Project1

Gefei Zhao

2020/11/16

Contents


```
Sys.setenv(LANGUAGE = "en")
# import data
hp <- read.csv("C:/Users/Gefei Zhao/Desktop/UCLA/430/Project/1/new.csv", header = TRUE)
```

The dataset is about house prices in Beijing from Kaggle.

```
# load the libraries
library(Boruta)
library(VIM)
library(leaps)
library(car)
library(caret)
library(corrplot)
library(caTools)
library(lubridate) # deal with date
library(lmtest)
library(effects)
library(margins)
```

Since the dataset has 318,851 rows which are too large for R to run the whole dataset, I randomly pick out about 10,000 rows for this project.

```
set.seed(123)
split <- sample.split(hp, SplitRatio = 0.02)
df <- subset(hp, split == TRUE)
```

We should do some data cleaning before regression. Firstly, I correct a naming error due to the translation. Then I convert the numerical but categorical variables to character variables and corret data types of some variables.

```
str(df)
```

```
## 'data.frame':    11809 obs. of  27 variables:
##  $ url                : chr  "https://bj.lianjia.com/chengjiao/101087957433.html" "https://bj.lianjia
##  $ id                 : chr  "1.01088E+11" "1.01089E+11" "1.01089E+11" "1.01089E+11" ...
##  $ Lng                : num  116 116 116 116 116 ...
##  $ Lat                : num  39.9 40 39.9 40 39.9 ...
##  $ Cid                : num  1.11e+12 1.11e+12 1.11e+12 1.11e+12 1.11e+12 ...
##  $ tradeTime          : chr  "2016/9/17" "2016/8/14" "2016/12/29" "2016/9/12" ...
##  $ DOM                : int  790 533 622 492 437 455 395 366 313 363 ...
##  $ followers          : int  6 114 110 33 59 110 13 233 97 81 ...
##  $ totalPrice         : num  212 610 392 1388 365 ...
##  $ price              : int  32981 103566 83582 53799 25000 47521 51087 65741 24728 101519 ...
##  $ square             : num  64.3 58.9 47 258 146 ...
##  $ livingRoom         : chr  "1" "2" "1" "5" ...
##  $ drawingRoom        : chr  "0" "1" "0" "2" ...
##  $ kitchen            : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ bathRoom           : int  1 1 1 3 2 2 1 1 1 1 ...
##  $ totalFloor         : chr  "19" "6" "17" "6" ...
##  $ floor              : chr  "Middle" "Middle" "Middle" "Top" ...
##  $ buildingType       : num  1 4 1 4 4 4 3 3 4 3 ...
##  $ constructionTime   : int  2009 1989 2003 1998 2000 2004 2007 2008 2001 1993 ...
##  $ renovationCondition: int  3 4 4 4 3 4 4 4 3 4 ...
##  $ buildingStructure  : int  6 2 6 2 2 4 6 6 2 2 ...
##  $ ladderRatio        : num  0.111 0.333 0.4 0.5 0.5 0.5 0.154 0.667 0.5 0.167 ...
##  $ elevator           : int  1 0 1 0 0 0 1 1 0 1 ...
##  $ fiveYearsProperty  : int  0 0 0 0 1 1 0 0 1 1 ...
##  $ subway             : int  1 1 1 0 0 0 1 1 0 0 ...
##  $ district           : int  8 8 7 8 12 6 7 8 4 10 ...
##  $ communityAverage   : int  70141 114107 83273 85111 36358 50000 62834 104613 35885 129541 ...
```

```
# Correct for the names of variables which are confusing due to translation
names(df)[names(df) == 'livingRoom'] <- 'bedRoom'
names(df)[names(df) == 'drawingRoom'] <- 'livingRoom'

# convert data types
df[, c("district", "renovationCondition")] <- data.frame(apply(df[, c("district", "renovationCondition")
                                                                  2, as.character))

df[, c("bedRoom", "livingRoom", "kitchen", "totalFloor")] <- apply(df[, c("bedRoom", "livingRoom",
                                                                          "kitchen", "totalFloor")], 2,
                                                                   as.numeric)
```

```
## Warning in apply(df[, c("bedRoom", "livingRoom", "kitchen", "totalFloor")], :
## NAs introduced by coercion
```

```
## Warning in apply(df[, c("bedRoom", "livingRoom", "kitchen", "totalFloor")], :
## NAs introduced by coercion

## Warning in apply(df[, c("bedRoom", "livingRoom", "kitchen", "totalFloor")], :
## NAs introduced by coercion
```

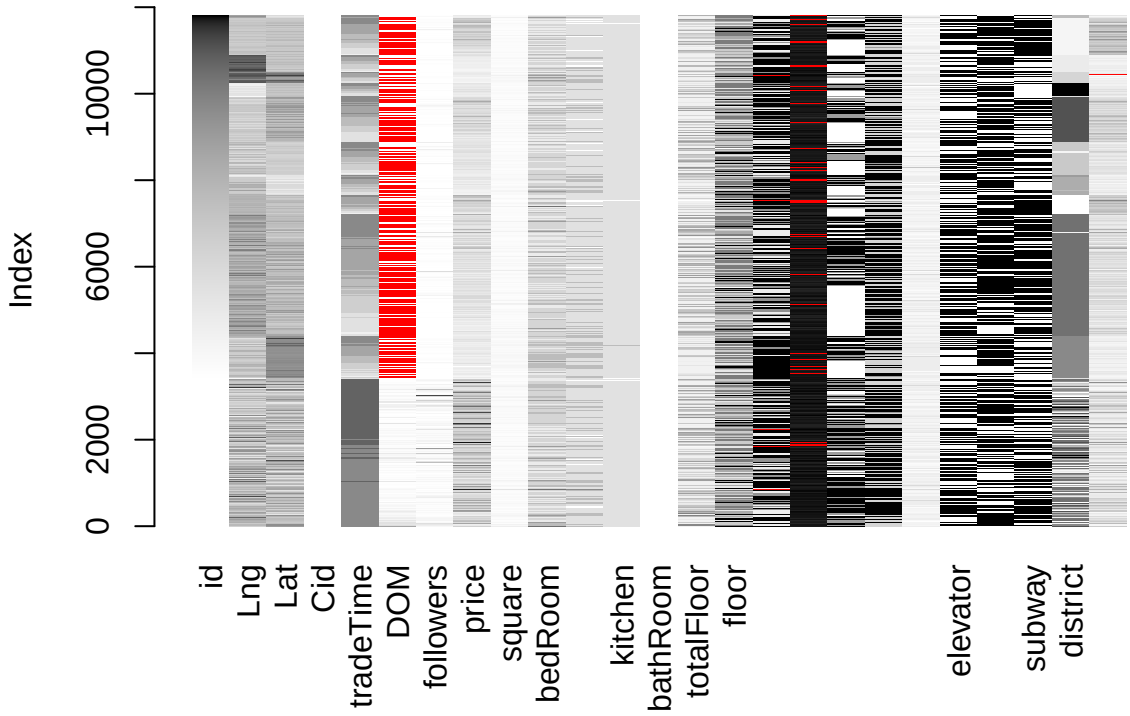
```
df <- subset(df, select = -c(url, totalPrice)) # remove the variable "url" and "totalPrice"

df[, 5] <- year(df[, 5]) # use years of trading times to do the regression instead of exact day
```

1. Variable Selection

(a) Boruta Algorithm

- Dealing with Missing Values

```
matrixplot(df) # look at the pattern of missing values
```



There are quite significant number of missing value in the variable DOM and serveral missing values in building type and construction time. I think we can directly remove the variable DOM and further check the missing values in other variables to decide how to deal with.

```
# count the number of NA in every variable
count_na <- function(x) {
  return(sum(is.na(x)))
```

```
}
apply(df, 2, count_na)
```

```
##                  id                 Lng                 Lat                 Cid
##                   0                   0                   0                   0
##           tradeTime                 DOM           followers               price
##                   0                5875                   0                   0
##              square             bedRoom          livingRoom             kitchen
##                   0                   1                   1                   0
##            bathRoom          totalFloor               floor        buildingType
##                   0                   1                   0                  78
##    constructionTime renovationCondition   buildingStructure         ladderRatio
##                 685                   0                   0                   0
##            elevator   fiveYearsProperty              subway            district
##                   1                   1                   1                   0
##    communityAverage
##                  21
```

We can found that the number of missing values in other variables are not significant compared to the sample size. And the pattern of missing values are kind of random in the matrixplot. Therefore, we can directly delete these rows containing missing values.

```
df <- df[,c(-6)] # remove the variable "DOM"
df <- df[complete.cases(df),] # remove other missing values

any(is.na(df)) # check if there still have NAs in dataset
```

```
## [1] FALSE
```

We can split the dataset into training data and testing data randomly.

```
split_train <- sample.split(df, SplitRatio = 0.7)
df_train <- subset(df, split_train == TRUE)
df_test <- subset(df, split_train == FALSE)
```

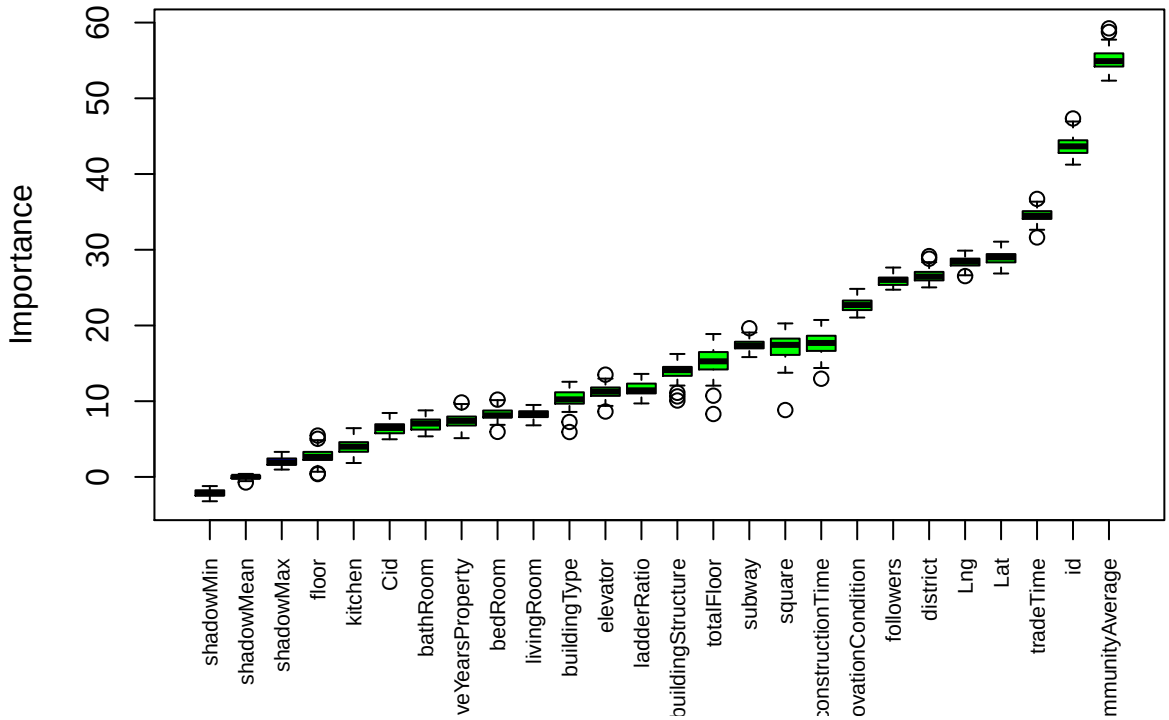
- Boruta Algorithm

```
set.seed(1)
boruta <- Boruta(price ~ ., data = df_train)
print(boruta)
```

```
## Boruta performed 53 iterations in 9.389359 mins.
##  23 attributes confirmed important: bathRoom, bedRoom,
## buildingStructure, buildingType, Cid and 18 more;
##  No attributes deemed unimportant.
```

We can plot the boruta results to find top 10 important predictors.

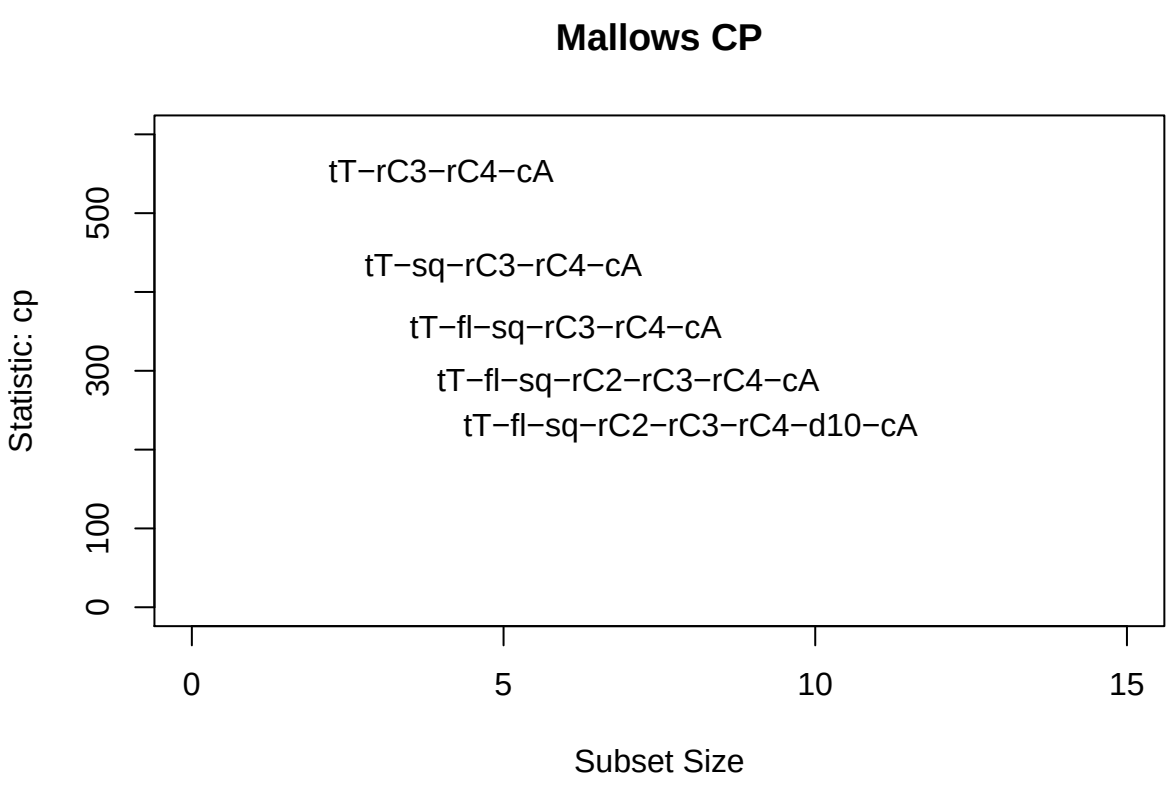
```
plot(boruta, xlab = "", xaxt = "n")

# add the feature labels to the x axis vertically to see it clearly
lz <- lapply(1 : ncol(boruta$ImpHistory), function(i)
boruta$ImpHistory[is.finite(boruta$ImpHistory[, i]), i])
names(lz) <- colnames(boruta$ImpHistory)
Labels <- sort(sapply(lz, median))
axis(side = 1, las=2, labels = names(Labels), at = 1:ncol(boruta$ImpHistory), cex.axis = 0.7)
```



Top 10 predictors selected by Boruta Algorithm: communityAverage, tradeTime, Lat, Lng, district, followers, renovationCondition, square, constructionTime, subway

(b) Mallows CP

```
reg.mod <- lm(price ~ . - id, data = df_train)
ss <- regsubsets(price ~ . - id, method = c("exhaustive"), nbest = 1, data = df_train, really.big=T)
subsets(ss, statistic = "cp", legend = F, main = "Mallows CP", xlim=c(0, 15), ylim=c(0, 600))
```




```
##                          Abbreviation
## Lng                                Ln
## Lat                                Lt
## Cid                                 C
## tradeTime                          tT
## followers                          fl
## square                             sq
## bedRoom                           bdR
## livingRoom                        lvR
## kitchen                             k
## bathRoom                          btR
## totalFloor                         tF
## floorGround                        fG
## floorHigh                          fH
## floorLow                           fL
## floorMiddle                        fM
## floorTop                           fT
## buildingType                       bT
## constructionTime                   cT
## renovationCondition2              rC2
## renovationCondition3              rC3
## renovationCondition4              rC4
## buildingStructure                  bS
## ladderRatio                       ldR
## elevator                            e
## fiveYearsProperty                  fY
```

```
## subway                        sb
## district10                   d10
## district11                   d11
## district12                   d12
## district13                   d13
## district2                     d2
## district3                     d3
## district4                     d4
## district5                     d5
## district6                     d6
## district7                     d7
## district8                     d8
## district9                     d9
## communityAverage              cA
```

Variables choosen by Mallows CP: tradeTime, followers, square, renovationCondition, district, communityAverage

(c) Choice of Predictors

I combine above results and remove the lattitude and longtitude which are not important from my point of view because Beijing is too small to measure by these variables. Therefore, the selected predictors are:

communityAverage: Average price per square of the community.

tradeTime: Trading year of the house.

district: District of the house located in.

followers: Number of people follow the house on webpages.

renovationCondition: Renovation conditions, including other(1), rough(2), Simplicity(3), hardcover(4)

constructionTime: Construction year.

square: square of the house.

subway: Near subway(1) or not(0).

