DATA605 Final Project

Bin Lin

2017-5-22

Pick one of the quantitative independent variables from the training data set (train.csv) , and define that variable as X. Pick SalePrice as the dependent variable, and define it as Y for the next analysis.

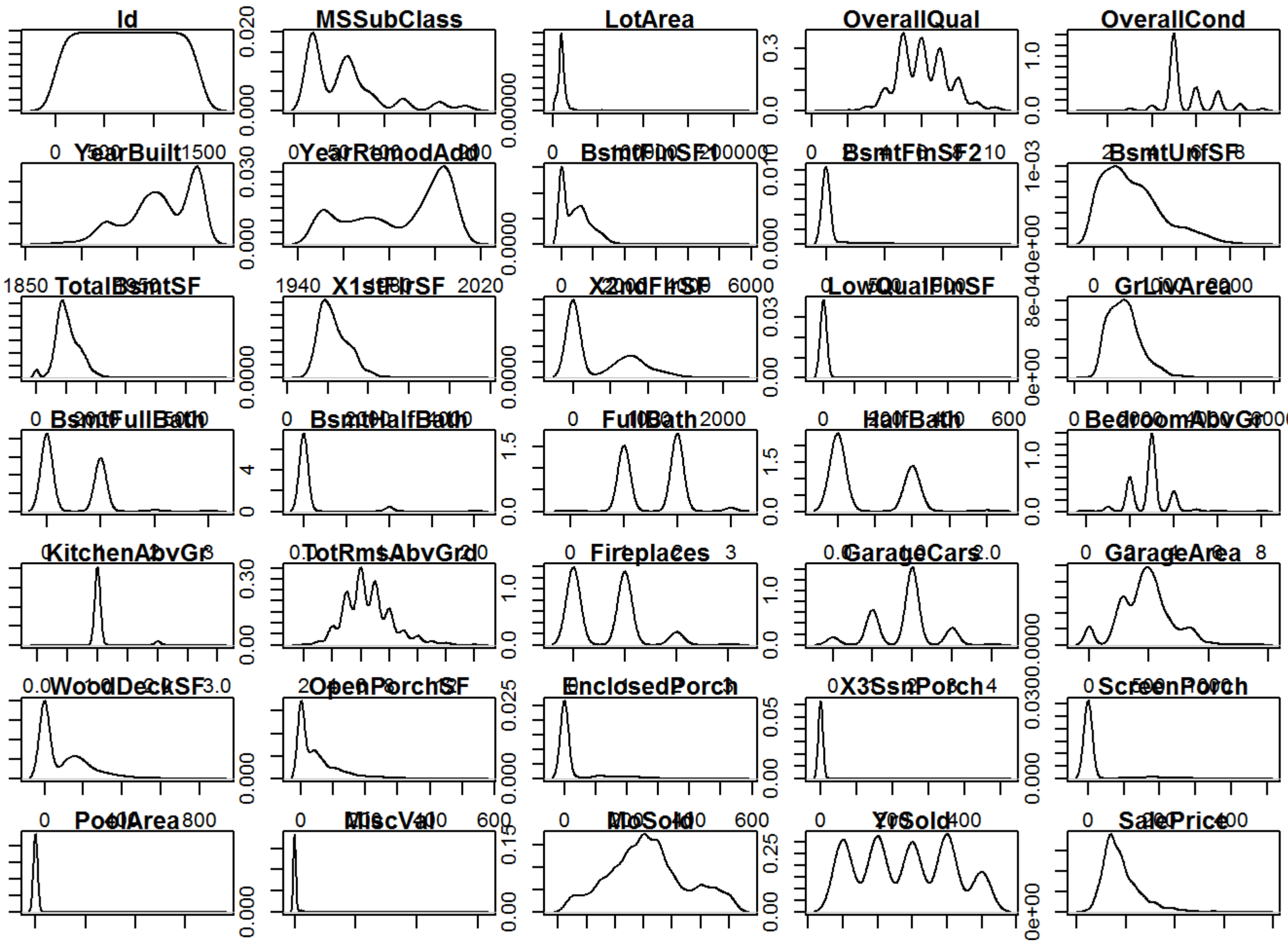
```
#Loading all necessary R packages.
suppressWarnings(suppressMessages(library(ggplot2)))
suppressWarnings(suppressMessages(library(mice)))
suppressWarnings(suppressMessages(library(dplyr)))
suppressWarnings(suppressMessages(library(reshape2)))
suppressWarnings(suppressMessages(library(MASS)))
suppressWarnings(suppressMessages(library(broom)))
suppressWarnings(suppressMessages(library(lmtest)))
suppressWarnings(suppressMessages(library(moments)))
suppressWarnings(suppressMessages(library(forecast)))
suppressWarnings(suppressMessages(library(Metrics)))



#Subset all the numeric variables, and the and filter out columns that have too many na.
raw_data <- read.csv("https://raw.githubusercontent.com/blin261/data-605/master/train.csv", head
er=TRUE, stringsAsFactors = FALSE)
train <- Filter(is.numeric, raw_data)
train <- train[ , colSums(is.na(train)) <= (0.05 * length(raw_data))]

str(train)
```

```
## 'data.frame':	1460 obs. of  35 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000
...
```

```
head(train)
```

```
##   Id MSSubClass LotArea OverallQual OverallCond YearBuilt YearRemodAdd
## 1  1         60    8450           7           5      2003         2003
## 2  2         20    9600           6           8      1976         1976
## 3  3         60   11250           7           5      2001         2002
## 4  4         70    9550           7           5      1915         1970
## 5  5         60   14260           8           5      2000         2000
## 6  6         50   14115           5           5      1993         1995
##   BsmtFinSF1 BsmtFinSF2 BsmtUnfSF TotalBsmtSF X1stFlrSF X2ndFlrSF
## 1        706          0       150         856       856       854
## 2        978          0       284        1262      1262         0
## 3        486          0       434         920       920       866
## 4        216          0       540         756       961       756
## 5        655          0       490        1145      1145      1053
## 6        732          0        64         796       796       566
##   LowQualFinSF GrLivArea BsmtFullBath BsmtHalfBath FullBath HalfBath
## 1            0      1710            1            0        2        1
## 2            0      1262            0            1        2        0
## 3            0      1786            1            0        2        1
## 4            0      1717            1            0        1        0
## 5            0      2198            1            0        2        1
## 6            0      1362            1            0        1        1
##   BedroomAbvGr KitchenAbvGr TotRmsAbvGrd Fireplaces GarageCars GarageArea
## 1            3            1            8          0          2        548
## 2            3            1            6          1          2        460
## 3            3            1            6          1          2        608
## 4            3            1            7          1          3        642
## 5            4            1            9          1          3        836
## 6            1            1            5          0          2        480
##   WoodDeckSF OpenPorchSF EnclosedPorch X3SsnPorch ScreenPorch PoolArea
## 1          0          61             0          0           0        0
## 2        298           0             0          0           0        0
## 3          0          42             0          0           0        0
## 4          0          35           272          0           0        0
## 5        192          84             0          0           0        0
## 6         40          30             0        320           0        0
##   MiscVal MoSold YrSold SalePrice
## 1       0      2   2008    208500
## 2       0      5   2007    181500
## 3       0      9   2008    223500
## 4       0      2   2006    140000
## 5       0     12   2008    250000
## 6     700     10   2009    143000
```

```
par(mar = rep(1, 4),mfrow=c(7, 5))
for (i in 1:length(train))
{
    plot(density(train[,i]), main = colnames(train)[i])
}
```




```
#I would like to pick LotArea to be my X and Sale Price to be my Y. Then I create a dataframe co
ntaining X and Y.
X <- train$LotArea
Y <- train$SalePrice
house_price <- data.frame(X, Y)
```

Probability.

Calculate as a minimum the below probabilities a through c. Assume the small letter “x” is estimated as the 4th quartile of the X variable, and the small letter “y” is estimated as the 2d quartile of the Y variable. Interpret the meaning of all probabilities.
a. P(X>x | Y>y) b. P(X>x, Y>y) c. P(Xy)

```
#P(X > x)
a <- subset(house_price, X > quantile(house_price$X, 0.75))


#P(X <= x)
b <- subset(house_price, X <= quantile(house_price$X, 0.75))


#P(Y > y)
c <- subset(house_price, Y > quantile(house_price$Y, 0.5))


#P(Y <= y)
d <- subset(house_price, Y <= quantile(house_price$Y, 0.5))



#a.  P(X>x | Y>y) = P(X>x, Y>y) / P(Y>y)
nrow(intersect(a, c)) / nrow(c)
```

```
## [1] 0.3791209
```

```
#b.  P(X>x, Y>y)
nrow(intersect(a, c)) / nrow(house_price)
```

```
## [1] 0.1890411
```

```
#c.  P(X<x | Y>y) = P(X<x, Y>y) / P(Y>y)
nrow(intersect(b, c)) / nrow(c)
```

```
## [1] 0.6208791
```

Does splitting the training data in this fashion make them independent? In other words, does P(X|Y)=P(X)P(Y))? Check mathematically, and then evaluate by running a Chi Square test for association. You might have to research this.

