# Final Project

CUNY MSDS DATA 605

*Duubar Villalobos Jimenez mydvtech@gmail.com*

*May 20, 2018*

## House Prices: Advanced Regression Techniques.

Predict sales prices and practice feature engineering, RFs, and gradient boosting.

### Read data

```
url <- paste( wd, '/data/', sep="")
train <- read.csv(paste(url, 'train.csv', sep = ''), header=TRUE, sep=",", stringsAsFactors=FALSE)
test <- read.csv(paste(url, 'test.csv', sep = ''), header=TRUE, sep=",", stringsAsFactors=FALSE)
```

### Picking quantitative independent variable

**Instruction:** Pick one of the quantitative independent variables from the training data set (train.csv), and define that variable as $X$. Make sure this variable is skewed to the right!

For this, I would like to answer the following question:

Is ________________, on average, higher or lower in home prices with high rates of value?

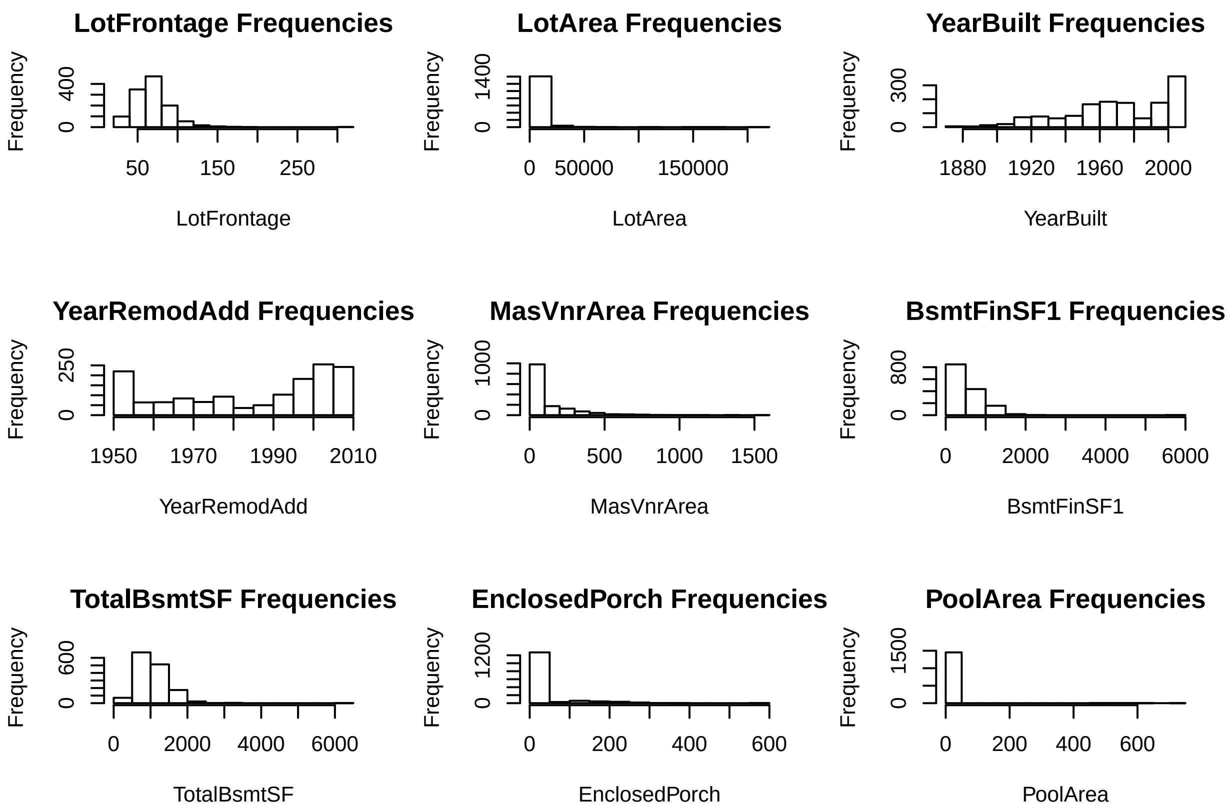
If we suspect __________________ might affect home prices, then __________________ is the independent (explanatory) variable and SalePrice is the dependent (response) variable in the relationship.

```
# Structure of the training dataset
str(train)
```

```
## 'data.frame':    1460 obs. of  81 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 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ 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 ...
## $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
## $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
## $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
## $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
## $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
## $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
## $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
## $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
## $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
## $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
## $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
## $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
## $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
## $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
## $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
## $ 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 ...
## $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
## $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
## $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
## $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
## $ 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 ...
## $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
## $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
## $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
## $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
## $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
## $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
## $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
## $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
## $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
## $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
## $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
## $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
## $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
## $ 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 ...
```

```
## $ PoolQC       : chr  NA NA NA NA ...
## $ Fence        : chr  NA NA NA NA ...
## $ MiscFeature  : chr  NA NA NA NA ...
## $ 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 ...
## $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
## $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
## $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```




I will pick **TotalBsmtSF** as $X$.

```
X <- train$TotalBsmtSF
```

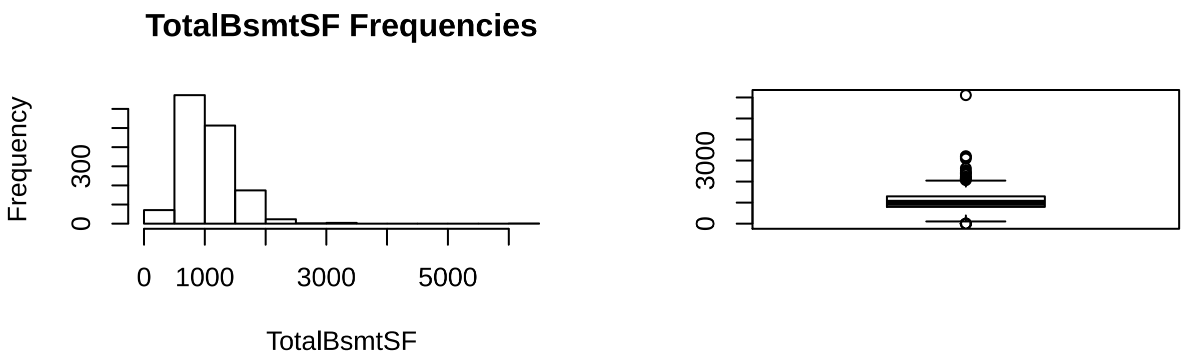
**Instruction:** Pick the dependent variable and define it as $Y$.

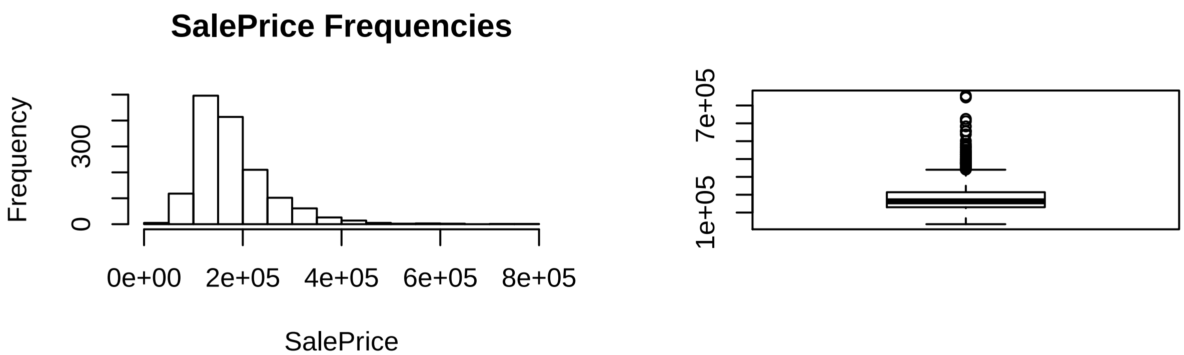
My dependent variable will be **SalePrice** as $Y$.

```
Y <- train$SalePrice
```

## Probability

Calculate as a minimum the below probabilities $a$ through $c$. Assume the small letter "x" is estimated as the 1st quartile of the X variable, and the small letter "y" is estimated as the 1st quartile of the Y variable. Interpret the meaning of all probabilities. In addition, make a table of counts as shown below.




```
#X[is.na(X)] <- 0 # No need, complete data readings
x <- quantile(X, c(.25))

#Y[is.na(Y)] <- 0 # No need, complete data readings
y <- quantile(Y, c(.25))
```

**a.** $P(X > x \,|\, Y > y)$

Let's evaluate the probability that X > 795.75 **given** Y > 129975.

```
p <- mean(Y[X > x] > y)
```

**Answer:** The probability of $P(X > x \,|\, Y > y) = 0.8301$ or 83.01%.

That is, 83.01 % of the houses will have a price over $129975 when the total square feet of basement area is over 795.75.

**b.** $P(X > x \,,\, Y > y)$

Let's evaluate the probability that X > 795.75 **and** Y > 129975.

```
px <- mean(X > x)
py <- mean(Y > y)
p <- px * py
```

**Answer:** The probability of $P(X > x \,,\, Y > y) = 0.5625$ or 56.25%.

That is, there's a 56.25 % probability of finding a house with a price over $129975 and a total square feet of basement area is over 795.75.

**c.** $P(X < x \mid Y > y)$

Let's evaluate the probability that X < 795.75 **given** Y > 129975.

```
p <- mean(Y[X < x] > y)
```

**Answer:** The probability of $P(X < x \mid Y > y) = 0.5096$ or 50.96%.

That is, 50.96 % of the houses will have a price over $129975 when the total square feet of basement area is less than 795.75.

**Table of Counts:**

Quartile Counts

<=2d quartile

>2d quartile

Total

<=3d quartile

696

399

1095

>3d quartile

36

329

365

Total

732

728

1460

**Question:** Does splitting the training data in this fashion make them independent?

**Answer:** By splitting the data in this fashion, it does not make the variables independent, but it offer a little bit more freedom in terms of working out with this data for insights purposes.

### Defining A & B from quartiles

Let A be the new variable counting those observations above the 1st quartile for X.

```
A <- sum(X > x)
```

Let B be the new variable counting those observations above the 1st quartile for Y.

```
B <- sum(Y > y)
```

**Does $P(AB) = P(A)P(B)$?**

Check mathematically, and then evaluate by running a Chi Square test for association.

In this case, A and B are not considered independent since B has influence from A due to the origin of our data.

```
p <- (A / 1460) * (B / 1460)
```

In this case the probability won't be considered to be P(AB) = 0.5625.

**Chi Square test for association.**

A and B Counts

A

B

Counts

1095

1095

The hypotheses can be described as the following:

$H_0$ : The counts for each column are the same.

$H_A$ : The counts for each column are not the same.

There are two conditions that must be checked before performing a chi-square test:

**Independence.** Each case that contributes a count to the table must be independent of all the other cases in the table. (This condition is not necessarily met since B comes from SalePrice and it's believed to be affected by TotalBsmtSF).

**Sample size / distribution.** Each particular scenario (i.e. cell count) must have at least 5 expected cases.

Failing to check conditions may affect the test's error rates.

```
##
##  Chi-squared test for given probabilities
##
## data:  AB
## X-squared = 0, df = 1, p-value = 1
```

In this case, the p-value for this test statistic is found by looking at the upper tail of this chisquare distribution. We consider the upper tail because larger values of 2 would provide greater evidence against the null hypothesis.

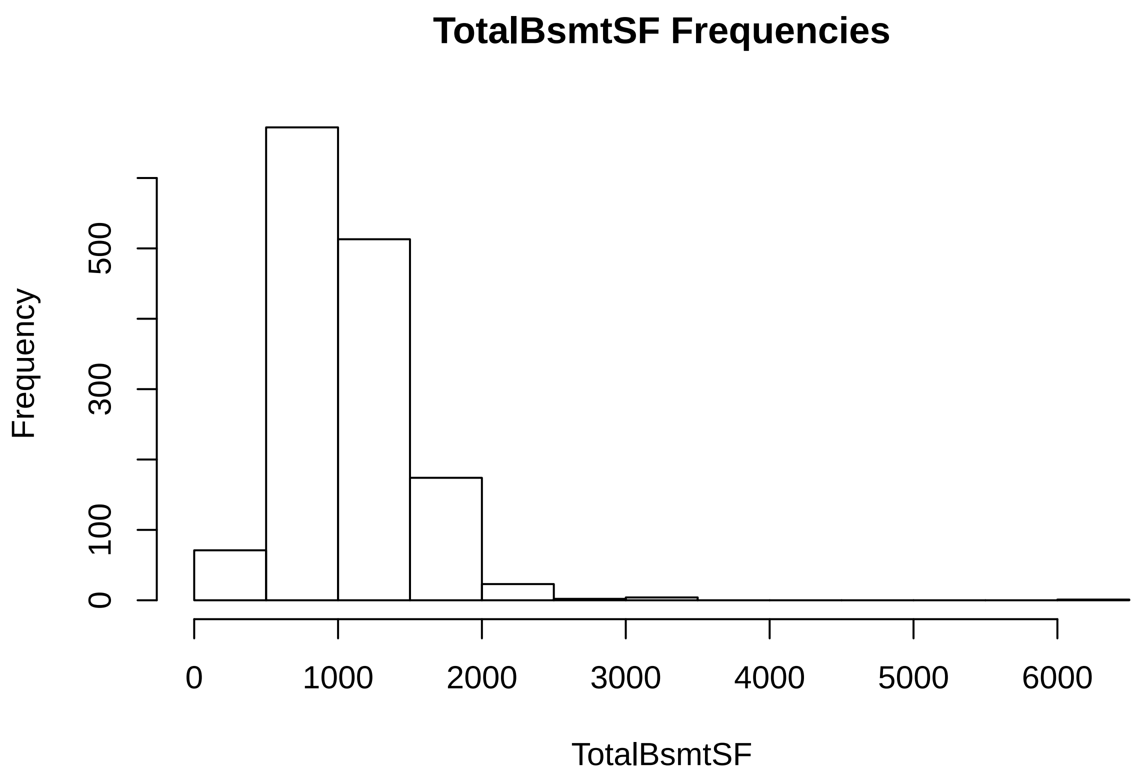
Due to the p value being so large, we reject the null hypothesis and accept the alternate hypothesis.

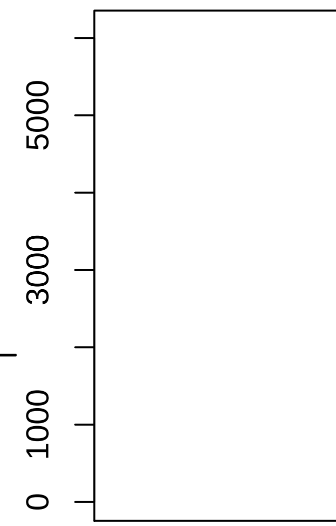
# Descriptive Statistics

Below is a description for our selected data sets X and Y.

```
summary(X)
```

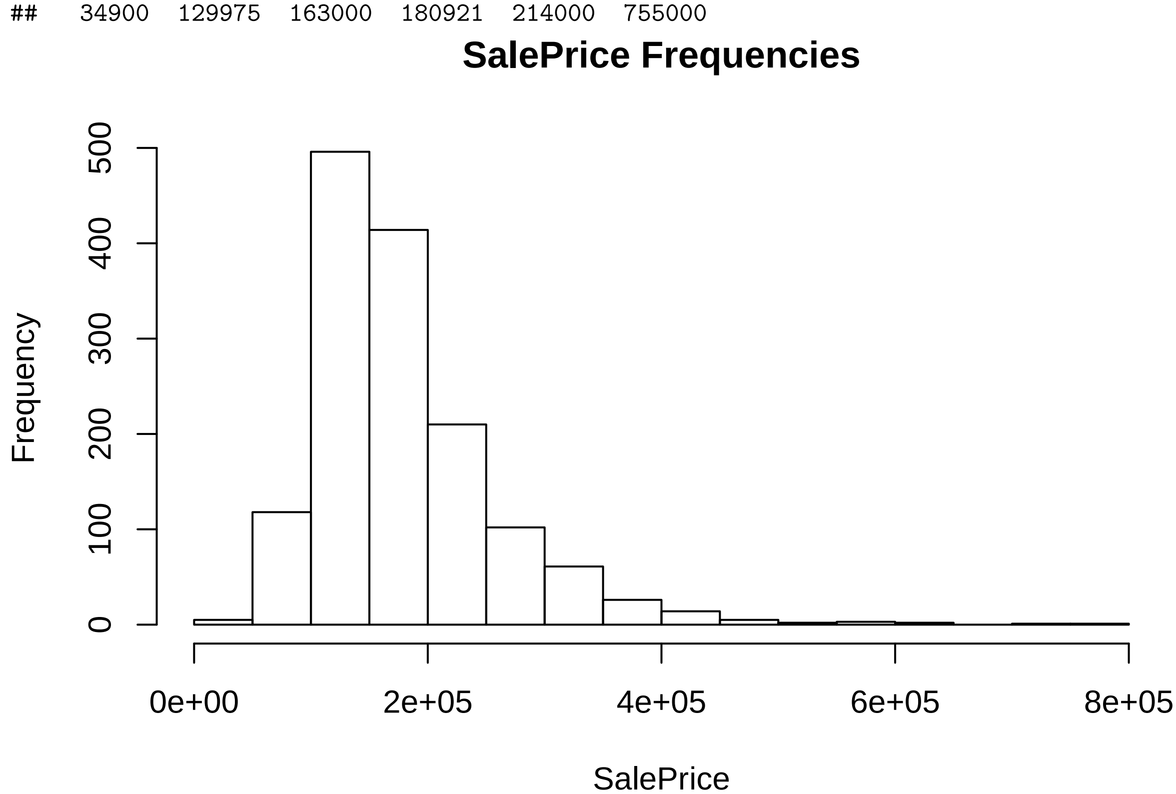
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   795.8   991.5  1057.4  1298.2  6110.0
```