The dependent variable is price per square of the house.

```
var <- c("price", "communityAverage", "tradeTime", "district", "followers", "renovationCondition", "cons
train <- subset(df_train, select = var)
```

2. Univariate Analysis

(a) Descriptive Analysis

- Summary Statistics

```
psych::describe(train)
```

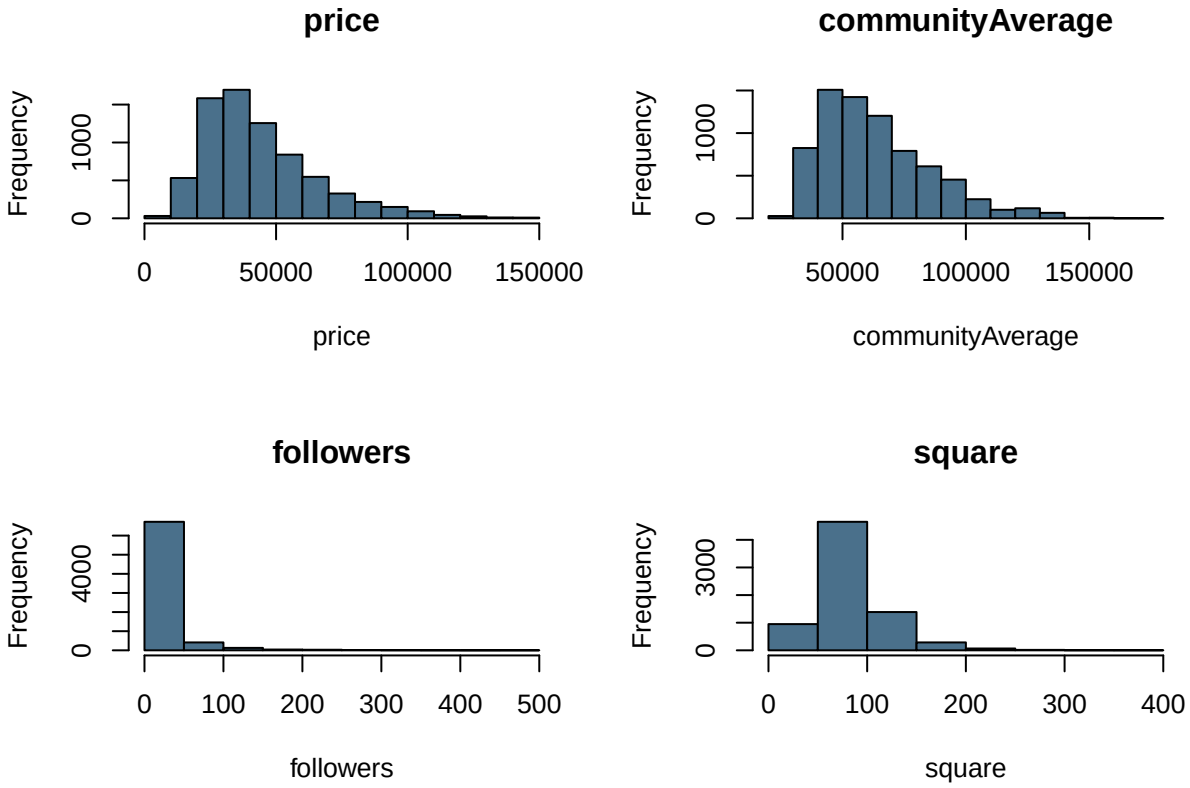
```
##                      vars    n     mean       sd   median  trimmed      mad
## price                   1 7362 43948.86 21744.08 39193.50 41301.40 18405.74
## communityAverage        2 7362 63845.16 22149.15 59366.00 61505.49 21036.61
## tradeTime               3 7362  2014.81     1.66  2015.00  2014.91     1.48
## district*               4 7362     8.47     3.76    10.50     8.86     2.22
```

```
## followers               5 7362    17.40    35.58     5.00     9.49     7.41
## renovationCondition*    6 7362     2.60     1.31     3.00     2.63     1.48
## constructionTime        7 7362  1999.19     8.77  2001.00  1999.92     8.90
## square                  8 7362    83.10    36.44    73.86    78.39    27.41
## subway                  9 7362     0.62     0.49     1.00     0.65     0.00
##                        min       max     range  skew kurtosis     se
## price                    3 149657.00 149654.00  1.22     1.75 253.42
## communityAverage     20483 171180.00 150697.00  0.99     0.83 258.14
## tradeTime             2010   2018.00      8.00 -0.52    -0.83   0.02
## district*                1     13.00     12.00 -0.84    -0.81   0.04
## followers                0    492.00    492.00  5.04    36.82   0.41
## renovationCondition*     1      4.00      3.00 -0.24    -1.68   0.02
## constructionTime      1954   2016.00     62.00 -0.89     1.04   0.10
## square                  10    399.15    389.15  1.72     5.33   0.42
## subway                   0      1.00      1.00 -0.48    -1.77   0.01
```

The table summary the variables regarding the home sales data in Beijing from 2010 to 2018. The dependent variable, price per square and a predicator, average price per square of community have large standard deviations which truely reflect the home price in Beijing recent years.

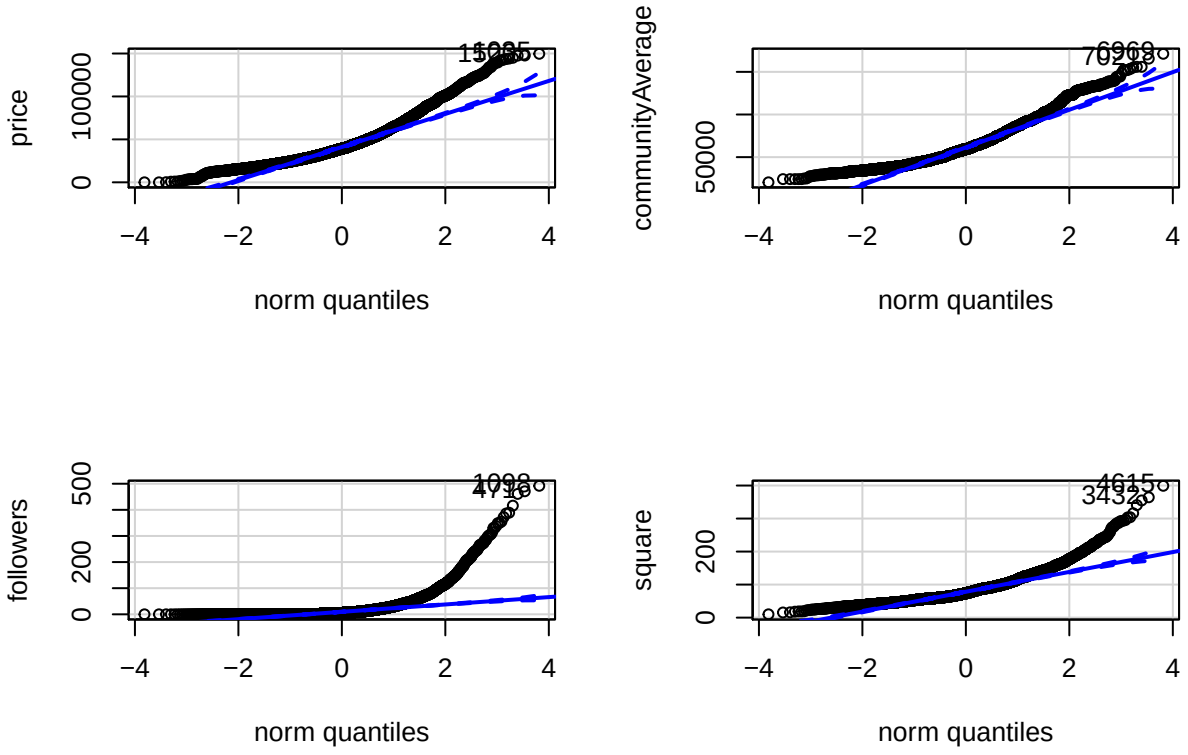
- Continous Variables

```
# Histograms
n <- length(train$price)
k <- 1 + log2(n)

par(mfrow = c(2, 2))
for (i in c(1, 2, 5, 8)) {
  hist(train[, i], breaks = k, col = 'skyblue4', main=names(train)[i], xlab=names(train)[i])
}
```



The histograms shows that the continous variables are left-skew that more obervations in small values and less in larger values. Besides, the histogram of follower has a really high kurotsis between 0 to 100.

```
# qqplots
par(mfrow = c(2, 2))
for (i in c(1, 2, 5, 8)){
  qqPlot(~ train[, i], ylab = names(train)[i])
}
```



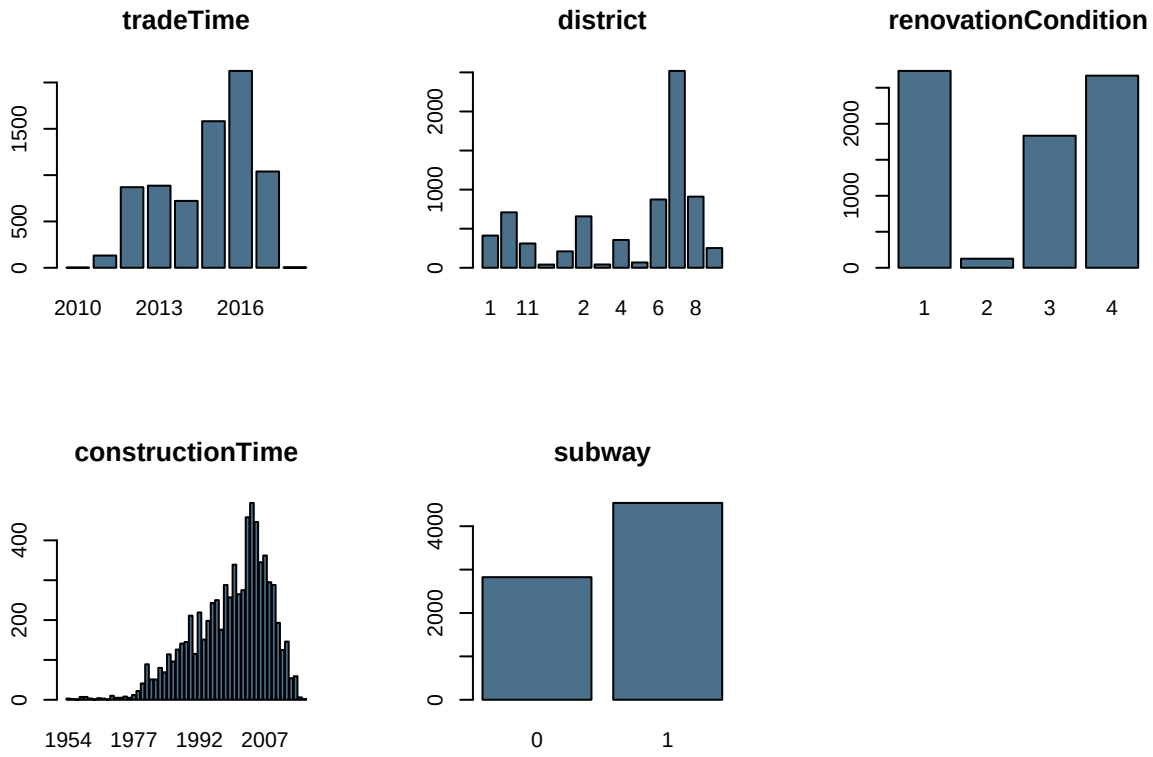
Qq plots indicate that the variables are not perfectly normal distributed and have outlier on the right tail.

```
# corelation plot
corrplot(cor(train[,c(-4,-6)]))
```

We can see that the correlation between denpendent variable and predictors are good and the correlation between predictors are low. So, it is good for us to build model.

- Categorial Variables

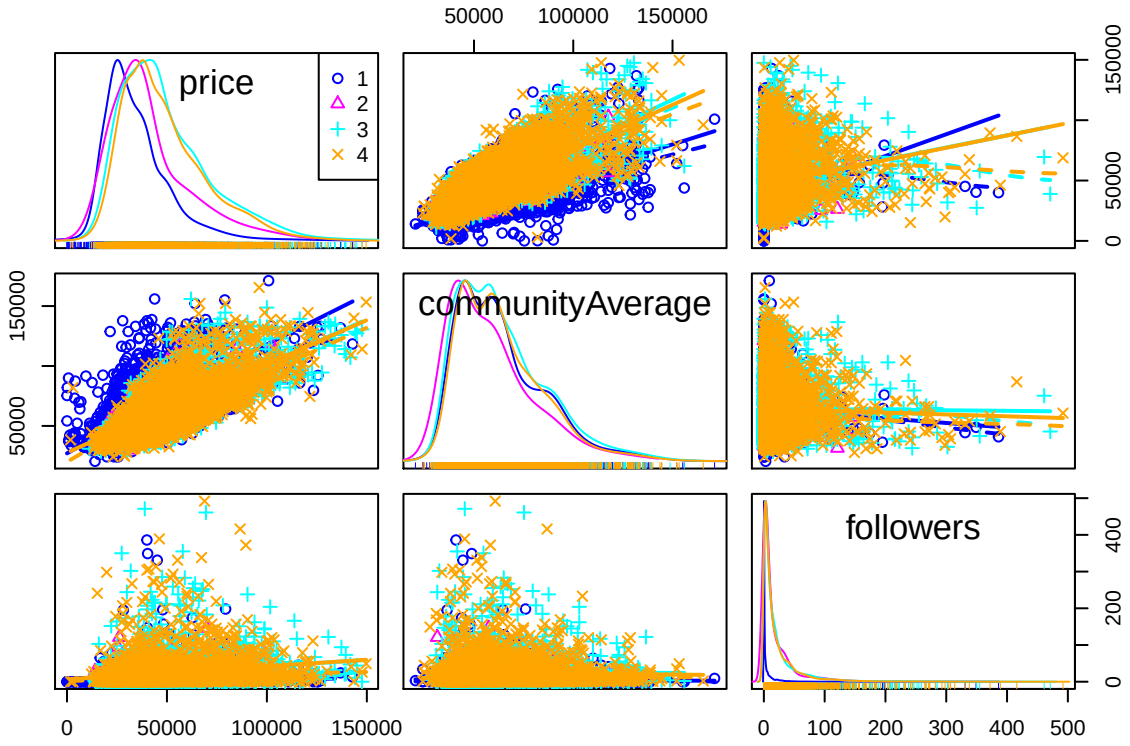
```
# Bar plot
par(mfrow = c(2, 3))
for (i in c(3, 4, 6, 7, 9)) {
  count = table(train[, i])
  barplot(as.numeric(count), names.arg = names(count), col = "skyblue4", main = names(train)[i])
}
```



The bar plots tells us the numer of observations in each categories of each variable.

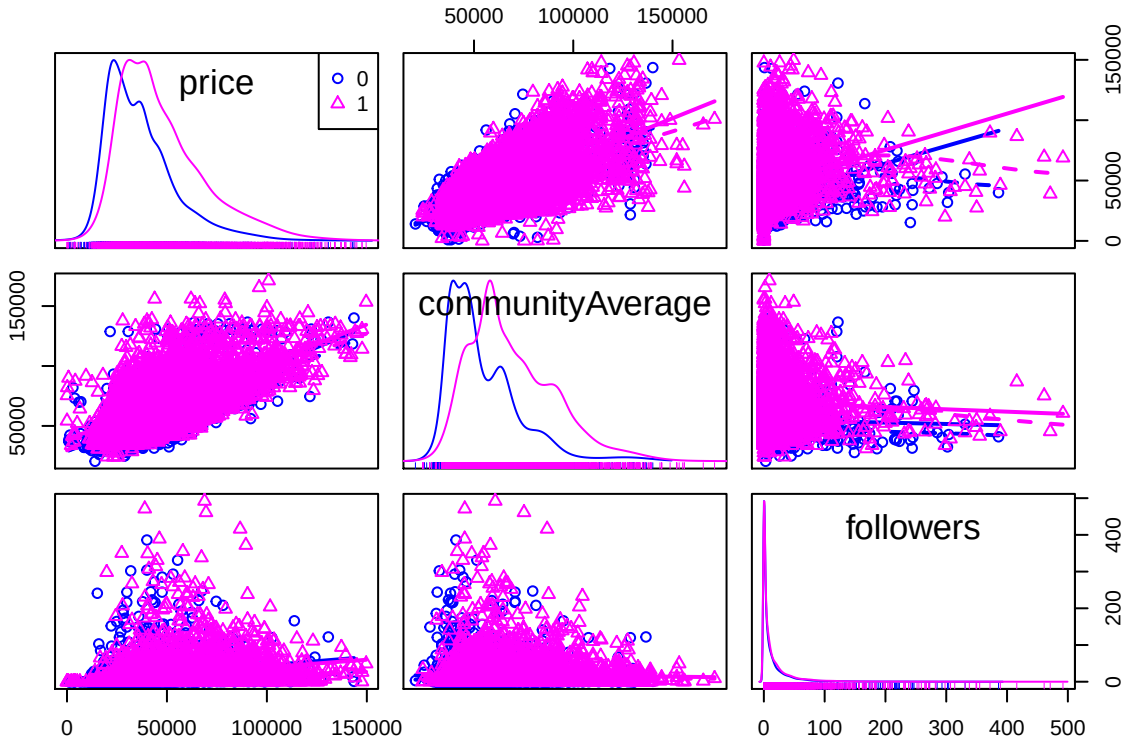
The most frequent traded year of housing in the dataset is 2016, indicating the real estate market is hot in 2015-2016 and then became calm in 2017. The construction of housing peaked in about 2008 to 2010 and declined recent years.The housing in number 8 district transcate more in market than other districts. And more than half of housing sold are near subway.

```
# scatterplot
scatterplotMatrix(~ price + communityAverage + followers | renovationCondition, train)
```



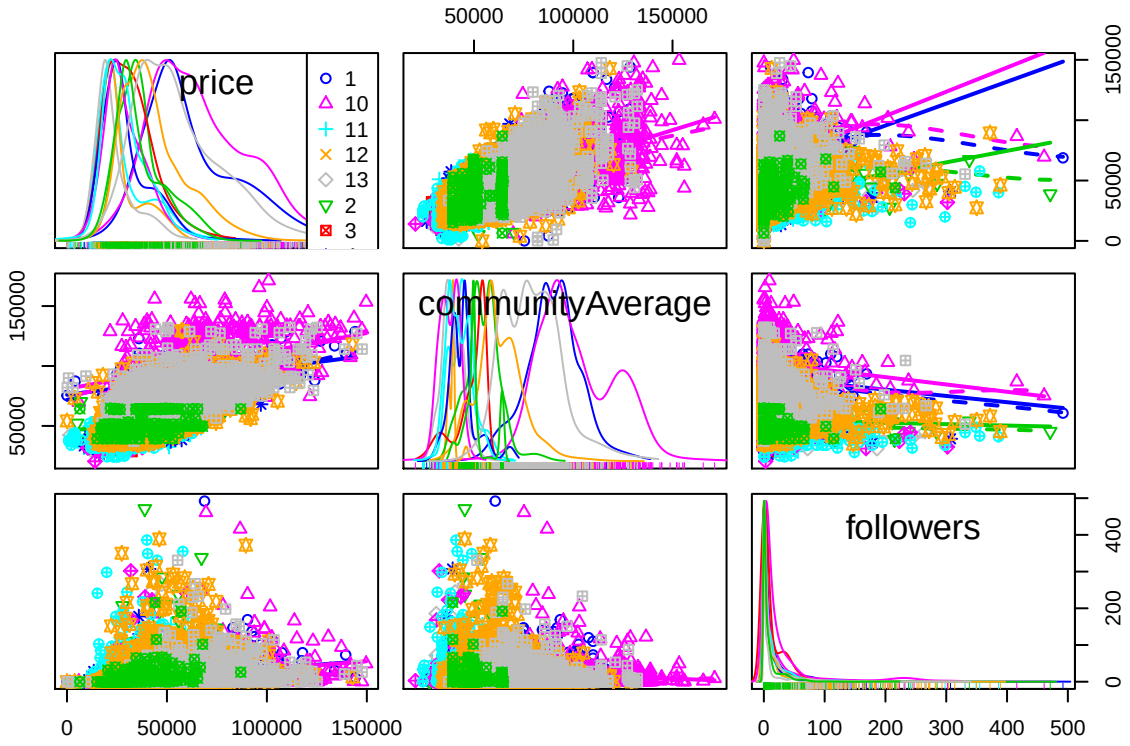
It seems that there is no much difference among the price, followers, community average price of different renovation conditons .

```
scatterplotMatrix(~ price + communityAverage + followers | subway, train)
```



There is significant difference between housing with subway and without subway from the scatter plots matrix.

