DATA 605 - Final Project

William Outcault

```
require(ggplot2)
```

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

```
require(corrplot)
```

```
## Loading required package: corrplot

## corrplot 0.84 loaded
```

```
require(dplyr)
```

```
## Loading required package: dplyr

## Warning: package 'dplyr' was built under R version 3.6.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

Creating the Distributions

```
n <- 10
X <- runif(10000, min=1, max=n)
mn <- (n+1)/2
Y <- rnorm(10000, mean = mn, sd = mn)
x <- median(X)
y <-  as.numeric(summary(Y)[2])
```

Distribution Probabilites

```
a <- round(length(X[X > x]) / length(X[X>y]),4)
b <- round(length(X[X > x & Y > y]) / length(X),4)
c <- round(length(X[X < x & X > y]) / length(X),4)
```

Contingency Table

```
rownames = c('P(X>x)','P(X<=x)','Total')
colnames = c('P(Y>y)','P(Y<=y)','Total')
r1c1 = length(X[X > x & Y > y])
r2c1 = length(X[X <= x & Y > y])
r3c1 = r1c1 + r2c1
r1c2 = length(X[X > x & Y <= y])
r2c2 = length(X[X <= x & Y <= y])
r3c2 = r1c2 + r2c2
r1c3 = r1c1 + r1c2
r2c3 = r2c1 + r2c2
r3c3 = r1c3 + r2c3
m <- matrix(c(r1c1,r2c1,r3c1,r1c2,r2c2,r3c2,r1c3,r2c3,r3c3),
            nrow = 3,byrow=TRUE, dimnames=list(rownames,colnames))
test <- (length(X[X>x])/length(X))*(length(X[Y>y])/length(Y))
m
```

```
##         P(Y>y) P(Y<=y) Total
## P(X>x)    3753    3747  7500
## P(X<=x)   1247    1253  2500
## Total     5000    5000 10000
```

Is $P(X > x, Y > y) = P(X > x)P(Y > y)$?

According to our contingency table $P(X > x, Y > y) = 0.3715$. However $P(X > x)P(Y > y) = 0.3741$

Testing Independence

Hypotheses

H0: The variables are independent, and there is no relationship between variables. H1: The variables are dependent, there is a relationship between variables.

Expected Frequencies

Fisher's exact test should be used given a small sample size (specifically when expected values of the contingecy table falls below 5). Adversely Chi-square test is used when you have a large enough sample size.

Fisher's exact test should not be used for larger sample sizes, over Chi-square tests largely because it is too conservative and can be misleading. However the conservative nature of the Fisher's exact test provides better feedback than using Chi-square test on the same small sample.

We see our frequency counts are well above 5 in our contigency table, therefore we will test using the chisq.test function. It is worth noting that if the sample size is to small, our function will produce a warning about inaccuracy, at which point we would use Fisher's exact test.

Chi-squared test

```
m2 <- m[-3,-3]
chisq.test(m2)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  m2
## X-squared = 0.013333, df = 1, p-value = 0.9081
```

As expected our function did not produce a warning. We see our p-value from the Chi-squared test was greater than our threshold 0.05 therefore we fail to reject our null-hypothesis.

Fisher's exact test

```
fisher.test(m2)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  m2
## p-value = 0.9081
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.9183507 1.1029174
## sample estimates:
## odds ratio
##    1.00642
```

Our p-value was the same using the Fisher's exact test. If our sample size was larger however, we would find this test to be computationally impractical.

Advanced Regression for Housing Prices

Descriptive and Inferenctial Statistics

```
train <- read.csv('https://raw.githubusercontent.com/willoutcault/DATA605_Final/master/train.csv',
                  TRUE, ",")
test <- read.csv('https://raw.githubusercontent.com/willoutcault/DATA605_Final/master/test.csv',
                 TRUE, ",")
```

Data Overview

