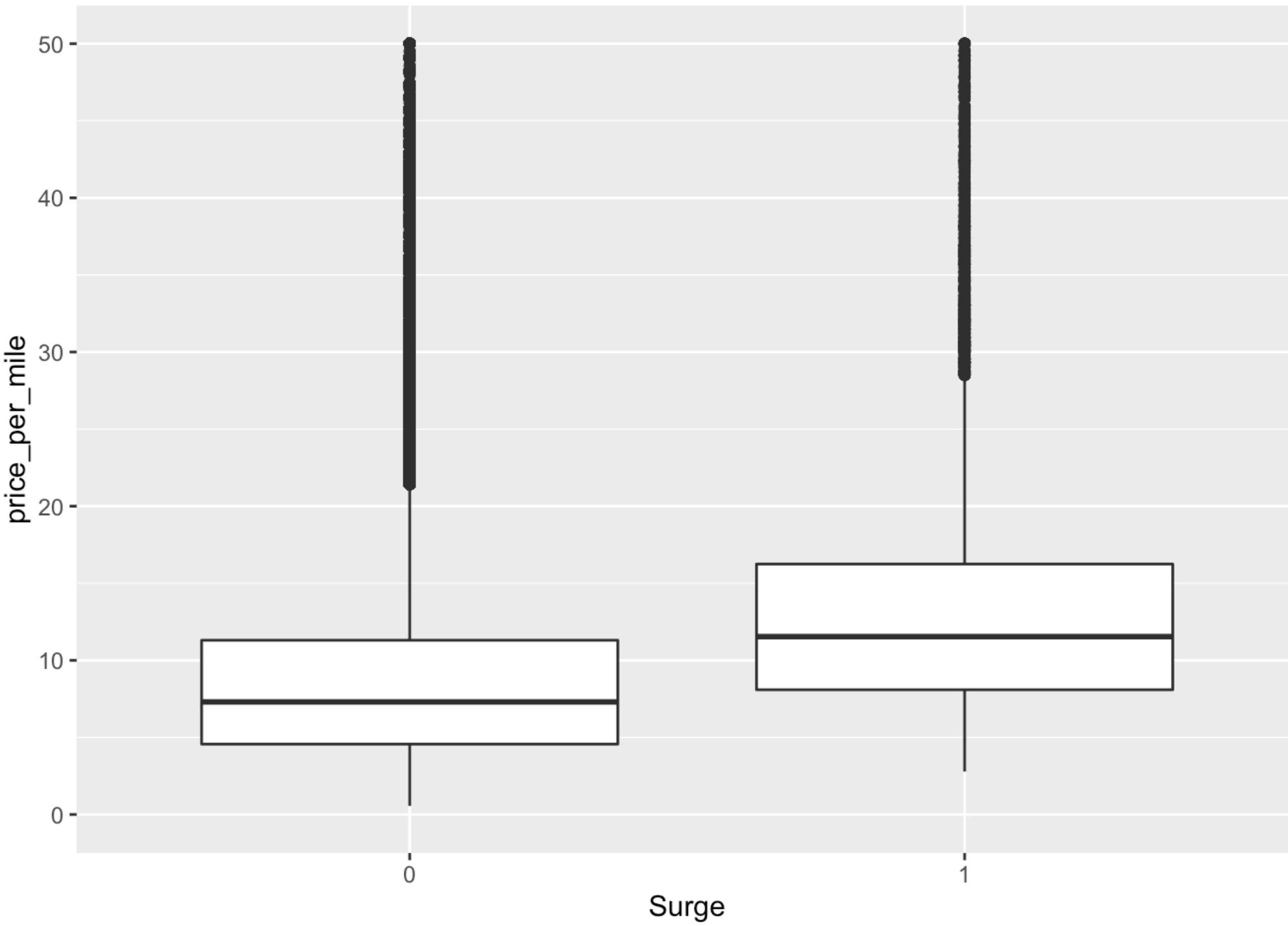
```
#precision rate
(sum((glm.pred==test_l$surge)&(glm.pred==1))/sum(glm.pred==1))
```

```
## [1] 0.2429752
```

We also wanted to explore at what unit price is surge pricing included in the price per mile cost. From the plots, we see that our unit price cut off is around 16. We used a loop to find the most precise pice cut off value by measuring the misclassification, precision, and true positives rates. When increasing the threshold, the misclassication rate will go down, making the minimum equal to a random guess, so we will use the precision and true positive rates to find the cut off price. We found that the best cut off price for a surge would be 15 with a percision rate 0.14, so if the price per mile is above $15, there will likely be a surge.

```
# from the plot, we can see that our unit price cut off should around 16.
qplot(x=surge,
      y=price_per_mile,
      data=df1,
      geom="boxplot",
      xlab="Surge",
      ylim = c(0,50))
```