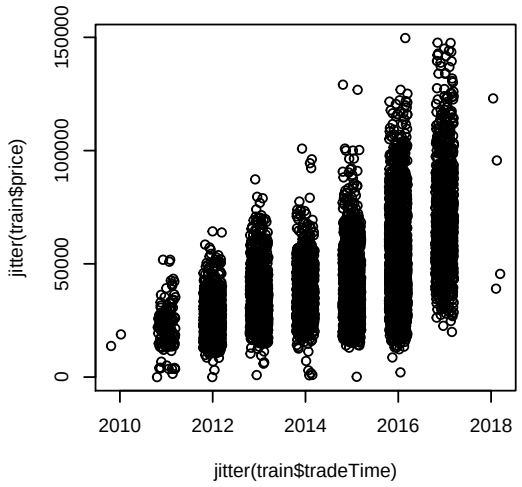
```
scatterplotMatrix(~ price + communityAverage + followers | district, train)
```

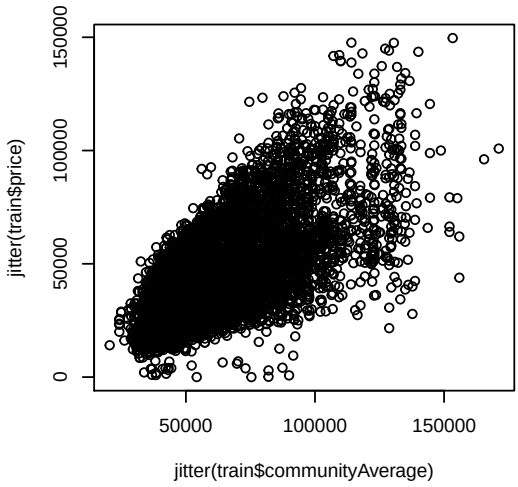


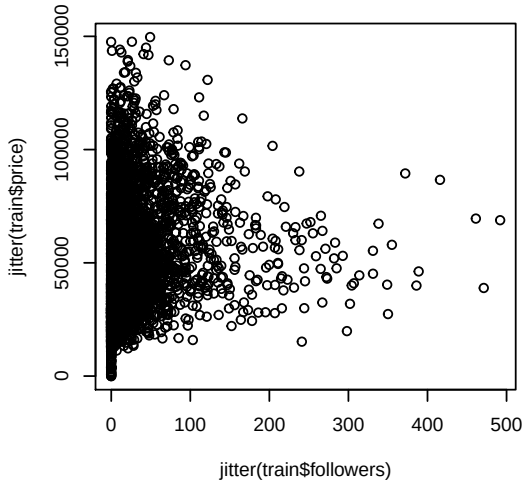
Different district have different prices and community average prices. The number 10 district is the most expensive on average.

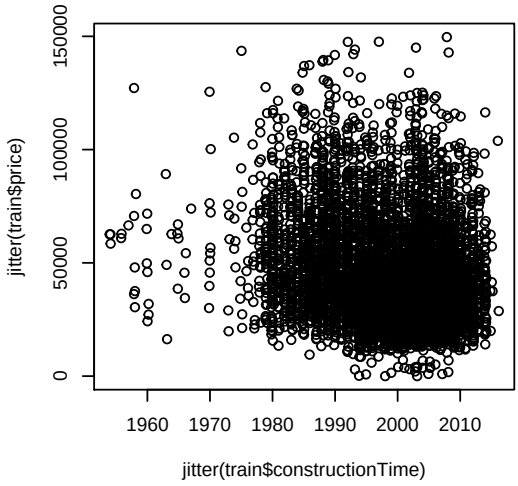
However, there are too much overlap in scatterplots because the large sample size. We can jitter the points to get clear relationship between variables.

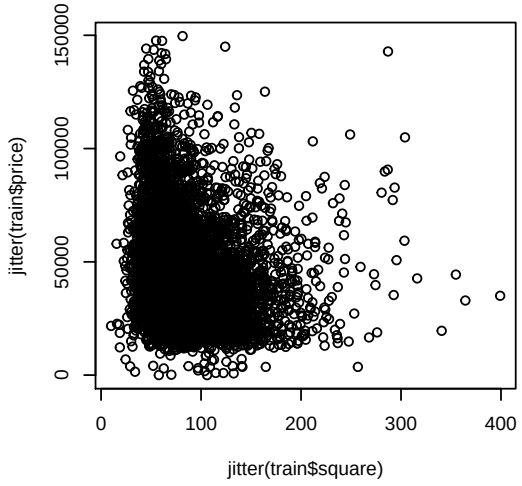
```
par(mfrow = c(3,2))
plot(jitter(train$price) ~ jitter(train$tradeTime))
plot(jitter(train$price) ~ jitter(train$communityAverage))
plot(jitter(train$price) ~ jitter(train$followers))
plot(jitter(train$price) ~ jitter(train$constructionTime))
plot(jitter(train$price) ~ jitter(train$square))
```


jitter(train$price)
0
50000
100000
150000
2010
2012
2014
2016
2018
jitter(train$tradeTime)


jitter(train$price)
0
50000
100000
150000
50000
100000
150000
jitter(train$communityAverage)


jitter(train$price)
0
50000
100000
150000
0
100
200
300
400
500
jitter(train$followers)


jitter(train$price)
0
50000
100000
150000
1960
1970
1980
1990
2000
2010
jitter(train$constructionTime)


jitter(train$price)
0
50000
100000
150000
0
100
200
300
400
jitter(train$square)

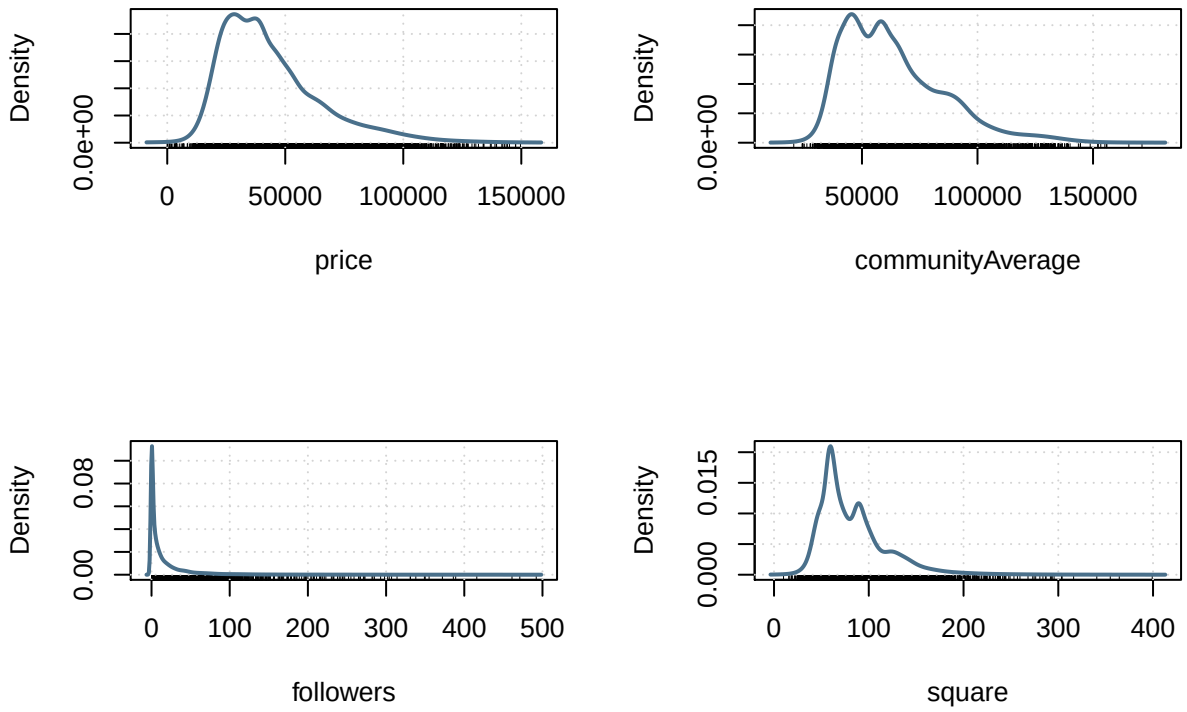
After

jittering the data, we can see more clear relationship between price and predictors.

Price increases with year but have larger variance in recent years. Price has linear regression with community average price but has larger variance as the average price increases. The followers and construction time still have no clear linear relationship with price. Square is negatively linear with price.

(b) Density Plots

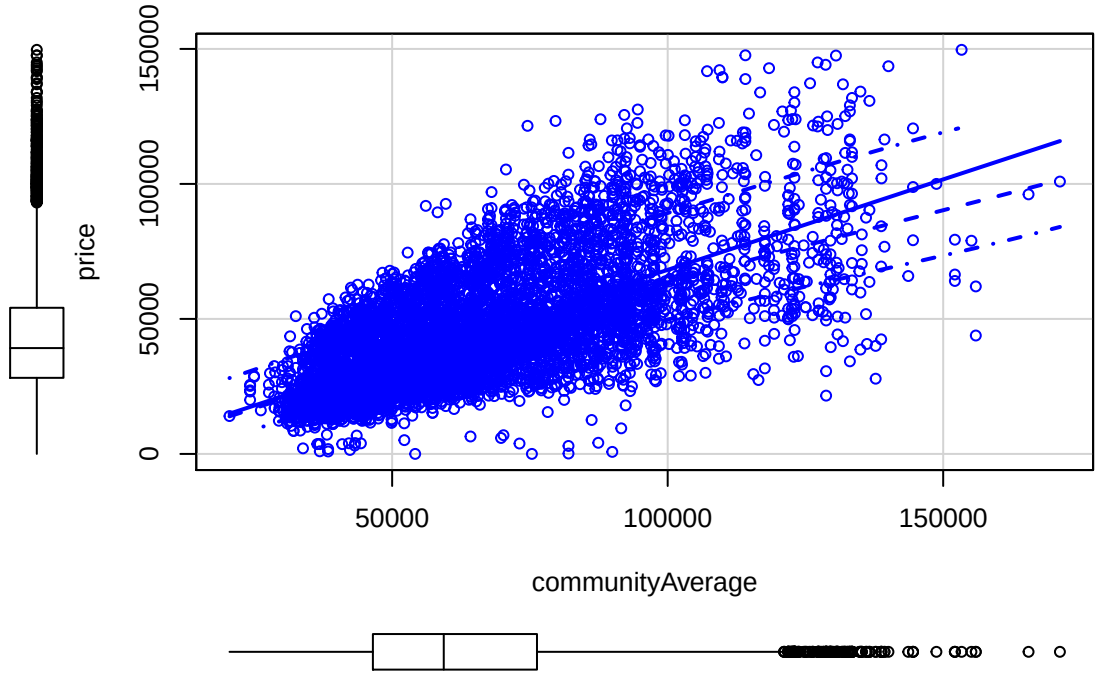
```
par(mfrow = c(2, 2))
for (i in c(1, 2, 5, 8)) {
  densityPlot(train[, i], xlab = names(train)[i], col = "skyblue4")
}
```



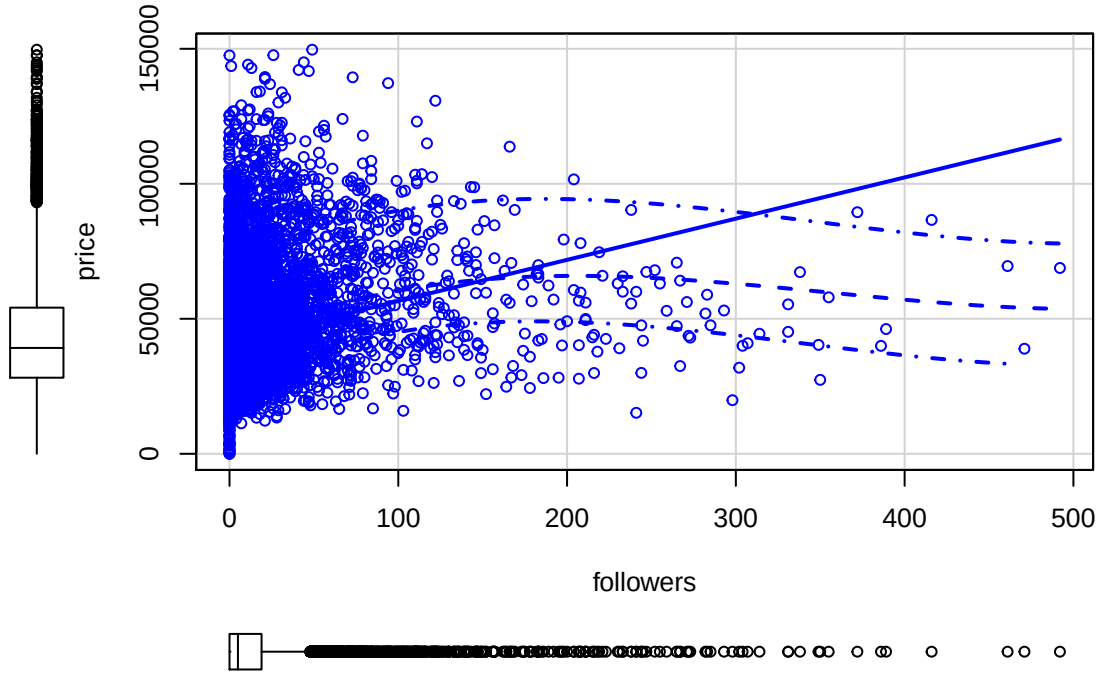
(c) Non-linearities and Transformations

- Identifying Non-linearities

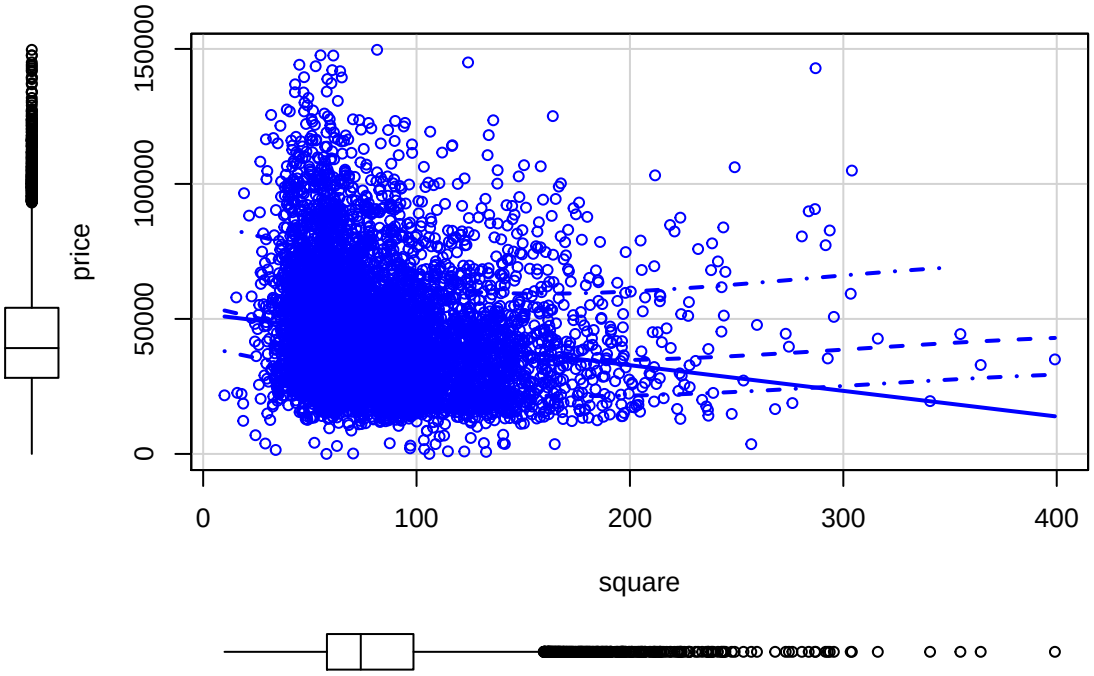
```
# scatterplots
scatterplot(price ~ communityAverage, data = train)
```




```
scatterplot(price ~ followers, data = train)
```




```
scatterplot(price ~ square, data = train)
```



The scatter plots and jitter plots in last parts both show that theses variables are not linear with price so that they need transformations.