```
glimpse(train)
```

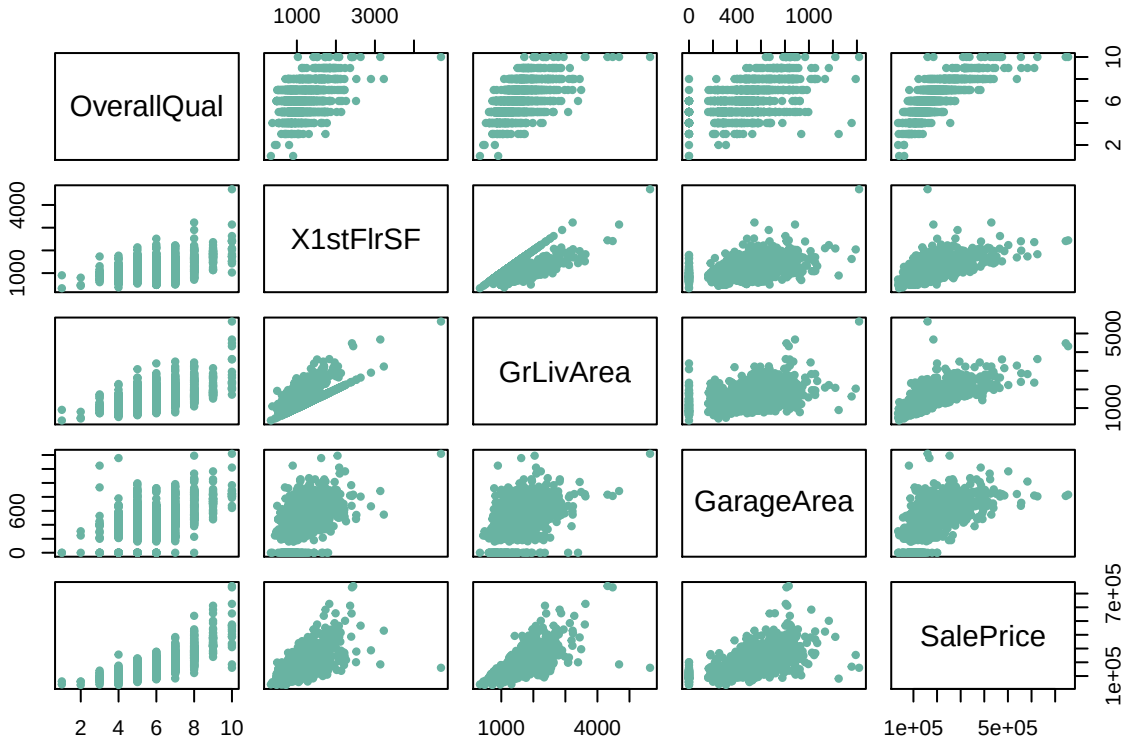
```
## Observations: 1,460
## Variables: 81
## $ Id            <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16...
## $ MSSubClass    <int> 60, 20, 60, 70, 60, 50, 20, 60, 50, 190, 20, 60, 20, ...
## $ MSZoning      <fct> RL, RL, RL, RL, RL, RL, RL, RL, RM, RL, RL, RL, RL, R...
## $ LotFrontage   <int> 65, 80, 68, 60, 84, 85, 75, NA, 51, 50, 70, 85, NA, 9...
## $ LotArea       <int> 8450, 9600, 11250, 9550, 14260, 14115, 10084, 10382, ...
## $ Street        <fct> Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pave,...
## $ Alley         <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ LotShape      <fct> Reg, Reg, IR1, IR1, IR1, IR1, Reg, IR1, Reg, Reg, Reg...
## $ LandContour   <fct> Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl...
## $ Utilities     <fct> AllPub, AllPub, AllPub, AllPub, AllPub, AllPub, AllPu...
## $ LotConfig     <fct> Inside, FR2, Inside, Corner, FR2, Inside, Inside, Cor...
## $ LandSlope     <fct> Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl...
## $ Neighborhood  <fct> CollgCr, Veenker, CollgCr, Crawfor, NoRidge, Mitchel,...
## $ Condition1    <fct> Norm, Feedr, Norm, Norm, Norm, Norm, Norm, PosN, Arte...
## $ Condition2    <fct> Norm, Norm, Norm, Norm, Norm, Norm, Norm, Norm, Norm,...
## $ BldgType      <fct> 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam,...
## $ HouseStyle    <fct> 2Story, 1Story, 2Story, 2Story, 2Story, 1.5Fin, 1Stor...
## $ OverallQual   <int> 7, 6, 7, 7, 8, 5, 8, 7, 7, 5, 5, 9, 5, 7, 6, 7, 6, 4,...
## $ OverallCond   <int> 5, 8, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 8, 7, 5,...
## $ YearBuilt     <int> 2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, 1931,...
## $ YearRemodAdd  <int> 2003, 1976, 2002, 1970, 2000, 1995, 2005, 1973, 1950,...
## $ RoofStyle     <fct> Gable, Gable, Gable, Gable, Gable, Gable, Gable, Gabl...
## $ RoofMatl      <fct> CompShg, CompShg, CompShg, CompShg, CompShg, CompShg,...
## $ Exterior1st   <fct> VinylSd, MetalSd, VinylSd, Wd Sdng, VinylSd, VinylSd,...
## $ Exterior2nd   <fct> VinylSd, MetalSd, VinylSd, Wd Shng, VinylSd, VinylSd,...
## $ MasVnrType    <fct> BrkFace, None, BrkFace, None, BrkFace, None, Stone, S...
## $ MasVnrArea    <int> 196, 0, 162, 0, 350, 0, 186, 240, 0, 0, 0, 286, 0, 30...