Splitting the training data in this fashion does not make them independent. Because P(X>x | Y>y) = 0.3791209, however, P(X>x, Y>y) = 0.1890411. The following code will perform the Chi Square test for association.

```
nrow(intersect(a, c))
```

```
## [1] 276
```

```
nrow(intersect(a, d))
```

```
## [1] 89
```

```
nrow(intersect(b, c))
```

```
## [1] 452
```

```
nrow(intersect(b, d))
```

```
## [1] 634
```

```
frequency_table <- matrix((c(nrow(intersect(a, c)), nrow(intersect(a, d)), nrow(intersect(b, c))
, nrow(intersect(b, d)))) , nrow = 2, ncol = 2, byrow = FALSE)

colnames(frequency_table) <- c("P(X>x)", "P(X<=x)")
rownames(frequency_table) <- c("P(Y>y)", "P(Y<=y)")
frequency_table <- as.table(frequency_table)
frequency_table
```

```
##         P(X>x) P(X<=x)
## P(Y>y)     276     452
## P(Y<=y)     89     634
```

```
chisq.test(frequency_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  frequency_table
## X-squared = 124.93, df = 1, p-value < 2.2e-16
```

According to Chi-Square test, the p-value of test is 2.2e-16 which is much less than the typical significance level alpha = 0.05. Therefore, we should reject the null-hypothesis and claim that two variables we studied are dependent on each other, meaning there exist some kind of relationship between them. Having a larger lot area is going to influence the final sale price(apparently a positive relationship here).

Descriptive and Inferential Statistics.

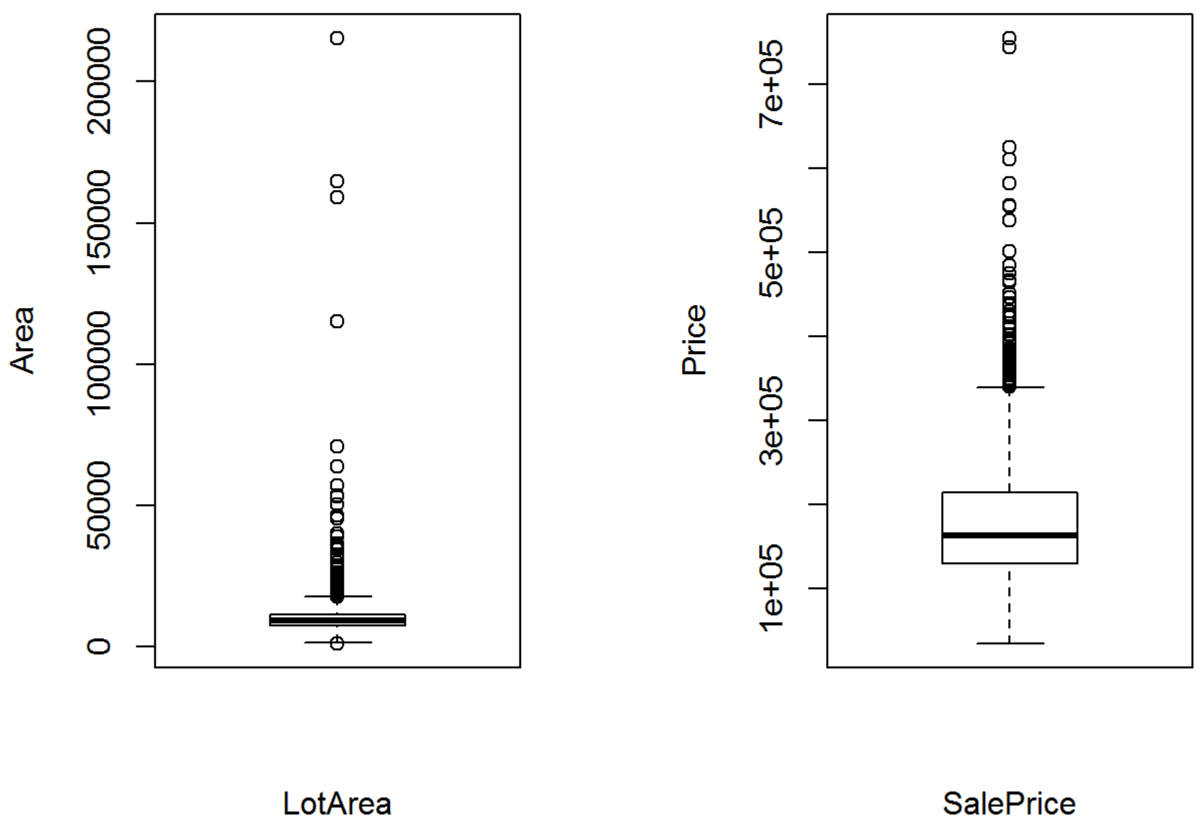
Provide univariate descriptive statistics and appropriate plots for both variables. Provide a scatterplot of X and Y. Transform both variables simultaneously using Box-Cox transformations. You might have to research this. Using the transformed variables, run a correlation analysis and interpret. Test the hypothesis that the correlation between these variables is 0 and provide a 99% confidence interval. Discuss the meaning of your analysis.

From the summary statistics, we can tell the distribution for both variables are skewed to the right, because their mean are larger than their median. It is also shown in the boxplot since there are a lot of outliers as the variables get larger. The histogram of LotArea looks like exponential distribution, while the histogram of SalePrice look more similar to possion distribution. And then the scatterplot of there two variables display a linear relationship.

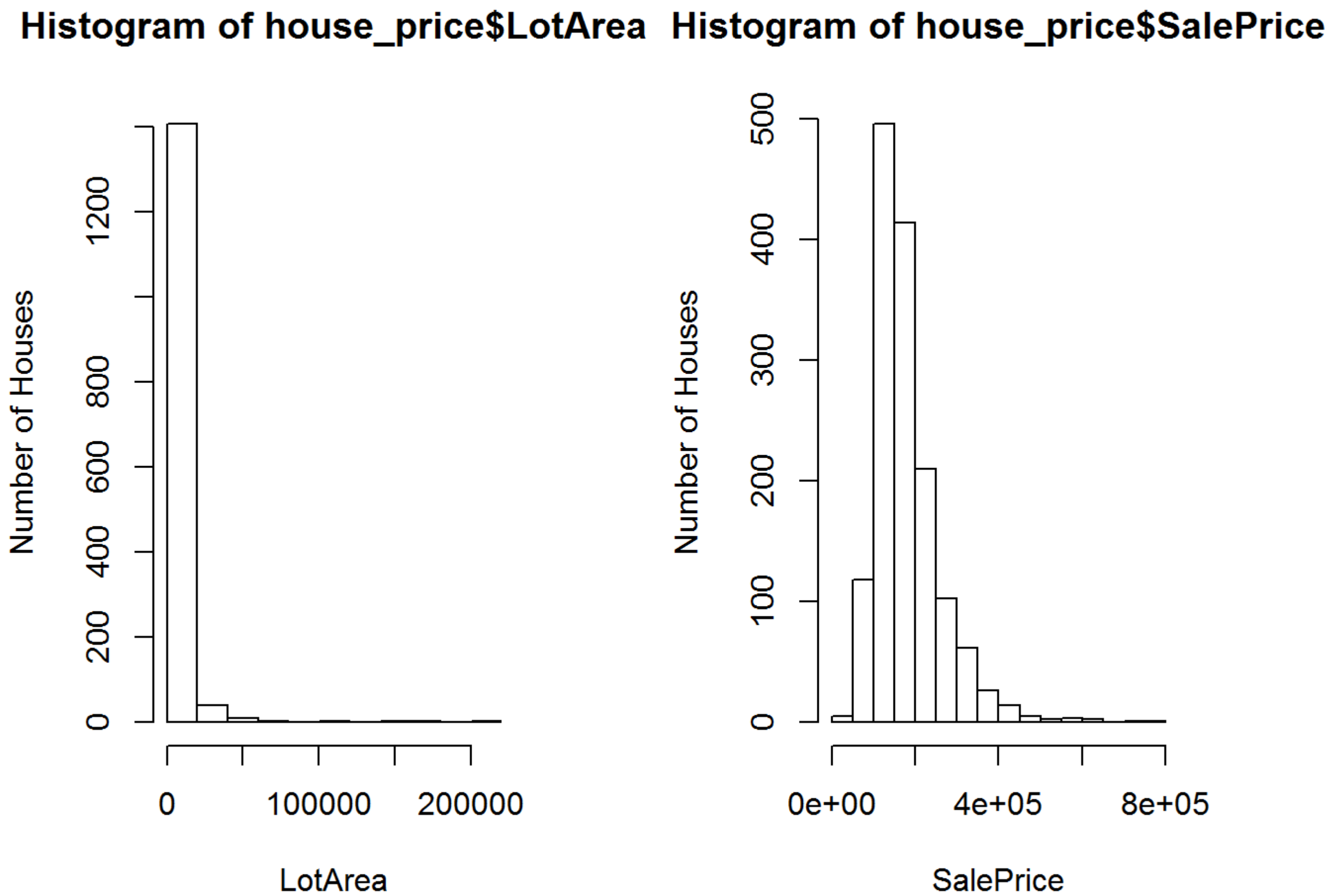
```
colnames(house_price) <- c("LotArea", "SalePrice")
summary(house_price)
```

```
##     LotArea         SalePrice
##  Min.   :  1300   Min.   : 34900
##  1st Qu.:  7554   1st Qu.:129975
##  Median :  9478   Median :163000
##  Mean   : 10517   Mean   :180921
##  3rd Qu.: 11602   3rd Qu.:214000
##  Max.   :215245   Max.   :755000
```