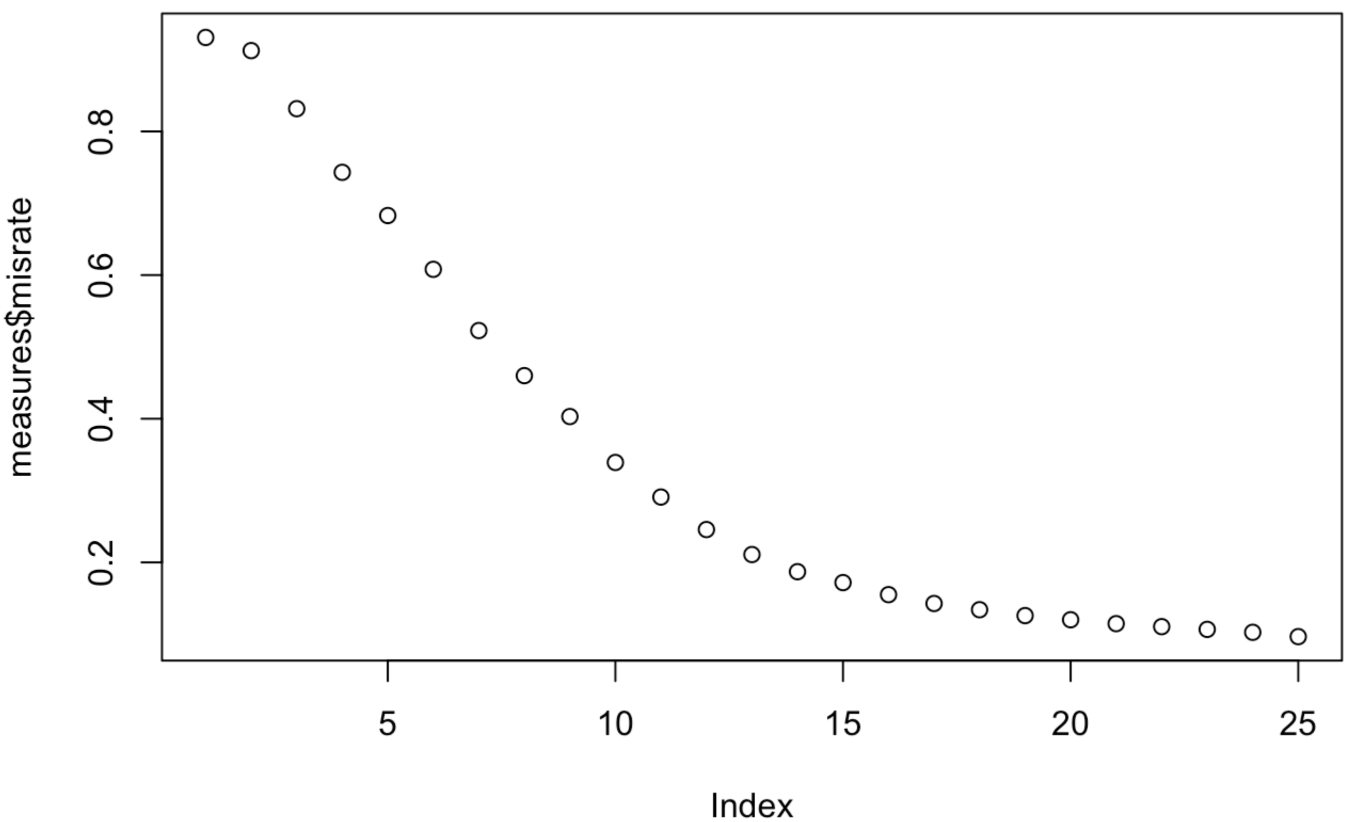
```
## Warning: Removed 58237 rows containing non-finite values (stat_boxplot).
```