## $ ExterQual     <fct> Gd, TA, Gd, TA, Gd, TA, Gd, TA, TA, TA, TA, Ex, TA, G...
## $ ExterCond     <fct> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, T...
## $ Foundation    <fct> PConc, CBlock, PConc, BrkTil, PConc, Wood, PConc, CBl...
## $ BsmtQual      <fct> Gd, Gd, Gd, TA, Gd, Gd, Ex, Gd, TA, TA, TA, Ex, TA, G...
## $ BsmtCond      <fct> TA, TA, TA, Gd, TA, TA, TA, TA, TA, TA, TA, TA, TA, T...
## $ BsmtExposure  <fct> No, Gd, Mn, No, Av, No, Av, Mn, No, No, No, No, No, A...
## $ BsmtFinType1  <fct> GLQ, ALQ, GLQ, ALQ, GLQ, GLQ, GLQ, ALQ, Unf, GLQ, Rec...
## $ BsmtFinSF1    <int> 706, 978, 486, 216, 655, 732, 1369, 859, 0, 851, 906,...
## $ BsmtFinType2  <fct> Unf, Unf, Unf, Unf, Unf, Unf, Unf, BLQ, Unf, Unf, Unf...
## $ BsmtFinSF2    <int> 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ BsmtUnfSF     <int> 150, 284, 434, 540, 490, 64, 317, 216, 952, 140, 134,...
## $ TotalBsmtSF   <int> 856, 1262, 920, 756, 1145, 796, 1686, 1107, 952, 991,...
## $ Heating       <fct> GasA, GasA, GasA, GasA, GasA, GasA, GasA, GasA, GasA,...
## $ HeatingQC     <fct> Ex, Ex, Ex, Gd, Ex, Ex, Ex, Ex, Gd, Ex, Ex, Ex, TA, E...
## $ CentralAir    <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y,...
## $ Electrical    <fct> SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrk...
## $ X1stFlrSF     <int> 856, 1262, 920, 961, 1145, 796, 1694, 1107, 1022, 107...
## $ X2ndFlrSF     <int> 854, 0, 866, 756, 1053, 566, 0, 983, 752, 0, 0, 1142,...
## $ LowQualFinSF  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ GrLivArea     <int> 1710, 1262, 1786, 1717, 2198, 1362, 1694, 2090, 1774,...
## $ BsmtFullBath  <int> 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,...
## $ BsmtHalfBath  <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```
## $ FullBath      <int> 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 3, 1, 2, 1, 1, 1, 2,...
## $ HalfBath      <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,...
## $ BedroomAbvGr  <int> 3, 3, 3, 3, 4, 1, 3, 3, 2, 2, 3, 4, 2, 3, 2, 2, 2, 2,...
## $ KitchenAbvGr  <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2,...
## $ KitchenQual   <fct> Gd, TA, Gd, Gd, Gd, TA, Gd, TA, TA, TA, TA, Ex, TA, G...
## $ TotRmsAbvGrd  <int> 8, 6, 6, 7, 9, 5, 7, 7, 8, 5, 5, 11, 4, 7, 5, 5, 5, 6...
## $ Functional    <fct> Typ, Typ, Typ, Typ, Typ, Typ, Typ, Typ, Min1, Typ, Ty...