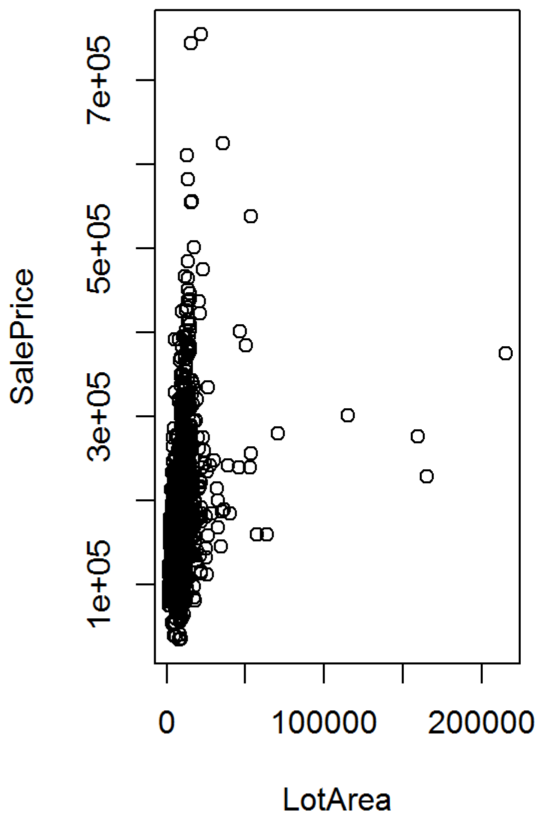
```
par(mar = rep(4, 4), mfrow=c(1,2))
boxplot(house_price$LotArea, xlab = "LotArea", ylab="Area")
boxplot(house_price$SalePrice, xlab = "SalePrice", ylab="Price")
```




```
par(mar = rep(4, 4), mfrow=c(1,2))
hist(house_price$LotArea, xlab = "LotArea", ylab = "Number of Houses")
hist(house_price$SalePrice, xlab = "SalePrice", ylab = "Number of Houses")
```

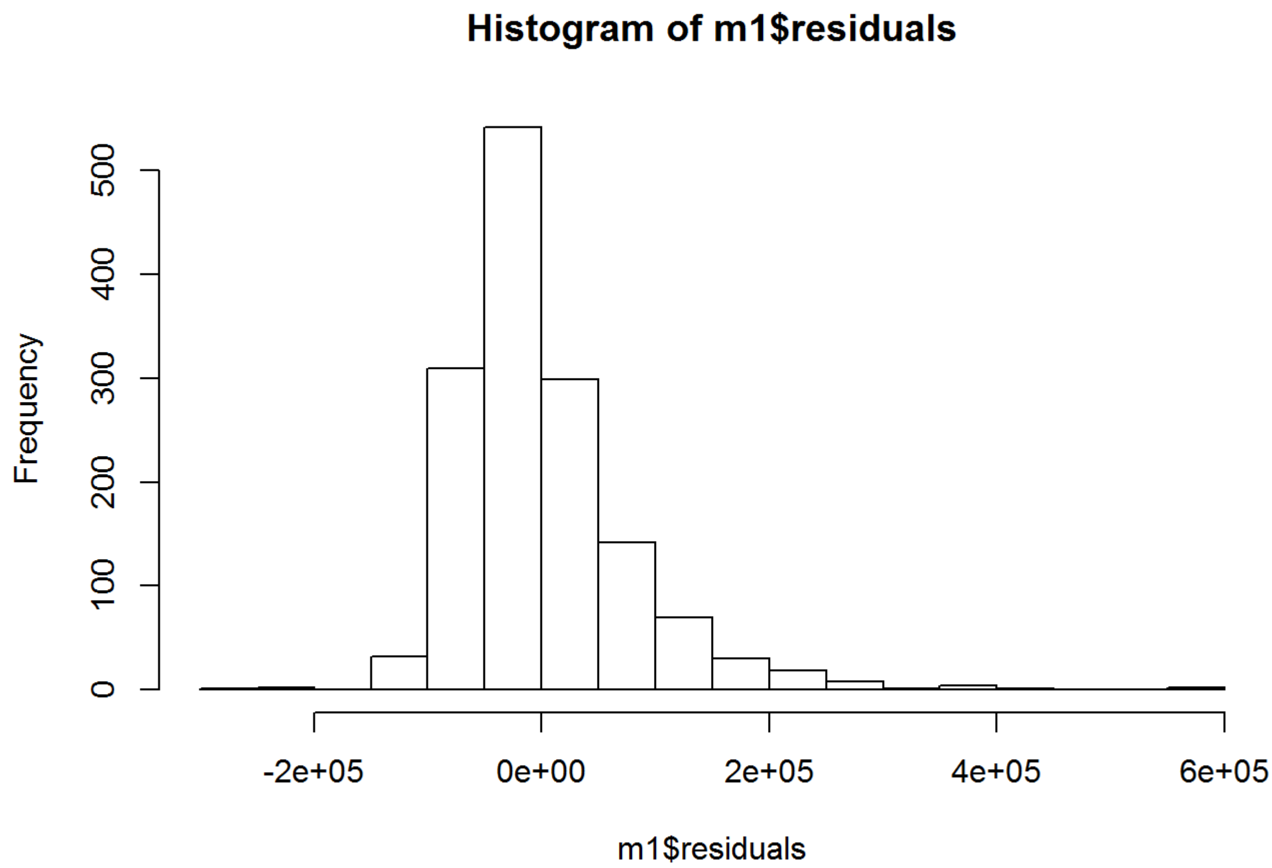



```
plot(x = house_price$LotArea, y = house_price$SalePrice, xlab = "LotArea", ylab = "SalePrice")
```



I created first linear model. The histogram of the residuals does not follow normal distribution. The qqplot of the residual also gets bended. Both phenomena have indicated the both of them may not be a good candidates for building linear regression model, because their variances differ significantly across x-axis. This is what we called “heteroscedasticity”.

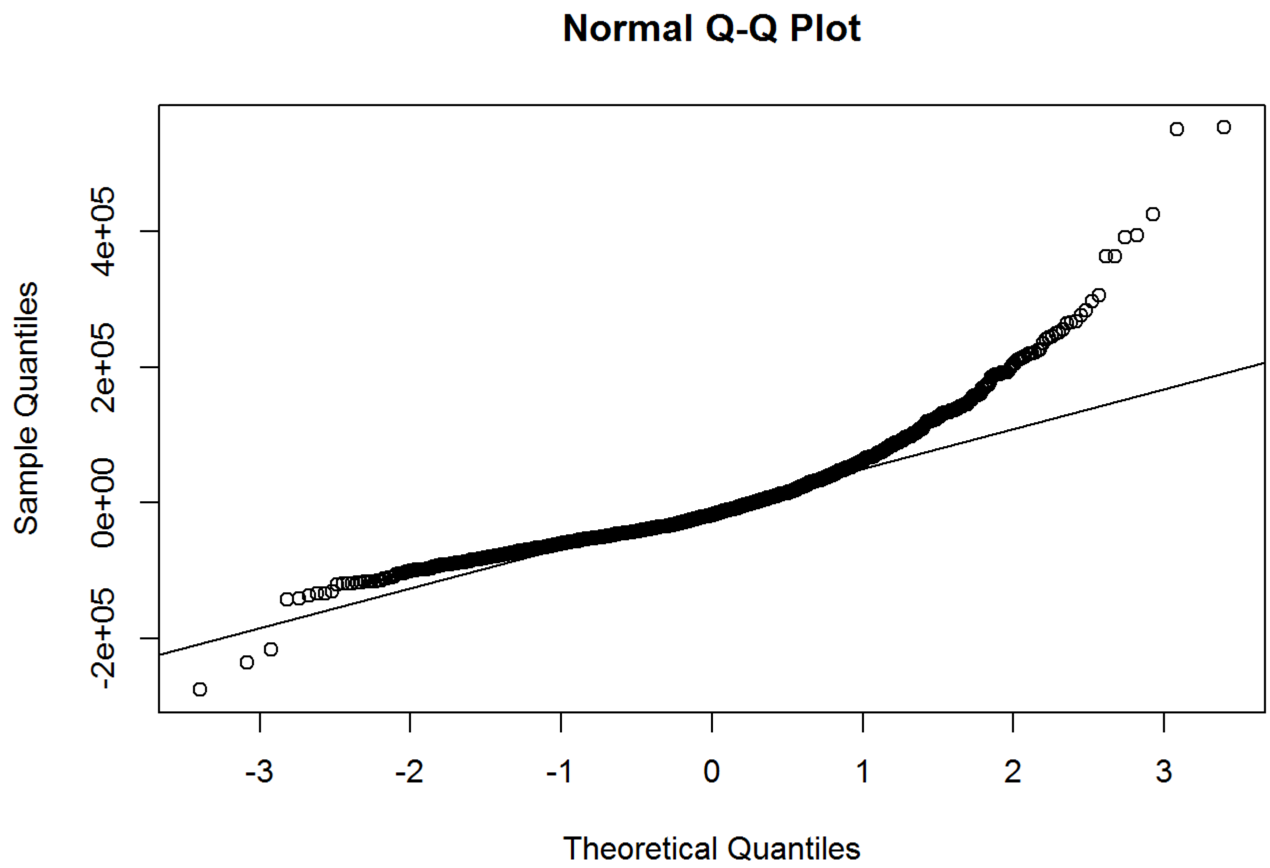
```
m1 <- lm(SalePrice ~ LotArea, house_price)
hist(m1$residuals)
```




```
skewness(m1$residuals)
```