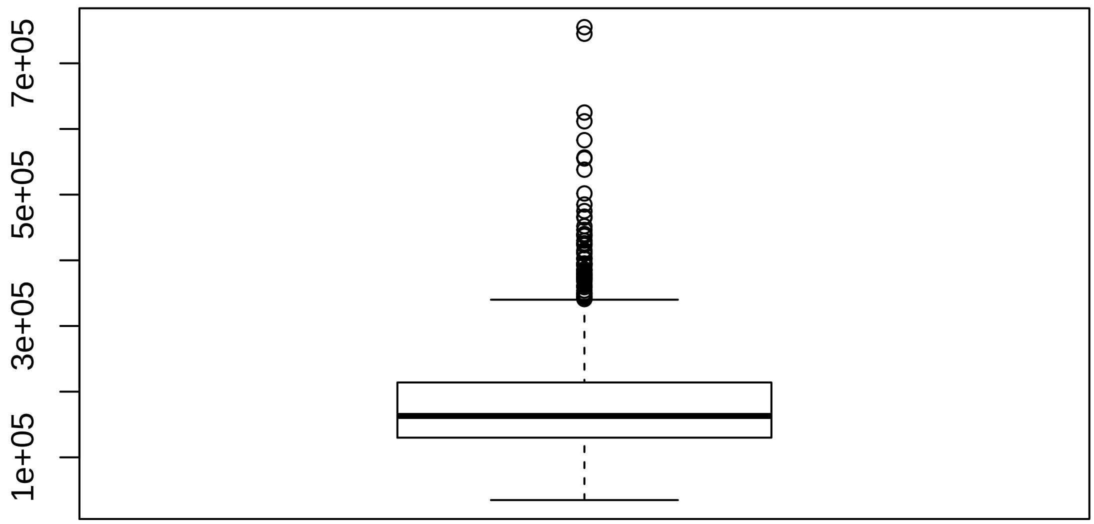


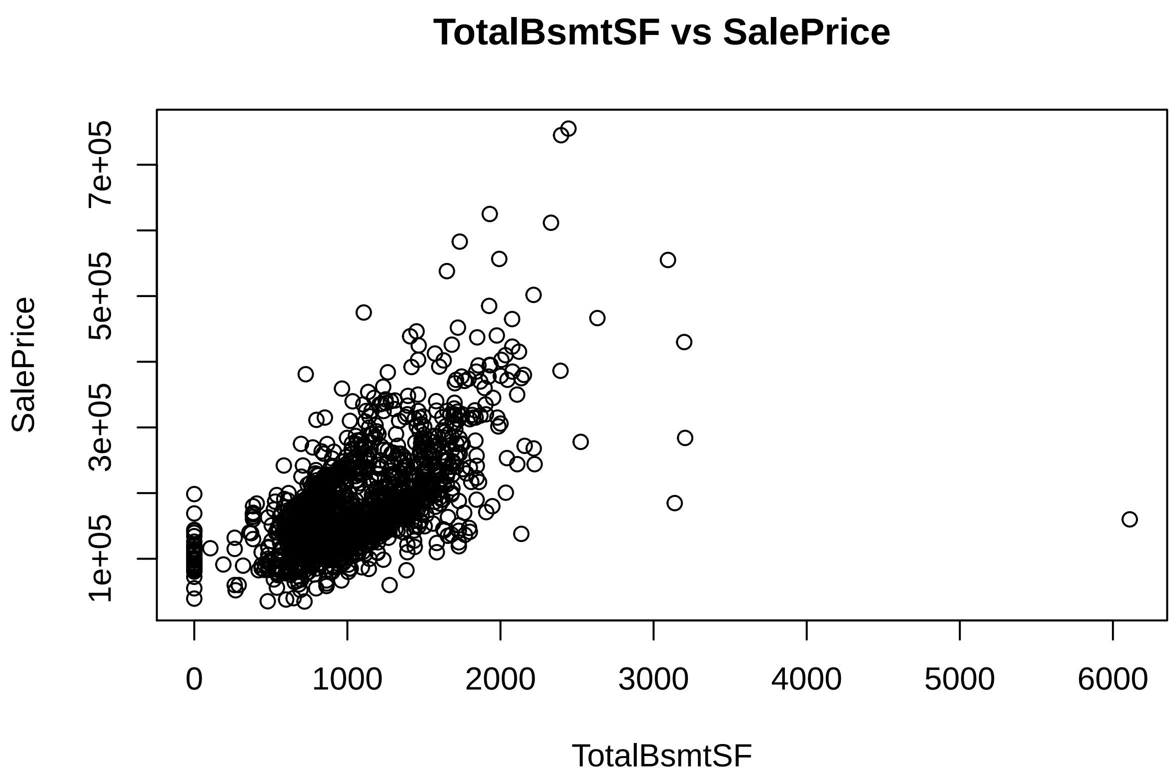




```r
summary(Y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```






**Correlation matrix**

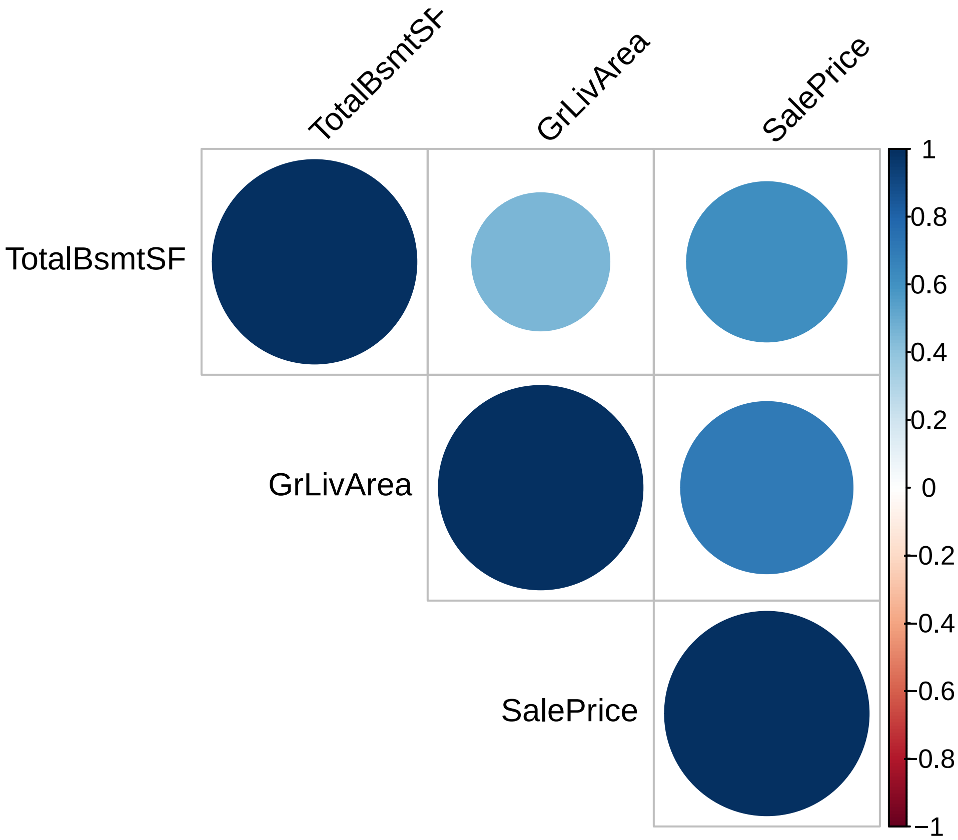
Derive a correlation matrix for any THREE quantitative variables in the data set.

My picks will be: `TotalBsmtSF`, `GrLivArea` and `SalePrice`.

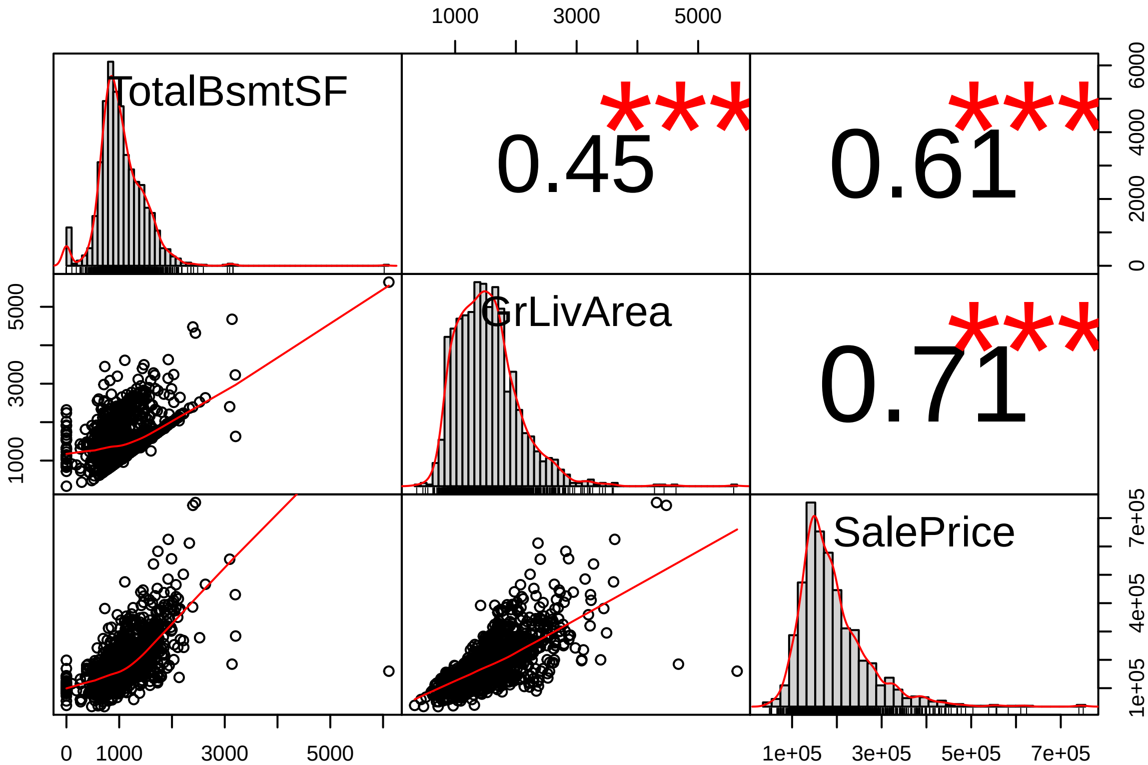
```
myvars <- c("TotalBsmtSF", "GrLivArea", "SalePrice")
my_matrix <- train[myvars]

cor_res <- cor(my_matrix)
round(cor_res, 2)
```

```
##             TotalBsmtSF GrLivArea SalePrice
```

```
## TotalBsmtSF        1.00      0.45      0.61
## GrLivArea          0.45      1.00      0.71
## SalePrice          0.61      0.71      1.00
```



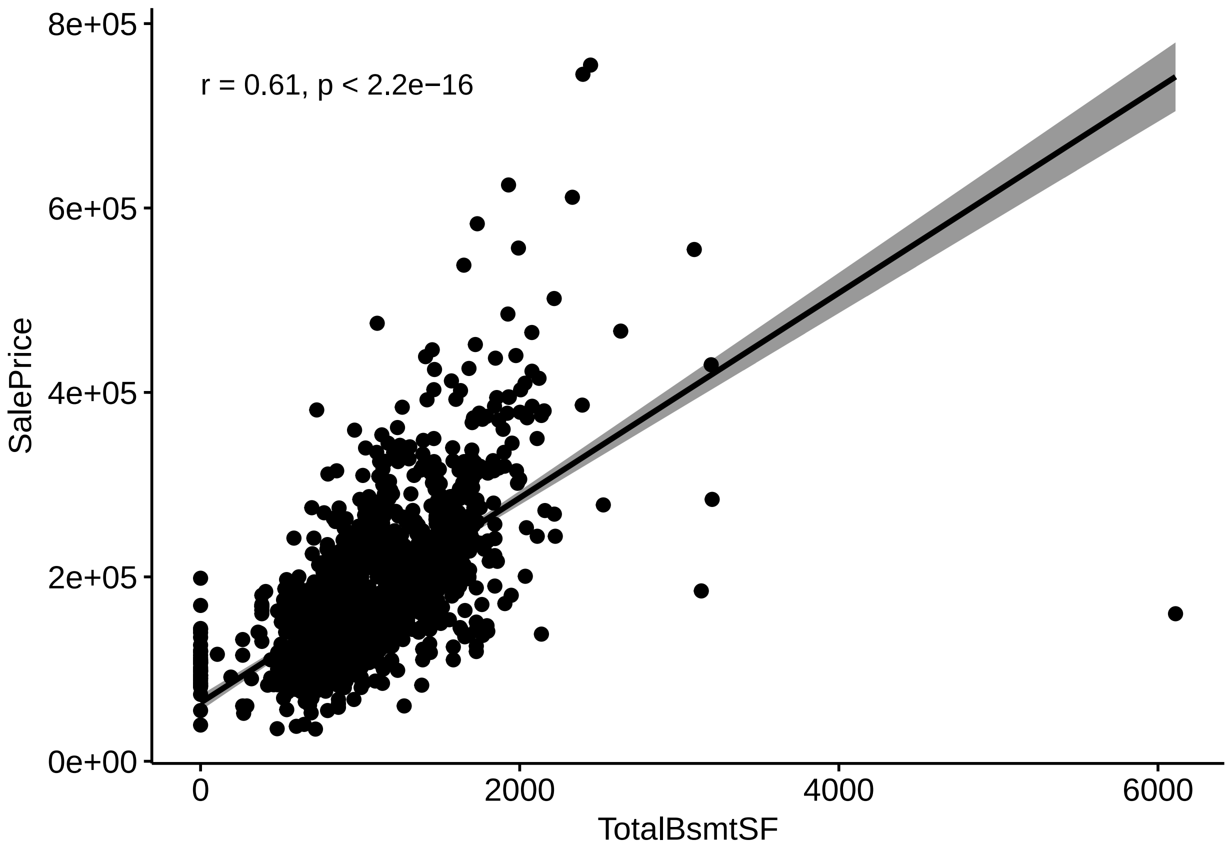

Positive correlations are displayed in blue and negative correlations in red color. Color intensity and the size of the circle are proportional to the correlation coefficients. In the right side of the correlogram, the legend color shows the correlation coefficients and the corresponding colors.



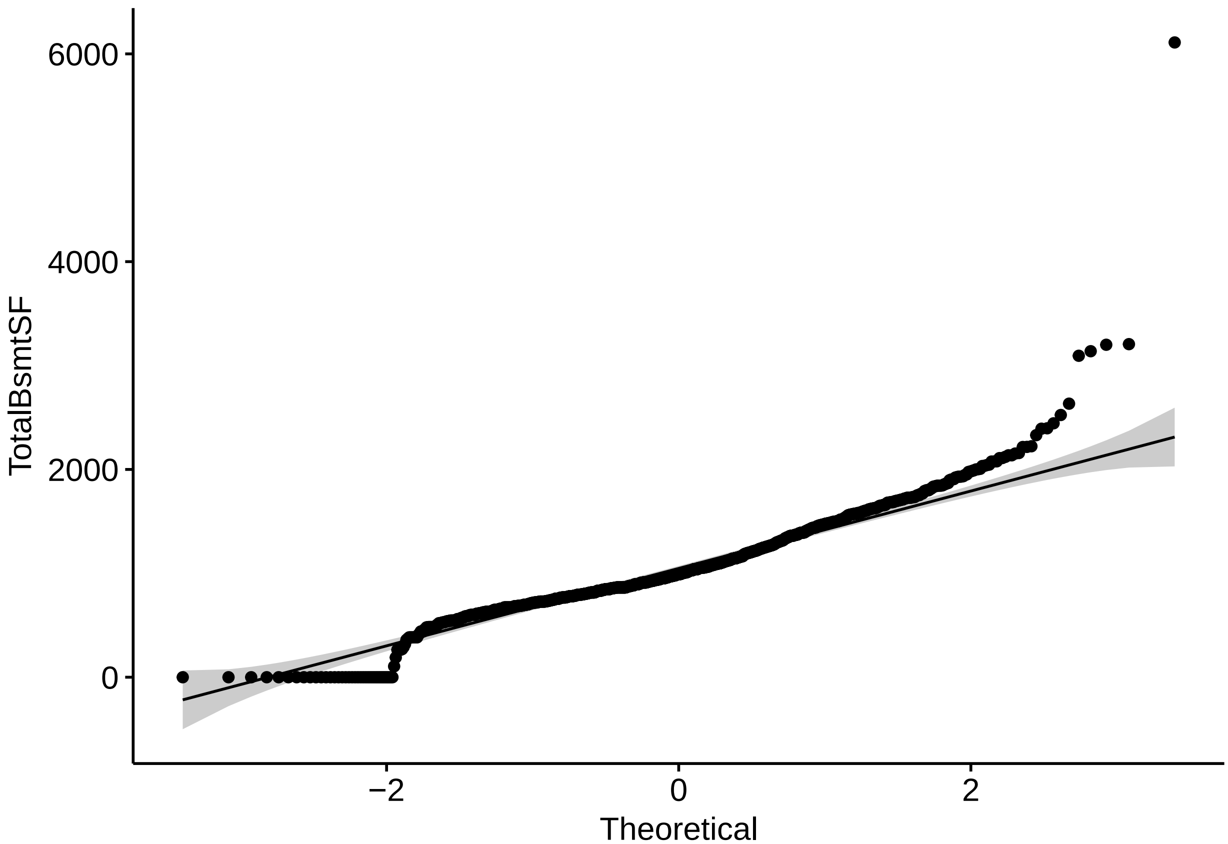

In the above plot:

```
The distribution of each variable is shown on the diagonal.
On the bottom of the diagonal : the bivariate scatter plots with a fitted line are displayed
On the top of the diagonal : the value of the correlation plus the significance level as stars
Each significance level is associated to a symbol : p-values(0, 0.001, 0.01, 0.05, 0.1, 1) <=> symbols(
```

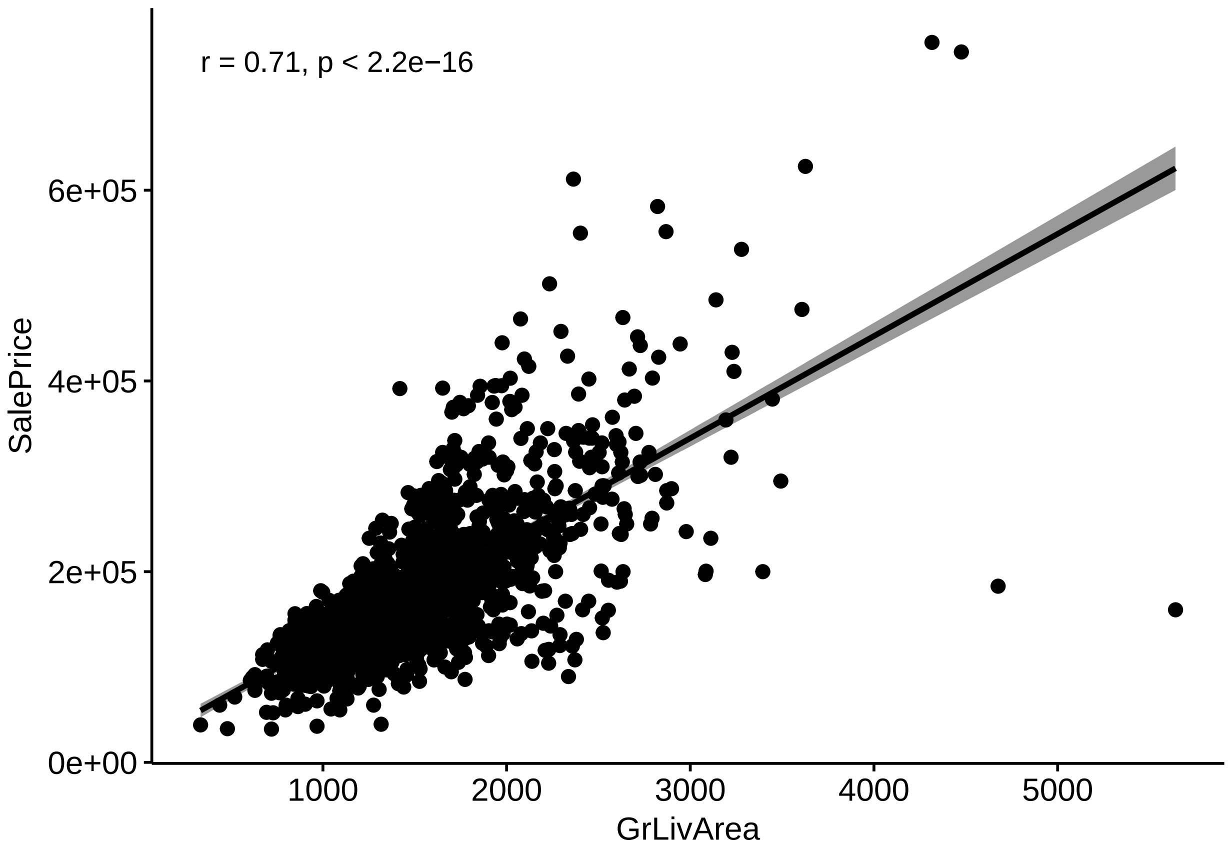
Visualization of `TotalBsmtSF` vs `SalePrice`.



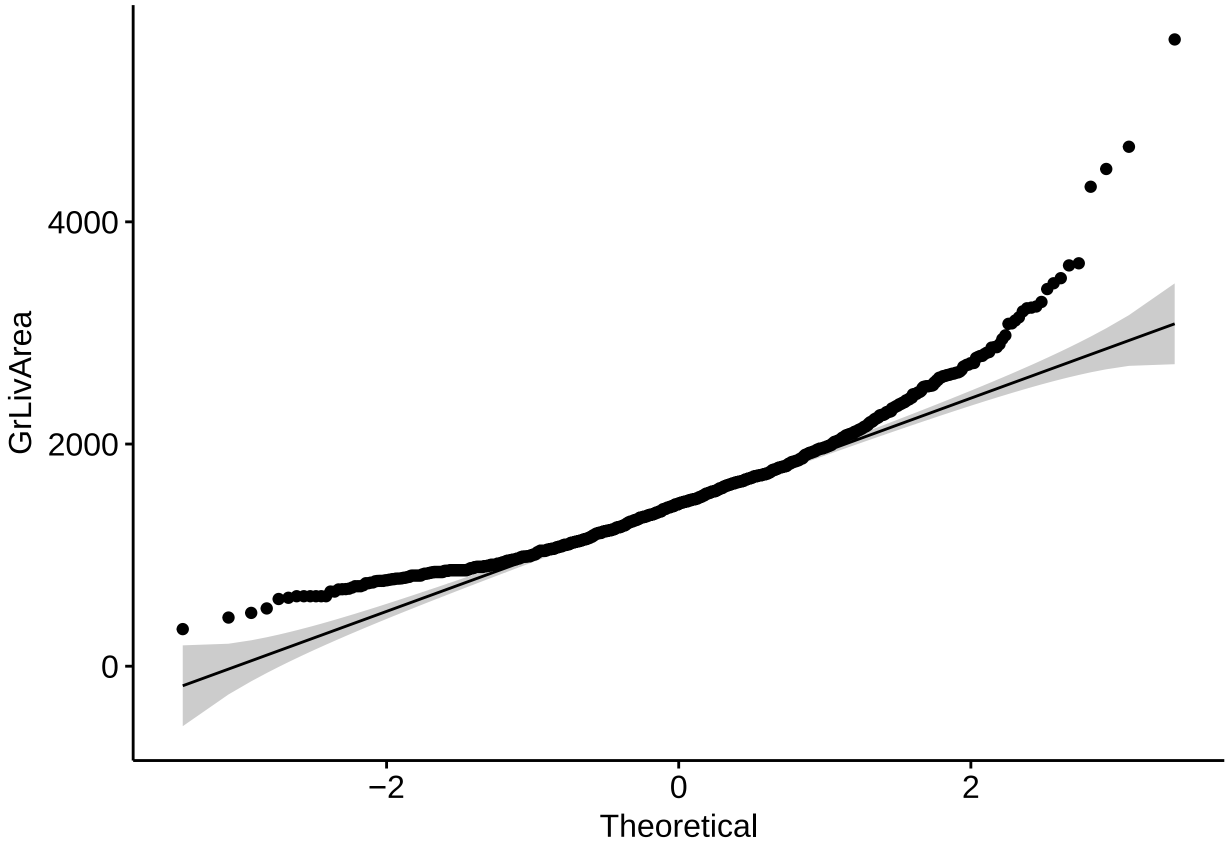

Q-Q plots (quantile-quantile plots) for `TotalBsmtSF`.



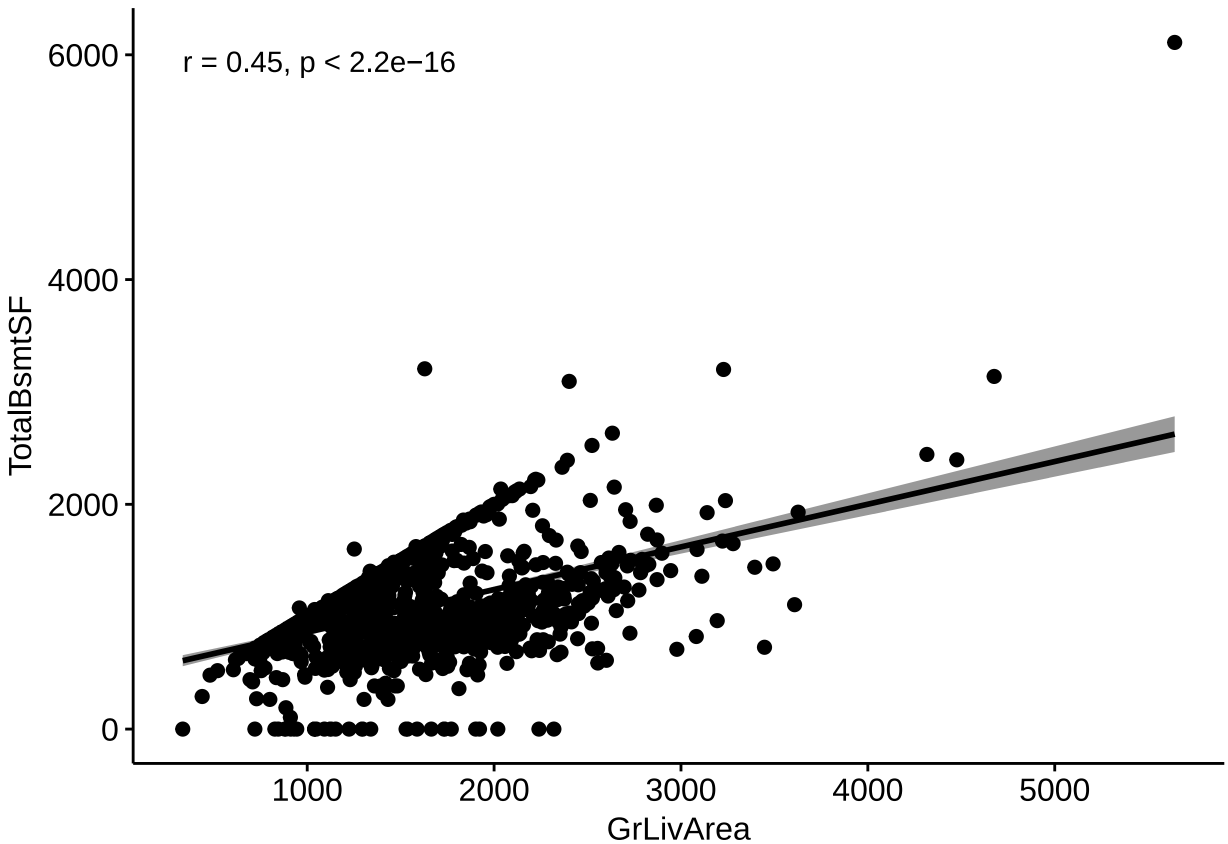

Visualization of `GrLivArea` vs `SalePrice`.



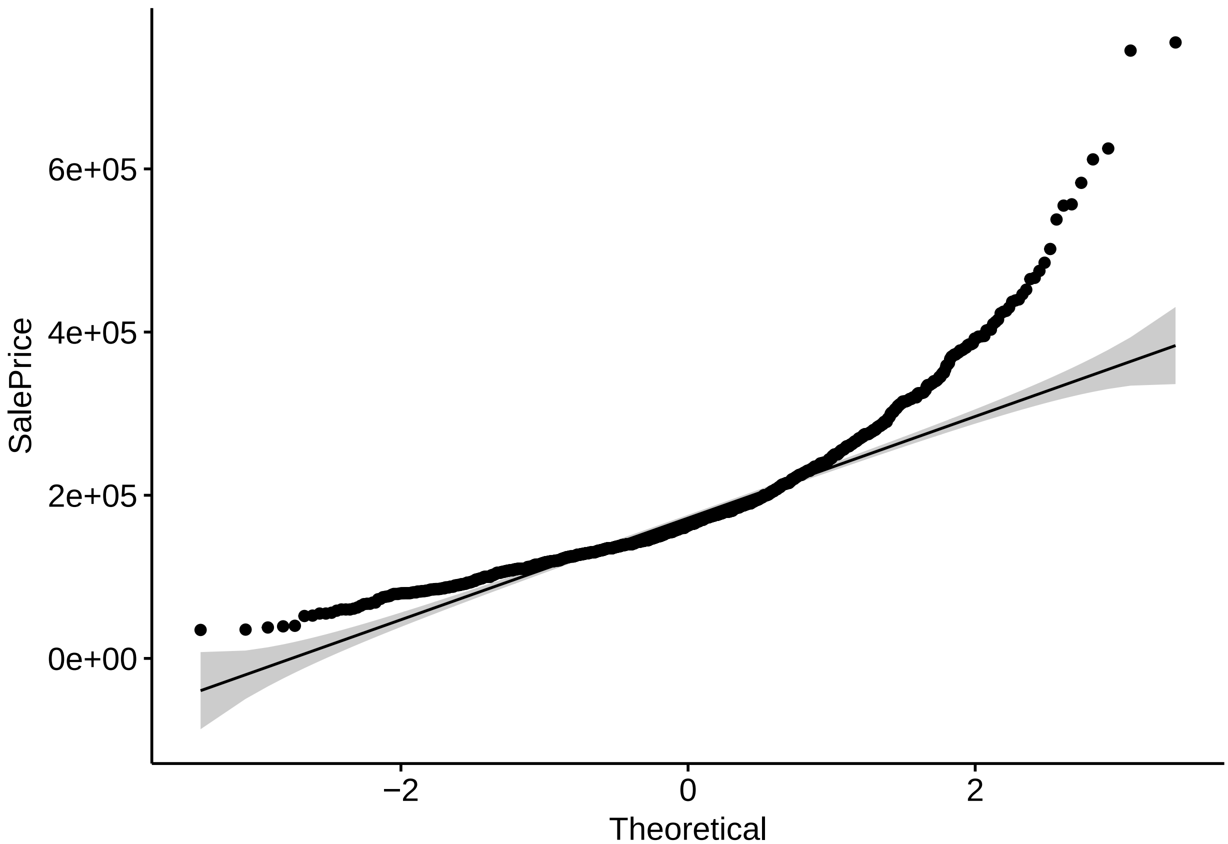

Q-Q plots (quantile-quantile plots) for `GrLivArea`.



Visualization of `GrLivArea` vs `TotalBsmtSF`.




Q-Q plots (quantile-quantile plots) for `SalePrice`.




**Testing hypotheses on correlations**

Testing: `TotalBsmtSF` vs `GrLivArea` with 92% confidence interval.

```
res <- cor.test(my_matrix$TotalBsmtSF, my_matrix$GrLivArea, conf.level = 0.92, method = "pearson")
res
```

```
##
##  Pearson's product-moment correlation
##
## data:  my_matrix$TotalBsmtSF and my_matrix$GrLivArea
## t = 19.503, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 92 percent confidence interval:
##  0.4177447 0.4904754
## sample estimates:
##       cor
## 0.4548682
```

In the result above :

```
t is the t-test statistic value (t = 19.503),
df is the degrees of freedom (df= 1458),
p-value is the significance level of the t-test (p-value = 2.2^{-16}).
conf.int is the confidence interval of the correlation coefficient at 92% (conf.int = [0.4177447, 0.490
sample estimates is the correlation coefficient (Cor.coeff = 0.45).
```

Testing: `TotalBsmtSF` vs `SalePrice` with 92% confidence interval.

```r
res <- cor.test(my_matrix$TotalBsmtSF, my_matrix$SalePrice, conf.level = 0.92, method = "pearson")
res
```

```
##
##  Pearson's product-moment correlation
##
## data:  my_matrix$TotalBsmtSF and my_matrix$SalePrice
## t = 29.671, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 92 percent confidence interval:
##  0.5841762 0.6413763
## sample estimates:
##       cor
## 0.6135806
```

In the result above :

```
t is the t-test statistic value (t = 29.671),
df is the degrees of freedom (df= 1458),
p-value is the significance level of the t-test (p-value = 2.2^{-16}).
conf.int is the confidence interval of the correlation coefficient at 92% (conf.int = [0.5841762, 0.641
sample estimates is the correlation coefficient (Cor.coeff = 0.61).
```

Testing: `GrLivArea` vs `SalePrice` with 92% confidence interval.

```r
res <- cor.test(my_matrix$GrLivArea, my_matrix$SalePrice, conf.level = 0.92, method = "pearson")
res
```

```
##
##  Pearson's product-moment correlation
##
## data:  my_matrix$GrLivArea and my_matrix$SalePrice
## t = 38.348, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 92 percent confidence interval:
##  0.6850407 0.7307245
## sample estimates:
##       cor
## 0.7086245
```

In the result above :

```
t is the t-test statistic value (t = 38.348),
df is the degrees of freedom (df= 1458),
p-value is the significance level of the t-test (p-value = 2.2^{-16}).
conf.int is the confidence interval of the correlation coefficient at 92% (conf.int = [0.6850407, 0.730
sample estimates is the correlation coefficient (Cor.coeff = 0.71).
```

In all three cases, our null hypothesis got discarded in favor of the alternate hypothesis.

**Would you be worried about familywise error? Why or why not?**

This relates to the Type I or Type II Errors.

In the above examples, I would not be too worry about family errors, and the reason is due to the nature of the data we can live with the possible uncertainty of False positives; in this case the extremely low p-value and the moderate “high” values of the correlation in between the variables offset the being worry part of it.

# Linear Algebra and Correlation

## Invert the Correlation matrix

**Instruction:** Invert your 3 x 3 correlation matrix from above. (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. Conduct LU decomposition on the matrix.

The original correlation matrix is as follows:

```
##             TotalBsmtSF GrLivArea SalePrice
## TotalBsmtSF        1.00      0.45      0.61
## GrLivArea          0.45      1.00      0.71
## SalePrice          0.61      0.71      1.00
```

**solve(A)** Inverse of A where A is a square matrix.

```
##             TotalBsmtSF GrLivArea SalePrice
## TotalBsmtSF        1.61     -0.06     -0.94
## GrLivArea         -0.06      2.01     -1.39
## SalePrice         -0.94     -1.39      2.56
```

Please note that the variance inflation factors (previously rounded to two decimals) are:

```
diag(precision_matrix)
```

```
## TotalBsmtSF   GrLivArea   SalePrice
##    1.605884    2.011242    2.558231
```

Multiply the correlation matrix by the precision matrix, and then multiply the precision matrix by the correlation matrix.

```
##             TotalBsmtSF GrLivArea SalePrice
## TotalBsmtSF           1         0         0
## GrLivArea             0         1         0
## SalePrice             0         0         1
```

```
round(precision_matrix  %*% cor_res, 0)
```

```
##             TotalBsmtSF GrLivArea SalePrice
## TotalBsmtSF           1         0         0
## GrLivArea             0         1         0
## SalePrice             0         0         1
```

We can quickly verify that this is the inverse by applying $A \cdot A^{-1} = I$ this returns the identity matrix.

## LU decomposition

Conduct LU decomposition on the matrix.