## $ Fireplaces    <int> 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 0, 2, 0, 1, 1, 0, 1, 0,...
## $ FireplaceQu   <fct> NA, TA, TA, Gd, TA, NA, Gd, TA, TA, TA, NA, Gd, NA, G...
## $ GarageType    <fct> Attchd, Attchd, Attchd, Detchd, Attchd, Attchd, Attch...
## $ GarageYrBlt   <int> 2003, 1976, 2001, 1998, 2000, 1993, 2004, 1973, 1931,...
## $ GarageFinish  <fct> RFn, RFn, RFn, Unf, RFn, Unf, RFn, RFn, Unf, RFn, Unf...
## $ GarageCars    <int> 2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 1, 3, 1, 2, 2, 2,...
## $ GarageArea    <int> 548, 460, 608, 642, 836, 480, 636, 484, 468, 205, 384...
## $ GarageQual    <fct> TA, TA, TA, TA, TA, TA, TA, TA, Fa, Gd, TA, TA, TA, T...
## $ GarageCond    <fct> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, T...
## $ PavedDrive    <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y,...
## $ WoodDeckSF    <int> 0, 298, 0, 0, 192, 40, 255, 235, 90, 0, 0, 147, 140, ...
## $ OpenPorchSF   <int> 61, 0, 42, 35, 84, 30, 57, 204, 0, 4, 0, 21, 0, 33, 2...
## $ EnclosedPorch <int> 0, 0, 0, 272, 0, 0, 0, 228, 205, 0, 0, 0, 0, 0, 176, ...
## $ X3SsnPorch    <int> 0, 0, 0, 0, 0, 320, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ ScreenPorch   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 176, 0, 0, 0, 0, ...
## $ PoolArea      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ PoolQC        <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ Fence         <fct> NA, NA, NA, NA, NA, MnPrv, NA, NA, NA, NA, NA, NA, NA...
## $ MiscFeature   <fct> NA, NA, NA, NA, NA, Shed, NA, Shed, NA, NA, NA, NA, N...
## $ MiscVal       <int> 0, 0, 0, 0, 0, 700, 0, 350, 0, 0, 0, 0, 0, 0, 0, 0, 7...
## $ MoSold        <int> 2, 5, 9, 2, 12, 10, 8, 11, 4, 1, 2, 7, 9, 8, 5, 7, 3,...
## $ YrSold        <int> 2008, 2007, 2008, 2006, 2008, 2009, 2007, 2009, 2008,...
## $ SaleType      <fct> WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, New, WD, ...
## $ SaleCondition <fct> Normal, Normal, Normal, Abnorml, Normal, Normal, Norm...
## $ SalePrice     <int> 208500, 181500, 223500, 140000, 250000, 143000, 30700...
```

Using Dplyr's glimpse function allows us to view each columns data type as well as the size of the data frame. We are working with a 1460x81 training set. After scrolling through each feature I found a few that I felt might have a correlation with Sales Price.

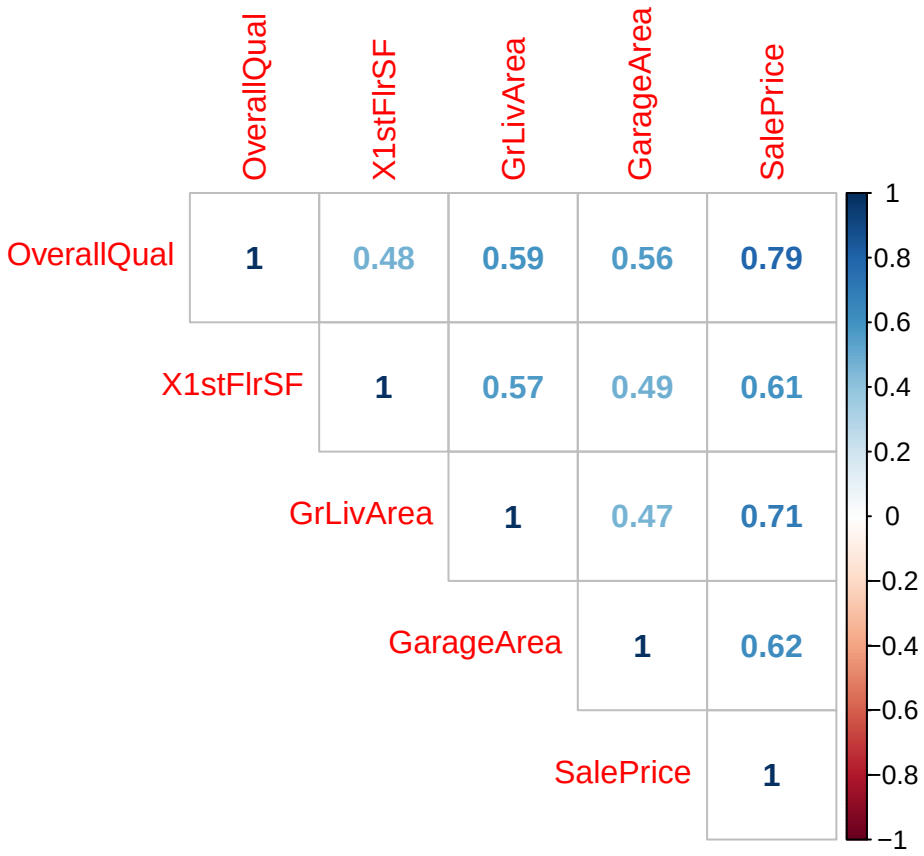
Correlation