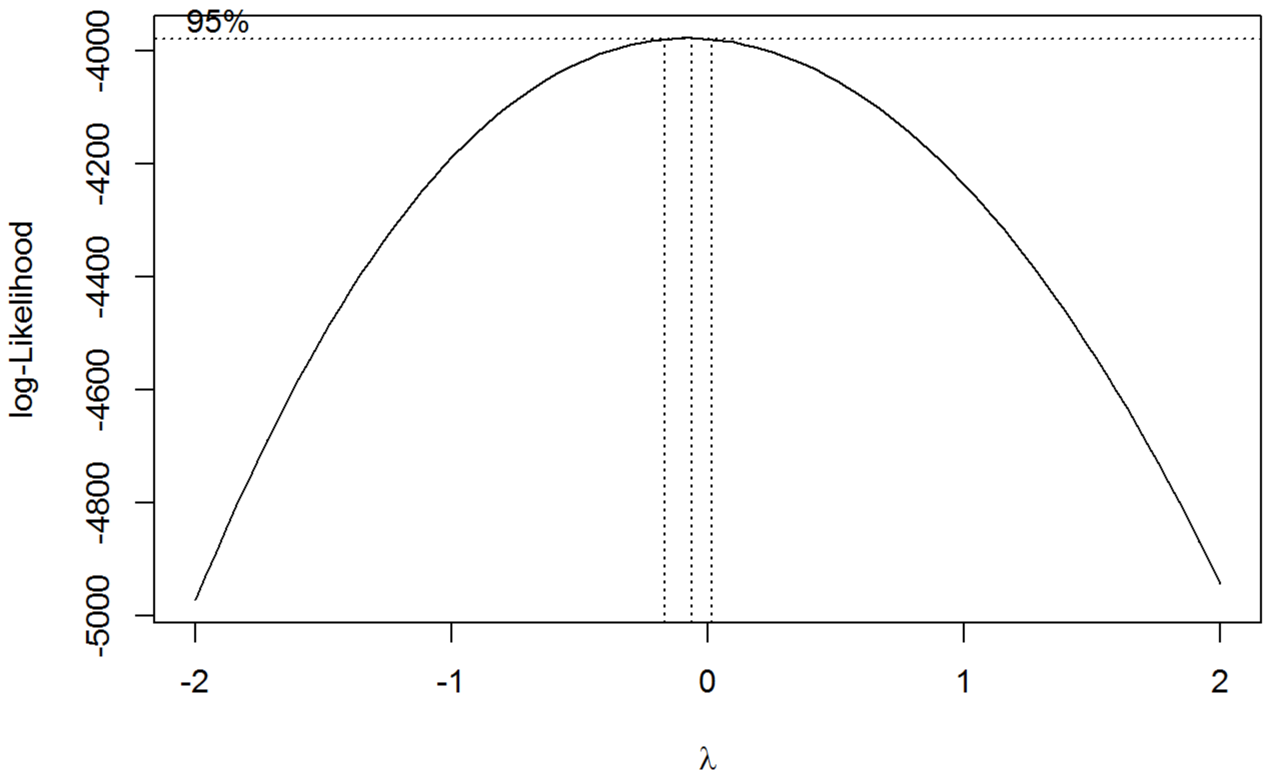
```
## [1] 1.788232
```

```
qqnorm(m1$residuals)
qqline(m1$residuals)
```



Using box-cox transformation on each of the two variables can yield normalized data variables which can be properly evaluated. Even residual histogram and qqplot are becoming normalized as show in the following figures. The skewedness is also reduced dramatically.

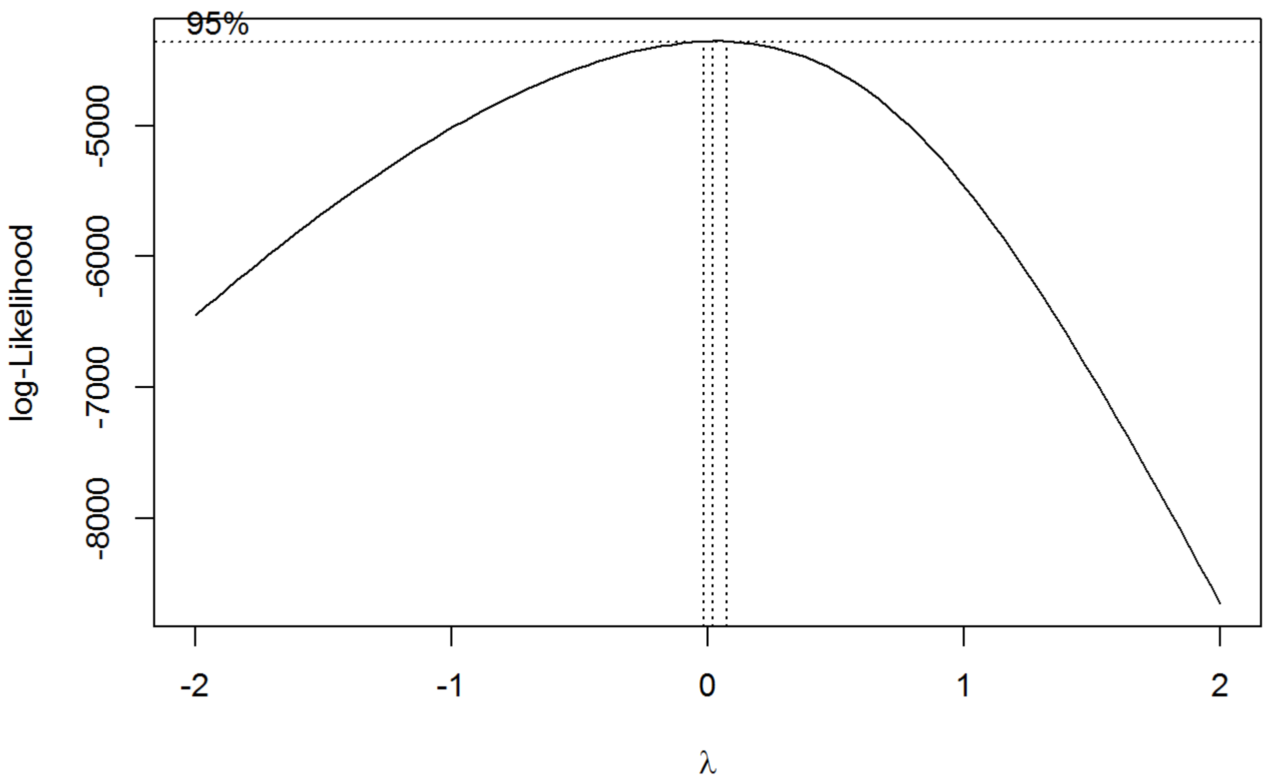
```
saleprice_bc <- boxcox(house_price$SalePrice ~ 1)
```




```
y_lambda <- with(saleprice_bc, x[which.max(y)])
y_lambda
```

```
## [1] -0.06060606
```

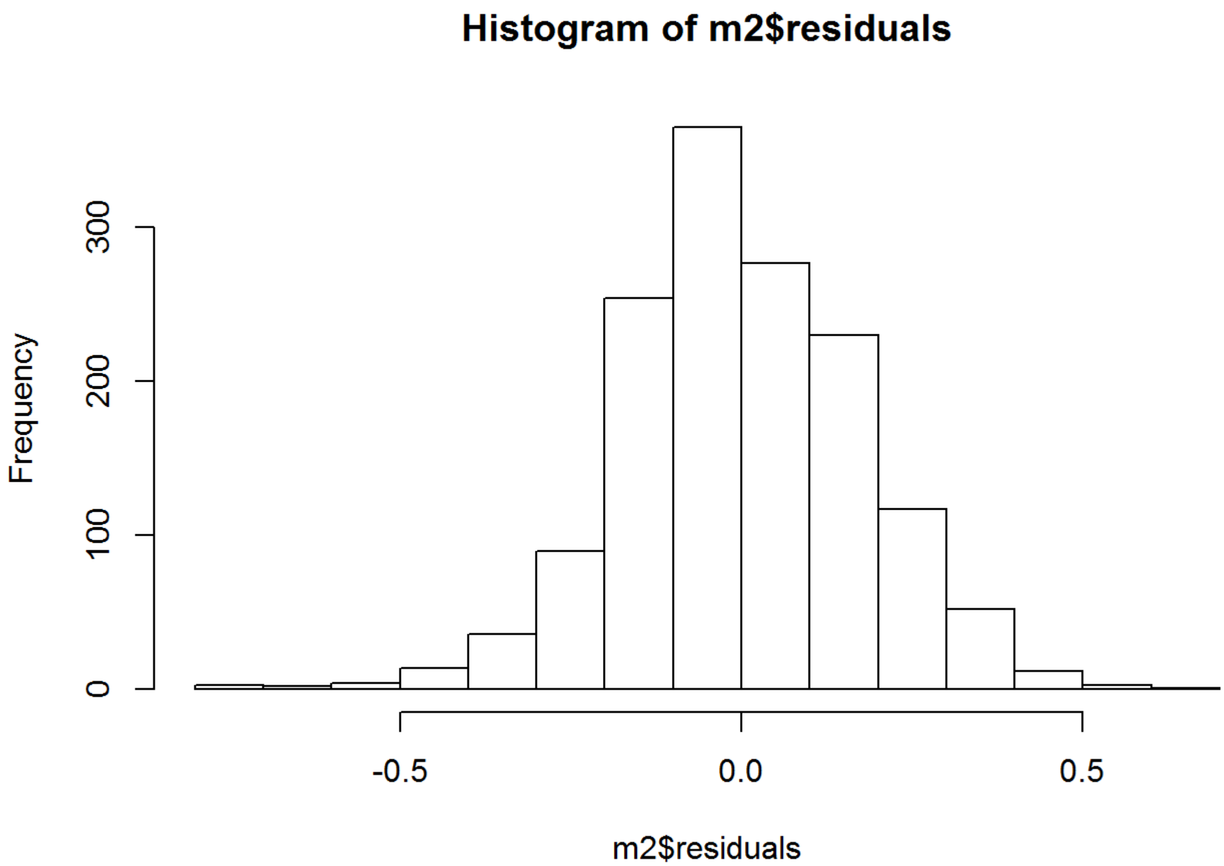
```
lotarea_bc <- boxcox(house_price$LotArea ~ 1)
```




```
x_lambda <- with(lotarea_bc, x[which.max(y)])
x_lambda
```

```
## [1] 0.02020202
```