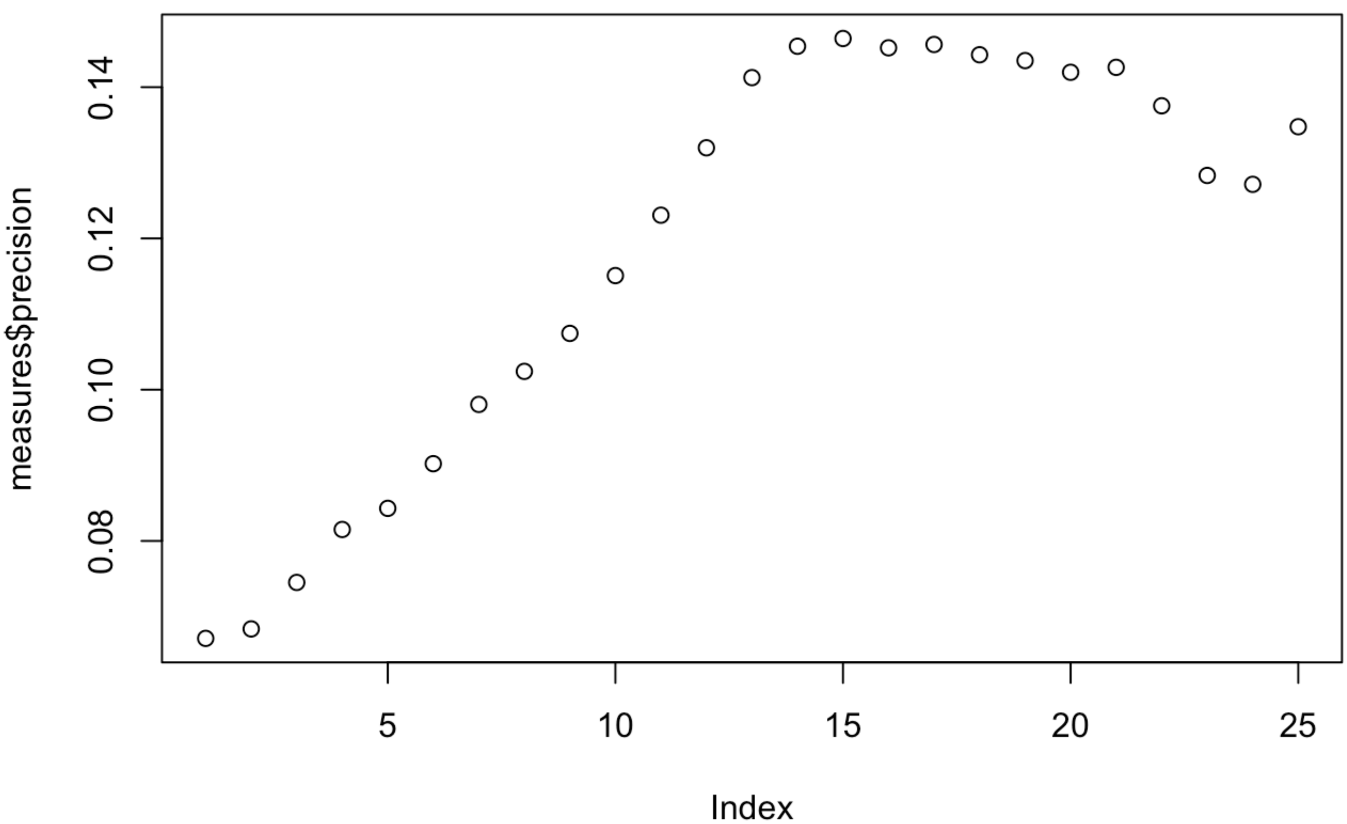

```
# we use a loop to find the perfect price by measuring misclassfication rate, precisi
on:true positives/ all positives, true positive rate: ture positivies / predicted pos
itives
cutoff.p <- 25
measures <- data.frame(misrate = rep(0,cutoff.p),precision = rep(0,cutoff.p),tpr = re
p(0,cutoff.p),fpr = rep(0,cutoff.p))
for (i in 1:cutoff.p) {
  pred <- ifelse(test_l$price_per_mile > i,1,0)
  measures$misrate[i] <- 1- mean(pred == test_l$surge)
  measures$precision[i] <- sum((pred==test_l$surge)&(pred==1))/sum(pred==1)
  measures$tpr[i] <- sum((pred==test_l$surge)&(pred==1))/sum(test_l$surge==1)
  measures$fpr[i] <- sum((pred!=test_l$surge)&(pred==1))/sum(test_l$surge==0)
}

which.min(measures$misrate)
```

```
## [1] 25
```

```
# from the misclassfication plot, we can see that if we increase threshold, the rate
will go down which means the minimum should equals to a random guess. This doesn't sa
tisfy our goal.
plot(measures$misrate)
```




```
# we can see this precision and true positive rate plot, the best option should be bo
th of the critera are high.
plot(measures$precision)
```




```
(index <- which.max(measures$precision))
```

```
## [1] 15
```

```
measures[index,]
```

```
##      misrate precision       tpr       fpr
## 15 0.1718773 0.1464306 0.3244219 0.1357264
```

The precision rate of simple model is lower than logistic model.

5.Conclusion

When predicting price per mile in Uber, we would use the regression tree based on the lower MSE, making distance the best predictor for price per mile. Additionally, the particular service being Uber Ecomony is also helpful in predicting price per mile. This makes sense because Uber sets different prices for different services, so these various set prices will lead to a difference in price per mile. The regression tree is also superior to the linear model in this case because it more clearly shows and explains the extremely large or extremely small price per mile values, so it allows us to make better predictions for why these prices are the way they are. For example, we were able to see that most often high priced rides had low distances, and we could make the assumption that these were cancelled rides.

Predicting price per mile in Lyft had a similar outcome to that of Uber. The Lyft regression tree did a better job than the linear model based on the MSE value. We were able to see that the Lyft service being Lyft Economy was the most important predictor for price per mile and that distance was also important in predicting this value.

When predicting if there is a price difference between the same level service for Uber and Lyft, we found that the regression tree was the best tool to predict the difference. The tree showed for both levels of service that a surge in price was the most important predictor to prove their is a difference in price between the two companies. Uber does not have this surge value, so it is logical that it would be the most deciding factor of price between Uber and Lyft.

From the surge logistic model, price per mile, distance, pickup ,and service are related to a surge or not. From the simple model, we found that when price per mile is over $15, there is likely a surge in unit price. However, the precision rate of simple model is lower than logistic model, which means price per mile is not the only factors driven a surge.