```
pairs(train[ , c(18,44,47,63,81)], pch=20, col = "#69b3a2")
```



We notice a positive correlation between out features and our dependent variable, SalePrice. Using these same variables lets visualize a correlation plot.

```
train_sub <- cor(train[, c(18,44,47,63,81)])
corrplot(train_sub, method = "number", type="upper")
```



Out of the independent variables from the correlation plot we see OverallQall with the strongest correlation, and all correlations are positive. Next we will test the significance of each correlation significant.

Hypothesis Testing

We want to test to see if these correlations are significant using the Pearson correlation test. We will use a confidence level of 80% therefore our significance level alpha = 0.20

Null Hypothesis

H0: The variables are independent, and there is no correlation between variables. H1: The variables are dependent, there is a correlation between variables.

Pearson Tests

```
cor.test(train$SalePrice,train$OverallQual,method = "pearson",conf.level = 0.80)
```

```
##
##  Pearson's product-moment correlation
##
## data:  train$SalePrice and train$OverallQual
## t = 49.364, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 80 percent confidence interval:
```

```
##  0.7780752 0.8032204
## sample estimates:
##       cor
## 0.7909816
```

```
cor.test(train$SalePrice,train$X1stFlrSF,method = "pearson",conf.level = 0.80)
```

```
##
##  Pearson's product-moment correlation
##
## data:  train$SalePrice and train$X1stFlrSF
## t = 29.078, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 80 percent confidence interval:
##  0.5841687 0.6266715
## sample estimates:
##       cor
## 0.6058522
```

```
cor.test(train$SalePrice,train$GrLivArea,method = "pearson",conf.level = 0.80)
```

```
##
##  Pearson's product-moment correlation
##
## data:  train$SalePrice and train$GrLivArea
## t = 38.348, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 80 percent confidence interval:
##  0.6915087 0.7249450
## sample estimates:
##       cor
## 0.7086245
```

```
cor.test(train$SalePrice,train$GarageArea,method = "pearson",conf.level = 0.80)
```

```
##
##  Pearson's product-moment correlation
##
## data:  train$SalePrice and train$GarageArea
## t = 30.446, df = 1458, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 80 percent confidence interval:
##  0.6024756 0.6435283
## sample estimates:
##       cor
## 0.6234314
```

Each feature had a p-value less than the significance level so we reject our null hypothesis, therefore we can say the correlation between the listed independent variables and dependent variable is significant.

FWE

```
1-(1-(.2))^4
```

```
## [1] 0.5904
```

Our FWE came in just under 60% which is very high considering we only ran four tests. This is due to our 80% confidence interval, however I am not worried about our chances for a FWE due to the fact that each p-value is extremely low. If we were to change our CI to 95% we would still reject our null hypothesis for each test and reduce our FWE to about 18%.Once again due to our low p-values a type-1 error is very low.

Linear Algebra and Correlation

Precision Matrix

We will calculate the precision matrix by inverting the correlation matrix.

```
library(Matrix)
train_sub_inv <- solve(train_sub)
train_sub_inv
```

```
##             OverallQual  X1stFlrSF   GrLivArea  GarageArea  SalePrice
## OverallQual   2.7480965  0.1115031 -0.19242376 -0.32159236 -1.9044012
## X1stFlrSF     0.1115031  1.7362287 -0.46706793 -0.31007012 -0.6158116
## GrLivArea    -0.1924238 -0.4670679  2.14880584  0.01164084 -1.0947759
## GarageArea   -0.3215924 -0.3100701  0.01164084  1.72833958 -0.6435199
## SalePrice    -1.9044012 -0.6158116 -1.09477590 -0.64351992  4.0564126
```

We notice a high correlation between our selected features and sales price, especially between OverallQual and SalesPrice.

To obtain our identity matrix we will multiply our precision matrix by our correlation matrix, and vice versa. We will also make sure these two results are equal.

```
zapsmall(train_sub_inv %*% train_sub)
```

```
##             OverallQual X1stFlrSF GrLivArea GarageArea SalePrice
## OverallQual           1         0         0          0         0
## X1stFlrSF             0         1         0          0         0
## GrLivArea             0         0         1          0         0
## GarageArea            0         0         0          1         0
## SalePrice             0         0         0          0         1
```

```
zapsmall(train_sub %*% train_sub_inv)
```

```
##             OverallQual X1stFlrSF GrLivArea GarageArea SalePrice
## OverallQual           1         0         0          0         0
## X1stFlrSF             0         1         0          0         0
## GrLivArea             0         0         1          0         0
## GarageArea            0         0         0          1         0
## SalePrice             0         0         0          0         1
```

```
zapsmall(train_sub %*% train_sub_inv) == zapsmall(train_sub_inv %*% train_sub)
```

```
##             OverallQual X1stFlrSF GrLivArea GarageArea SalePrice
## OverallQual        TRUE      TRUE      TRUE       TRUE      TRUE
## X1stFlrSF          TRUE      TRUE      TRUE       TRUE      TRUE
## GrLivArea          TRUE      TRUE      TRUE       TRUE      TRUE
## GarageArea         TRUE      TRUE      TRUE       TRUE      TRUE
## SalePrice          TRUE      TRUE      TRUE       TRUE      TRUE
```

LU Decomposition

Finally we will perform LU decomposition. Because LU = A by multiplying our lower and upper triangualar matrices it will return our original correlation matrix.