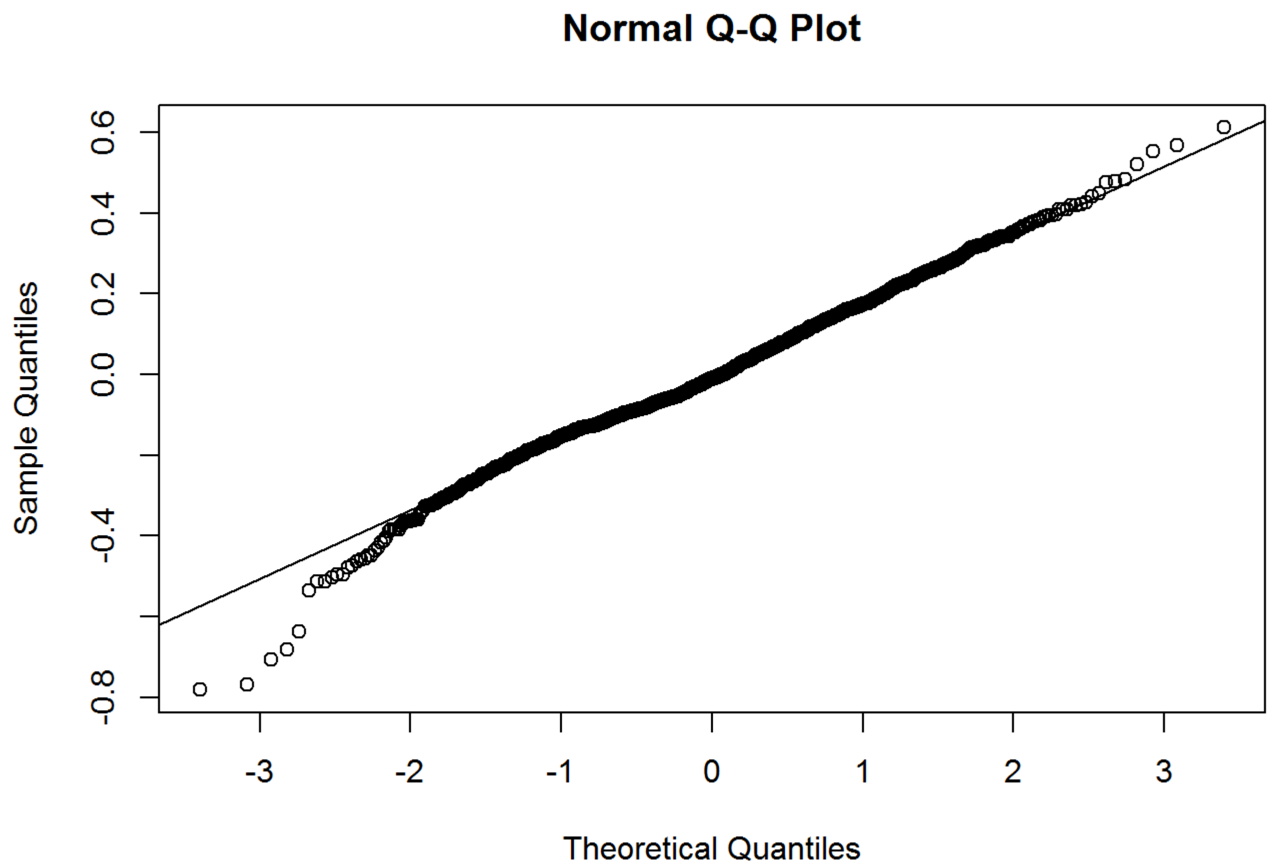
```
saleprice_new <- BoxCox(house_price$SalePrice, y_lambda)
lotarea_new <- BoxCox(house_price$LotArea, x_lambda)

m2 <- lm(saleprice_new ~ lotarea_new)
hist(m2$residuals)
```




```
skewness(m2$residuals)
```

```
## [1] -0.1123381
```

```
qqnorm(m2$residuals)
qqline(m2$residuals)
```



The correlation coefficient between two variables are 0.3992109, which means moderate positive relationship. The relationship is statistically significant since the p-value is almost 0 (less than significance level 0.05). Therefore, we reject the null hypothesis, while accepting the alternative hypothesis to be true (true correlation is not equal to 0). The 99 percent confidence interval is (0.3410038, 0.4543686). If we perform the experiment 100 times by drawing sample distribution from the population, 99 or more times, the correlation between LotArea and SalePrice will fall in the range.

```
house_price2 <- data.frame(lotarea_new, saleprice_new)
colnames(house_price2) <- c("LotArea", "SalePrice" )
head(house_price2)
```

```
##     LotArea SalePrice
## 1  9.920409  8.645584
## 2 10.073775  8.579289
## 3 10.264964  8.678585
## 4 10.067491  8.453678
## 5 10.551907  8.731520
## 6 10.539510  8.464011
```

```
cor(house_price2)
```

```
##              LotArea SalePrice
## LotArea   1.0000000 0.3992109
## SalePrice 0.3992109 1.0000000
```

```
cor.test(house_price2$LotArea, house_price2$SalePrice, conf.level = 0.99)
```

```
##
##  Pearson's product-moment correlation
##
## data:  house_price2$LotArea and house_price2$SalePrice
## t = 16.626, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 99 percent confidence interval:
##  0.3410038 0.4543686
## sample estimates:
##       cor
## 0.3992109
```

Linear Algebra and Correlation.

Invert your correlation matrix. (This is known as the precision matrix and contains variance inflation factors on the diagonal.) Multiply the correlation matrix by the precision matrix, and then multiply the precision matrix by the correlation matrix.

```
A <- round(cor(house_price2))
B <- round(solve(cor(house_price2)))
A %*% B
```

```
##           LotArea SalePrice
## LotArea         1         0
## SalePrice       0         1
```

```
B %*% A
```

```
##           LotArea SalePrice
## LotArea         1         0
## SalePrice       0         1
```

Calculus-Based Probability & Statistics.

Many times, it makes sense to fit a closed form distribution to data. For your non-transformed independent variable, location shift it so that the minimum value is above zero. Then load the MASS package and run fitdistr to fit a density function of your choice. (See https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/fitdistr.html (https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/fitdistr.html)). Find the optimal value of the parameters for this distribution, and then take 1000 samples from this distribution (e.g., rexp(1000, ???) for an exponential). Plot a histogram and compare it with a histogram of your non-transformed original variable.

fitdistr is function which will find the best fitting univariate distribution in terms of maximum-likelihood and return the optimal parameters. In this case, lambda is the parameter for exponential distribution. The sample data was simulated using lambda. Apparently, the sample data follows exponential distribution more closedly. It is also less skewed and smooth.

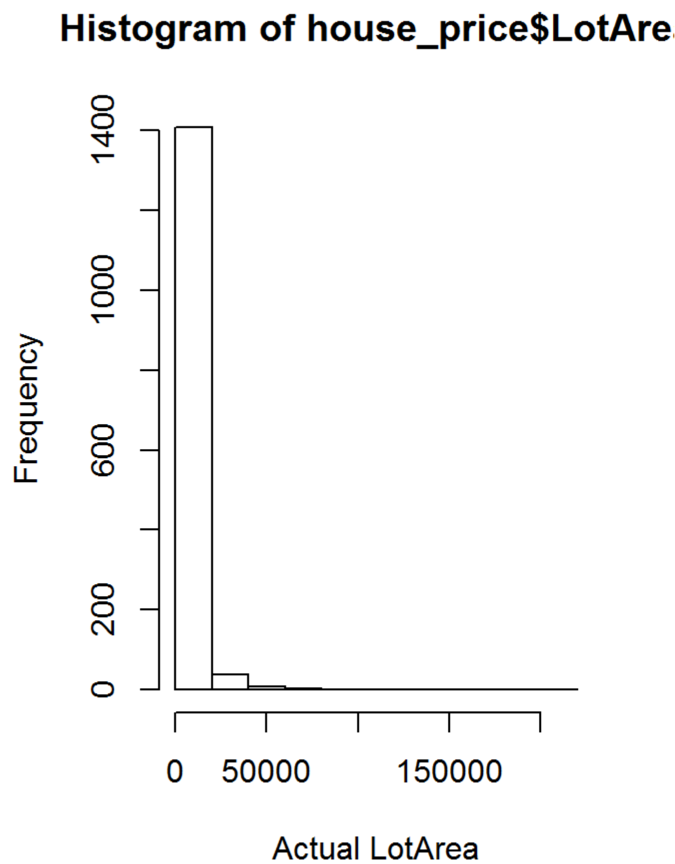
```
#Both Variables are above 0.
summary(house_price)
```