```
LU_decomp <- lu.decomposition(precision_matrix)
```

### L Matrix

```
##           [,1]      [,2] [,3]
## [1,]  1.000000  0.000000    0
## [2,] -0.040313  1.000000    0
## [3,] -0.585014 -0.708624    1
```

### U Matrix

```
##          [,1]      [,2]      [,3]
## [1,] 1.605884 -0.064738 -0.939464
## [2,] 0.000000  2.008632 -1.423366
## [3,] 0.000000  0.000000  1.000000
```

## Calculus-Based Probability & Statistics

Many times, it makes sense to fit a closed form distribution to data. For the first variable that you selected which is skewed to the right, shift it so that the minimum value is above zero as necessary. Then load the MASS package and run fitdistr to fit an exponential probability density function. (See https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/fitdistr.html ).

### Finding $\lambda$

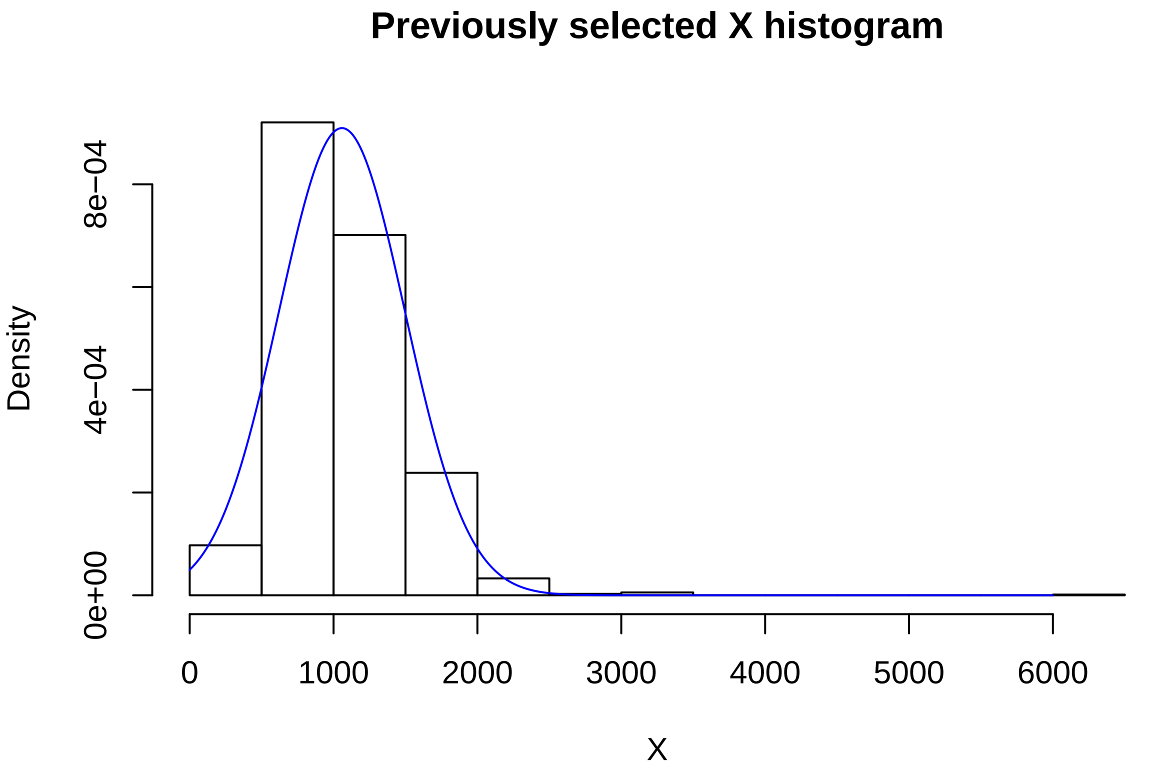
```r
# Adding 1 in order to bring the values above zero for the minimum values.
X <- X + 1

lamda <- fitdistr(X, 'exponential')
```

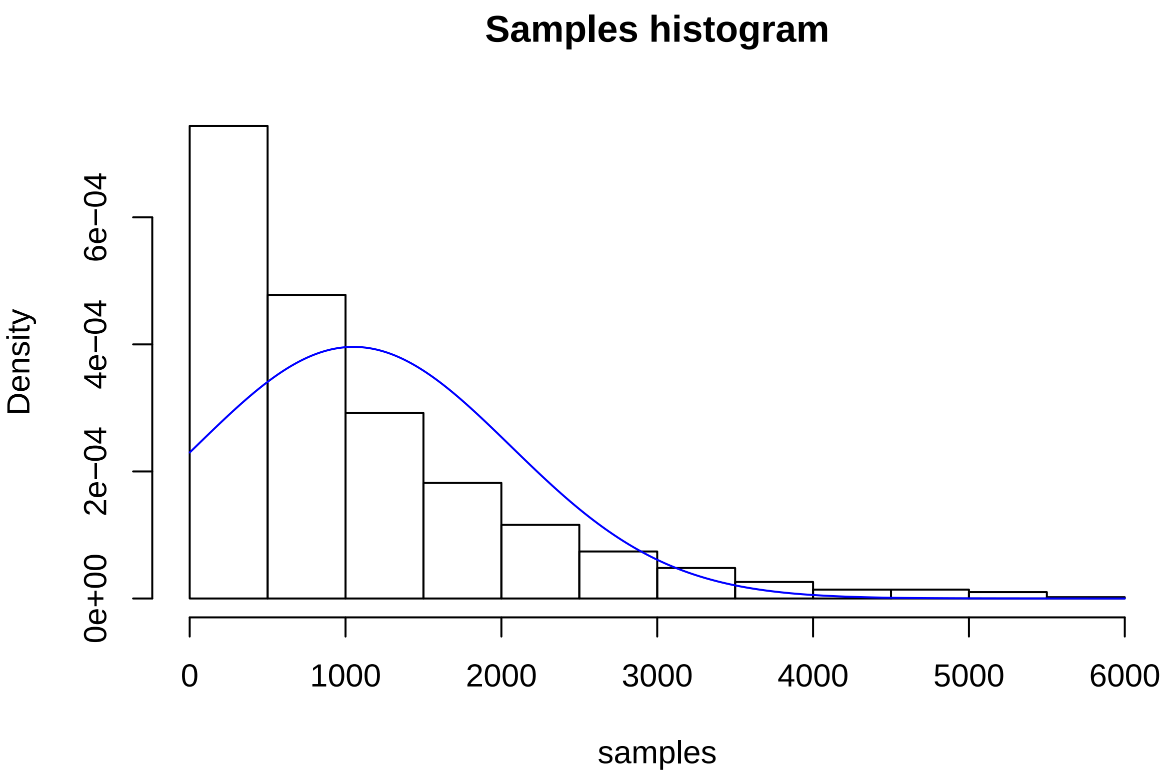
The optimal value $\lambda$ will be: 0.0009447961.

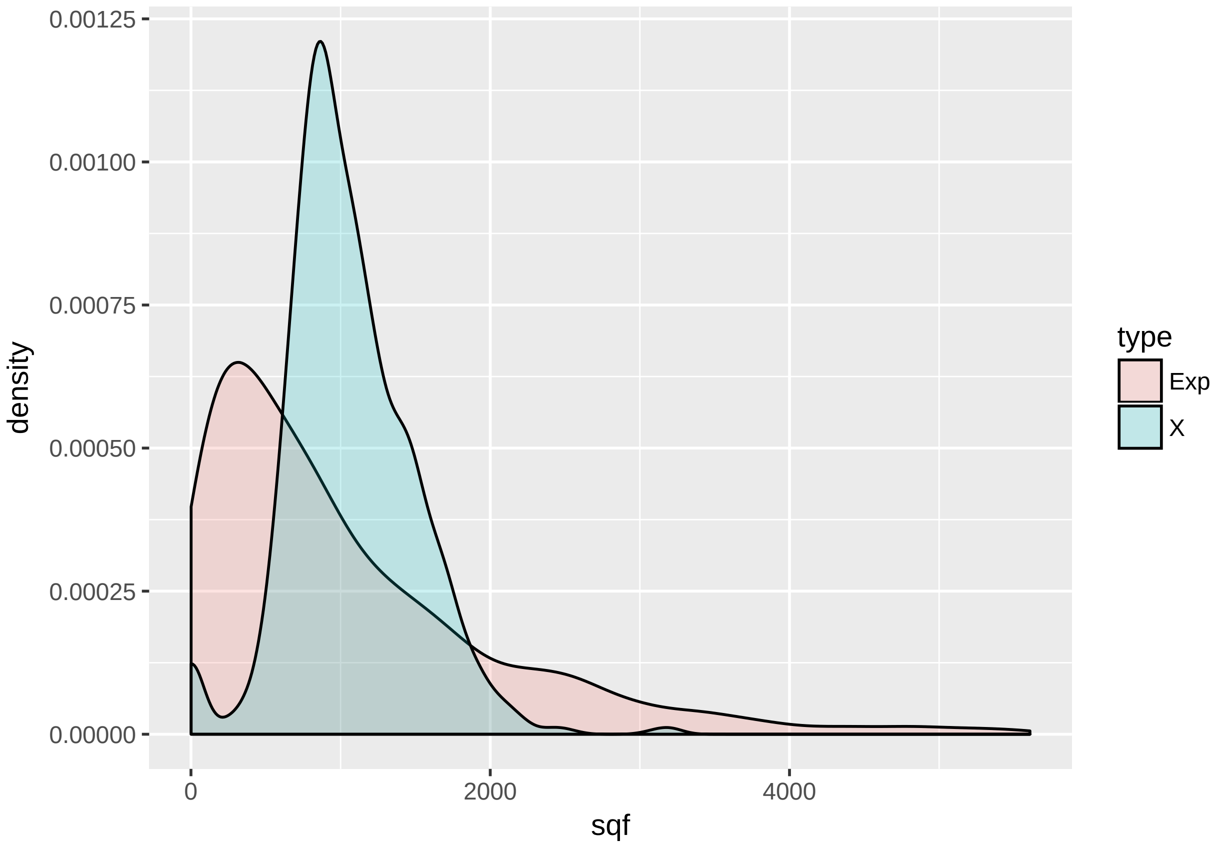
### 1000 Samples

Take 1000 samples using this $\lambda$.

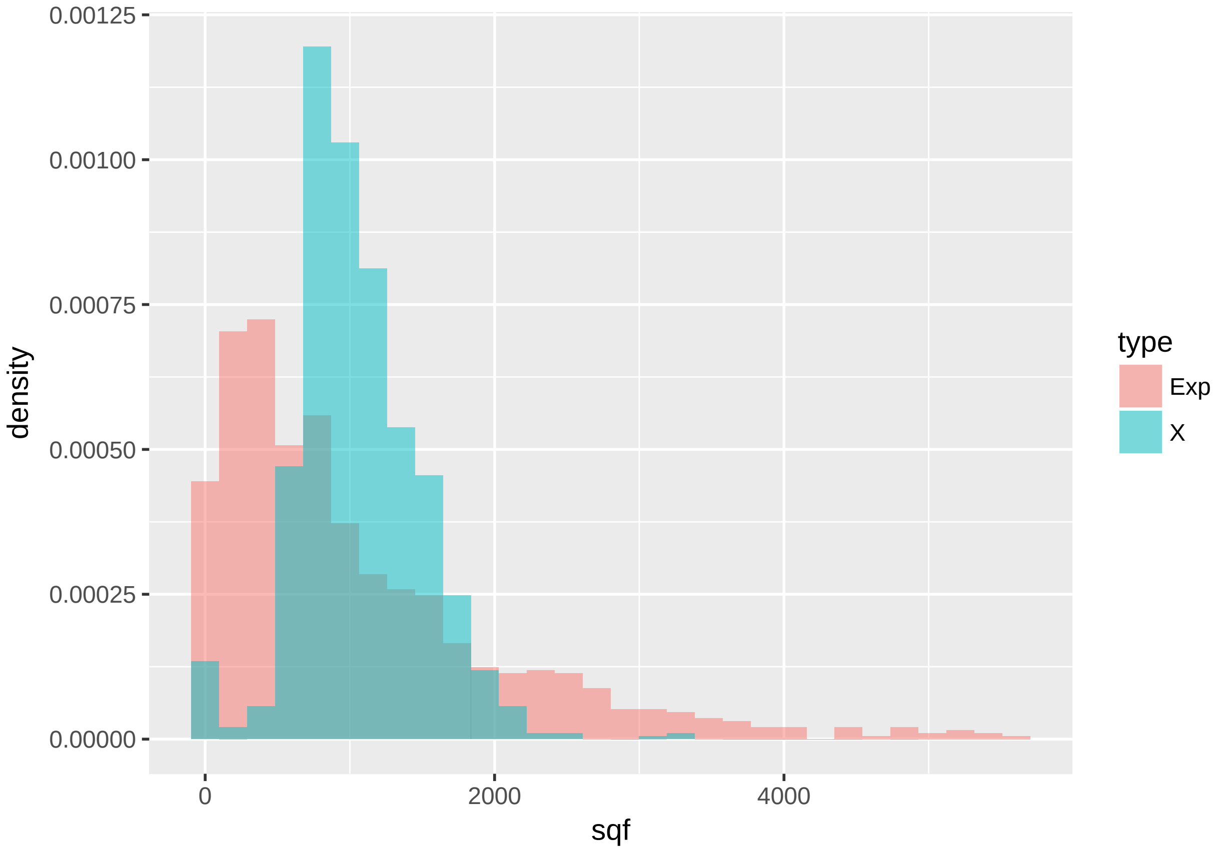
```r
fitdistr_sample <- rexp(1000, lamda$estimate)
```


Previously selected X histogram
Density
0e+00
4e−04
8e−04
0
1000
2000
3000
4000
5000
6000
X


Samples histogram
Density
0e+00
2e−04
4e−04
6e−04
0
1000
2000
3000
4000
5000
6000
samples




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




### Find Percentiles using the exponential PDF.

```
qlow <- qexp(0.05,lamda$estimate)
qup <- qexp(0.95,lamda$estimate)
```

The 5th percentile will be 54.29.

The 95th percentile will be 3170.77.

Comparing to quantiles for `X`.

```
quantile(X, c(.05, 0.95))
```

```
##     5%    95%
##  520.3 1754.0
```

Comparing to quantiles for `fitdistr_sample`.

```
round(quantile(fitdistr_sample, c(.05, 0.95)),2)
```

```
##      5%     95%
##   56.66 3086.72
```

## Confidence Interval

#### Confidence interval on Exponential sample data.

```r
#se <- mean(fitdistr_sample) / sqrt(1000)
se <- 1 / (lamda$estimate * sqrt(1000))
lower = mean(fitdistr_sample) - 1.96 * se
upper = mean(fitdistr_sample) + 1.96 * se
```

The 95% Confidence interval on the exponential approximation will be from 986 to 1117.

**Confidence interval on provided training data assuming normality.**

```r
lower <- qnorm(.05,mean(X),sd(X))
upper <- qnorm(.95,mean(X),sd(X))
```

The 95% Confidence interval on the provided training data set will be from 337 to 1780.

From the above we can conclude that we could approximate this data sets in a "Normalized fashion", even though the analysis was performed with exponential distribution; the above due to we quickly identify that in our particular case, the exponential distribution does not follow or perform a good match for our data set since the tentative values are way off from our real values from the data set. The normal distribution seems to provide better approximations.

# Modeling

## Removing "duplicate" or "non valuable information" columns

```r
# Removing basement "duplicate" columns
removevars <- names(train) %in% c("Id", "BsmtExposure","BsmtFinType1", "BsmtFinSF1", "BsmtFinType2", "B
my_train <- train[!removevars]
```

```r
# Removing Bath "duplicate" columns
removevars <- names(my_train) %in% c("HalfBath")
my_train <- my_train[!removevars]
```

```r
# Removing Fireplace "duplicate" columns
removevars <- names(my_train) %in% c("FireplaceQu")
my_train <- my_train[!removevars]
```

```r
# Removing Garage "duplicate" columns
removevars <- names(my_train) %in% c("GarageYrBlt", "GarageYrBlt", "GarageCars", "GarageQual", "GarageC
my_train <- my_train[!removevars]
```

```r
# Removing Floors "duplicate" columns
removevars <- names(my_train) %in% c("X2ndFlrSF", "LowQualFinSF")
my_train <- my_train[!removevars]
```

```r
# Removing Kitchen "duplicate" columns
removevars <- names(my_train) %in% c("KitchenAbvGr")
my_train <- my_train[!removevars]
```

```r
# Removing Porch "duplicate" columns
my_train$PorchSF <- my_train$OpenPorchSF + my_train$EnclosedPorch + my_train$X3SsnPorch + my_train$Scre
removevars <- names(my_train) %in% c("OpenPorchSF", "EnclosedPorch", "X3SsnPorch", "ScreenPorch")
my_train <- my_train[!removevars]
```

```r
# Removing Misc "duplicate" columns
removevars <- names(my_train) %in% c("PoolQC", "Fence", "MiscFeature", "MiscVal")
```

```
my_train <- my_train[!removevars]
```

After some "clean up" I end up with 57 columns; from here onward I will start to do some extra clean up in terms of creating "dummy" values for the categorical values and taking care for NA values.

## Taking care of NA Values

```
# MSSubClass
# No need to assign, already has correct data
```

```
# Assigning "Dummy values" MSZoning
#my_train$MSZoning[is.na(my_train$MSZoning) == TRUE] <- 0 # Treating NA as non existent since there wer
my_train$MSZoning[my_train$MSZoning == 'A'] <- 0
my_train$MSZoning[my_train$MSZoning == 'C'] <- 1
my_train$MSZoning[my_train$MSZoning == 'C (all)'] <- 1
my_train$MSZoning[my_train$MSZoning == 'FV'] <- 2
my_train$MSZoning[my_train$MSZoning == 'I'] <- 3
my_train$MSZoning[my_train$MSZoning == 'RH'] <- 4
my_train$MSZoning[my_train$MSZoning == 'RL'] <- 5
my_train$MSZoning[my_train$MSZoning == 'RP'] <- 6
my_train$MSZoning[my_train$MSZoning == 'RM'] <- 7
my_train$MSZoning <- as.numeric(my_train$MSZoning)