- Testing For Transformation

```
# box-cox transform for postitive predictors
summary(p1 <- powerTransform(cbind(price, communityAverage, square) ~ 1, data = train, family = "bcPowe
```

```
## bcPower Transformations to Multinormality
##                   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## price               0.3310        0.33       0.3079       0.3541
## communityAverage   -0.2158       -0.22      -0.2727      -0.1590
## square             -0.2306       -0.23      -0.2752      -0.1861
##
## Likelihood ratio test that transformation parameters are equal to 0
##  (all log transformations)
##                                 LRT df       pval
## LR test, lambda = (0 0 0) 1749.715  3 < 2.22e-16
##
## Likelihood ratio test that no transformations are needed
##                                 LRT df       pval
## LR test, lambda = (1 1 1) 6337.607  3 < 2.22e-16
```

```
# Yj transform for non-positive predictors
summary(p2 <- powerTransform(followers ~ 1, data = train, family = "yjPower")) # followers contains 0
```

```
## yjPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1   -0.1173       -0.12      -0.1343      -0.1003
##
##  Likelihood ratio test that transformation parameter is equal to 0
##                             LRT df       pval
```

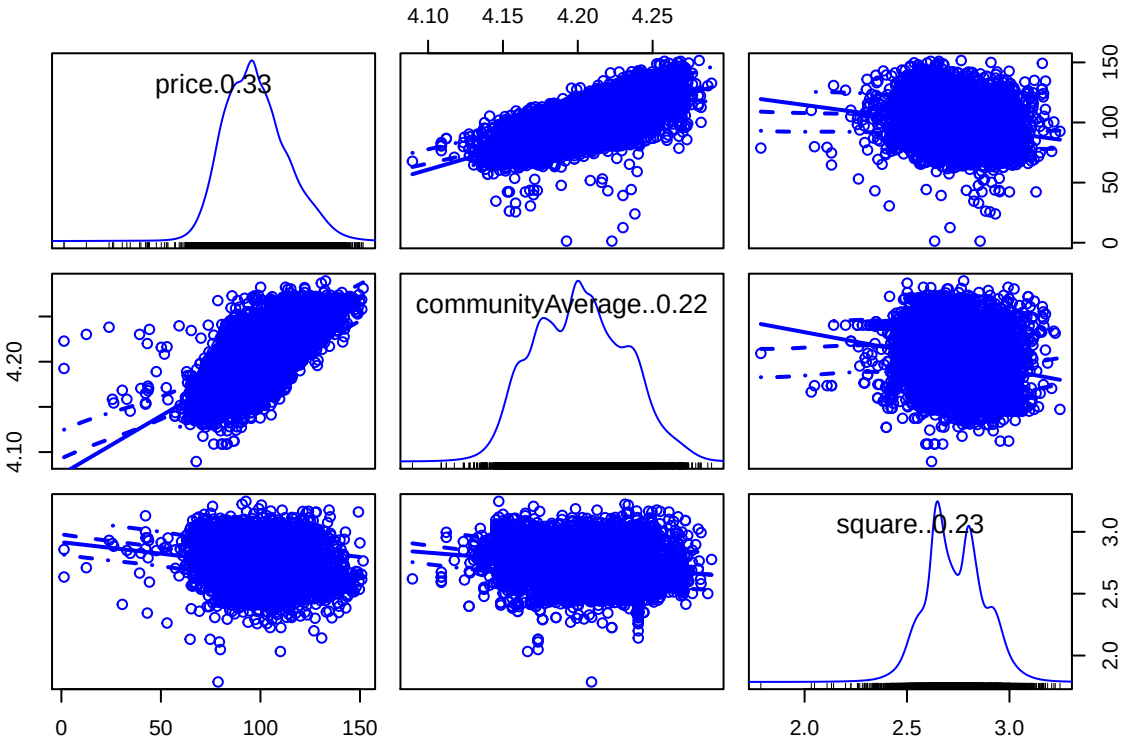
```
## LR test, lambda = (0) 190.5573  1 < 2.22e-16
```

```
testTransform(p1, c(0, -0.22, -0.23))
```

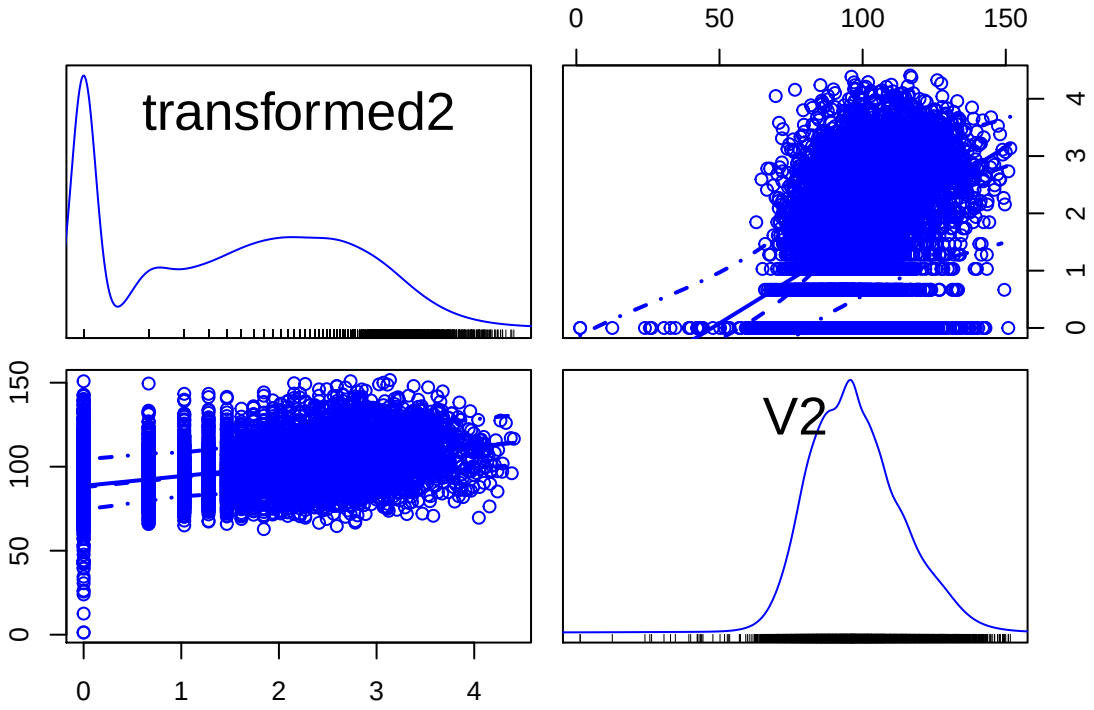
```
##                                           LRT df       pval
## LR test, lambda = (0 -0.22 -0.23) 1559.203  3 < 2.22e-16
```

The powerTransform test fail to reject the null hypothesis, so it need power transform.

```
# bc transform
transformed <- bcPower(with(train, cbind(price, communityAverage, square)), coef(p1, round = T))
scatterplotMatrix(transformed)
```




```
# yj transfrom
transformed2 <- yjPower(train$followers, coef(p2, round = T))
scatterplotMatrix(cbind(transformed2, bcPower(train$price, 0.33)))
```



After transformation, the data points are more spread out instread of concentrated on the left side so that linearity between price and predictors are improved a lot. If we do not transfer the variables before regression and we still use OLS to measure their influence, the residual would have non-constant pattern which is heteroskedasticity. The non-constant variance would mess up the t-value and make our estimate not BLUE.

```
# put the transformed variables into the new dataset
transformed_train = as.data.frame(cbind(transformed, transformed2, data.frame(with(train, cbind(tradeTi

names(transformed_train)[1: 4] = c("price", "communityAverage", "square", "followers")

transformed_train[, c("constructionTime", "tradeTime")] <- data.frame(apply(transformed_train[, c("cons
```

(d) Outliers and/or Unusual Features

```
par(mfrow = c(2, 3))
Boxplot( ~ price, transformed_train)
```

```
##  [1] "3013" "4563" "6080" "5869" "2441" "2508" "6427" "2459" "759"  "5894"
## [11] "1035" "1520" "1728" "1537" "1421" "2154" "218"  "1734" "1702" "1663"
```

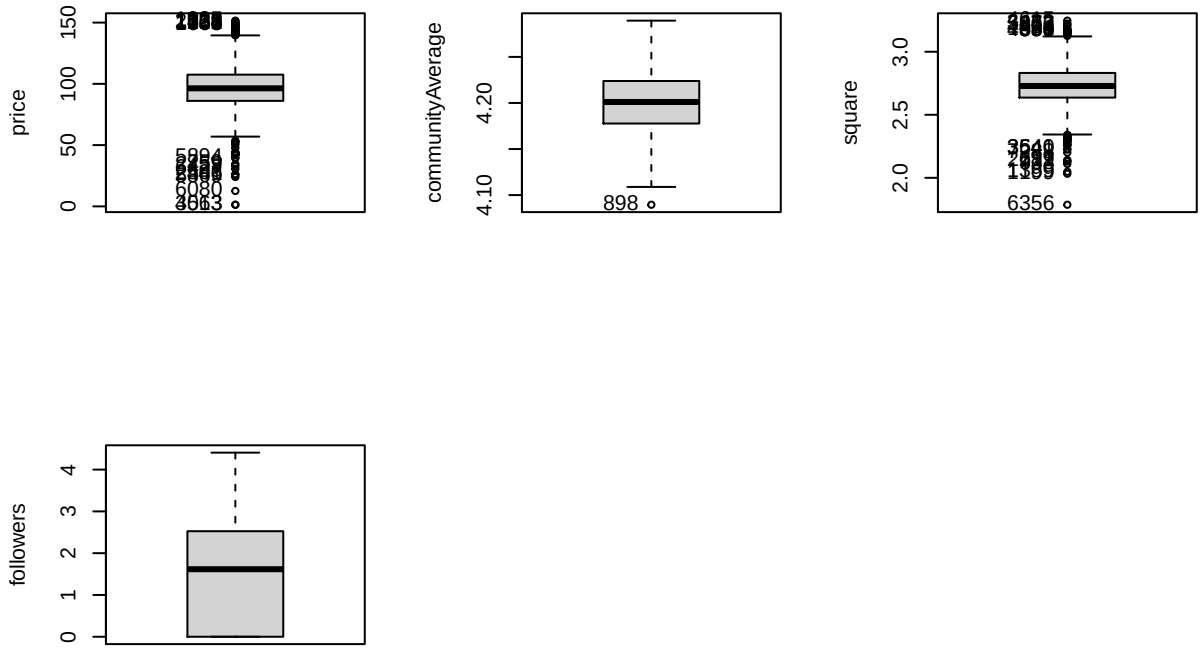
```
Boxplot( ~ communityAverage, transformed_train)
```

```
## [1] "898"
```

```
Boxplot( ~ square, transformed_train)
```

```
##  [1] "6356" "1199" "1389" "933"  "211"  "2689" "192"  "7288" "3541" "3640"
## [11] "4615" "3432" "5676" "857"  "4494" "1381" "676"  "4094" "390"  "589"
```

```
Boxplot( ~ followers, transformed_train)
```



Cause we have a relatively large dataset, the number of outliers is plausible and we can deal with it after building the simple OLS model.

(e) Missing Values

We already done this question in part 1, variable selection because Boruta Algorithm requires no-missing value datasets.

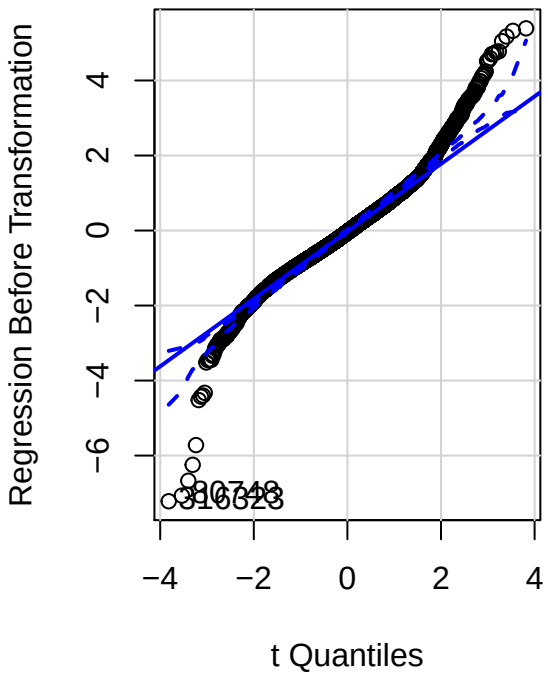
3. Model Building

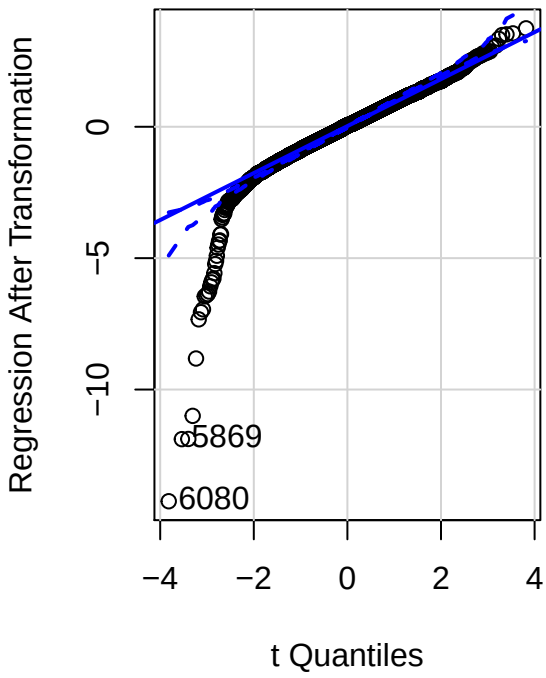
(a) Evaluate Transformations of Variables

```
# qqplot
par(mfrow = c(1,2))
# before transformation
qqPlot(lm(price ~ ., data = train), envelope = 0.99, ylab = "Regression Before Transformation")
```

```
##  80748 316323
##   1932   7311
```

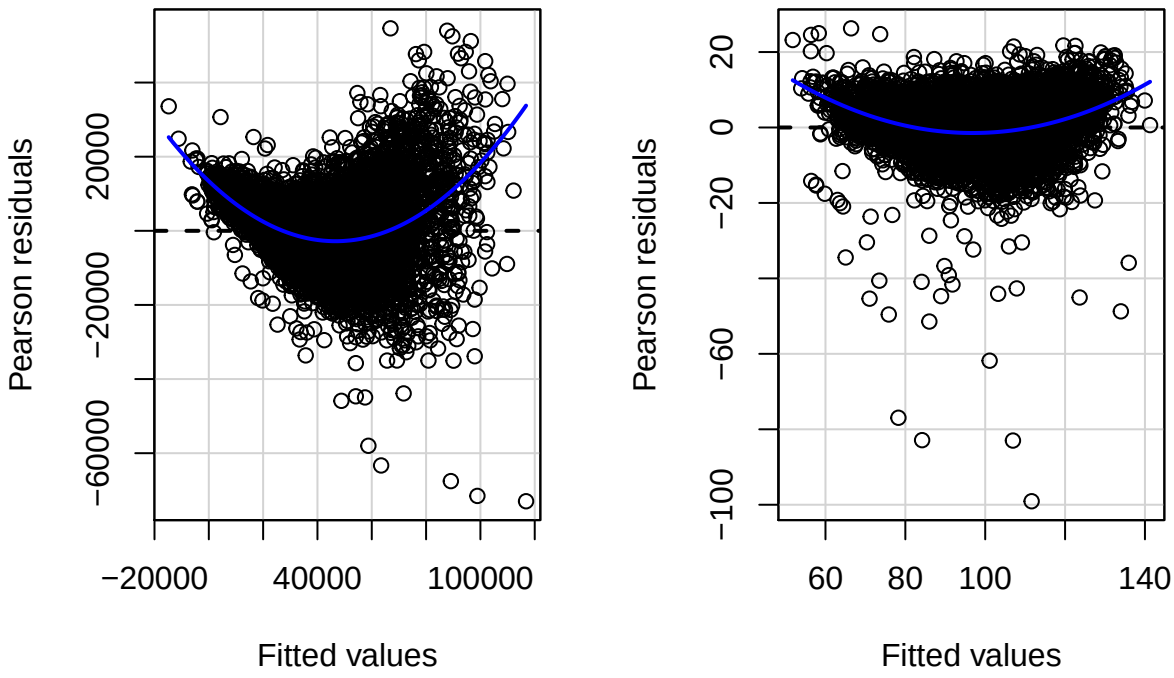
```
# after transformation
qqPlot(lm(price ~ ., data = transformed_train), envelope = 0.99, ylab = "Regression After Transformatio
```






```
## [1] 5869 6080
```

```
# residual
residualPlot(lm(price ~ ., data = train))
residualPlot(lm(price ~ ., data = transformed_train))
```



It is obvious that the fitted value is more normal distributed and heteroskedasticity of residual is mitgated to some degree after transformation. Therefore, our transformation is effective.