```
train_sub_lu <- lu(train_sub)
elu <- expand(train_sub_lu)
elu$L
```

```
## 5 x 5 Matrix of class "dtrMatrix" (unitriangular)
##      [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 1.00000000          .          .          .          .
## [2,] 0.47622383 1.00000000          .          .          .
## [3,] 0.59300743 0.36680770 1.00000000          .          .
## [4,] 0.56202176 0.28728709 0.09963861 1.00000000          .
## [5,] 0.79098160 0.29638474 0.28569464 0.15864262 1.00000000
```

```
elu$U
```

```
## 5 x 5 Matrix of class "dtrMatrix"
##      [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 1.00000000 0.47622383 0.59300743 0.56202176 0.79098160
## [2,]          . 0.77321086 0.28361970 0.22213350 0.22916790
## [3,]          .          . 0.54430830 0.05423412 0.15550596
## [4,]          .          .          . 0.61491165 0.09755119
## [5,]          .          .          .          . 0.24652324
```

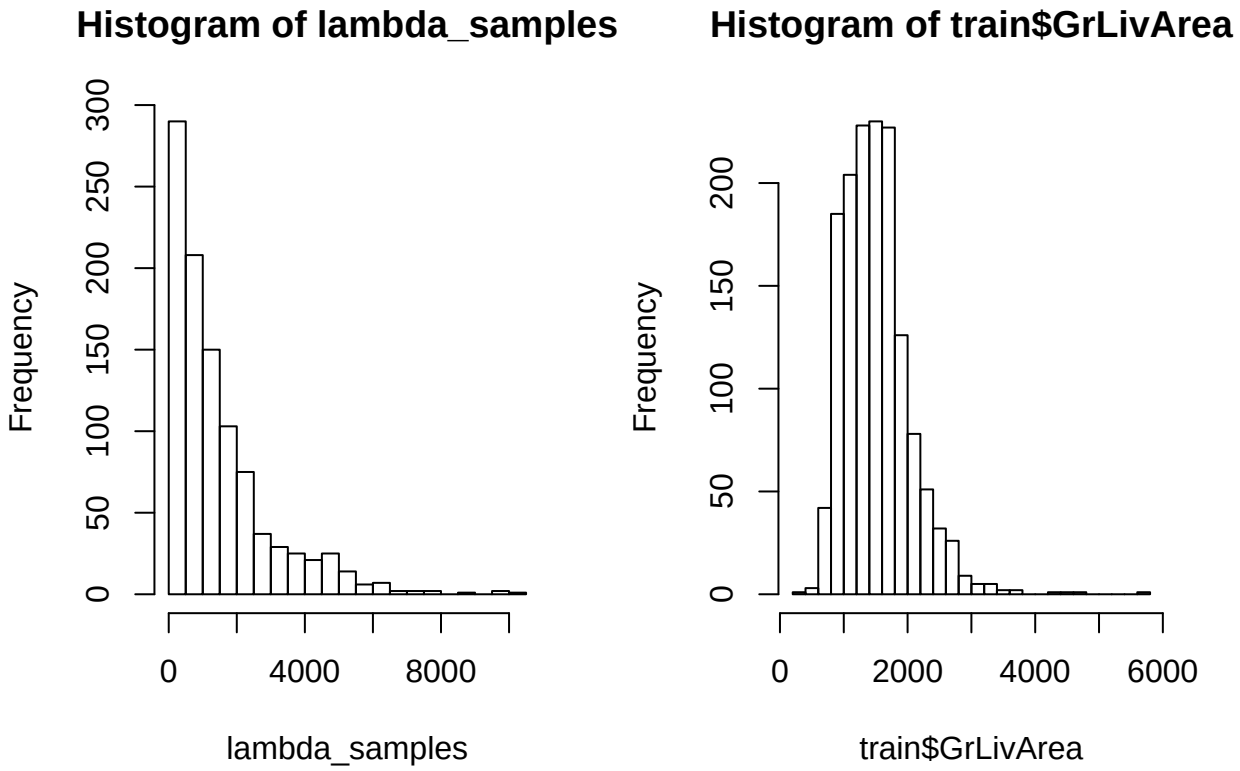
```
elu$L %*% elu$U == train_sub
```

```
## 5 x 5 Matrix of class "lgeMatrix"
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

Calculus-Based Probability & Statistics

Exponential Distribution

```r
fd <- fitdistr(train$GrLivArea, "exponential")
lambda_samples <- rexp(1000, fd$estimate)
par(mfrow=c(1,2))
hist(lambda_samples, breaks=20)
hist(train$GrLivArea, breaks=20)
```



5th and 95th Percentiles

Cumulative Distribution Function

```r
qexp(c(0.05, 0.95), rate = fd$estimate)
```

```
## [1]   77.73313 4539.92351
```

Confidence Interval

```r
me <- qnorm(0.975) * (fd$sd) / sqrt(fd$n)
c(1 - me, 1 + me)
```

```
##      rate      rate
## 0.9999991 1.0000009
```

Empirical Distribution

```
quantile(train$GrLivArea, c(0.05, 0.95))
```

```
##     5%    95%
##  848.0 2466.1
```