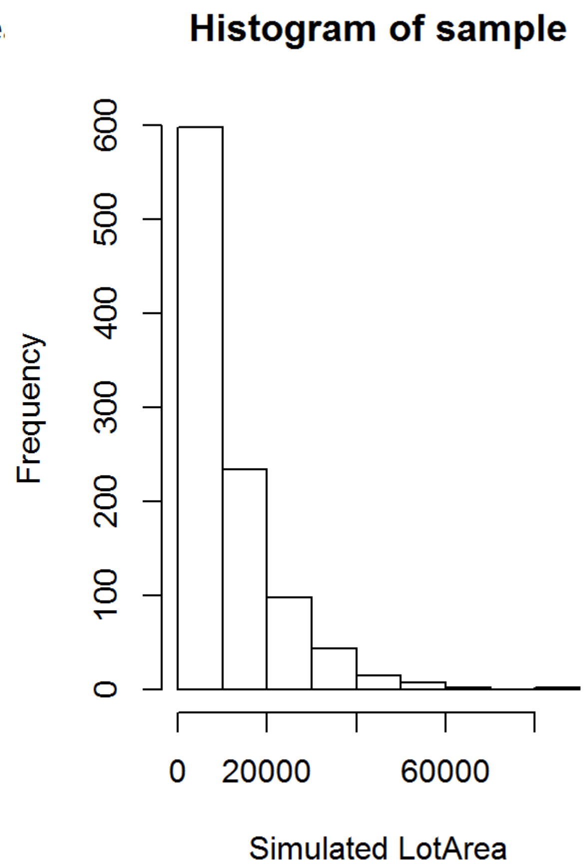
```
##     LotArea         SalePrice
##  Min.   :  1300   Min.   : 34900
##  1st Qu.:  7554   1st Qu.:129975
##  Median :  9478   Median :163000
##  Mean   : 10517   Mean   :180921
##  3rd Qu.: 11602   3rd Qu.:214000
##  Max.   :215245   Max.   :755000
```

```
fit <- fitdistr(house_price$LotArea, densfun = "exponential")
lambda_best <- fit$estimate
lambda_best
```

```
##        rate
## 9.50857e-05
```

```
sample <- rexp(1000, lambda_best)

par(mfrow=c(1,2))
hist(house_price$LotArea, xlab = "Actual LotArea")
hist(sample, xlab = "Simulated LotArea")
```





Modeling.

Build some type of regression model and submit your model to the competition board. Provide your complete model summary and results with analysis. Report your Kaggle.com user name and score.

My model's prediction of sale price for the training dataset is very similar to the actual sale price. This is manifest in the both histogram and boxplot of the estimation residual. However, there still exist some outliers in the boxplot, that means for certain observations, the predicted value differ quite a lot from the actual value. But overall it is well performed model.

```
#Remove both Id and SalePrice variables.
m3 <- lm(SalePrice ~. - Id - SalePrice, data = train)
m3_back <- step(m3, trace = 0)
summary(m3_back)
```

```
##
## Call:
## lm(formula = SalePrice ~ MSSubClass + LotArea + OverallQual +
##     OverallCond + YearBuilt + YearRemodAdd + BsmtFinSF1 + BsmtUnfSF +
##     X1stFlrSF + X2ndFlrSF + BsmtFullBath + FullBath + BedroomAbvGr +
##     KitchenAbvGr + TotRmsAbvGrd + Fireplaces + GarageCars + WoodDeckSF +
##     ScreenPorch + PoolArea, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -479173  -17000   -2217   13418  289817
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.610e+05  1.249e+05  -7.692 2.67e-14 ***
## MSSubClass   -1.585e+02  2.608e+01  -6.078 1.56e-09 ***
## LotArea       4.080e-01  1.008e-01   4.048 5.44e-05 ***
## OverallQual   1.808e+04  1.178e+03  15.346  < 2e-16 ***
## OverallCond   4.305e+03  1.014e+03   4.245 2.33e-05 ***
## YearBuilt     3.195e+02  5.334e+01   5.989 2.66e-09 ***
## YearRemodAdd  1.348e+02  6.567e+01   2.053 0.040280 *
## BsmtFinSF1    1.953e+01  3.906e+00   5.000 6.43e-07 ***
## BsmtUnfSF     8.454e+00  3.652e+00   2.315 0.020760 *
## X1stFlrSF     5.252e+01  5.300e+00   9.909  < 2e-16 ***
## X2ndFlrSF     4.879e+01  4.256e+00  11.465  < 2e-16 ***
## BsmtFullBath  8.608e+03  2.437e+03   3.532 0.000426 ***
## FullBath      3.973e+03  2.605e+03   1.525 0.127414
## BedroomAbvGr -1.034e+04  1.691e+03  -6.115 1.24e-09 ***
## KitchenAbvGr -1.476e+04  5.158e+03  -2.861 0.004281 **
## TotRmsAbvGrd  5.255e+03  1.223e+03   4.296 1.85e-05 ***
## Fireplaces    3.347e+03  1.756e+03   1.906 0.056801 .
## GarageCars    1.095e+04  1.702e+03   6.431 1.72e-10 ***
## WoodDeckSF    2.629e+01  7.944e+00   3.310 0.000957 ***
## ScreenPorch   5.671e+01  1.702e+01   3.332 0.000883 ***
## PoolArea     -3.703e+01  2.350e+01  -1.575 0.115408
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35040 on 1439 degrees of freedom
## Multiple R-squared:  0.8081, Adjusted R-squared:  0.8055
## F-statistic: 303.1 on 20 and 1439 DF,  p-value: < 2.2e-16
```

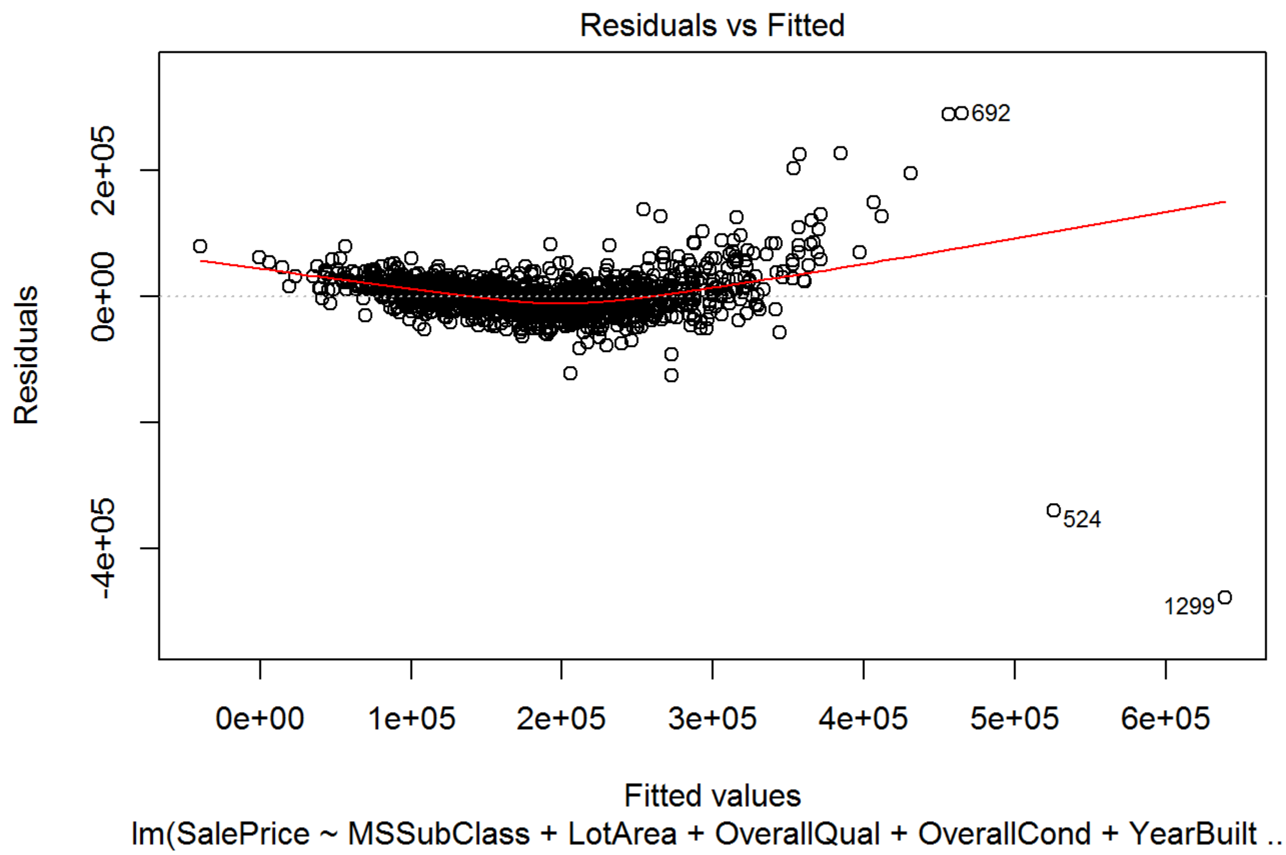
```
#Just want to see how well the model perform.
AIC(m3_back)
```