(b) Modelling and Improvement

Simple Model with Transformed Variables

```
mod1 <- lm(price ~ ., data = transformed_train)
summary(mod1)
```

```
##
## Call:
## lm(formula = price ~ ., data = transformed_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.072  -4.035   0.287   4.416  26.279
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.361e+04  1.602e+02 -84.963  < 2e-16 ***
## communityAverage      3.481e+02  5.054e+00  68.866  < 2e-16 ***
## square               -6.082e+00  6.243e-01  -9.743  < 2e-16 ***
## followers             6.841e-01  9.537e-02   7.173 8.05e-13 ***
## tradeTime             6.098e+00  7.886e-02  77.337  < 2e-16 ***
## district10            1.240e-01  4.424e-01   0.280 0.779255
```

```
## district11           -7.569e-01  6.280e-01  -1.205 0.228164
## district12            3.758e+00  1.226e+00   3.066 0.002175 **
## district13           -8.951e-01  7.137e-01  -1.254 0.209858
## district2             2.266e-01  4.981e-01   0.455 0.649255
## district3            -2.659e+00  1.182e+00  -2.250 0.024450 *
## district4            -1.099e-01  6.084e-01  -0.181 0.856680
## district5            -1.490e+00  1.022e+00  -1.458 0.144959
## district6            -5.484e-01  5.487e-01  -0.999 0.317610
## district7            -2.045e-01  4.093e-01  -0.500 0.617322
## district8             7.882e-01  4.250e-01   1.854 0.063710 .
## district9             9.705e-01  6.230e-01   1.558 0.119302
## renovationCondition2 -5.493e+00  6.634e-01  -8.280  < 2e-16 ***
## renovationCondition3 -4.479e+00  2.732e-01 -16.392  < 2e-16 ***
## renovationCondition4 -3.973e+00  2.524e-01 -15.745  < 2e-16 ***
## constructionTime     -1.346e-02  1.097e-02  -1.227 0.219892
## subway1               7.234e-01  1.859e-01   3.891 0.000101 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.054 on 7340 degrees of freedom
## Multiple R-squared:  0.8103, Adjusted R-squared:  0.8097
## F-statistic:  1493 on 21 and 7340 DF,  p-value: < 2.2e-16
```

The degree of freedom, adjusted R square and p-value suggest the model performs good. And variables are statistically and economically significant except construction time. We can further re-estimate the model without irrelevent terms.

```
# remove the insignificant predictors and re-estimate the model
mod2 <- lm(price ~ communityAverage + square + followers + tradeTime + district + renovationCondition
           + subway, data = transformed_train)
summary(mod2)
```

```
##
## Call:
## lm(formula = price ~ communityAverage + square + followers +
##     tradeTime + district + renovationCondition + subway, data = transformed_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.016  -4.045   0.287   4.426  26.349
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.362e+04  1.598e+02 -85.219  < 2e-16 ***
## communityAverage      3.480e+02  5.054e+00  68.857  < 2e-16 ***
## square               -6.317e+00  5.941e-01 -10.633  < 2e-16 ***
## followers             6.885e-01  9.531e-02   7.224 5.57e-13 ***
## tradeTime             6.092e+00  7.867e-02  77.432  < 2e-16 ***
## district10            1.542e-01  4.417e-01   0.349  0.72695
## district11           -8.502e-01  6.234e-01  -1.364  0.17268
## district12            3.694e+00  1.224e+00   3.017  0.00256 **
## district13           -9.434e-01  7.127e-01  -1.324  0.18565
## district2             1.595e-01  4.951e-01   0.322  0.74734
## district3            -2.757e+00  1.179e+00  -2.338  0.01940 *
## district4            -1.844e-01  6.053e-01  -0.305  0.76061
```

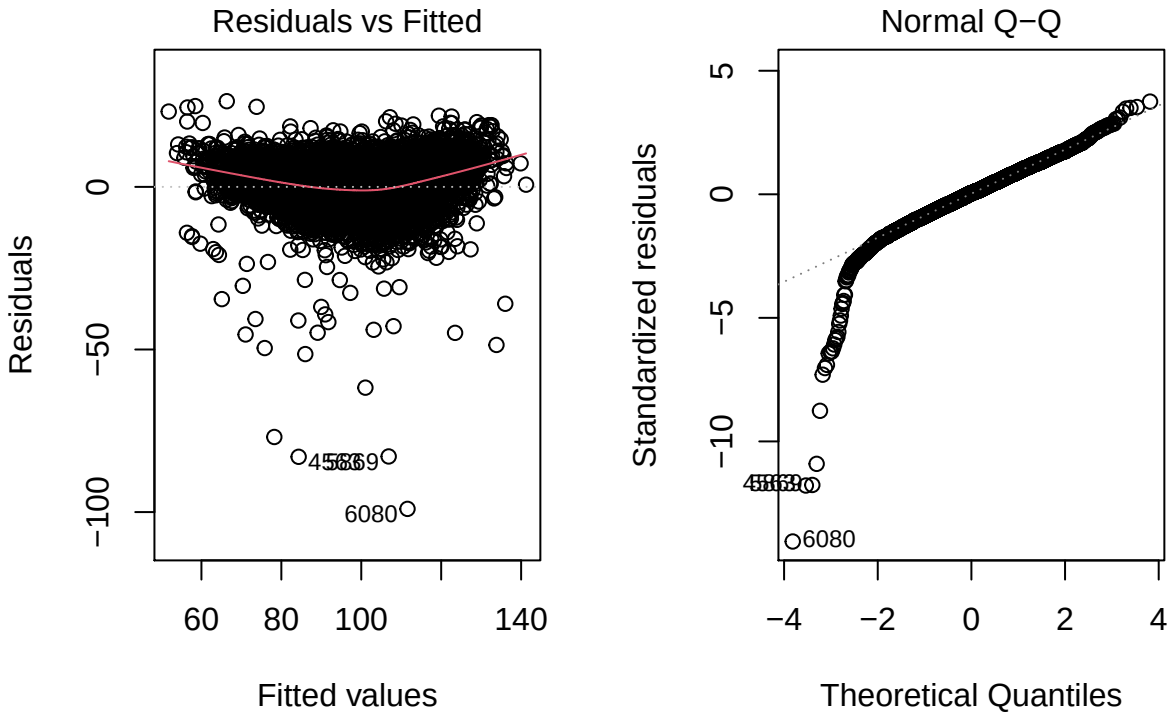
```
## district5            -1.651e+00  1.014e+00  -1.629  0.10333
## district6            -6.185e-01  5.457e-01  -1.133  0.25714
## district7            -2.340e-01  4.086e-01  -0.573  0.56689
## district8             7.729e-01  4.249e-01   1.819  0.06892 .
## district9             9.774e-01  6.230e-01   1.569  0.11672
## renovationCondition2 -5.506e+00  6.633e-01  -8.301  < 2e-16 ***
## renovationCondition3 -4.434e+00  2.708e-01 -16.375  < 2e-16 ***
## renovationCondition4 -3.979e+00  2.523e-01 -15.769  < 2e-16 ***
## subway1               7.408e-01  1.854e-01   3.996 6.51e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.055 on 7341 degrees of freedom
## Multiple R-squared:  0.8102, Adjusted R-squared:  0.8097
## F-statistic:  1567 on 20 and 7341 DF,  p-value: < 2.2e-16
```

After removing the irrelevant terms, the stardard errors of estimates are smaller.

Removing Unusual Observations

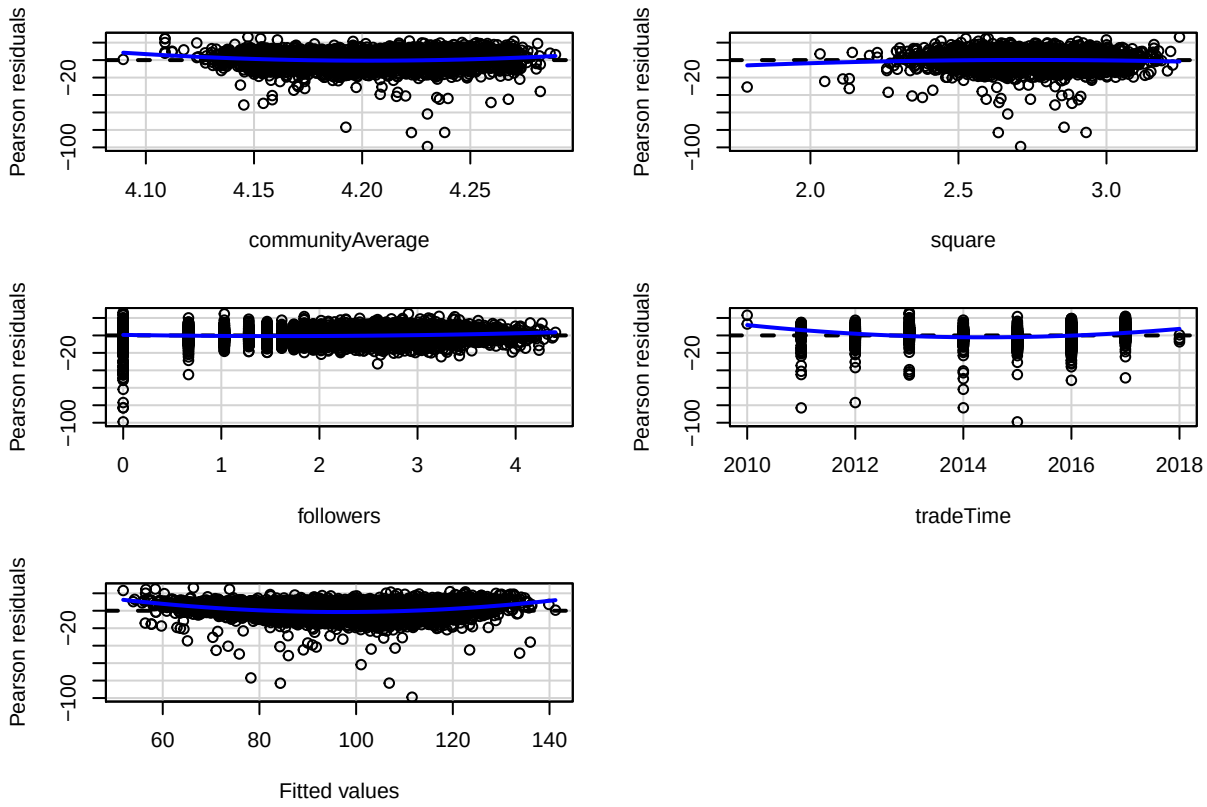
- Outliers

```
par(mfrow = c(1,2))
plot(mod2, 1:2)
```



The residual plot and qq plot suggest that there are some outliers in our model. Besides, the model has heteroskedasticity problems and we would try to mitgate in the later part.

```
residualPlots(mod2)
```




```
##                  Test stat Pr(>|Test stat|)
## communityAverage   12.1453        < 2.2e-16 ***
## square             -2.7226         0.006493 **
## followers           8.5290        < 2.2e-16 ***
## tradeTime          25.1676        < 2.2e-16 ***
## Tukey test         25.7243        < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the residual plot of each variables, the mean of residual is around zero and the spread is almost constant. However, the values are crowed in the middle so that less data points in the left and right sides which makes the model heteroskedasticity.

We can do the outlier test to identify outliers specifically and remove them in the dataset.

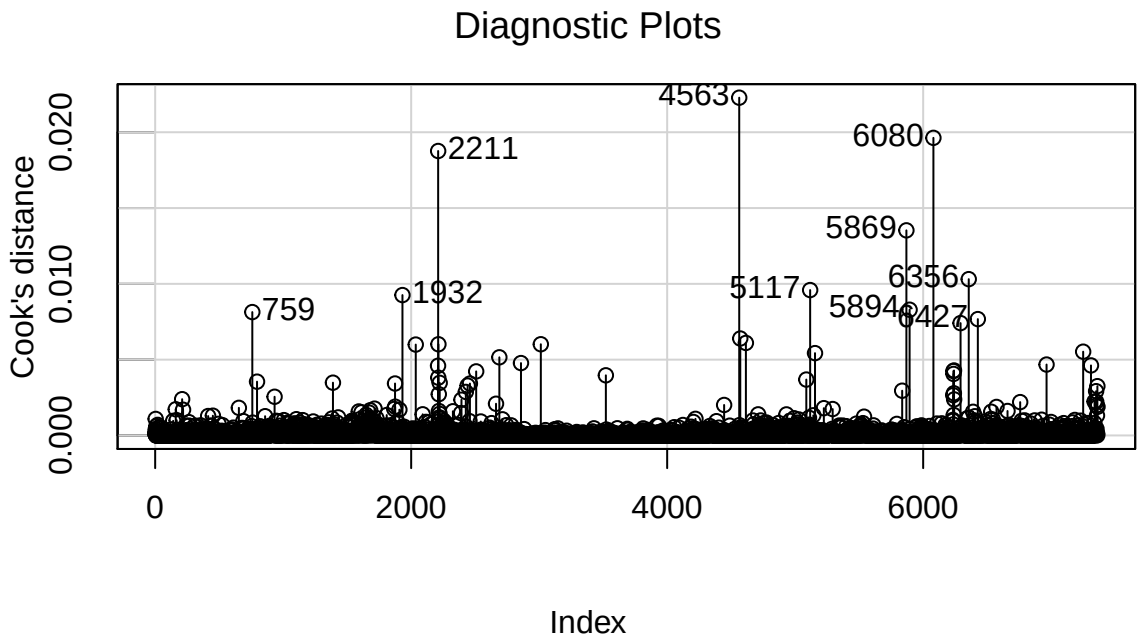
```
outlierTest(mod2)
```

```
##        rstudent unadjusted p-value Bonferroni p
## 6080 -14.241986         2.0178e-45   1.4855e-41
## 4563 -11.897987         2.4013e-32   1.7678e-28
## 5869 -11.874048         3.1807e-32   2.3416e-28
## 3013 -10.996915         6.5250e-28   4.8037e-24
## 5894  -8.807525         1.5764e-18   1.1605e-14
## 759   -7.325939         2.6251e-13   1.9326e-09
## 2508  -7.055154         1.8815e-12   1.3851e-08
## 1932  -6.924994         4.7288e-12   3.4814e-08
```

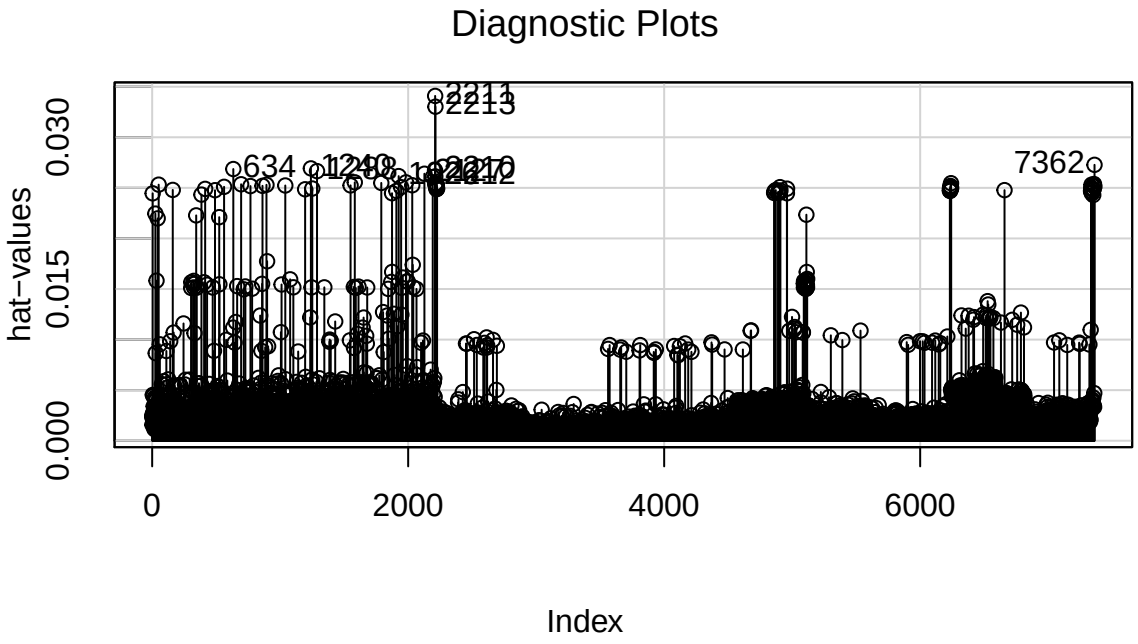
```
## 2441  -6.457033          1.1358e-10   8.3619e-07
## 7250  -6.389517          1.7658e-10   1.3000e-06
```

- Influential Observations Apart from outliers, influential and high leverage observations matter as well. We should identify these data points and remove them to make our model more precise.

```
influenceIndexPlot(mod2, id = list(n = 10), vars = "Cook")
```




```
influenceIndexPlot(mod2, id = list(n = 10), vars = "hat")
```



The model has several influential points and large amount of high leverage points. We can directly delete the influential points but we can not do this to high leverage datas. Since there would be a quite large amout of observations loss if I use the threshold 2P/n, I just delete the points that hat-values are latger than 0.02.

```
# delete unusual observations from dataset
outliers <- c("6080", "4563", "5869", "3013", "5894", "759", "2508", "1932", "2441", "7250")

influentials <- c("759", "1932", "2211", "4564", "5117", "5869", "5894", "6080", "6356", "6247")

hats <- as.character(which(hatvalues(mod2) > 0.02))
transformed_train_noOutliers <- transformed_train[!(rownames(transformed_train)
                                                    %in%  c(outliers, influentials, hats)), ]
```

```
# update the model
mod3 <- update(mod2, data = transformed_train_noOutliers)
summary(mod3)
```

```
##
## Call:
## lm(formula = price ~ communityAverage + square + followers +
##     tradeTime + district + renovationCondition + subway, data = transformed_train_noOutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.955  -4.035   0.215   4.296  26.128
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)          -1.389e+04  1.515e+02 -91.675  < 2e-16 ***
## communityAverage      3.485e+02  4.746e+00  73.435  < 2e-16 ***
## square               -6.490e+00  5.609e-01 -11.571  < 2e-16 ***
## followers             5.531e-01  8.976e-02   6.161 7.59e-10 ***
## tradeTime             6.227e+00  7.460e-02  83.464  < 2e-16 ***
## district10            1.283e-01  4.133e-01   0.310   0.7563
## district11           -9.645e-01  5.836e-01  -1.653   0.0984 .
## district13           -1.078e+00  6.672e-01  -1.615   0.1063
## district2             1.619e-02  4.636e-01   0.035   0.9721
## district4            -3.100e-01  5.667e-01  -0.547   0.5844
## district5            -1.266e+00  9.800e-01  -1.292   0.1965
## district6            -5.659e-01  5.114e-01  -1.107   0.2685
## district7            -3.427e-01  3.827e-01  -0.896   0.3705
## district8             8.895e-01  3.977e-01   2.237   0.0253 *
## district9             9.446e-01  5.841e-01   1.617   0.1058
## renovationCondition2 -6.117e+00  6.367e-01  -9.607  < 2e-16 ***
## renovationCondition3 -4.797e+00  2.554e-01 -18.781  < 2e-16 ***
## renovationCondition4 -4.311e+00  2.383e-01 -18.093  < 2e-16 ***
## subway1               7.652e-01  1.740e-01   4.398 1.11e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.588 on 7241 degrees of freedom
## Multiple R-squared:  0.8302, Adjusted R-squared:  0.8297
## F-statistic:  1966 on 18 and 7241 DF,  p-value: < 2.2e-16
```

Compared with the model with outliers, we have a larger adjusted R square and smaller standard errors.

Model Specification (Nonlinear Predictors)

```
resettest(mod2, power = 2)
```

```
##
##  RESET test
##
## data:  mod2
## RESET = 661.74, df1 = 1, df2 = 7340, p-value < 2.2e-16
```

```
resettest(mod2, power = 3)
```

```
##
##  RESET test
##
## data:  mod2
## RESET = 0.1648, df1 = 1, df2 = 7340, p-value = 0.6848
```

The test results suggest we should include quardractic terms or interactions and do not need to include cubic terms in our model.