#unique(my_train$MSZoning)
```

```
# Assigning mean value instead of NA
my_train$LotFrontage[is.na(my_train$LotFrontage) == TRUE] <- mean(my_train$LotFrontage, na.rm=TRUE)

# Test needs to be aggregated
test$LotFrontage[is.na(test$LotFrontage) == TRUE] <- mean(test$LotFrontage, na.rm=TRUE)
```

```
# Assigning mean value instead of NA
my_train$MasVnrArea[is.na(my_train$MasVnrArea) == TRUE] <- mean(my_train$MasVnrArea, na.rm=TRUE)
```

```
# Assining "Dummy values" MSZoning
my_train$Street[my_train$Street == 'Grvl'] <- 0
my_train$Street[my_train$Street == 'Pave'] <- 1
my_train$Street <- as.numeric(my_train$Street)
#unique(my_train$Street)
```

```
# Assining "Dummy values" Alley
my_train$Alley[is.na(my_train$Alley) == TRUE] <- 0
my_train$Alley[my_train$Alley == 'Grvl'] <- 1
my_train$Alley[my_train$Alley == 'Pave'] <- 2
my_train$Alley <- as.numeric(my_train$Alley)

#unique(my_train$Alley)
```

```
# Assining "Dummy values" LotShape
#my_train$LotShape[is.na(my_train$LotShape) == TRUE] <- 0
my_train$LotShape[my_train$LotShape == 'Reg'] <- 1
my_train$LotShape[my_train$LotShape == 'IR1'] <- 2
my_train$LotShape[my_train$LotShape == 'IR2'] <- 3
my_train$LotShape[my_train$LotShape == 'IR3'] <- 4
```

```r
my_train$LotShape <- as.numeric(my_train$LotShape)

#unique(train$LotShape)
#unique(test$LotShape)

# Need to transform in test as well
# test$LotShape[is.na(test$LotShape) == TRUE] <- 0
test$LotShape[test$LotShape == 'Reg'] <- 1
test$LotShape[test$LotShape == 'IR1'] <- 2
test$LotShape[test$LotShape == 'IR2'] <- 3
test$LotShape[test$LotShape == 'IR3'] <- 4
test$LotShape <- as.numeric(test$LotShape)
```

```r
# Assining "Dummy values" LandContour
#my_train$LandContour[is.na(my_train$LandContour) == TRUE] <- 0
my_train$LandContour[my_train$LandContour == 'Lvl'] <- 1
my_train$LandContour[my_train$LandContour == 'Bnk'] <- 2
my_train$LandContour[my_train$LandContour == 'HLS'] <- 3
my_train$LandContour[my_train$LandContour == 'Low'] <- 4
my_train$LandContour <- as.numeric(my_train$LandContour)

#unique(my_train$LandContour)
```

```r
# Assining "Dummy values" Utilities
my_train$Utilities[is.na(my_train$Utilities) == TRUE] <- 0
my_train$Utilities[my_train$Utilities == 'AllPub'] <- 1
my_train$Utilities[my_train$Utilities == 'NoSewr'] <- 2
my_train$Utilities[my_train$Utilities == 'NoSeWa'] <- 3
my_train$Utilities[my_train$Utilities == 'ELO'] <- 4
my_train$Utilities <- as.numeric(my_train$Utilities)

#unique(my_train$Utilities)
```

```r
# Assining "Dummy values" LotConfig
#my_train$LotConfig[is.na(my_train$LotConfig) == TRUE] <- 0
my_train$LotConfig[my_train$LotConfig == 'Inside'] <- 1
my_train$LotConfig[my_train$LotConfig == 'Corner'] <- 2
my_train$LotConfig[my_train$LotConfig == 'CulDSac'] <- 3
my_train$LotConfig[my_train$LotConfig == 'FR2'] <- 4
my_train$LotConfig[my_train$LotConfig == 'FR3'] <- 5
my_train$LotConfig <- as.numeric(my_train$LotConfig)

#unique(my_train$LotConfig)
```

```r
# Assining "Dummy values" LandSlope
#my_train$LandSlope[is.na(my_train$LandSlope) == TRUE] <- 0
my_train$LandSlope[my_train$LandSlope == 'Gtl'] <- 0
my_train$LandSlope[my_train$LandSlope == 'Mod'] <- 1
my_train$LandSlope[my_train$LandSlope == 'Sev'] <- 2
my_train$LandSlope <- as.numeric(my_train$LandSlope)

#unique(my_train$LandSlope)
```

```r
# Assining "Dummy values" Neighborhood
#my_train$Neighborhood[is.na(my_train$Neighborhood) == TRUE] <- 0
```

```r
my_train$Neighborhood[my_train$Neighborhood == 'Blmngtn'] <- 0
my_train$Neighborhood[my_train$Neighborhood == 'Blueste'] <- 1
my_train$Neighborhood[my_train$Neighborhood == 'BrDale'] <- 2
my_train$Neighborhood[my_train$Neighborhood == 'BrkSide'] <- 3
my_train$Neighborhood[my_train$Neighborhood == 'ClearCr'] <- 4
my_train$Neighborhood[my_train$Neighborhood == 'CollgCr'] <- 5
my_train$Neighborhood[my_train$Neighborhood == 'Crawfor'] <- 6
my_train$Neighborhood[my_train$Neighborhood == 'Edwards'] <- 7
my_train$Neighborhood[my_train$Neighborhood == 'Gilbert'] <- 8
my_train$Neighborhood[my_train$Neighborhood == 'IDOTRR'] <- 9
my_train$Neighborhood[my_train$Neighborhood == 'MeadowV'] <- 10
my_train$Neighborhood[my_train$Neighborhood == 'Mitchel'] <- 11
my_train$Neighborhood[my_train$Neighborhood == 'Names'] <- 12
my_train$Neighborhood[my_train$Neighborhood == 'NoRidge'] <- 13
my_train$Neighborhood[my_train$Neighborhood == 'NPkVill'] <- 14
my_train$Neighborhood[my_train$Neighborhood == 'NridgHt'] <- 15
my_train$Neighborhood[my_train$Neighborhood == 'NWAmes'] <- 16
my_train$Neighborhood[my_train$Neighborhood == 'NAmes'] <- 16
my_train$Neighborhood[my_train$Neighborhood == 'OldTown'] <- 17
my_train$Neighborhood[my_train$Neighborhood == 'SWISU'] <- 18
my_train$Neighborhood[my_train$Neighborhood == 'Sawyer'] <- 19
my_train$Neighborhood[my_train$Neighborhood == 'SawyerW'] <- 20
my_train$Neighborhood[my_train$Neighborhood == 'Somerst'] <- 21
my_train$Neighborhood[my_train$Neighborhood == 'StoneBr'] <- 22
my_train$Neighborhood[my_train$Neighborhood == 'Timber'] <- 23
my_train$Neighborhood[my_train$Neighborhood == 'Veenker'] <- 24
my_train$Neighborhood <- as.numeric(my_train$Neighborhood)

#unique(my_train$Neighborhood)
```

```r
# Assining "Dummy values" Condition1
#my_train$Condition1[is.na(my_train$Condition1) == TRUE] <- 0
my_train$Condition1[my_train$Condition1 == 'Artery'] <- 0
my_train$Condition1[my_train$Condition1 == 'Feedr'] <- 1
my_train$Condition1[my_train$Condition1 == 'Norm'] <- 2
my_train$Condition1[my_train$Condition1 == 'RRNn'] <- 3
my_train$Condition1[my_train$Condition1 == 'RRAn'] <- 4
my_train$Condition1[my_train$Condition1 == 'PosN'] <- 5
my_train$Condition1[my_train$Condition1 == 'PosA'] <- 6
my_train$Condition1[my_train$Condition1 == 'RRNe'] <- 7
my_train$Condition1[my_train$Condition1 == 'RRAe'] <- 8
my_train$Condition1 <- as.numeric(my_train$Condition1)

#unique(my_train$Condition1)
```

```r
# Assining "Dummy values" Condition2
#my_train$Condition2[is.na(my_train$Condition2) == TRUE] <- 0
my_train$Condition2[my_train$Condition2 == 'Artery'] <- 0
my_train$Condition2[my_train$Condition2 == 'Feedr'] <- 1
my_train$Condition2[my_train$Condition2 == 'Norm'] <- 2
my_train$Condition2[my_train$Condition2 == 'RRNn'] <- 3
my_train$Condition2[my_train$Condition2 == 'RRAn'] <- 4
my_train$Condition2[my_train$Condition2 == 'PosN'] <- 5
my_train$Condition2[my_train$Condition2 == 'PosA'] <- 6
```

```r
my_train$Condition2[my_train$Condition2 == 'RRNe'] <- 7
my_train$Condition2[my_train$Condition2 == 'RRAe'] <- 8
my_train$Condition2 <- as.numeric(my_train$Condition2)

#unique(my_train$Condition2)
```

```r
# Assining "Dummy values" BldgType
#my_train$BldgType[is.na(my_train$BldgType) == TRUE] <- 0
my_train$BldgType[my_train$BldgType == '1Fam'] <- 0
my_train$BldgType[my_train$BldgType == '2FmCon'] <- 1
my_train$BldgType[my_train$BldgType == '2fmCon'] <- 1
my_train$BldgType[my_train$BldgType == 'Duplx'] <- 2
my_train$BldgType[my_train$BldgType == 'Duplex'] <- 2
my_train$BldgType[my_train$BldgType == 'TwnhsE'] <- 3
my_train$BldgType[my_train$BldgType == 'TwnhsI'] <- 4
my_train$BldgType[my_train$BldgType == 'Twnhs'] <- 4
my_train$BldgType <- as.numeric(my_train$BldgType)

#my_train$BldgType <- train$BldgType
#unique(my_train$BldgType)
```

```r
# Assining "Dummy values" HouseStyle
#my_train$HouseStyle[is.na(my_train$HouseStyle) == TRUE] <- 0
my_train$HouseStyle[my_train$HouseStyle == '1Story'] <- 0
my_train$HouseStyle[my_train$HouseStyle == '1.5Fin'] <- 1
my_train$HouseStyle[my_train$HouseStyle == '1.5Unf'] <- 1
my_train$HouseStyle[my_train$HouseStyle == '2Story'] <- 2
my_train$HouseStyle[my_train$HouseStyle == '2.5Fin'] <- 2
my_train$HouseStyle[my_train$HouseStyle == '2.5Unf'] <- 3
my_train$HouseStyle[my_train$HouseStyle == 'SFoyer'] <- 4
my_train$HouseStyle[my_train$HouseStyle == 'SLvl'] <- 4
my_train$HouseStyle <- as.numeric(my_train$HouseStyle)

#my_train$HouseStyle <- train$HouseStyle
#unique(my_train$HouseStyle)
```

```r
# Assining "Dummy values" OverallQual
# Complete set, nothing to do.
#unique(my_train$OverallQual)

#test$OverallQual[is.na(test$OverallQual) == TRUE] <- 0
```

```r
# Assining "Dummy values" OverallCond
# Seems to be complete dataset

#my_train$OverallCond <- train$OverallCond
#unique(my_train$OverallCond)
```

```r
# Assining "Dummy values" YearBuilt
# Seems to be complete dataset

#my_train$YearBuilt <- train$YearBuilt
#unique(my_train$YearBuilt)

#test$YearBuilt[is.na(test$YearBuilt) == TRUE]
```

```r
# Assining "Dummy values" YearRemodAdd
# Seems to be complete dataset

#my_train$YearRemodAdd <- train$YearRemodAdd
#unique(my_train$YearRemodAdd)

#test$YearRemodAdd[is.na(test$YearRemodAdd) == TRUE]
```

```r
# Assining "Dummy values" RoofStyle
#my_train$RoofStyle[is.na(my_train$RoofStyle) == TRUE] <- 0
my_train$RoofStyle[my_train$RoofStyle == 'Flat'] <- 0
my_train$RoofStyle[my_train$RoofStyle == 'Gable'] <- 1
my_train$RoofStyle[my_train$RoofStyle == 'Gambrel'] <- 2
my_train$RoofStyle[my_train$RoofStyle == 'Hip'] <- 3
my_train$RoofStyle[my_train$RoofStyle == 'Mansard'] <- 4
my_train$RoofStyle[my_train$RoofStyle == 'Shed'] <- 5
my_train$RoofStyle <- as.numeric(my_train$RoofStyle)

#my_train$RoofStyle <- train$RoofStyle
#unique(my_train$RoofStyle)
```

```r
# Assining "Dummy values" RoofMatl
#my_train$RoofMatl[is.na(my_train$RoofMatl) == TRUE] <- 0
my_train$RoofMatl[my_train$RoofMatl == 'ClyTile'] <- 0
my_train$RoofMatl[my_train$RoofMatl == 'CompShg'] <- 1
my_train$RoofMatl[my_train$RoofMatl == 'Membran'] <- 2
my_train$RoofMatl[my_train$RoofMatl == 'Metal'] <- 3
my_train$RoofMatl[my_train$RoofMatl == 'Roll'] <- 4
my_train$RoofMatl[my_train$RoofMatl == 'Tar&Grv'] <- 5
my_train$RoofMatl[my_train$RoofMatl == 'WdShake'] <- 6
my_train$RoofMatl[my_train$RoofMatl == 'WdShngl'] <- 7
my_train$RoofMatl <- as.numeric(my_train$RoofMatl)

#my_train$RoofMatl <- train$RoofMatl
#unique(my_train$RoofMatl)
```

```r
# Assining "Dummy values" Exterior1st
#my_train$Exterior1st[is.na(my_train$Exterior1st) == TRUE] <- 0
my_train$Exterior1st[my_train$Exterior1st == 'AsbShng'] <- 0
my_train$Exterior1st[my_train$Exterior1st == 'AsphShn'] <- 1
my_train$Exterior1st[my_train$Exterior1st == 'BrkComm'] <- 2
my_train$Exterior1st[my_train$Exterior1st == 'BrkFace'] <- 3
my_train$Exterior1st[my_train$Exterior1st == 'CBlock'] <- 4
my_train$Exterior1st[my_train$Exterior1st == 'CemntBd'] <- 5
my_train$Exterior1st[my_train$Exterior1st == 'HdBoard'] <- 6
my_train$Exterior1st[my_train$Exterior1st == 'ImStucc'] <- 7
my_train$Exterior1st[my_train$Exterior1st == 'MetalSd'] <- 8
my_train$Exterior1st[my_train$Exterior1st == 'Other'] <- 9
my_train$Exterior1st[my_train$Exterior1st == 'Plywood'] <- 10
my_train$Exterior1st[my_train$Exterior1st == 'PreCast'] <- 11
my_train$Exterior1st[my_train$Exterior1st == 'Stone'] <- 12
my_train$Exterior1st[my_train$Exterior1st == 'Stucco'] <- 13
my_train$Exterior1st[my_train$Exterior1st == 'VinylSd'] <- 14
my_train$Exterior1st[my_train$Exterior1st == 'Wd Sdng'] <- 15
```

```r
my_train$Exterior1st[my_train$Exterior1st == 'WdShing'] <- 16

my_train$Exterior1st <- as.numeric(my_train$Exterior1st)

#my_train$Exterior1st <- train$Exterior1st
#unique(my_train$Exterior1st)
```

```r
# Assining "Dummy values" Exterior2nd
#my_train$Exterior2nd[is.na(my_train$Exterior2nd) == TRUE] <- 0
my_train$Exterior2nd[my_train$Exterior2nd == 'AsbShng'] <- 0
my_train$Exterior2nd[my_train$Exterior2nd == 'AsphShn'] <- 1
my_train$Exterior2nd[my_train$Exterior2nd == 'BrkComm'] <- 2
my_train$Exterior2nd[my_train$Exterior2nd == 'Brk Cmn'] <- 2
my_train$Exterior2nd[my_train$Exterior2nd == 'BrkFace'] <- 3
my_train$Exterior2nd[my_train$Exterior2nd == 'CBlock'] <- 4
my_train$Exterior2nd[my_train$Exterior2nd == 'CemntBd'] <- 5
my_train$Exterior2nd[my_train$Exterior2nd == 'CmentBd'] <- 5
my_train$Exterior2nd[my_train$Exterior2nd == 'HdBoard'] <- 6
my_train$Exterior2nd[my_train$Exterior2nd == 'ImStucc'] <- 7
my_train$Exterior2nd[my_train$Exterior2nd == 'MetalSd'] <- 8
my_train$Exterior2nd[my_train$Exterior2nd == 'Other'] <- 9
my_train$Exterior2nd[my_train$Exterior2nd == 'Plywood'] <- 10
my_train$Exterior2nd[my_train$Exterior2nd == 'PreCast'] <- 11
my_train$Exterior2nd[my_train$Exterior2nd == 'Stone'] <- 12
my_train$Exterior2nd[my_train$Exterior2nd == 'Stucco'] <- 13
my_train$Exterior2nd[my_train$Exterior2nd == 'VinylSd'] <- 14
my_train$Exterior2nd[my_train$Exterior2nd == 'Wd Sdng'] <- 15
my_train$Exterior2nd[my_train$Exterior2nd == 'WdShing'] <- 16
my_train$Exterior2nd[my_train$Exterior2nd == 'Wd Shng'] <- 16

my_train$Exterior2nd <- as.numeric(my_train$Exterior2nd)

#my_train$Exterior2nd <- train$Exterior2nd
#unique(my_train$Exterior2nd)
```

```r
# Assining "Dummy values" MasVnrType
my_train$MasVnrType[is.na(my_train$MasVnrType) == TRUE] <- 0
my_train$MasVnrType[my_train$MasVnrType == 'None'] <- 0
my_train$MasVnrType[my_train$MasVnrType == 'Stone'] <- 1
my_train$MasVnrType[my_train$MasVnrType == 'CBlock'] <- 2
my_train$MasVnrType[my_train$MasVnrType == 'BrkFace'] <- 3
my_train$MasVnrType[my_train$MasVnrType == 'BrkCmn'] <- 4


my_train$MasVnrType <- as.numeric(my_train$MasVnrType)