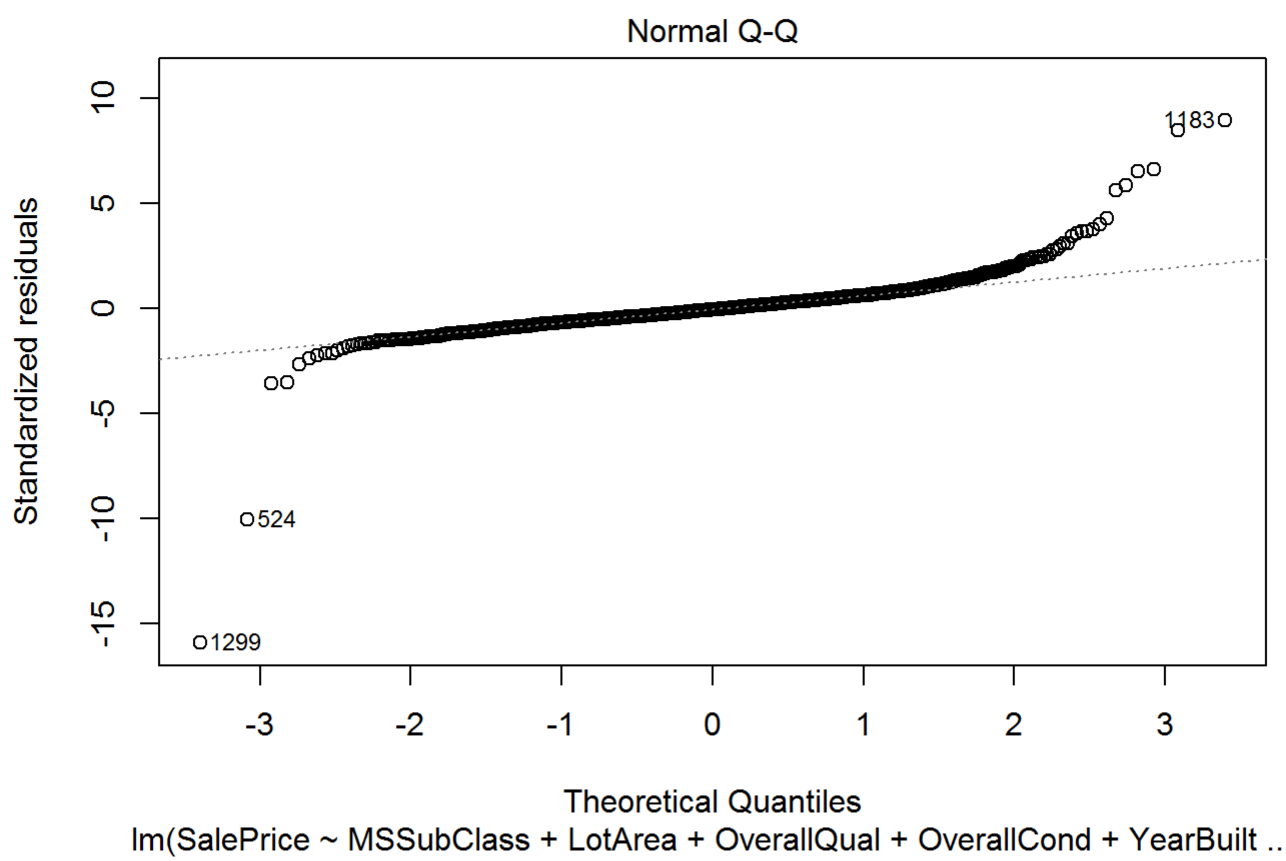
```
## [1] 34721.6
```

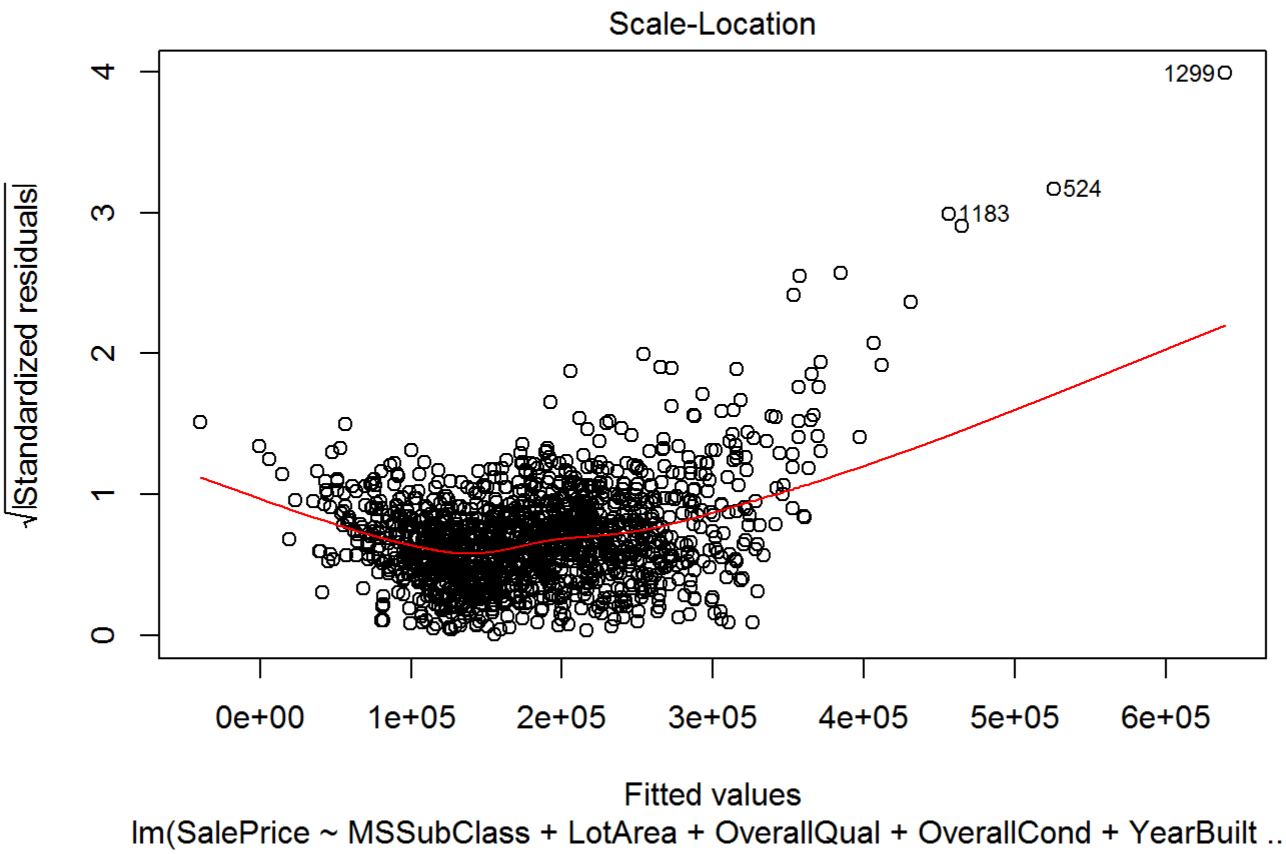
```
logLik(m3_back)
```

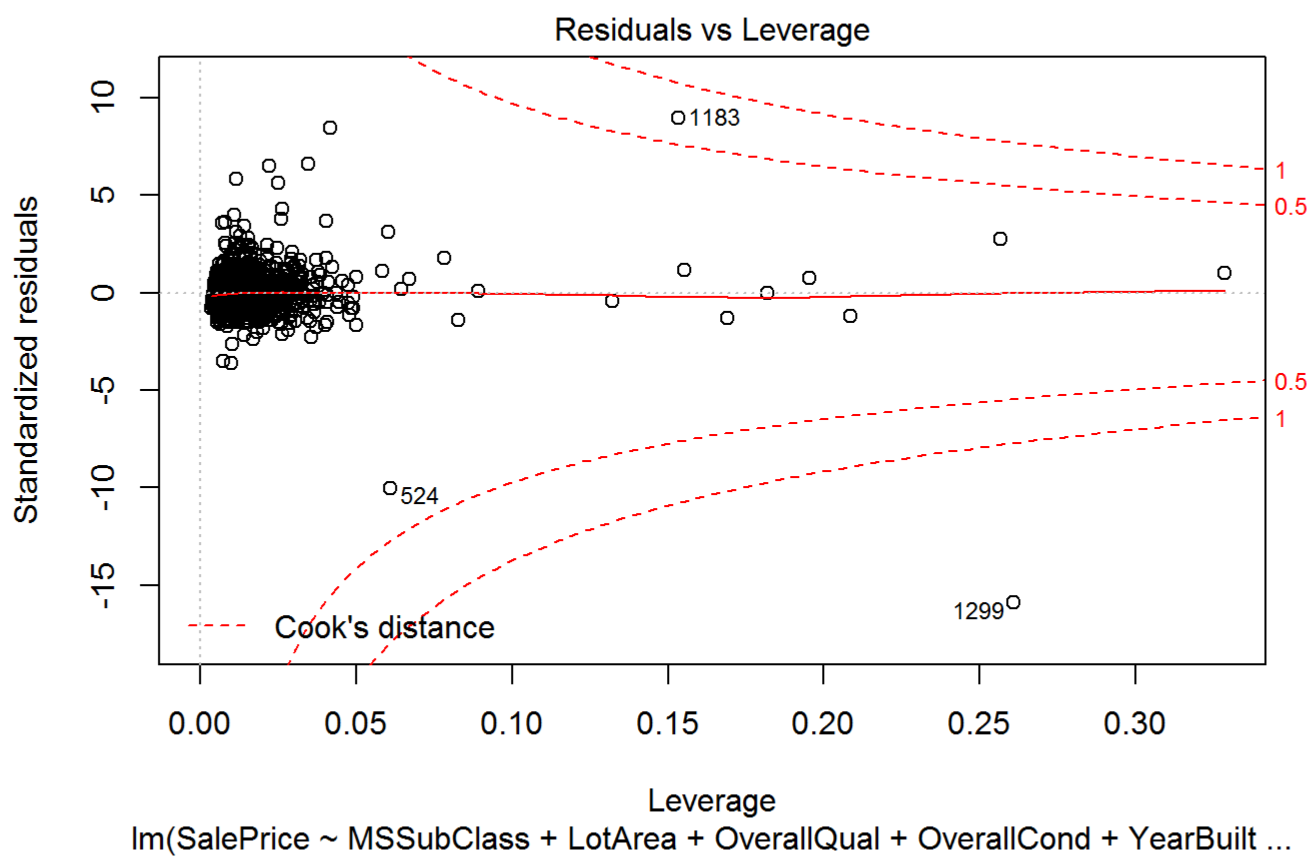
```
## 'log Lik.' -17338.8 (df=22)
```

```
plot(m3_back)
```