```
# test for the quardratic model
mod_reset2 <- lm(price ~ (communityAverage + square + followers + tradeTime + district +
                            renovationCondition + subway)^2, data = transformed_train_noOutliers)
summary(mod_reset2)
```

```
##
## Call:
```

```
## lm(formula = price ~ (communityAverage + square + followers +
##     tradeTime + district + renovationCondition + subway)^2, data = transformed_train_noOutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.085  -3.443   0.127   3.782  28.287
##
## Coefficients: (1 not defined because of singularities)
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             1.728e+05  3.379e+04   5.113 3.25e-07
## communityAverage                       -4.474e+04  7.989e+03  -5.601 2.22e-08
## square                                  8.485e+02  9.358e+02   0.907 0.364602
## followers                               1.506e+02  1.238e+02   1.216 0.223999
## tradeTime                              -8.593e+01  1.678e+01  -5.121 3.12e-07
## district10                              7.546e+02  7.407e+02   1.019 0.308301
## district11                              6.817e+02  1.097e+03   0.621 0.534496
## district13                              1.927e+03  1.185e+03   1.626 0.103953
## district2                               2.765e+03  8.194e+02   3.375 0.000742
## district4                               2.316e+03  9.601e+02   2.413 0.015864
## district5                              -8.667e+02  2.426e+03  -0.357 0.720893
## district6                               3.053e+03  8.988e+02   3.396 0.000686
## district7                               2.818e+03  6.824e+02   4.130 3.68e-05
## district8                               1.814e+03  7.057e+02   2.570 0.010186
## district9                               3.244e+03  1.010e+03   3.212 0.001326
## renovationCondition2                   -9.125e+03  1.352e+03  -6.750 1.59e-11
## renovationCondition3                   -8.270e+03  4.134e+02 -20.005  < 2e-16
## renovationCondition4                   -8.194e+03  3.798e+02 -21.578  < 2e-16
## subway1                                 7.701e+01  3.066e+02   0.251 0.801659
## communityAverage:square                 9.033e+01  2.890e+01   3.125 0.001783
## communityAverage:followers              3.286e+00  4.894e+00   0.671 0.502018
## communityAverage:tradeTime              2.226e+01  3.967e+00   5.611 2.08e-08
## communityAverage:district10             5.371e+01  2.478e+01   2.167 0.030255
## communityAverage:district11            -1.725e+02  3.189e+01  -5.410 6.52e-08
## communityAverage:district13            -1.782e+02  3.841e+01  -4.639 3.56e-06
## communityAverage:district2             -9.157e+01  2.586e+01  -3.542 0.000400
## communityAverage:district4             -6.658e+01  2.992e+01  -2.225 0.026083
## communityAverage:district5             -6.539e+01  7.026e+01  -0.931 0.352050
## communityAverage:district6             -1.303e+02  2.523e+01  -5.163 2.50e-07
## communityAverage:district7             -1.072e+01  2.180e+01  -0.492 0.622964
## communityAverage:district8              8.025e+01  2.410e+01   3.329 0.000875
## communityAverage:district9             -8.267e+01  3.832e+01  -2.157 0.031005
## communityAverage:renovationCondition2 -1.859e+00  3.262e+01  -0.057 0.954562
## communityAverage:renovationCondition3  1.454e+01  1.410e+01   1.032 0.302315
## communityAverage:renovationCondition4  1.381e+01  1.268e+01   1.088 0.276447
## communityAverage:subway1               -7.956e+00  9.285e+00  -0.857 0.391533
## square:followers                       -1.602e+00  5.635e-01  -2.842 0.004491
## square:tradeTime                       -6.118e-01  4.605e-01  -1.329 0.184016
## square:district10                      -3.356e+00  2.680e+00  -1.252 0.210546
## square:district11                      -2.728e+00  3.851e+00  -0.709 0.478599
## square:district13                       3.544e+00  4.239e+00   0.836 0.403100
## square:district2                        4.815e+00  2.995e+00   1.608 0.107899
## square:district4                       -3.559e+00  3.854e+00  -0.923 0.355790
## square:district5                        8.464e+00  1.234e+01   0.686 0.492638
## square:district6                       -1.269e+00  3.250e+00  -0.391 0.696166
```

```
## square:district7                      2.765e+00  2.416e+00   1.145 0.252336
## square:district8                     -2.352e+00  2.585e+00  -0.910 0.362912
## square:district9                      5.644e+00  4.506e+00   1.252 0.210484
## square:renovationCondition2          -1.912e-01  4.333e+00  -0.044 0.964798
## square:renovationCondition3           4.231e-01  1.709e+00   0.248 0.804447
## square:renovationCondition4           2.828e+00  1.511e+00   1.872 0.061231
## square:subway1                       -3.618e+00  1.156e+00  -3.131 0.001749
## followers:tradeTime                  -7.936e-02  6.020e-02  -1.318 0.187443
## followers:district10                  1.242e-01  4.425e-01   0.281 0.778903
## followers:district11                  1.607e+00  6.336e-01   2.536 0.011241
## followers:district13                  2.091e+00  7.338e-01   2.849 0.004393
## followers:district2                  -2.105e-02  4.937e-01  -0.043 0.965994
## followers:district4                   1.177e+00  5.919e-01   1.988 0.046884
## followers:district5                   1.423e+00  9.818e-01   1.450 0.147201
## followers:district6                   1.231e+00  5.480e-01   2.247 0.024681
## followers:district7                   6.421e-01  4.206e-01   1.527 0.126867
## followers:district8                   4.111e-01  4.349e-01   0.945 0.344563
## followers:district9                   9.802e-01  6.397e-01   1.532 0.125485
## followers:renovationCondition2       -1.166e+00  5.833e-01  -1.999 0.045607
## followers:renovationCondition3       -8.077e-01  2.349e-01  -3.438 0.000589
## followers:renovationCondition4       -7.163e-01  2.197e-01  -3.260 0.001120
## followers:subway1                     5.974e-02  1.820e-01   0.328 0.742717
## tradeTime:district10                 -4.841e-01  3.637e-01  -1.331 0.183217
## tradeTime:district11                  2.093e-02  5.377e-01   0.039 0.968954
## tradeTime:district13                 -5.948e-01  5.880e-01  -1.012 0.311730
## tradeTime:district2                  -1.189e+00  4.019e-01  -2.957 0.003116
## tradeTime:district4                  -1.008e+00  4.796e-01  -2.101 0.035678
## tradeTime:district5                   5.492e-01  1.189e+00   0.462 0.644222
## tradeTime:district6                  -1.245e+00  4.432e-01  -2.809 0.004984
## tradeTime:district7                  -1.381e+00  3.363e-01  -4.105 4.09e-05
## tradeTime:district8                  -1.066e+00  3.484e-01  -3.059 0.002232
## tradeTime:district9                  -1.446e+00  4.968e-01  -2.910 0.003625
## tradeTime:renovationCondition2        4.533e+00  6.603e-01   6.865 7.22e-12
## tradeTime:renovationCondition3        4.072e+00  2.033e-01  20.032  < 2e-16
## tradeTime:renovationCondition4        4.033e+00  1.859e-01  21.691  < 2e-16
## tradeTime:subway1                    -1.680e-02  1.510e-01  -0.111 0.911449
## district10:renovationCondition2      -3.050e+00  3.444e+00  -0.886 0.375882
## district11:renovationCondition2      -4.114e+00  4.176e+00  -0.985 0.324663
## district13:renovationCondition2      -4.988e+00  4.316e+00  -1.156 0.247857
## district2:renovationCondition2       -1.122e+00  4.015e+00  -0.279 0.779887
## district4:renovationCondition2       -6.294e+00  3.849e+00  -1.635 0.102036
## district5:renovationCondition2               NA         NA      NA       NA
## district6:renovationCondition2       -2.619e+00  3.833e+00  -0.683 0.494561
## district7:renovationCondition2       -3.821e+00  3.195e+00  -1.196 0.231799
## district8:renovationCondition2       -2.050e+00  3.359e+00  -0.610 0.541645
## district9:renovationCondition2       -4.169e+00  4.464e+00  -0.934 0.350370
## district10:renovationCondition3       6.610e-01  1.171e+00   0.565 0.572419
## district11:renovationCondition3      -8.987e-01  1.856e+00  -0.484 0.628220
## district13:renovationCondition3      -1.494e-01  1.970e+00  -0.076 0.939526
## district2:renovationCondition3       -4.012e-01  1.329e+00  -0.302 0.762837
## district4:renovationCondition3       -5.299e-01  1.620e+00  -0.327 0.743615
## district5:renovationCondition3        4.973e+00  3.930e+00   1.265 0.205790
## district6:renovationCondition3       -4.093e-01  1.491e+00  -0.275 0.783678
## district7:renovationCondition3       -1.330e-01  1.096e+00  -0.121 0.903381
```

```
## district8:renovationCondition3        -1.737e-01  1.129e+00  -0.154 0.877762
## district9:renovationCondition3        -2.444e+00  1.642e+00  -1.488 0.136688
## district10:renovationCondition4        7.222e-01  1.138e+00   0.635 0.525627
## district11:renovationCondition4        3.264e-01  1.693e+00   0.193 0.847104
## district13:renovationCondition4       -1.265e+00  1.844e+00  -0.686 0.492811
## district2:renovationCondition4         2.071e-01  1.295e+00   0.160 0.872886
## district4:renovationCondition4        -1.137e-02  1.580e+00  -0.007 0.994262
## district5:renovationCondition4         4.462e+00  3.271e+00   1.364 0.172604
## district6:renovationCondition4         2.826e-01  1.387e+00   0.204 0.838599
## district7:renovationCondition4        -3.753e-02  1.061e+00  -0.035 0.971775
## district8:renovationCondition4        -6.944e-03  1.111e+00  -0.006 0.995012
## district9:renovationCondition4         1.803e-01  1.588e+00   0.114 0.909610
## district10:subway1                     1.581e+00  1.261e+00   1.254 0.209960
## district11:subway1                     1.497e+00  1.458e+00   1.027 0.304591
## district13:subway1                     1.762e+00  1.614e+00   1.092 0.274819
## district2:subway1                      9.418e-01  1.298e+00   0.725 0.468192
## district4:subway1                      7.661e-01  1.435e+00   0.534 0.593548
## district5:subway1                      2.164e+00  2.474e+00   0.875 0.381659
## district6:subway1                      1.990e+00  1.366e+00   1.457 0.145133
## district7:subway1                      6.524e-01  1.206e+00   0.541 0.588701
## district8:subway1                      6.087e-01  1.215e+00   0.501 0.616321
## district9:subway1                      4.437e-01  1.516e+00   0.293 0.769808
## renovationCondition2:subway1          -6.548e-01  1.530e+00  -0.428 0.668714
## renovationCondition3:subway1          -5.299e-01  5.233e-01  -1.013 0.311247
## renovationCondition4:subway1          -3.029e-01  4.783e-01  -0.633 0.526634
##
## (Intercept)                           ***
## communityAverage                      ***
## square
## followers
## tradeTime                             ***
## district10
## district11
## district13
## district2                             ***
## district4                             *
## district5
## district6                             ***
## district7                             ***
## district8                             *
## district9                             **
## renovationCondition2                  ***
## renovationCondition3                  ***
## renovationCondition4                  ***
## subway1
## communityAverage:square               **
## communityAverage:followers
## communityAverage:tradeTime            ***
## communityAverage:district10           *
## communityAverage:district11           ***
## communityAverage:district13           ***
## communityAverage:district2            ***
## communityAverage:district4            *
## communityAverage:district5
```

```
## communityAverage:district6            ***
## communityAverage:district7
## communityAverage:district8            ***
## communityAverage:district9            *
## communityAverage:renovationCondition2
## communityAverage:renovationCondition3
## communityAverage:renovationCondition4
## communityAverage:subway1
## square:followers                      **
## square:tradeTime
## square:district10
## square:district11
## square:district13
## square:district2
## square:district4
## square:district5
## square:district6
## square:district7
## square:district8
## square:district9
## square:renovationCondition2
## square:renovationCondition3
## square:renovationCondition4           .
## square:subway1                        **
## followers:tradeTime
## followers:district10
## followers:district11                  *
## followers:district13                  **
## followers:district2
## followers:district4                   *
## followers:district5
## followers:district6                   *
## followers:district7
## followers:district8
## followers:district9
## followers:renovationCondition2        *
## followers:renovationCondition3        ***
## followers:renovationCondition4        **
## followers:subway1
## tradeTime:district10
## tradeTime:district11
## tradeTime:district13
## tradeTime:district2                   **
## tradeTime:district4                   *
## tradeTime:district5
## tradeTime:district6                   **
## tradeTime:district7                   ***
## tradeTime:district8                   **
## tradeTime:district9                   **
## tradeTime:renovationCondition2        ***
## tradeTime:renovationCondition3        ***
## tradeTime:renovationCondition4        ***
## tradeTime:subway1
## district10:renovationCondition2
```

```
## district11:renovationCondition2
## district13:renovationCondition2
## district2:renovationCondition2
## district4:renovationCondition2
## district5:renovationCondition2
## district6:renovationCondition2
## district7:renovationCondition2
## district8:renovationCondition2
## district9:renovationCondition2
## district10:renovationCondition3
## district11:renovationCondition3
## district13:renovationCondition3
## district2:renovationCondition3
## district4:renovationCondition3
## district5:renovationCondition3
## district6:renovationCondition3
## district7:renovationCondition3
## district8:renovationCondition3
## district9:renovationCondition3
## district10:renovationCondition4
## district11:renovationCondition4
## district13:renovationCondition4
## district2:renovationCondition4
## district4:renovationCondition4
## district5:renovationCondition4
## district6:renovationCondition4
## district7:renovationCondition4
## district8:renovationCondition4
## district9:renovationCondition4
## district10:subway1
## district11:subway1
## district13:subway1
## district2:subway1
## district4:subway1
## district5:subway1
## district6:subway1
## district7:subway1
## district8:subway1
## district9:subway1
## renovationCondition2:subway1
## renovationCondition3:subway1
## renovationCondition4:subway1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.923 on 7137 degrees of freedom
## Multiple R-squared:  0.8647, Adjusted R-squared:  0.8624
## F-statistic: 373.8 on 122 and 7137 DF,  p-value: < 2.2e-16
```

I select both statistically significant and economica significant terms to add in our model. Trade time is a negative quardratic model that the price of housing firstly increased to a peak then begin to decline. And the interaction between trade time and renoavtion condition also have economic meaning.

```
mod4 <- lm(price ~ communityAverage + square + followers + tradeTime + I(tradeTime^2)
           + district + renovationCondition + subway + tradeTime:renovationCondition,
```

```
          data = transformed_train_noOutliers)
summary(mod4)
```

```
##
## Call:
## lm(formula = price ~ communityAverage + square + followers +
##     tradeTime + I(tradeTime^2) + district + renovationCondition +
##     subway + tradeTime:renovationCondition, data = transformed_train_noOutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.042  -3.620   0.190   3.994  26.441
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.248e+06  1.796e+05   6.948 4.01e-12 ***
## communityAverage                3.499e+02  4.483e+00  78.062  < 2e-16 ***
## square                         -6.939e+00  5.301e-01 -13.092  < 2e-16 ***
## followers                       3.585e-01  8.569e-02   4.183 2.91e-05 ***
## tradeTime                      -1.246e+03  1.784e+02  -6.984 3.12e-12 ***
## I(tradeTime^2)                  3.105e-01  4.428e-02   7.012 2.56e-12 ***
## district10                      1.876e-01  3.903e-01   0.481   0.6308
## district11                     -1.212e+00  5.512e-01  -2.199   0.0279 *
## district13                     -1.118e+00  6.301e-01  -1.774   0.0761 .
## district2                      -8.541e-02  4.377e-01  -0.195   0.8453
## district4                      -4.171e-01  5.351e-01  -0.779   0.4358
## district5                      -1.926e+00  9.258e-01  -2.080   0.0375 *
## district6                      -6.566e-01  4.829e-01  -1.360   0.1740
## district7                      -4.160e-01  3.614e-01  -1.151   0.2498
## district8                       7.275e-01  3.756e-01   1.937   0.0528 .
## district9                       9.707e-01  5.515e-01   1.760   0.0784 .
## renovationCondition2           -6.136e+03  1.280e+03  -4.795 1.66e-06 ***
## renovationCondition3           -5.377e+03  4.609e+02 -11.665  < 2e-16 ***
## renovationCondition4           -5.372e+03  4.270e+02 -12.581  < 2e-16 ***
## subway1                         6.805e-01  1.643e-01   4.141 3.50e-05 ***
## tradeTime:renovationCondition2  3.043e+00  6.349e-01   4.792 1.68e-06 ***
## tradeTime:renovationCondition3  2.666e+00  2.287e-01  11.658  < 2e-16 ***
## tradeTime:renovationCondition4  2.664e+00  2.119e-01  12.576  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.22 on 7237 degrees of freedom
## Multiple R-squared:  0.8487, Adjusted R-squared:  0.8482
## F-statistic:  1845 on 22 and 7237 DF,  p-value: < 2.2e-16
```

We have a higher R square and all estimates of coefficients are statistically significant.

(c) Testing and Mitgating Multicollinearity

```
vif(mod4)
```

```
##                                     GVIF Df GVIF^(1/(2*Df))
## communityAverage              3.501648e+00  1        1.871269
## square                        1.101756e+00  1        1.049646
```

```
## followers                     1.904890e+00  1        1.380177
## tradeTime                     1.640231e+07  1     4049.976533
## I(tradeTime^2)                1.640805e+07  1     4050.685190
## district                      3.762510e+00 10        1.068498
## renovationCondition           1.519389e+20  3     2309.997546
## subway                        1.192666e+00  1        1.092093
## tradeTime:renovationCondition 1.520206e+20  3     2310.204526
```

```
vif(mod3)
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## communityAverage    3.499218  1        1.870620
## square              1.099724  1        1.048677
## followers           1.863148  1        1.364972
## tradeTime           2.558056  1        1.599393
## district            3.744465 10        1.068242
## renovationCondition 2.018268  3        1.124164
## subway              1.192027  1        1.091800
```

Unfortunately, mddel with higher terms has highly collinearity. I found that as long as I include the higher terms in model, there would exist multicollinearity. So, I have to abandon the mod4 and use model 3.

(d) Testing and Mitgating Heteroskedasticity

```
# test heteroskedasticity
bptest(mod3)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mod3
## BP = 183.54, df = 18, p-value < 2.2e-16
```

Combining the residual plots in the former parts, the white test shows the model do have heteroskedasticity. We can use WLS to try to remove the heteroskedasticity.