#my_train$MasVnrType <- train$MasVnrType
#unique(my_train$MasVnrType)
```

```r
# Assining "Dummy values" MasVnrArea
# Nothing to do seems to be complete
#unique(my_train$MasVnrArea)

test$MasVnrArea[is.na(test$MasVnrArea) == TRUE] <- mean(test$MasVnrArea, na.rm = TRUE)
```

```r
# Assining "Dummy values" ExterQual
#my_train$ExterQual[is.na(my_train$ExterQual) == TRUE] <- 0
my_train$ExterQual[my_train$ExterQual == 'Po'] <- 0
my_train$ExterQual[my_train$ExterQual == 'Fa'] <- 1
my_train$ExterQual[my_train$ExterQual == 'TA'] <- 2
my_train$ExterQual[my_train$ExterQual == 'Gd'] <- 3
my_train$ExterQual[my_train$ExterQual == 'Ex'] <- 4

my_train$ExterQual <- as.numeric(my_train$ExterQual)

#my_train$ExterQual <- train$ExterQual
#unique(my_train$ExterQual)

# Need to tranform in test
#test$ExterQual[is.na(test$ExterQual) == TRUE]
test$ExterQual[test$ExterQual == 'Po'] <- 0
test$ExterQual[test$ExterQual == 'Fa'] <- 1
test$ExterQual[test$ExterQual == 'TA'] <- 2
test$ExterQual[test$ExterQual == 'Gd'] <- 3
test$ExterQual[test$ExterQual == 'Ex'] <- 4

test$ExterQual <- as.numeric(test$ExterQual)
```

```r
# Assining "Dummy values" ExterCond
#my_train$ExterCond[is.na(my_train$ExterCond) == TRUE] <- 0
my_train$ExterCond[my_train$ExterCond == 'Po'] <- 0
my_train$ExterCond[my_train$ExterCond == 'Fa'] <- 1
my_train$ExterCond[my_train$ExterCond == 'TA'] <- 2
my_train$ExterCond[my_train$ExterCond == 'Gd'] <- 3
my_train$ExterCond[my_train$ExterCond == 'Ex'] <- 4

my_train$ExterCond <- as.numeric(my_train$ExterCond)

#my_train$ExterCond <- train$ExterCond
#unique(my_train$ExterCond)
```

```r
# Assining "Dummy values" Foundation
#my_train$Foundation[is.na(my_train$Foundation) == TRUE] <- 0
my_train$Foundation[my_train$Foundation == 'Wood'] <- 0
my_train$Foundation[my_train$Foundation == 'Stone'] <- 1
my_train$Foundation[my_train$Foundation == 'Slab'] <- 2
my_train$Foundation[my_train$Foundation == 'PConc'] <- 3
my_train$Foundation[my_train$Foundation == 'CBlock'] <- 4
my_train$Foundation[my_train$Foundation == 'BrkTil'] <- 5

my_train$Foundation <- as.numeric(my_train$Foundation)

#my_train$Foundation <- train$Foundation
#unique(my_train$Foundation)
```

```r
# Assining "Dummy values" BsmtQual
my_train$BsmtQual[is.na(my_train$BsmtQual) == TRUE] <- 0
my_train$BsmtQual[my_train$BsmtQual == 'Po'] <- 0
my_train$BsmtQual[my_train$BsmtQual == 'Fa'] <- 1
```

```r
my_train$BsmtQual[my_train$BsmtQual == 'TA'] <- 2
my_train$BsmtQual[my_train$BsmtQual == 'Gd'] <- 3
my_train$BsmtQual[my_train$BsmtQual == 'Ex'] <- 4

my_train$BsmtQual <- as.numeric(my_train$BsmtQual)

#my_train$BsmtQual <- train$BsmtQual
#unique(my_train$BsmtQual)

# Need to transform in test as well
test$BsmtQual[is.na(test$BsmtQual) == TRUE] <- 0
test$BsmtQual[test$BsmtQual == 'Po'] <- 0
test$BsmtQual[test$BsmtQual == 'Fa'] <- 1
test$BsmtQual[test$BsmtQual == 'TA'] <- 2
test$BsmtQual[test$BsmtQual == 'Gd'] <- 3
test$BsmtQual[test$BsmtQual == 'Ex'] <- 4

test$BsmtQual <- as.numeric(test$BsmtQual)
```

```r
# Assining "Dummy values" BsmtCond
my_train$BsmtCond[is.na(my_train$BsmtCond) == TRUE] <- 0
my_train$BsmtCond[my_train$BsmtCond == 'Po'] <- 0
my_train$BsmtCond[my_train$BsmtCond == 'Fa'] <- 1
my_train$BsmtCond[my_train$BsmtCond == 'TA'] <- 2
my_train$BsmtCond[my_train$BsmtCond == 'Gd'] <- 3
my_train$BsmtCond[my_train$BsmtCond == 'Ex'] <- 4

my_train$BsmtCond <- as.numeric(my_train$BsmtCond)

#my_train$BsmtCond <- train$BsmtCond
#unique(my_train$BsmtCond)
```

```r
# Assining "Dummy values" BsmtUnfSF
# Nothing to do, seems to be complete
#unique(my_train$BsmtUnfSF)
```

```r
# Assining "Dummy values" TotalBsmtSF
# Nothing to do, seems to be complete
#unique(my_train$TotalBsmtSF)

# Need to adjust for test
test$TotalBsmtSF[is.na(test$TotalBsmtSF) == TRUE] <- mean(test$TotalBsmtSF, na.rm = TRUE)
```

```r
# Assining "Dummy values" Heating
#my_train$Heating[is.na(my_train$Heating) == TRUE] <- 0
my_train$Heating[my_train$Heating == 'Floor'] <- 0
my_train$Heating[my_train$Heating == 'GasA'] <- 1
my_train$Heating[my_train$Heating == 'GasW'] <- 2
my_train$Heating[my_train$Heating == 'Grav'] <- 3
my_train$Heating[my_train$Heating == 'Wall'] <- 4
my_train$Heating[my_train$Heating == 'OthW'] <- 5

my_train$Heating <- as.numeric(my_train$Heating)

#my_train$Heating <- train$Heating
```

```r
#unique(my_train$Heating)
```

```r
# Assining "Dummy values" HeatingQC
#my_train$HeatingQC[is.na(my_train$HeatingQC) == TRUE] <- 0
my_train$HeatingQC[my_train$HeatingQC == 'Po'] <- 0
my_train$HeatingQC[my_train$HeatingQC == 'Fa'] <- 1
my_train$HeatingQC[my_train$HeatingQC == 'TA'] <- 2
my_train$HeatingQC[my_train$HeatingQC == 'Gd'] <- 3
my_train$HeatingQC[my_train$HeatingQC == 'Ex'] <- 4
my_train$HeatingQC <- as.numeric(my_train$HeatingQC)

#my_train$HeatingQC <- train$HeatingQC
#unique(my_train$HeatingQC)

# Need to transform in test too.
#test$HeatingQC[is.na(test$HeatingQC) == TRUE] <- 0
test$HeatingQC[test$HeatingQC == 'Po'] <- 0
test$HeatingQC[test$HeatingQC == 'Fa'] <- 1
test$HeatingQC[test$HeatingQC == 'TA'] <- 2
test$HeatingQC[test$HeatingQC == 'Gd'] <- 3
test$HeatingQC[test$HeatingQC == 'Ex'] <- 4
test$HeatingQC <- as.numeric(test$HeatingQC)
```

```r
# Assining "Dummy values" CentralAir
#my_train$CentralAir[is.na(my_train$CentralAir) == TRUE] <- 0
my_train$CentralAir[my_train$CentralAir == 'N'] <- 0
my_train$CentralAir[my_train$CentralAir == 'Y'] <- 1
my_train$CentralAir <- as.numeric(my_train$CentralAir)

#my_train$CentralAir <- train$CentralAir
#unique(my_train$CentralAir)
```

```r
# Assining "Dummy values" Electrical
my_train$Electrical[is.na(my_train$Electrical) == TRUE] <- 0 # Use as standard
my_train$Electrical[my_train$Electrical == 'SBrkr'] <- 0
my_train$Electrical[my_train$Electrical == 'FuseA'] <- 1
my_train$Electrical[my_train$Electrical == 'FuseF'] <- 2
my_train$Electrical[my_train$Electrical == 'FuseP'] <- 3
my_train$Electrical[my_train$Electrical == 'Mix'] <- 4

my_train$Electrical <- as.numeric(my_train$Electrical)

#my_train$CentralAir <- train$CentralAir
#unique(my_train$Electrical)
```

```r
# Assining "Dummy values" X1stFlrSF
# Seems to be complete data

#my_train$X1stFlrSF <- train$X1stFlrSF
#unique(my_train$X1stFlrSF)

#Test
#test$X1stFlrSF[is.na(test$X1stFlrSF) == TRUE] <- 0
```

```r
# Assining "Dummy values" GrLivArea
# Seems to be complete data

#my_train$GrLivArea <- train$GrLivArea
#unique(my_train$GrLivArea)

# Test
#test$GrLivArea[is.na(test$GrLivArea) == TRUE] <- 0
```

```r
# Assining "Dummy values" FullBath
# Seems to be complete data

#my_train$FullBath <- train$FullBath
#unique(my_train$FullBath)

# Test
#test$FullBath[is.na(test$FullBath) == TRUE] <- 0
```

```r
# Assining "Dummy values" BedroomAbvGr
# Seems to be complete data

#my_train$BedroomAbvGr <- train$BedroomAbvGr
#unique(my_train$BedroomAbvGr)
```

```r
# Assining "Dummy values" KitchenQual
#my_train$KitchenQual[is.na(my_train$KitchenQual) == TRUE] <- 0
my_train$KitchenQual[my_train$KitchenQual == 'Po'] <- 0
my_train$KitchenQual[my_train$KitchenQual == 'Fa'] <- 1
my_train$KitchenQual[my_train$KitchenQual == 'TA'] <- 2
my_train$KitchenQual[my_train$KitchenQual == 'Gd'] <- 3
my_train$KitchenQual[my_train$KitchenQual == 'Ex'] <- 4
my_train$KitchenQual <- as.numeric(my_train$KitchenQual)

#my_train$KitchenQual <- train$KitchenQual
#unique(my_train$KitchenQual)

# Need to transform in test as well
test$KitchenQual[is.na(test$KitchenQual) == TRUE]  <- 2 # USe typical
test$KitchenQual[test$KitchenQual == 'Po'] <- 0
test$KitchenQual[test$KitchenQual == 'Fa'] <- 1
test$KitchenQual[test$KitchenQual == 'TA'] <- 2
test$KitchenQual[test$KitchenQual == 'Gd'] <- 3
test$KitchenQual[test$KitchenQual == 'Ex'] <- 4
test$KitchenQual <- as.numeric(test$KitchenQual)
```

```r
# Assining "Dummy values" TotRmsAbvGrd
# Data seems to be complete

#unique(my_train$TotRmsAbvGrd)

# Test
#test$TotRmsAbvGrd[is.na(test$TotRmsAbvGrd) == TRUE] <- 1
```

```r
# Assining "Dummy values" Functional
my_train$Functional[is.na(my_train$Functional) == TRUE] <- 7
```

```r
my_train$Functional[my_train$Functional == 'Sal'] <- 0
my_train$Functional[my_train$Functional == 'Sev'] <- 1
my_train$Functional[my_train$Functional == 'Maj2'] <- 2
my_train$Functional[my_train$Functional == 'Maj1'] <- 3
my_train$Functional[my_train$Functional == 'Mod'] <- 4
my_train$Functional[my_train$Functional == 'Min2'] <- 5
my_train$Functional[my_train$Functional == 'Min1'] <- 6
my_train$Functional[my_train$Functional == 'Typ'] <- 7
my_train$Functional <- as.numeric(my_train$Functional)

#my_train$Functional <- train$Functional
#unique(my_train$Functional)
```

```r
# Assining "Dummy values" Fireplaces
my_train$Fireplaces[is.na(my_train$Fireplaces) == TRUE] <- 0

#my_train$Fireplaces <- train$Fireplaces
#unique(my_train$Fireplaces)

# Test
test$Fireplaces[is.na(test$Fireplaces) == TRUE] <- 0
```

```r
# Assining "Dummy values" GarageType
my_train$GarageType[is.na(my_train$GarageType) == TRUE] <- 0
my_train$GarageType[my_train$GarageType == 'Detchd'] <- 1
my_train$GarageType[my_train$GarageType == 'CarPort'] <- 2
my_train$GarageType[my_train$GarageType == 'BuiltIn'] <- 3
my_train$GarageType[my_train$GarageType == 'Basment'] <- 4
my_train$GarageType[my_train$GarageType == 'Attchd'] <- 5
my_train$GarageType[my_train$GarageType == '2Types'] <- 6

my_train$GarageType <- as.numeric(my_train$GarageType)

#my_train$GarageType <- train$GarageType
#unique(my_train$GarageType)

# Need to transform in test too
test$GarageType[is.na(test$GarageType) == TRUE] <- 0
test$GarageType[test$GarageType == 'Detchd'] <- 1
test$GarageType[test$GarageType == 'CarPort'] <- 2
test$GarageType[test$GarageType == 'BuiltIn'] <- 3
test$GarageType[test$GarageType == 'Basment'] <- 4
test$GarageType[test$GarageType == 'Attchd'] <- 5
test$GarageType[test$GarageType == '2Types'] <- 6

test$GarageType <- as.numeric(test$GarageType)
```

```r
# Assining "Dummy values" GarageFinish
my_train$GarageFinish[is.na(my_train$GarageFinish) == TRUE] <- 0
my_train$GarageFinish[my_train$GarageFinish == 'Unf'] <- 1
my_train$GarageFinish[my_train$GarageFinish == 'RFn'] <- 2
my_train$GarageFinish[my_train$GarageFinish == 'Fin'] <- 3

my_train$GarageFinish <- as.numeric(my_train$GarageFinish)
```

```
#my_train$GarageFinish <- train$GarageFinish
#unique(my_train$GarageFinish)

# Test
test$GarageFinish[is.na(test$GarageFinish) == TRUE] <- 0
test$GarageFinish[test$GarageFinish == 'Unf'] <- 1
test$GarageFinish[test$GarageFinish == 'RFn'] <- 2
test$GarageFinish[test$GarageFinish == 'Fin'] <- 3

test$GarageFinish <- as.numeric(test$GarageFinish)
```

```
# Assining "Dummy values" GarageArea
# Seems to have complete data

#my_train$GarageArea <- train$GarageArea
#unique(my_train$GarageArea)

# Test
test$GarageArea[is.na(test$GarageArea) == TRUE] <- mean(test$GarageArea, na.rm=TRUE) # Need to use mean
```

```
# Assining "Dummy values" PavedDrive
#my_train$PavedDrive[is.na(my_train$PavedDrive) == TRUE] <- 0
my_train$PavedDrive[my_train$PavedDrive == 'N'] <- 0
my_train$PavedDrive[my_train$PavedDrive == 'P'] <- 1
my_train$PavedDrive[my_train$PavedDrive == 'Y'] <- 2

my_train$PavedDrive <- as.numeric(my_train$PavedDrive)

#my_train$PavedDrive <- train$PavedDrive
#unique(my_train$PavedDrive)
```

```
# Assining "Dummy values" WoodDeckSF
# Seems to be complete data

#my_train$WoodDeckSF <- train$WoodDeckSF
#unique(my_train$WoodDeckSF)

#test$WoodDeckSF[is.na(test$WoodDeckSF) == TRUE] <- 0
```

```
# Assining "Dummy values" PoolArea
# Seems to be complete data

#my_train$PoolArea <- train$PoolArea
#unique(my_train$PoolArea)
```

```
# Assining "Dummy values" MoSold
# Seems to be complete data

#my_train$MoSold <- train$MoSold
#unique(my_train$MoSold)
```

```
# Assining "Dummy values" YrSold
# Seems to be complete data

#my_train$YrSold <- train$YrSold
```

```r
#unique(my_train$YrSold)
```

```r
# Assining "Dummy values" SaleType
#my_train$SaleType[is.na(my_train$SaleType) == TRUE] <- 0
my_train$SaleType[my_train$SaleType == 'Oth'] <- 0
my_train$SaleType[my_train$SaleType == 'ConLD'] <- 1
my_train$SaleType[my_train$SaleType == 'ConLI'] <- 2
my_train$SaleType[my_train$SaleType == 'ConLw'] <- 3
my_train$SaleType[my_train$SaleType == 'Con'] <- 4
my_train$SaleType[my_train$SaleType == 'COD'] <- 5
my_train$SaleType[my_train$SaleType == 'New'] <- 6
my_train$SaleType[my_train$SaleType == 'VWD'] <- 7
my_train$SaleType[my_train$SaleType == 'CWD'] <- 8
my_train$SaleType[my_train$SaleType == 'WD'] <- 9

my_train$SaleType <- as.numeric(my_train$SaleType)

#my_train$SaleType <- train$SaleType
#unique(my_train$SaleType)
```

```r
# Assining "Dummy values" SaleCondition
#my_train$SaleCondition[is.na(my_train$SaleCondition) == TRUE] <- 0
my_train$SaleCondition[my_train$SaleCondition == 'Partial'] <- 0
my_train$SaleCondition[my_train$SaleCondition == 'Family'] <- 1
my_train$SaleCondition[my_train$SaleCondition == 'Alloca'] <- 2
my_train$SaleCondition[my_train$SaleCondition == 'AdjLand'] <- 3
my_train$SaleCondition[my_train$SaleCondition == 'Abnorml'] <- 4
my_train$SaleCondition[my_train$SaleCondition == 'Normal'] <- 5

my_train$SaleCondition <- as.numeric(my_train$SaleCondition)

#my_train$SaleCondition <- train$SaleCondition
#unique(my_train$SaleCondition)
```

```r
# Assining "Dummy values" PorchSF
#my_train$PorchSF[is.na(my_train$PorchSF) == TRUE] <- 0