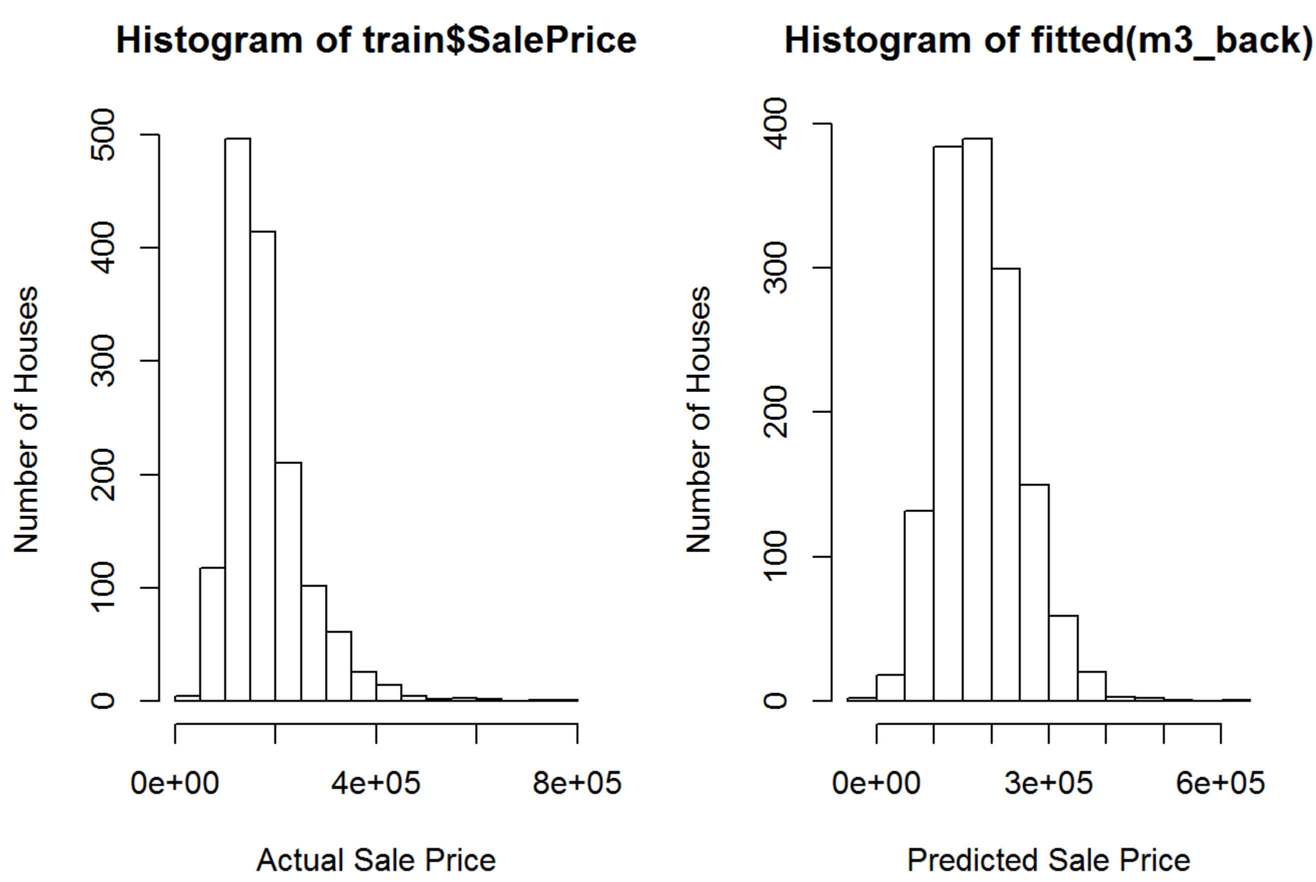

Residuals vs Fitted
Residuals
Fitted values
lm(SalePrice ~ MSSubClass + LotArea + OverallQual + OverallCond + YearBuilt ...
692
524
1299
2e+05
0e+00
-4e+05
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
6e+05


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(SalePrice ~ MSSubClass + LotArea + OverallQual + OverallCond + YearBuilt ...
1183
524
1299
10
5
0
-5
-10
-15
-3
-2
-1
0
1
2
3


Scale-Location
√|Standardized residuals|
Fitted values
lm(SalePrice ~ MSSubClass + LotArea + OverallQual + OverallCond + YearBuilt ...
1299
524
1183
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
6e+05


Residuals vs Leverage
Standardized residuals
Leverage
lm(SalePrice ~ MSSubClass + LotArea + OverallQual + OverallCond + YearBuilt ...
Cook's distance
1183
524
1299
1
0.5
0.00
0.05
0.10
0.15
0.20
0.25
0.30

```
par(mfcol=c(1,2))
hist(train$SalePrice, xlab = "Actual Sale Price", ylab = "Number of Houses")
hist(fitted(m3_back), xlab = "Predicted Sale Price", ylab = "Number of Houses")
```

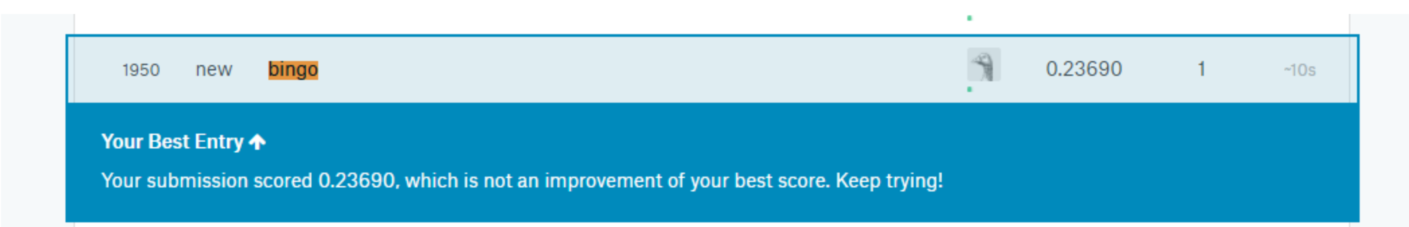


My final predicted results.

```
#Prepared the test data for evaluation
raw_data <- read.csv("https://raw.githubusercontent.com/blin261/data-605/master/test.csv", heade
r = TRUE, stringsAsFactors = FALSE)
test <- Filter(is.numeric, raw_data)
test <- test[ , colSums(is.na(test)) <= (0.05 * length(raw_data))]
str(test)
```

```
## 'data.frame':    1459 obs. of  34 variables:
##  $ Id           : int  1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 ...
##  $ MSSubClass   : int  20 20 60 60 120 60 20 60 20 20 ...
##  $ LotArea      : int  11622 14267 13830 9978 5005 10000 7980 8402 10176 8400 ...
##  $ OverallQual  : int  5 6 5 6 8 6 6 6 7 4 ...
##  $ OverallCond  : int  6 6 5 6 5 5 7 5 5 5 ...
##  $ YearBuilt    : int  1961 1958 1997 1998 1992 1993 1992 1998 1990 1970 ...
##  $ YearRemodAdd : int  1961 1958 1998 1998 1992 1994 2007 1998 1990 1970 ...
##  $ BsmtFinSF1   : int  468 923 791 602 263 0 935 0 637 804 ...
##  $ BsmtFinSF2   : int  144 0 0 0 0 0 0 0 0 78 ...
##  $ BsmtUnfSF    : int  270 406 137 324 1017 763 233 789 663 0 ...
##  $ TotalBsmtSF  : int  882 1329 928 926 1280 763 1168 789 1300 882 ...
##  $ X1stFlrSF    : int  896 1329 928 926 1280 763 1187 789 1341 882 ...
##  $ X2ndFlrSF    : int  0 0 701 678 0 892 0 676 0 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  896 1329 1629 1604 1280 1655 1187 1465 1341 882 ...
##  $ BsmtFullBath : int  0 0 0 0 0 0 1 0 1 1 ...
##  $ BsmtHalfBath : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  1 1 2 2 2 2 2 2 1 1 ...
##  $ HalfBath     : int  0 1 1 1 0 1 0 1 1 0 ...
##  $ BedroomAbvGr : int  2 3 3 3 2 3 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ TotRmsAbvGrd : int  5 6 6 7 5 7 6 7 5 4 ...
##  $ Fireplaces   : int  0 0 1 1 0 1 0 1 1 0 ...
##  $ GarageCars   : int  1 1 2 2 2 2 2 2 2 2 ...
##  $ GarageArea   : int  730 312 482 470 506 440 420 393 506 525 ...
##  $ WoodDeckSF   : int  140 393 212 360 0 157 483 0 192 240 ...
##  $ OpenPorchSF  : int  0 36 34 36 82 84 21 75 0 0 ...
##  $ EnclosedPorch: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ScreenPorch  : int  120 0 0 0 144 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MiscVal      : int  0 12500 0 0 0 0 500 0 0 0 ...
##  $ MoSold       : int  6 6 3 6 1 4 3 5 2 4 ...
##  $ YrSold       : int  2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
```

```
#Create the data frame with the final result
saleprice <- predict(m3_back, newdata = test, type = "response")
final <- data.frame(test$Id, saleprice)
colnames(final) <- c("Id", "SalePrice")
write.table(final, row.names = FALSE, file = "C:/Users/blin261/Desktop/DATA605/Final Project/kaggle_final.csv", sep=",")
```



Kaggle Final Result