```
# mitigate the heteroskedasticity
ehatsq <- resid(mod3)^2 # variance of the model

# regression between variance and predictors
sighatsq.ols <- lm(log(ehatsq) ~ log(communityAverage) + log(square) + log(followers + 1)
                   + log(tradeTime) + district + renovationCondition + subway,
                   data = transformed_train_noOutliers)

# weight of the WLS model
vari <- exp(fitted(sighatsq.ols))

# WLS
mod5 <- lm(price ~ communityAverage + square + I(followers + 1) + tradeTime + district
           + renovationCondition + subway, weights = 1 / vari,
           data = transformed_train_noOutliers)
summary(mod5)
```

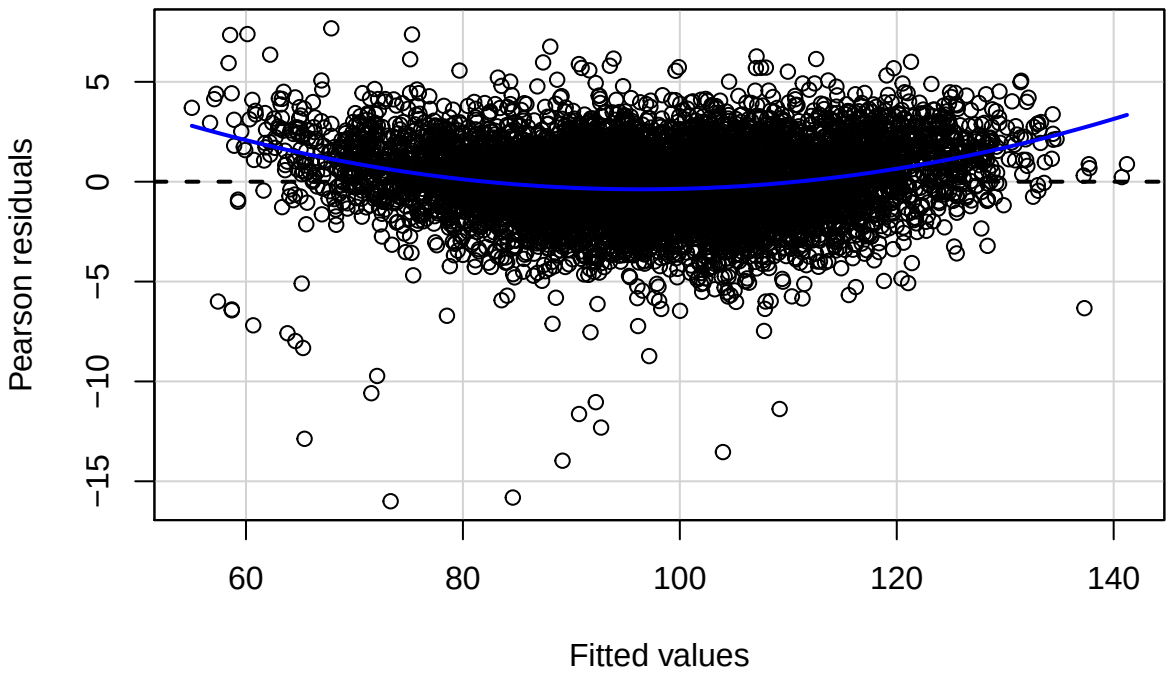
```
##
## Call:
## lm(formula = price ~ communityAverage + square + I(followers +
```

```
##     1) + tradeTime + district + renovationCondition + subway,
##     data = transformed_train_noOutliers, weights = 1/vari)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -15.9990  -1.2291   0.0567   1.2631   7.6776
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.418e+04  1.615e+02 -87.760  < 2e-16 ***
## communityAverage      3.391e+02  4.501e+00  75.343  < 2e-16 ***
## square               -6.261e+00  5.364e-01 -11.673  < 2e-16 ***
## I(followers + 1)      5.684e-01  8.922e-02   6.371 1.99e-10 ***
## tradeTime             6.387e+00  7.968e-02  80.166  < 2e-16 ***
## district10            2.655e-01  4.797e-01   0.554   0.5799
## district11           -1.405e+00  5.733e-01  -2.450   0.0143 *
## district13           -1.325e+00  6.761e-01  -1.961   0.0500 *
## district2             3.539e-01  5.028e-01   0.704   0.4816
## district4            -2.077e-01  5.861e-01  -0.354   0.7230
## district5            -1.272e+00  7.821e-01  -1.627   0.1039
## district6            -4.180e-01  5.470e-01  -0.764   0.4448
## district7            -9.143e-02  4.351e-01  -0.210   0.8336
## district8             9.783e-01  4.472e-01   2.188   0.0287 *
## district9             1.308e+00  6.290e-01   2.080   0.0376 *
## renovationCondition2 -7.915e+00  6.303e-01 -12.557  < 2e-16 ***
## renovationCondition3 -6.591e+00  2.728e-01 -24.158  < 2e-16 ***
## renovationCondition4 -6.033e+00  2.502e-01 -24.113  < 2e-16 ***
## subway1               7.623e-01  1.629e-01   4.680 2.92e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.935 on 7241 degrees of freedom
## Multiple R-squared:  0.8319, Adjusted R-squared:  0.8315
## F-statistic:  1991 on 18 and 7241 DF,  p-value: < 2.2e-16
```

```
bptest(mod5)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mod5
## BP = 183.54, df = 18, p-value < 2.2e-16
```

```
residualPlot(mod5)
```



The residual plot and result of the White test suggest that the WLS model fail to mitigate heteroskedasticity.

We can try GLS model to deal with the heteroskedasticity. The dependent variable is a non-negative continous variable, so I use the gaussian(normal) distribution to be the link function.

```
mod6 <- glm(price ~ communityAverage + square + followers + tradeTime + district
          + renovationCondition + subway, family = "gaussian",
          data = transformed_train_noOutliers)
summary(mod6)
```

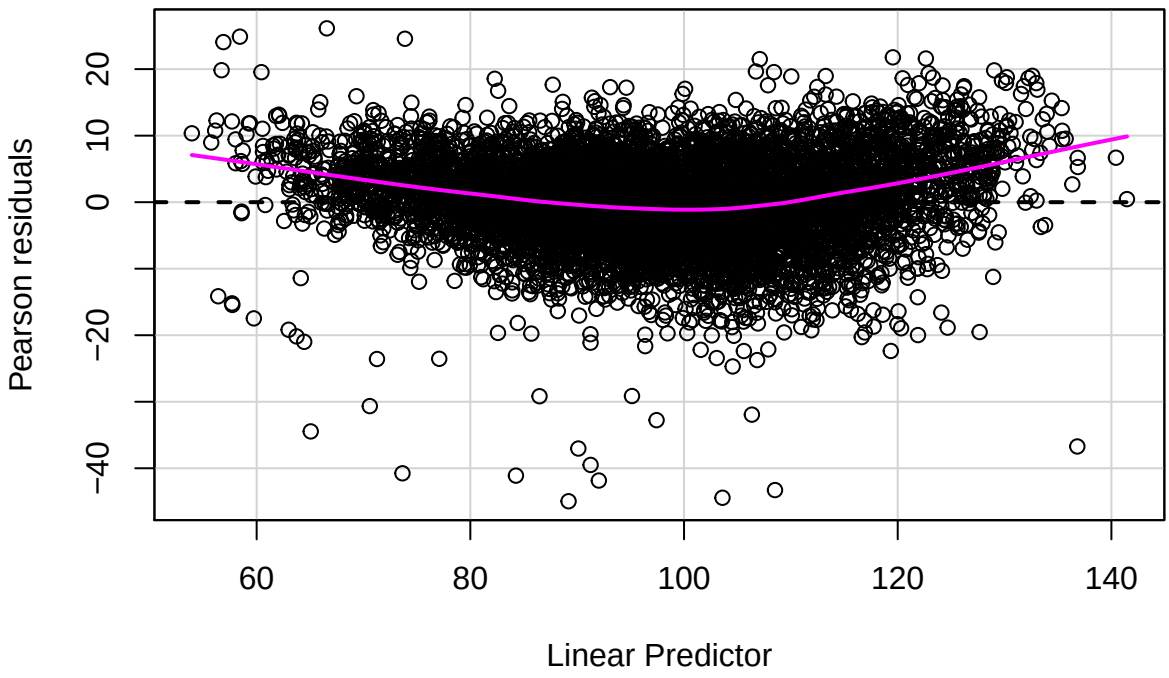
```
##
## Call:
## glm(formula = price ~ communityAverage + square + followers +
##     tradeTime + district + renovationCondition + subway, family = "gaussian",
##     data = transformed_train_noOutliers)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -44.955   -4.035    0.215    4.296   26.128
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.389e+04  1.515e+02 -91.675  < 2e-16 ***
## communityAverage     3.485e+02  4.746e+00  73.435  < 2e-16 ***
## square              -6.490e+00  5.609e-01 -11.571  < 2e-16 ***
## followers            5.531e-01  8.976e-02   6.161 7.59e-10 ***
## tradeTime            6.227e+00  7.460e-02  83.464  < 2e-16 ***
## district10           1.283e-01  4.133e-01   0.310   0.7563
```

```
## district11           -9.645e-01  5.836e-01  -1.653   0.0984 .
## district13           -1.078e+00  6.672e-01  -1.615   0.1063
## district2             1.619e-02  4.636e-01   0.035   0.9721
## district4            -3.100e-01  5.667e-01  -0.547   0.5844
## district5            -1.266e+00  9.800e-01  -1.292   0.1965
## district6            -5.659e-01  5.114e-01  -1.107   0.2685
## district7            -3.427e-01  3.827e-01  -0.896   0.3705
## district8             8.895e-01  3.977e-01   2.237   0.0253 *
## district9             9.446e-01  5.841e-01   1.617   0.1058
## renovationCondition2 -6.117e+00  6.367e-01  -9.607  < 2e-16 ***
## renovationCondition3 -4.797e+00  2.554e-01 -18.781  < 2e-16 ***
## renovationCondition4 -4.311e+00  2.383e-01 -18.093  < 2e-16 ***
## subway1               7.652e-01  1.740e-01   4.398 1.11e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 43.40226)
##
##     Null deviance: 1850514  on 7259  degrees of freedom
## Residual deviance:  314276  on 7241  degrees of freedom
## AIC: 47998
##
## Number of Fisher Scoring iterations: 2
```

```
bptest(mod6)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mod6
## BP = 183.54, df = 18, p-value < 2.2e-16
```

```
residualPlot(mod6)
```



However, the GLS model still have heteroskedasticity.

(e) Model Selection

```
AIC(mod1, mod2, mod3, mod4, mod5, mod6)
```

```
## Warning in AIC.default(mod1, mod2, mod3, mod4, mod5, mod6): models are not all
## fitted to the same number of observations
```

```
##      df      AIC
## mod1 23 49682.08
## mod2 22 49681.59
## mod3 20 47997.88
## mod4 24 47167.82
## mod5 20 47652.79
## mod6 20 47997.88
```

```
BIC(mod1, mod2, mod3, mod4, mod5, mod6)
```

```
## Warning in BIC.default(mod1, mod2, mod3, mod4, mod5, mod6): models are not all
## fitted to the same number of observations
```

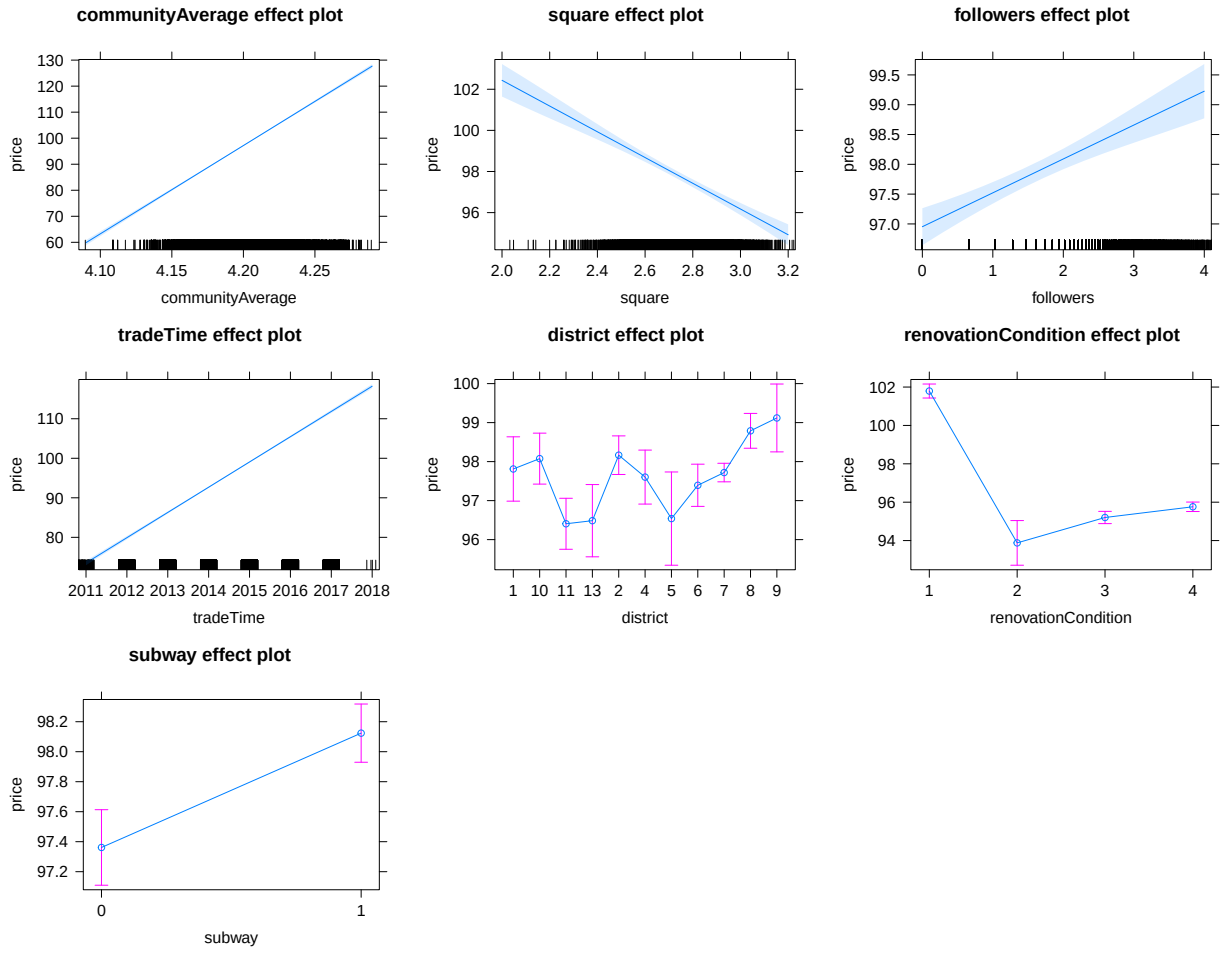
```
##      df      BIC
## mod1 23 49840.87
## mod2 22 49833.48
## mod3 20 48135.68
## mod4 24 47333.18
## mod5 20 47790.59
```

```
## mod6 20 48135.68
```

Considering there is multicollinearity in mod4, we choose mod5 for our final model because it has smallest AIC and BIC values.

(f) Overall Findings of the Preferred Model

```
plot(allEffects(mod5))
```




```
m <- margins(mod5)
summary(m)
```

```
##              factor      AME     SE        z      p    lower    upper
##  communityAverage 339.1171 4.5005  75.3510 0.0000 330.2963 347.9379
##       district10   0.2655 0.4797   0.5535 0.5799  -0.6747   1.2057
##       district11  -1.4048 0.5733  -2.4504 0.0143  -2.5284  -0.2811
##       district13  -1.3255 0.6761  -1.9606 0.0499  -2.6506  -0.0004
##        district2   0.3539 0.5028   0.7038 0.4816  -0.6316   1.3394
##        district4  -0.2077 0.5861  -0.3545 0.7230  -1.3564   0.9409
##        district5  -1.2721 0.7820  -1.6266 0.1038  -2.8049   0.2607
##        district6  -0.4180 0.5470  -0.7642 0.4448  -1.4900   0.6541
##        district7  -0.0914 0.4351  -0.2101 0.8336  -0.9443   0.7614
##        district8   0.9783 0.4472   2.1877 0.0287   0.1018   1.8548
```

```
##             district9   1.3084 0.6290   2.0801 0.0375   0.0756   2.5413
##             followers   0.5684 0.0891   6.3806 0.0000   0.3938   0.7430
##  renovationCondition2  -7.9150 0.6303 -12.5569 0.0000  -9.1504  -6.6796
##  renovationCondition3  -6.5907 0.2728 -24.1577 0.0000  -7.1254  -6.0559
##  renovationCondition4  -6.0331 0.2502 -24.1126 0.0000  -6.5235  -5.5427
##                square  -6.2611 0.5364 -11.6734 0.0000  -7.3123  -5.2099
##               subway1   0.7623 0.1629   4.6801 0.0000   0.4431   1.0816
##             tradeTime   6.3873 0.0797  80.1656 0.0000   6.2311   6.5434
```

Note: Since our predictors and dependent variable in the regression are after transformation, the model is a predictive model instead of causl model to estimate the relationship between variables. The coefficient interpretation is based on the transformed data instead of real data.

Intercept: If all else terms are zero, the fixed cost for the house is 1418.

communityAverage: From economic aspect, communityAverage is an indicator of price, so it is positively linear with price and has a narrow confidence interval. Additional 1 unit of average community housing price is associated with an increase of 339.1171 on average, holding else constant.

square: The estimate is economic significant because when the total square of housing increase, the price per square would be relatively cheap, holding other control variables constant. However, there might be some special cases, the confidence interval is large in small and large regions. The estimate of coefficient indicate that addtional 1 unit of area increase is associated with a decrease of 6.261 on price on average, holding all else constant.

followers: More followers means more attractive the house is. Addtional 1 follower would cause an increase of 0.5684 on price on average, holding all else constant.

tradeTime: In recent years, housing price is continously raising. The house in this year would sell more by 6.377 on average, holding all else constant.

district: Among the districts, the number 8 and 9 district are the most expensive districts and number 11 and 13 district are the most cheap districts. Other districts have no significant difference. The coefficient of the district is the difference of price compared to the baseline, district 1.

renovationCondition: This estimate seems not economic significant because people always prefer hardcover house.

subway: Compared to the house without subway nearing, the houses with subway worth 0.7623 per area on average, holding all else constant.

(g) Model Evaluation

Robustness Test (Boostrapping)

We can use the bootstrap to enhance the robustness of the model.