#my_train$PorchSF <- train$PorchSF
#unique(my_train$PorchSF)
```

```r
str(my_train)
```

```
## 'data.frame':    1459 obs. of  57 variables:
##  $ MSSubClass   : int  20 60 70 60 50 20 60 50 190 20 ...
##  $ MSZoning     : num  5 5 5 5 5 5 5 7 5 5 ...
##  $ LotFrontage  : num  80 68 60 84 85 ...
##  $ LotArea      : int  9600 11250 9550 14260 14115 10084 10382 6120 7420 11200 ...
##  $ Street       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Alley        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ LotShape     : num  1 2 2 2 2 1 2 1 1 1 ...
##  $ LandContour  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Utilities    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : num  4 1 2 4 1 1 2 1 2 1 ...
##  $ LandSlope    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Neighborhood : num  24 5 6 13 11 21 16 17 3 19 ...
```

```
##  $ Condition1   : num  1 2 2 2 2 2 5 0 0 2 ...
##  $ Condition2   : num  2 2 2 2 2 2 2 2 0 2 ...
##  $ BldgType     : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ HouseStyle   : num  0 2 2 2 1 0 2 1 1 0 ...
##  $ OverallQual  : int  6 7 7 8 5 8 7 7 5 5 ...
##  $ OverallCond  : int  8 5 5 5 5 5 6 5 6 5 ...
##  $ YearBuilt    : int  1976 2001 1915 2000 1993 2004 1973 1931 1939 1965 ...
##  $ YearRemodAdd : int  1976 2002 1970 2000 1995 2005 1973 1950 1950 1965 ...
##  $ RoofStyle    : num  1 1 1 1 1 1 1 1 1 3 ...
##  $ RoofMatl     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Exterior1st  : num  8 14 15 14 14 14 6 3 8 6 ...
##  $ Exterior2nd  : num  8 14 16 14 14 14 6 16 8 6 ...
##  $ MasVnrType   : num  0 3 0 3 0 1 1 0 0 0 ...
##  $ MasVnrArea   : num  0 162 0 350 0 186 240 0 0 0 ...
##  $ ExterQual    : num  2 3 2 3 2 3 2 2 2 2 ...
##  $ ExterCond    : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ Foundation   : num  4 3 5 3 0 3 4 5 5 4 ...
##  $ BsmtQual     : num  3 3 2 3 3 4 3 2 2 2 ...
##  $ BsmtCond     : num  2 2 3 2 2 2 2 2 2 2 ...
##  $ BsmtUnfSF    : int  284 434 540 490 64 317 216 952 140 134 ...
##  $ TotalBsmtSF  : num  1262 920 756 1145 796 ...
##  $ Heating      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ HeatingQC    : num  4 4 3 4 4 4 4 3 4 4 ...
##  $ CentralAir   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Electrical   : num  0 0 0 0 0 0 0 2 0 0 ...
##  $ X1stFlrSF    : num  1262 920 961 1145 796 ...
##  $ GrLivArea    : num  1262 1786 1717 2198 1362 ...
##  $ FullBath     : int  2 2 1 2 1 2 2 2 1 1 ...
##  $ BedroomAbvGr : int  3 3 3 4 1 3 3 2 2 3 ...
##  $ KitchenQual  : num  2 3 3 3 2 3 2 2 2 2 ...
##  $ TotRmsAbvGrd : int  6 6 7 9 5 7 7 8 5 5 ...
##  $ Functional   : num  7 7 7 7 7 7 7 6 7 7 ...
##  $ Fireplaces   : num  1 1 1 1 0 1 2 2 2 0 ...
##  $ GarageType   : num  5 5 1 5 5 5 5 1 5 1 ...
##  $ GarageFinish : num  2 2 1 2 1 2 2 1 2 1 ...
##  $ GarageArea   : num  460 608 642 836 480 636 484 468 205 384 ...
##  $ PavedDrive   : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ WoodDeckSF   : int  298 0 0 192 40 255 235 90 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MoSold       : int  5 9 2 12 10 8 11 4 1 2 ...
##  $ YrSold       : int  2007 2008 2006 2008 2009 2007 2009 2008 2008 2008 ...
##  $ SaleType     : num  9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: num  5 5 4 5 5 5 5 4 5 5 ...
##  $ SalePrice    : int  181500 223500 140000 250000 143000 307000 200000 129900 118000 129500 ...
##  $ PorchSF      : int  0 42 307 84 350 57 432 205 4 0 ...
```

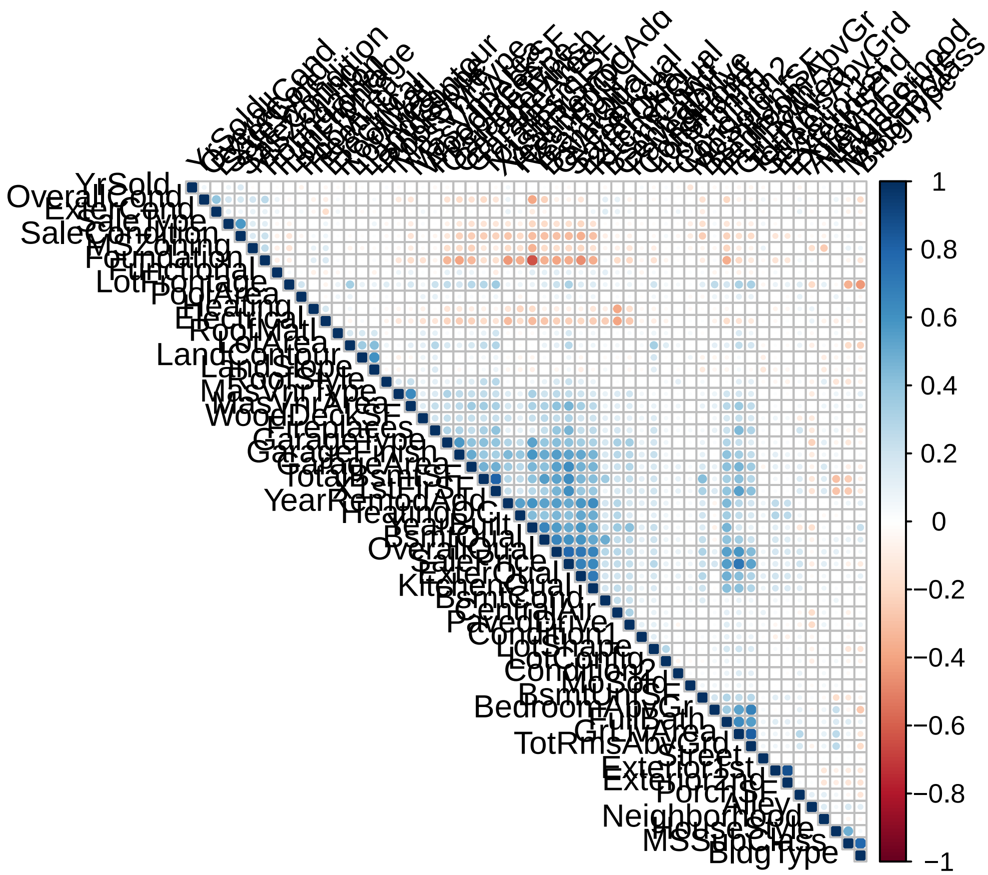
### Sampling train (80%) / test (20%) data

```r
# Utilities do not add any extra info.
my_train <- subset(my_train, select = names(my_train) != 'Utilities')
```

```r
set.seed(101) # Set Seed so that same sample can be reproduced in future also
# Now Selecting 80% of data as sample from total 'n' rows of the data
```

```r
sample <- sample(nrow(my_train), round(nrow(my_train) * 0.8,0), replace = FALSE)
t_train <- my_train[sample, ]
t_test <- my_train[-sample, ]
```

```r
cor_res <- cor(t_train, method="pearson")
#round(cor_res, 2)
```




From the above graph we can visualize some strong correlations, some positive and some negative.

From now on, I will focus on the positive correlations only.

```r
cor_res1 <- data.frame(cor_res)
cor_res1 <- cor_res1['SalePrice']
cor_res1
```

```
##                 SalePrice
## MSSubClass    -0.07920688
## MSZoning      -0.19399872
## LotFrontage    0.31945728
## LotArea        0.27202307
## Street         0.05768576
## Alley         -0.09294711
## LotShape       0.27463538
## LandContour    0.07402610
## LotConfig      0.09210081
## LandSlope      0.05021254
## Neighborhood   0.12462575
## Condition1     0.09581421
```

```
## Condition2     0.07295329
## BldgType      -0.10633383
## HouseStyle     0.06461708
## OverallQual    0.78431627
## OverallCond   -0.06347098
## YearBuilt      0.50070784
## YearRemodAdd   0.50136735
## RoofStyle      0.21590129
## RoofMatl       0.16038341
## Exterior1st    0.12820707
## Exterior2nd    0.10757869
## MasVnrType     0.24082198
## MasVnrArea     0.46479060
## ExterQual      0.67057998
## ExterCond      0.02872189
## Foundation    -0.36654064
## BsmtQual       0.60604617
## BsmtCond       0.21950379
## BsmtUnfSF      0.21178527
## TotalBsmtSF    0.62809739
## Heating       -0.08782199
## HeatingQC      0.41675229
## CentralAir     0.24925478
## Electrical    -0.23145072
## X1stFlrSF      0.61965603
## GrLivArea      0.72453191
## FullBath       0.55103108
## BedroomAbvGr   0.18274615
## KitchenQual    0.64729832
## TotRmsAbvGrd   0.53999358
## Functional     0.10344810
## Fireplaces     0.46162225
## GarageType     0.40465894
## GarageFinish   0.53802235
## GarageArea     0.62613419
## PavedDrive     0.23779357
## WoodDeckSF     0.30804202
## PoolArea       0.10564620
## MoSold         0.06552097
## YrSold        -0.03477067
## SaleType      -0.16773032
## SaleCondition -0.31464583
## SalePrice      1.00000000
## PorchSF        0.19834330
```

The list of variable that I will start this model is listed below and I will take the variable with a “moderate” correlation related to SalePrice.

```r
cor_res1 <- subset(cor_res1, SalePrice > .25)
cor_res1
```

```
##             SalePrice
## LotFrontage 0.3194573
## LotArea     0.2720231
## LotShape    0.2746354
```

```
## OverallQual  0.7843163
## YearBuilt    0.5007078
## YearRemodAdd 0.5013673
## MasVnrArea   0.4647906
## ExterQual    0.6705800
## BsmtQual     0.6060462
## TotalBsmtSF  0.6280974
## HeatingQC    0.4167523
## X1stFlrSF    0.6196560
## GrLivArea    0.7245319
## FullBath     0.5510311
## KitchenQual  0.6472983
## TotRmsAbvGrd 0.5399936
## Fireplaces   0.4616223
## GarageType   0.4046589
## GarageFinish 0.5380224
## GarageArea   0.6261342
## WoodDeckSF   0.3080420
## SalePrice    1.0000000
```

```
lm_columns <- rownames(cor_res1)
lm_columns
```

```
##  [1] "LotFrontage"  "LotArea"      "LotShape"     "OverallQual"
##  [5] "YearBuilt"    "YearRemodAdd" "MasVnrArea"   "ExterQual"
##  [9] "BsmtQual"     "TotalBsmtSF"  "HeatingQC"    "X1stFlrSF"
## [13] "GrLivArea"    "FullBath"     "KitchenQual"  "TotRmsAbvGrd"
## [17] "Fireplaces"   "GarageType"   "GarageFinish" "GarageArea"
## [21] "WoodDeckSF"   "SalePrice"
```

I will exclude SalePrice since that's the variable we want to predict.

```
t_train1 <- t_train[lm_columns]
home_price.lm <- lm(SalePrice ~ LotFrontage + LotShape + OverallQual + YearBuilt + YearRemodAdd + MasVn
```

```
summary(home_price.lm)
```

```
##
## Call:
## lm(formula = SalePrice ~ LotFrontage + LotShape + OverallQual +
##     YearBuilt + YearRemodAdd + MasVnrArea + ExterQual + BsmtQual +
##     TotalBsmtSF + HeatingQC + X1stFlrSF + GrLivArea + FullBath +
##     KitchenQual + TotRmsAbvGrd + Fireplaces + GarageType + GarageFinish +
##     GarageArea + WoodDeckSF, data = t_train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -361034  -17071   -1001   15383  262878
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.048e+05  1.635e+05  -3.700 0.000226 ***
## LotFrontage   1.238e+02  4.961e+01   2.495 0.012731 *
## LotShape      5.515e+03  1.804e+03   3.058 0.002281 **
## OverallQual   1.131e+04  1.368e+03   8.271 3.66e-16 ***
## YearBuilt     1.126e+02  5.906e+01   1.906 0.056848 .
```

```
## YearRemodAdd  1.430e+02  7.327e+01   1.952 0.051161 .
## MasVnrArea    2.945e+01  6.306e+00   4.671 3.36e-06 ***
## ExterQual     1.124e+04  3.007e+03   3.738 0.000195 ***
## BsmtQual      5.673e+03  2.173e+03   2.611 0.009134 **
## TotalBsmtSF   1.934e+01  5.039e+00   3.837 0.000131 ***
## HeatingQC     1.563e+03  1.322e+03   1.183 0.237157
## X1stFlrSF     1.274e+01  5.489e+00   2.322 0.020404 *
## GrLivArea     5.297e+01  4.461e+00  11.872  < 2e-16 ***
## FullBath     -8.772e+03  2.735e+03  -3.208 0.001375 **
## KitchenQual   1.142e+04  2.390e+03   4.779 1.99e-06 ***
## TotRmsAbvGrd -1.499e+03  1.128e+03  -1.329 0.184144
## Fireplaces    8.305e+03  1.906e+03   4.357 1.44e-05 ***
## GarageType   -1.306e+03  7.087e+02  -1.842 0.065662 .
## GarageFinish  1.581e+03  1.639e+03   0.965 0.334961
## GarageArea    4.028e+01  6.389e+00   6.305 4.12e-10 ***
## WoodDeckSF    2.333e+01  8.810e+00   2.648 0.008209 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33960 on 1146 degrees of freedom
## Multiple R-squared:  0.8171, Adjusted R-squared:  0.8139
## F-statistic: 255.9 on 20 and 1146 DF,  p-value: < 2.2e-16
```

From the above, we can notice how there's a good Multiple R-squared of 0.841, also, the p-value is very low and the Median is not too far odd from zero.

I will proceed to continue with backward elimination.

```r
home_price.lm <- update(home_price.lm, .~. -GarageFinish, data = t_train1)
summary(home_price.lm)
```

```
##
## Call:
## lm(formula = SalePrice ~ LotFrontage + LotShape + OverallQual +
##     YearBuilt + YearRemodAdd + MasVnrArea + ExterQual + BsmtQual +
##     TotalBsmtSF + HeatingQC + X1stFlrSF + GrLivArea + FullBath +
##     KitchenQual + TotRmsAbvGrd + Fireplaces + GarageType + GarageArea +
##     WoodDeckSF, data = t_train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -360278  -17331    -828   15389  263065
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.239e+05  1.623e+05  -3.845 0.000127 ***
## LotFrontage   1.233e+02  4.960e+01   2.486 0.013062 *
## LotShape      5.603e+03  1.801e+03   3.111 0.001913 **
## OverallQual   1.138e+04  1.366e+03   8.334  < 2e-16 ***
## YearBuilt     1.210e+02  5.842e+01   2.071 0.038602 *
## YearRemodAdd  1.442e+02  7.326e+01   1.968 0.049301 *
## MasVnrArea    2.935e+01  6.305e+00   4.655 3.62e-06 ***
## ExterQual     1.141e+04  3.002e+03   3.799 0.000153 ***
## BsmtQual      5.842e+03  2.165e+03   2.698 0.007084 **
## TotalBsmtSF   1.903e+01  5.029e+00   3.784 0.000162 ***
## HeatingQC     1.675e+03  1.317e+03   1.272 0.203510
```

```
## X1stFlrSF     1.270e+01  5.488e+00   2.314 0.020853 *
## GrLivArea     5.308e+01  4.460e+00  11.902  < 2e-16 ***
## FullBath     -8.754e+03  2.734e+03  -3.201 0.001405 **
## KitchenQual   1.143e+04  2.390e+03   4.782 1.96e-06 ***
## TotRmsAbvGrd -1.495e+03  1.128e+03  -1.326 0.185108
## Fireplaces    8.489e+03  1.897e+03   4.476 8.36e-06 ***
## GarageType   -1.088e+03  6.718e+02  -1.620 0.105567
## GarageArea    4.144e+01  6.275e+00   6.604 6.11e-11 ***
## WoodDeckSF    2.329e+01  8.810e+00   2.644 0.008305 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33960 on 1147 degrees of freedom
## Multiple R-squared:  0.8169, Adjusted R-squared:  0.8139
## F-statistic: 269.4 on 19 and 1147 DF,  p-value: < 2.2e-16
```

```r
home_price.lm <- update(home_price.lm, .~. -HeatingQC, data = t_train1)
summary(home_price.lm)
```