Regression Model

```
train <- read.csv('https://raw.githubusercontent.com/willoutcault/DATA605_Final/master/train.csv', TRUE
test <- read.csv('https://raw.githubusercontent.com/willoutcault/DATA605_Final/master/test.csv', TRUE, '

glimpse(train)
```

```
## Observations: 1,460
## Variables: 81
## $ Id            <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16...
## $ MSSubClass    <int> 60, 20, 60, 70, 60, 50, 20, 60, 50, 190, 20, 60, 20, ...
## $ MSZoning      <fct> RL, RL, RL, RL, RL, RL, RL, RL, RM, RL, RL, RL, RL, R...
## $ LotFrontage   <int> 65, 80, 68, 60, 84, 85, 75, NA, 51, 50, 70, 85, NA, 9...
## $ LotArea       <int> 8450, 9600, 11250, 9550, 14260, 14115, 10084, 10382, ...
## $ Street        <fct> Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pave,...
## $ Alley         <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ LotShape      <fct> Reg, Reg, IR1, IR1, IR1, IR1, Reg, IR1, Reg, Reg, Reg...
## $ LandContour   <fct> Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl...
## $ Utilities     <fct> AllPub, AllPub, AllPub, AllPub, AllPub, AllPub, AllPu...
## $ LotConfig     <fct> Inside, FR2, Inside, Corner, FR2, Inside, Inside, Cor...
## $ LandSlope     <fct> Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl...
## $ Neighborhood  <fct> CollgCr, Veenker, CollgCr, Crawfor, NoRidge, Mitchel,...
## $ Condition1    <fct> Norm, Feedr, Norm, Norm, Norm, Norm, Norm, PosN, Arte...
## $ Condition2    <fct> Norm, Norm, Norm, Norm, Norm, Norm, Norm, Norm, Norm,...
## $ BldgType      <fct> 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam,...
## $ HouseStyle    <fct> 2Story, 1Story, 2Story, 2Story, 2Story, 1.5Fin, 1Stor...
## $ OverallQual   <int> 7, 6, 7, 7, 8, 5, 8, 7, 7, 5, 5, 9, 5, 7, 6, 7, 6, 4,...
## $ OverallCond   <int> 5, 8, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 8, 7, 5,...
## $ YearBuilt     <int> 2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, 1931,...
## $ YearRemodAdd  <int> 2003, 1976, 2002, 1970, 2000, 1995, 2005, 1973, 1950,...
## $ RoofStyle     <fct> Gable, Gable, Gable, Gable, Gable, Gable, Gable, Gabl...
## $ RoofMatl      <fct> CompShg, CompShg, CompShg, CompShg, CompShg, CompShg,...
## $ Exterior1st   <fct> VinylSd, MetalSd, VinylSd, Wd Sdng, VinylSd, VinylSd,...
## $ Exterior2nd   <fct> VinylSd, MetalSd, VinylSd, Wd Shng, VinylSd, VinylSd,...
## $ MasVnrType    <fct> BrkFace, None, BrkFace, None, BrkFace, None, Stone, S...
## $ MasVnrArea    <int> 196, 0, 162, 0, 350, 0, 186, 240, 0, 0, 0, 286, 0, 30...
## $ ExterQual     <fct> Gd, TA, Gd, TA, Gd, TA, Gd, TA, TA, TA, TA, Ex, TA, G...
## $ ExterCond     <fct> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, T...
## $ Foundation    <fct> PConc, CBlock, PConc, BrkTil, PConc, Wood, PConc, CBl...
## $ BsmtQual      <fct> Gd, Gd, Gd, TA, Gd, Gd, Ex, Gd, TA, TA, TA, Ex, TA, G...
## $ BsmtCond      <fct> TA, TA, TA, Gd, TA, TA, TA, TA, TA, TA, TA, TA, TA, T...
## $ BsmtExposure  <fct> No, Gd, Mn, No, Av, No, Av, Mn, No, No, No, No, No, A...
## $ BsmtFinType1  <fct> GLQ, ALQ, GLQ, ALQ, GLQ, GLQ, GLQ, ALQ, Unf, GLQ, Rec...
```

```
## $ BsmtFinSF1    <int> 706, 978, 486, 216, 655, 732, 1369, 859, 0, 851, 906,...
## $ BsmtFinType2  <fct> Unf, Unf, Unf, Unf, Unf, Unf, Unf, BLQ, Unf, Unf, Unf...
## $ BsmtFinSF2    <int> 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ BsmtUnfSF     <int> 150, 284, 434, 540, 490, 64, 317, 216, 952, 140, 134,...
## $ TotalBsmtSF   <int> 856, 1262, 920, 756, 1145, 796, 1686, 1107, 952, 991,...
## $ Heating       <fct> GasA, GasA, GasA, GasA, GasA, GasA, GasA, GasA, GasA,...
## $ HeatingQC     <fct> Ex, Ex, Ex, Gd, Ex, Ex, Ex, Ex, Gd, Ex, Ex, Ex, TA, E...
## $ CentralAir    <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y,...