```
betahat.boot = Boot(mod3, R=100)
usualEsts = summary(mod3)$coef[, 1:2]
summary(betahat.boot)
```

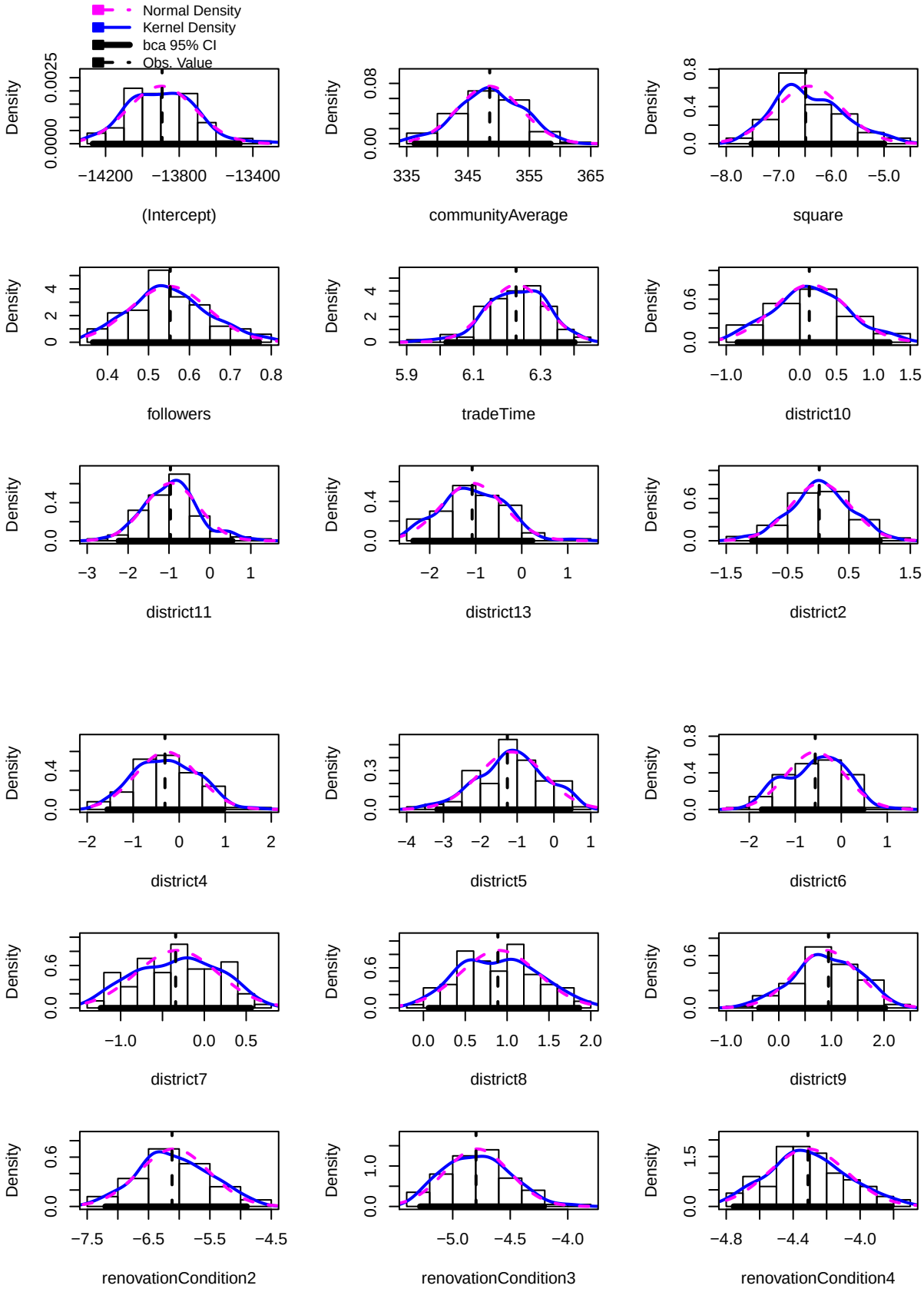
```
##
## Number of bootstrap replications R = 100
##                          original    bootBias     bootSE     bootMed
## (Intercept)          -1.3893e+04 -3.50024132 183.374551 -1.3902e+04
## communityAverage      3.4853e+02 -0.08210554   5.235007  3.4838e+02
## square               -6.4896e+00  0.06749842   0.646184 -6.5673e+00
## followers             5.5305e-01 -0.00313289   0.095139  5.4020e-01
```

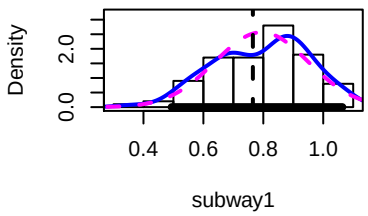
```
## tradeTime             6.2267e+00  0.00180547   0.089012  6.2303e+00
## district10            1.2827e-01  0.01944438   0.497910  1.2199e-01
## district11           -9.6453e-01  0.02221962   0.660374 -9.4834e-01
## district13           -1.0777e+00  0.00555814   0.684012 -1.0940e+00
## district2             1.6194e-02  0.01861828   0.487730  4.3864e-02
## district4            -3.0999e-01  0.02991193   0.670493 -2.8224e-01
## district5            -1.2660e+00  0.11404864   0.898938 -1.1243e+00
## district6            -5.6588e-01  0.00531295   0.635183 -5.2373e-01
## district7            -3.4274e-01  0.01739834   0.491124 -2.7698e-01
## district8             8.8951e-01  0.01424806   0.464618  9.3432e-01
## district9             9.4464e-01 -0.04712359   0.605066  8.4770e-01
## renovationCondition2 -6.1168e+00  0.03430696   0.570687 -6.1276e+00
## renovationCondition3 -4.7974e+00  0.00907557   0.280916 -4.7815e+00
## renovationCondition4 -4.3114e+00 -0.00064883   0.230962 -4.3295e+00
## subway1               7.6518e-01  0.02905427   0.155412  8.1935e-01
```

```
confint(betahat.boot)
```

```
## Bootstrap percent confidence intervals
##
##                              2.5 %        97.5 %
## (Intercept)          -1.426965e+04 -1.347169e+04
## communityAverage      3.362846e+02  3.584594e+02
## square               -7.520817e+00 -4.994477e+00
## followers             3.640734e-01  7.702072e-01
## tradeTime             6.018322e+00  6.401700e+00
## district10           -8.468764e-01  1.217380e+00
## district11           -2.230306e+00  5.506277e-01
## district13           -2.356972e+00  2.296863e-01
## district2            -1.077909e+00  1.004024e+00
## district4            -1.560521e+00  9.564451e-01
## district5            -3.168644e+00  4.497812e-01
## district6            -1.726156e+00  4.806006e-01
## district7            -1.238077e+00  5.585124e-01
## district8             6.166305e-02  1.861326e+00
## district9            -3.748292e-01  2.022747e+00
## renovationCondition2 -7.203229e+00 -4.898259e+00
## renovationCondition3 -5.283562e+00 -4.206650e+00
## renovationCondition4 -4.755412e+00 -3.812550e+00
## subway1               4.940397e-01  1.063902e+00
```

```
hist(betahat.boot)
```


Normal Density
Kernel Density
bca 95% CI
Obs. Value
Density
(Intercept)
communityAverage
square
followers
tradeTime
district10
district11
district13
district2
district4
district5
district6
district7
district8
district9
renovationCondition2
renovationCondition3
renovationCondition4




```
# bootstrap for variance
sigmahat.boot <- Boot(mod3, R=100, f=sigmaHat, labels="sigmaHat")
summary(sigmahat.boot)
```

```
##            R original   bootBias   bootSE bootMed
## sigmaHat 100    6.588 -0.0026512 0.078925  6.5886
```

```
confint(sigmahat.boot)
```

```
## Bootstrap percent confidence intervals
##
##            2.5 %   97.5 %
## sigmaHat 6.41406 6.755082
```

The boot method indicate that the estimates of districts are not robust. And the confidence intervals estimated by bootstrap is more robust than the ordinary methods. The histograms shows us the robustness of estimation intuitively.

Five-fold Cross-Validation

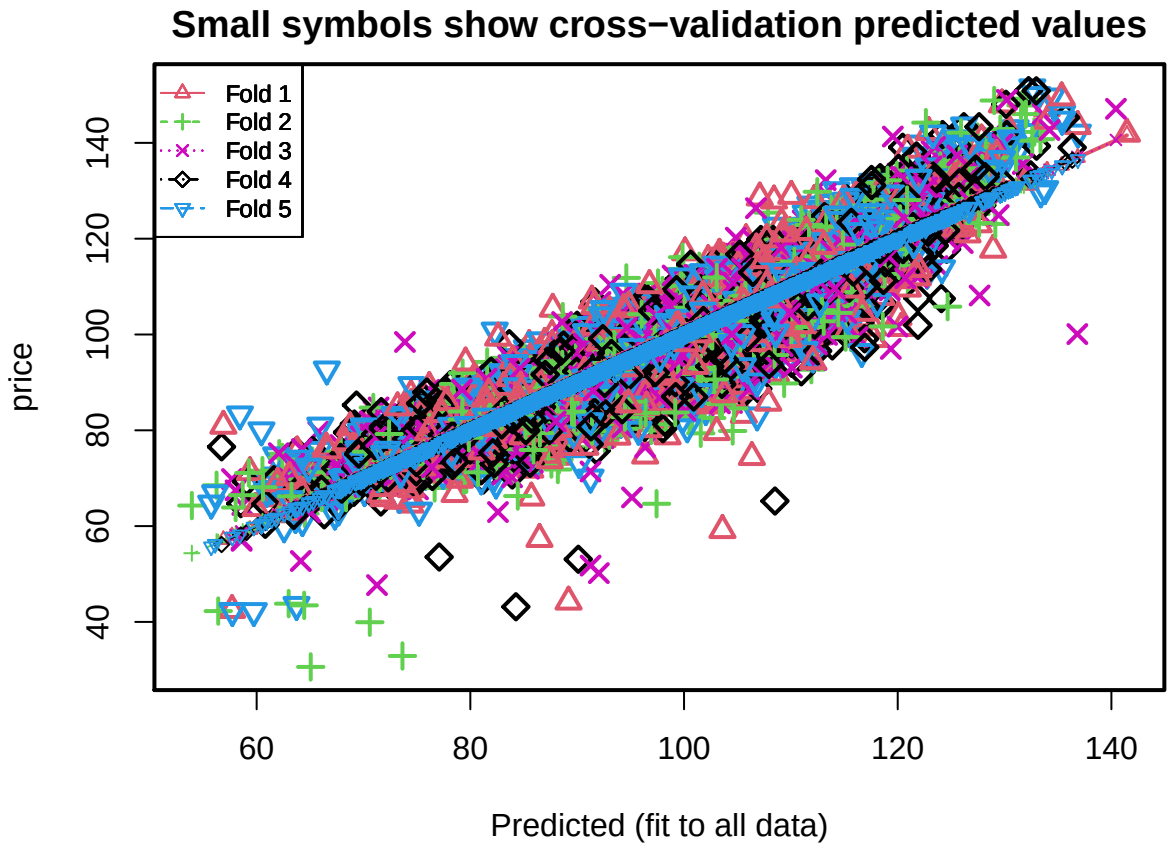
```
train_control <- trainControl(method = "cv", number = 5, savePredictions = TRUE, returnResamp = "all")

train(price ~ communityAverage + square + followers + tradeTime + district + renovationCondition
      + subway, weights = 1 / vari, data = transformed_train_noOutliers,
      trControl = train_control, method = "lm")
```

```
## Linear Regression
```

```
##
## 7260 samples
##    7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5808, 5808, 5808, 5808, 5808
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   6.63309  0.8277662  5.072502
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

RMSE is small and R square is large. Mod5 performs good according to the Five-folds cross vaildation test.

```
cvResults <- suppressWarnings(DAAG::CVlm(data = transformed_train_noOutliers, form.lm = mod5, m=5,
                                         dots = FALSE, seed = 1, legend.pos = "topleft",
                                         printit = FALSE))
```



The spread is balanced and narrow so that our model performs good.

Training and Testing

```
# select variable in test dataset
var <- c("price", "communityAverage", "tradeTime", "district", "followers", "renovationCondition", "cons
test <- subset(df_test, select = var)
```

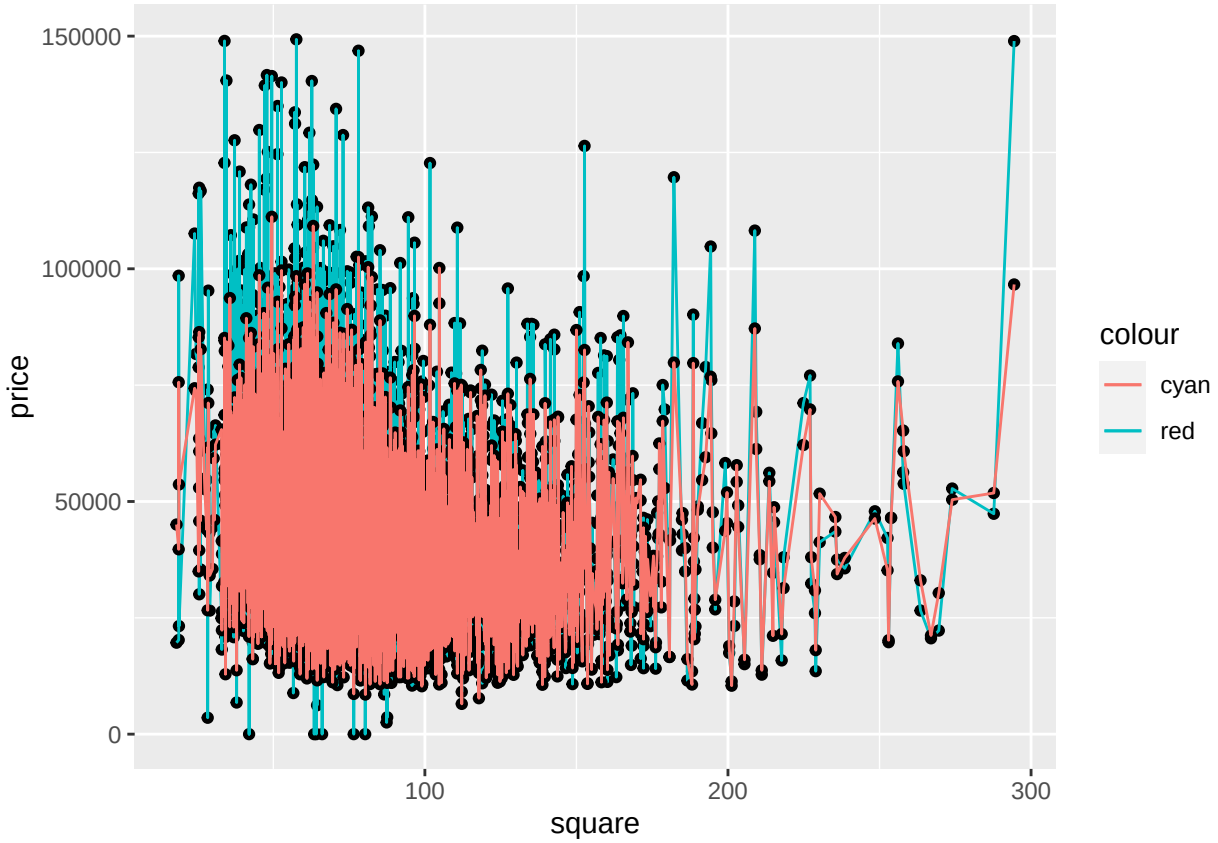
```r
test <- test[!(test$district %in% c("3", "12")), ]

# transform variable using training set coefficients
transformed1_test <- bcPower(with(test, cbind(price, communityAverage, square)), coef(p1, round = T))
transformed2_test <- yjPower(test$followers, coef(p2, round = T))

# new transformed test dataset
transformed_test = as.data.frame(cbind(transformed1_test, transformed2_test, data.frame(with(test, cbin
names(transformed_test)[1: 4] = c("price", "communityAverage", "square", "followers")

transformed_test[, c("constructionTime", "tradeTime")] <- data.frame(apply(transformed_test[, c("constr

# predict using model with testing data
predict_test <- predict(mod5, newdata = transformed_test)

# compare the fitted values with real data
lambda <- p1$lambda[1]
predict_test_ori <- (predict_test * lambda + 1) ^ (1 / lambda)
forecast::accuracy(predict_test_ori, test$price)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##                ME     RMSE      MAE       MPE     MAPE
## Test set 1640.173 9687.546 6853.722 -1590.169 1605.576
```

Cause the mean of price is 43948 RMB per square and model has heteroskedasticity which is hard to mitigate, the 9687.546 RMB error.

```r
result <- as.data.frame(cbind(test$square, test$price, predict_test_ori))
names(result) <- c("square", "price", "predicted_price")


ggplot(result, aes(x = square)) + geom_point(aes(y = price)) + geom_line(aes(y=price, color="red"))+ ge
```



The red line is fitted value and cyan lineis the real data(I do not know why ggplot mess up the label). Our prediction better in middle level price and do bad in relatively large prices.

Reference of Code

- Professor Rojas's Lecture Notes
- Boruta Algorithm: https://www.datacamp.com/community/tutorials/feature-selection-R-boruta
- GLM: https://stats.stackexchange.com/questions/190763/how-to-decide-which-glm-family-to-use