```
##
## Call:
## lm(formula = SalePrice ~ LotFrontage + LotShape + OverallQual +
##     YearBuilt + YearRemodAdd + MasVnrArea + ExterQual + BsmtQual +
##     TotalBsmtSF + X1stFlrSF + GrLivArea + FullBath + KitchenQual +
##     TotRmsAbvGrd + Fireplaces + GarageType + GarageArea + WoodDeckSF,
##     data = t_train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -361019  -17657    -552   15057  262893
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.738e+05  1.575e+05  -4.278 2.04e-05 ***
## LotFrontage   1.242e+02  4.961e+01   2.504 0.012434 *
## LotShape      5.581e+03  1.802e+03   3.098 0.001998 **
## OverallQual   1.142e+04  1.366e+03   8.362  < 2e-16 ***
## YearBuilt     1.247e+02  5.836e+01   2.137 0.032828 *
## YearRemodAdd  1.672e+02  7.100e+01   2.355 0.018693 *
## MasVnrArea    2.912e+01  6.304e+00   4.620 4.27e-06 ***
## ExterQual     1.197e+04  2.970e+03   4.032 5.90e-05 ***
## BsmtQual      5.760e+03  2.165e+03   2.661 0.007911 **
## TotalBsmtSF   1.935e+01  5.024e+00   3.852 0.000124 ***
## X1stFlrSF     1.223e+01  5.477e+00   2.232 0.025794 *
## GrLivArea     5.325e+01  4.459e+00  11.943  < 2e-16 ***
## FullBath     -8.620e+03  2.733e+03  -3.154 0.001652 **
## KitchenQual   1.173e+04  2.379e+03   4.931 9.41e-07 ***
## TotRmsAbvGrd -1.525e+03  1.128e+03  -1.352 0.176705
## Fireplaces    8.462e+03  1.897e+03   4.461 8.97e-06 ***
## GarageType   -1.076e+03  6.719e+02  -1.602 0.109492
## GarageArea    4.124e+01  6.275e+00   6.573 7.49e-11 ***
## WoodDeckSF    2.300e+01  8.809e+00   2.611 0.009132 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 33970 on 1148 degrees of freedom
## Multiple R-squared:  0.8167, Adjusted R-squared:  0.8138
## F-statistic: 284.1 on 18 and 1148 DF,  p-value: < 2.2e-16
```

```r
home_price.lm <- update(home_price.lm, .~. -TotRmsAbvGrd, data = t_train1)
summary(home_price.lm)
```

```
##
## Call:
## lm(formula = SalePrice ~ LotFrontage + LotShape + OverallQual +
##     YearBuilt + YearRemodAdd + MasVnrArea + ExterQual + BsmtQual +
##     TotalBsmtSF + X1stFlrSF + GrLivArea + FullBath + KitchenQual +
##     Fireplaces + GarageType + GarageArea + WoodDeckSF, data = t_train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -355450  -17427    -281   14635  269730
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.820e+05  1.574e+05  -4.332 1.61e-05 ***
## LotFrontage   1.137e+02  4.902e+01   2.320 0.020516 *
## LotShape      5.611e+03  1.802e+03   3.113 0.001897 **
## OverallQual   1.143e+04  1.366e+03   8.368  < 2e-16 ***
## YearBuilt     1.259e+02  5.838e+01   2.157 0.031240 *
## YearRemodAdd  1.682e+02  7.102e+01   2.369 0.018012 *
## MasVnrArea    2.944e+01  6.302e+00   4.671 3.34e-06 ***
## ExterQual     1.200e+04  2.971e+03   4.041 5.68e-05 ***
## BsmtQual      5.886e+03  2.164e+03   2.720 0.006624 **
## TotalBsmtSF   1.940e+01  5.026e+00   3.859 0.000120 ***
## X1stFlrSF     1.241e+01  5.477e+00   2.266 0.023657 *
## GrLivArea     4.933e+01  3.384e+00  14.575  < 2e-16 ***
## FullBath     -9.066e+03  2.714e+03  -3.340 0.000863 ***
## KitchenQual   1.181e+04  2.379e+03   4.966 7.88e-07 ***
## Fireplaces    8.652e+03  1.892e+03   4.572 5.35e-06 ***
## GarageType   -1.044e+03  6.718e+02  -1.554 0.120405
## GarageArea    4.135e+01  6.276e+00   6.588 6.76e-11 ***
## WoodDeckSF    2.297e+01  8.812e+00   2.607 0.009256 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33980 on 1149 degrees of freedom
## Multiple R-squared:  0.8164, Adjusted R-squared:  0.8137
## F-statistic: 300.5 on 17 and 1149 DF,  p-value: < 2.2e-16
```

```r
home_price.lm <- update(home_price.lm, .~. -GarageType, data = t_train1)
summary(home_price.lm)
```

```
##
## Call:
## lm(formula = SalePrice ~ LotFrontage + LotShape + OverallQual +
##     YearBuilt + YearRemodAdd + MasVnrArea + ExterQual + BsmtQual +
##     TotalBsmtSF + X1stFlrSF + GrLivArea + FullBath + KitchenQual +
##     Fireplaces + GarageArea + WoodDeckSF, data = t_train1)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -352261  -17706    -174   14714  271218
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.295e+05  1.539e+05  -4.091 4.59e-05 ***
## LotFrontage   1.035e+02  4.860e+01   2.129 0.033498 *
## LotShape      5.619e+03  1.803e+03   3.116 0.001879 **
## OverallQual   1.126e+04  1.363e+03   8.264 3.84e-16 ***
## YearBuilt     9.497e+01  5.491e+01   1.730 0.083985 .
## YearRemodAdd  1.716e+02  7.103e+01   2.415 0.015882 *
## MasVnrArea    2.999e+01  6.296e+00   4.764 2.14e-06 ***
## ExterQual     1.223e+04  2.969e+03   4.120 4.07e-05 ***
## BsmtQual      6.044e+03  2.163e+03   2.795 0.005281 **
## TotalBsmtSF   1.912e+01  5.026e+00   3.805 0.000150 ***
## X1stFlrSF     1.163e+01  5.458e+00   2.131 0.033323 *
## GrLivArea     4.978e+01  3.373e+00  14.758  < 2e-16 ***
## FullBath     -9.091e+03  2.716e+03  -3.347 0.000842 ***
## KitchenQual   1.191e+04  2.379e+03   5.006 6.41e-07 ***
## Fireplaces    8.132e+03  1.864e+03   4.363 1.40e-05 ***
## GarageArea    4.048e+01  6.255e+00   6.472 1.43e-10 ***
## WoodDeckSF    2.163e+01  8.776e+00   2.465 0.013835 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34000 on 1150 degrees of freedom
## Multiple R-squared:  0.816,  Adjusted R-squared:  0.8134
## F-statistic: 318.7 on 16 and 1150 DF,  p-value: < 2.2e-16
```

```r
home_price.lm <- update(home_price.lm, .~. -YearBuilt, data = t_train1)
summary(home_price.lm)
```

```
##
## Call:
## lm(formula = SalePrice ~ LotFrontage + LotShape + OverallQual +
##     YearRemodAdd + MasVnrArea + ExterQual + BsmtQual + TotalBsmtSF +
##     X1stFlrSF + GrLivArea + FullBath + KitchenQual + Fireplaces +
##     GarageArea + WoodDeckSF, data = t_train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -350282  -17433    -242   14829  274283
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.978e+05  1.338e+05  -3.720 0.000209 ***
## LotFrontage   1.016e+02  4.863e+01   2.088 0.036985 *
## LotShape      5.990e+03  1.792e+03   3.343 0.000857 ***
## OverallQual   1.145e+04  1.360e+03   8.422  < 2e-16 ***
## YearRemodAdd  1.963e+02  6.964e+01   2.818 0.004914 **
## MasVnrArea    3.166e+01  6.227e+00   5.085 4.29e-07 ***
## ExterQual     1.287e+04  2.948e+03   4.366 1.38e-05 ***
## BsmtQual      7.295e+03  2.040e+03   3.576 0.000363 ***
## TotalBsmtSF   1.882e+01  5.027e+00   3.743 0.000191 ***
```

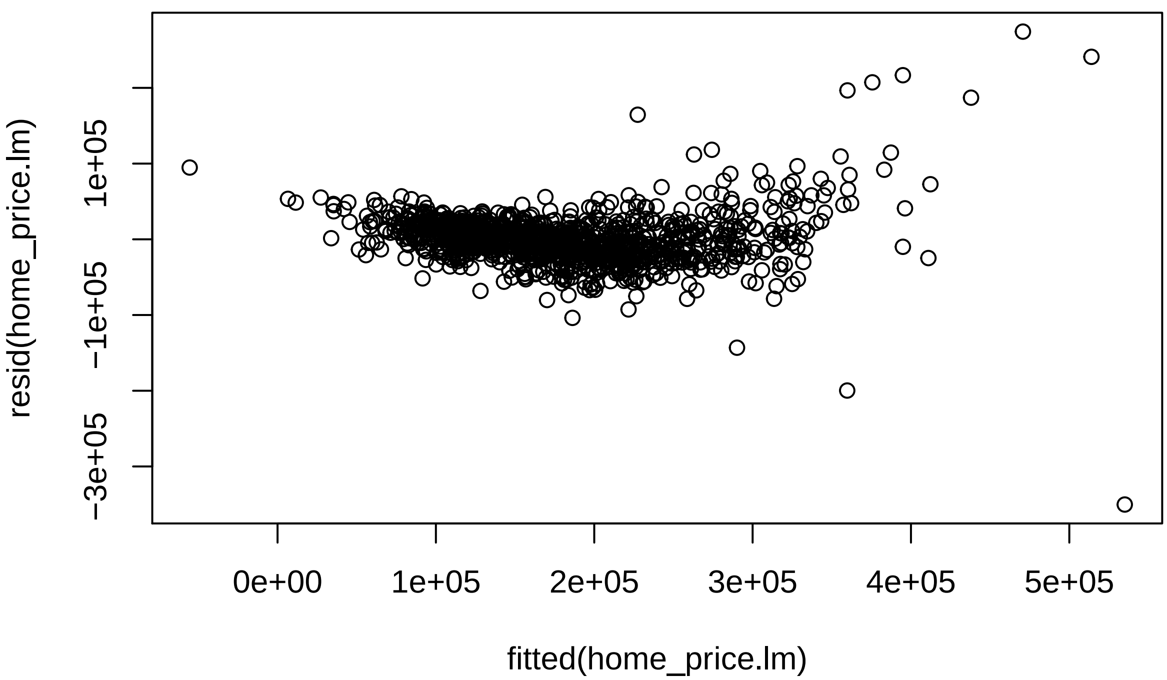
```
## X1stFlrSF     1.211e+01  5.455e+00   2.219 0.026663 *
## GrLivArea     4.748e+01  3.101e+00  15.311  < 2e-16 ***
## FullBath     -7.652e+03  2.587e+03  -2.957 0.003165 **
## KitchenQual   1.192e+04  2.381e+03   5.007 6.40e-07 ***
## Fireplaces    8.106e+03  1.865e+03   4.346 1.51e-05 ***
## GarageArea    4.264e+01  6.135e+00   6.951 6.07e-12 ***
## WoodDeckSF    2.236e+01  8.773e+00   2.548 0.010962 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34030 on 1151 degrees of freedom
## Multiple R-squared:  0.8155, Adjusted R-squared:  0.8131
## F-statistic: 339.2 on 15 and 1151 DF,  p-value: < 2.2e-16
```

### First Linear Model

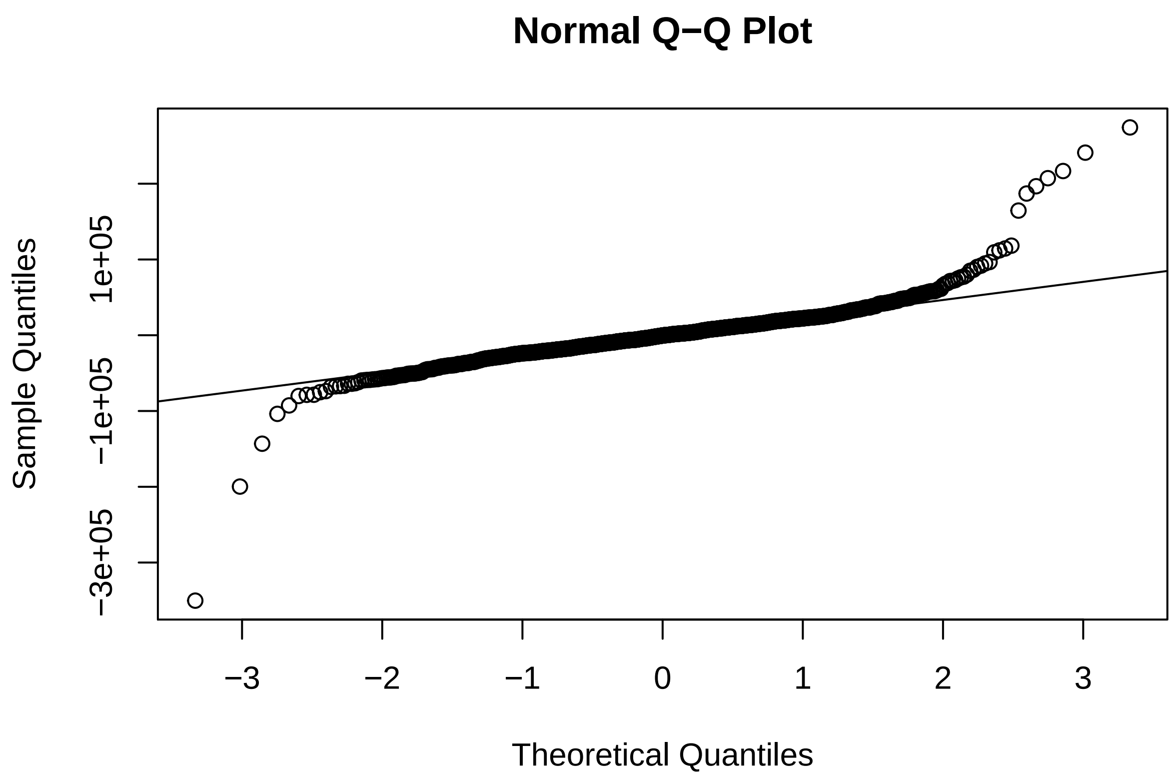
My first cut for the final model in this case has a "near zero" Median which is slightly under performing than others but I will consider this as "near" zero. A very good R squared and very to extremely good p-values.

**Plots**

```
plot(fitted(home_price.lm),resid(home_price.lm))
```




```
qqnorm(resid(home_price.lm))
qqline(resid(home_price.lm))
```




As we can notice the majority of points are "near" zero and follow the Quantile to Quantile line with an exception to some extreme (This linear model is not perfect but it does a good job over all).

**Predicting results from the model**

```
predicted.SalePrice <- predict(home_price.lm, newdata=t_test)
summary(predicted.SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9682  128105  176104  183864  227134  422077
```

```
delta <- predicted.SalePrice - t_test$SalePrice
summary(delta)
```

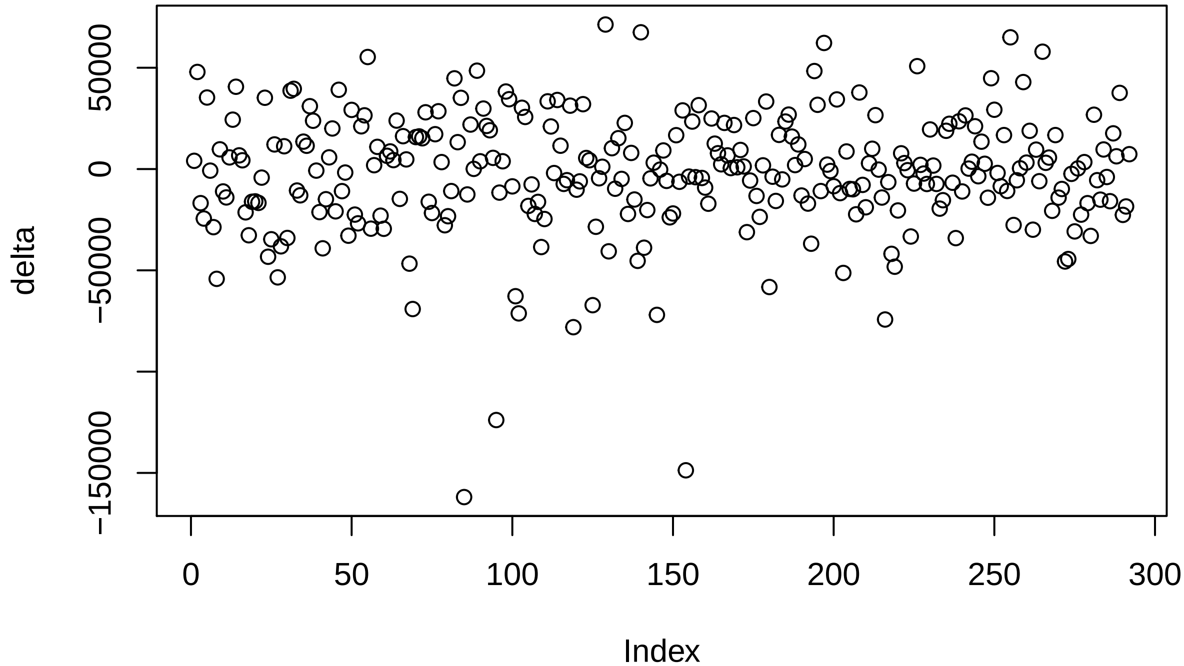
```
##      Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
## -161922.3  -16337.0    -425.2   -2127.6   16723.2   71334.6
```

**Considence interval**

```
t.test(predicted.SalePrice, conf.level = 0.95)
```

```
##
##  One Sample t-test
##
## data:  predicted.SalePrice
## t = 42.606, df = 291, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  175370.2 192357.0
## sample estimates:
## mean of x
##  183863.6
```

```r
plot(delta)
```




## Final Test

### Predicting results from the model

```r
#colnames(test)
```

```r
predicted.SalePrice <- predict(home_price.lm, newdata=test)
summary(predicted.SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -9861  126112  164504  177816  219932  664892
```

```r
predicted.SalePrice[predicted.SalePrice < 0]
```

```
##       388       757
## -9860.674 -7513.935
```

```r
#test[388,]
```

```r
test$SalePrice <- predicted.SalePrice
```

```r
summary(predicted.SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -9861  126112  164504  177816  219932  664892
```

```r
summary(train$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

### Confidence interval

```r
t.test(predicted.SalePrice, conf.level = 0.95)
```

```
##
##  One Sample t-test
##
## data:  predicted.SalePrice
## t = 92.605, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  174049.2 181582.4
## sample estimates:
## mean of x
##  177815.8
```

### Export csv

```r
my_file <- test[c('Id', 'SalePrice')]

write.csv(my_file, file = "MyPrediction.csv",row.names=FALSE)
```

## Kaggle Score

Your submission scored 0.45304.