## $ Electrical    <fct> SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrk...
## $ X1stFlrSF     <int> 856, 1262, 920, 961, 1145, 796, 1694, 1107, 1022, 107...
## $ X2ndFlrSF     <int> 854, 0, 866, 756, 1053, 566, 0, 983, 752, 0, 0, 1142,...
## $ LowQualFinSF  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ GrLivArea     <int> 1710, 1262, 1786, 1717, 2198, 1362, 1694, 2090, 1774,...
## $ BsmtFullBath  <int> 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,...
## $ BsmtHalfBath  <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ FullBath      <int> 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 3, 1, 2, 1, 1, 1, 2,...
## $ HalfBath      <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,...
## $ BedroomAbvGr  <int> 3, 3, 3, 3, 4, 1, 3, 3, 2, 2, 3, 4, 2, 3, 2, 2, 2, 2,...
## $ KitchenAbvGr  <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2,...
## $ KitchenQual   <fct> Gd, TA, Gd, Gd, Gd, TA, Gd, TA, TA, TA, TA, Ex, TA, G...
## $ TotRmsAbvGrd  <int> 8, 6, 6, 7, 9, 5, 7, 7, 8, 5, 5, 11, 4, 7, 5, 5, 5, 6...
## $ Functional    <fct> Typ, Typ, Typ, Typ, Typ, Typ, Typ, Typ, Min1, Typ, Ty...
## $ Fireplaces    <int> 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 0, 2, 0, 1, 1, 0, 1, 0,...
## $ FireplaceQu   <fct> NA, TA, TA, Gd, TA, NA, Gd, TA, TA, TA, NA, Gd, NA, G...
## $ GarageType    <fct> Attchd, Attchd, Attchd, Detchd, Attchd, Attchd, Attch...
## $ GarageYrBlt   <int> 2003, 1976, 2001, 1998, 2000, 1993, 2004, 1973, 1931,...
## $ GarageFinish  <fct> RFn, RFn, RFn, Unf, RFn, Unf, RFn, RFn, Unf, RFn, Unf...
## $ GarageCars    <int> 2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 1, 3, 1, 2, 2, 2,...
## $ GarageArea    <int> 548, 460, 608, 642, 836, 480, 636, 484, 468, 205, 384...
## $ GarageQual    <fct> TA, TA, TA, TA, TA, TA, TA, TA, Fa, Gd, TA, TA, TA, T...
## $ GarageCond    <fct> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, T...
## $ PavedDrive    <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y,...
## $ WoodDeckSF    <int> 0, 298, 0, 0, 192, 40, 255, 235, 90, 0, 0, 147, 140, ...
## $ OpenPorchSF   <int> 61, 0, 42, 35, 84, 30, 57, 204, 0, 4, 0, 21, 0, 33, 2...
## $ EnclosedPorch <int> 0, 0, 0, 272, 0, 0, 0, 228, 205, 0, 0, 0, 0, 0, 176, ...
## $ X3SsnPorch    <int> 0, 0, 0, 0, 0, 320, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ ScreenPorch   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 176, 0, 0, 0, 0, ...
## $ PoolArea      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ PoolQC        <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ Fence         <fct> NA, NA, NA, NA, NA, MnPrv, NA, NA, NA, NA, NA, NA, NA...
## $ MiscFeature   <fct> NA, NA, NA, NA, NA, Shed, NA, Shed, NA, NA, NA, NA, N...
## $ MiscVal       <int> 0, 0, 0, 0, 0, 700, 0, 350, 0, 0, 0, 0, 0, 0, 0, 7...
## $ MoSold        <int> 2, 5, 9, 2, 12, 10, 8, 11, 4, 1, 2, 7, 9, 8, 5, 7, 3,...
## $ YrSold        <int> 2008, 2007, 2008, 2006, 2008, 2009, 2007, 2009, 2008,...
## $ SaleType      <fct> WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, New, WD, ...
## $ SaleCondition <fct> Normal, Normal, Normal, Abnorml, Normal, Normal, Norm...
## $ SalePrice     <int> 208500, 181500, 223500, 140000, 250000, 143000, 30700...
```

Formatting the Data

```
train$SalePrice <- log(train$SalePrice)
test$SalePrice <- 0
```

```
asNumeric <- function(x) as.numeric(factor(x))
factorsNumeric <- function(d) modifyList(d, lapply(d[, sapply(d, is.factor)],
                                                   asNumeric))

train <- factorsNumeric(train)
test <- factorsNumeric(test)

train[is.na(train)] <- 0
test[is.na(test)] <- 0

anyNA(train)
```

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

```
anyNA(test)
```

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

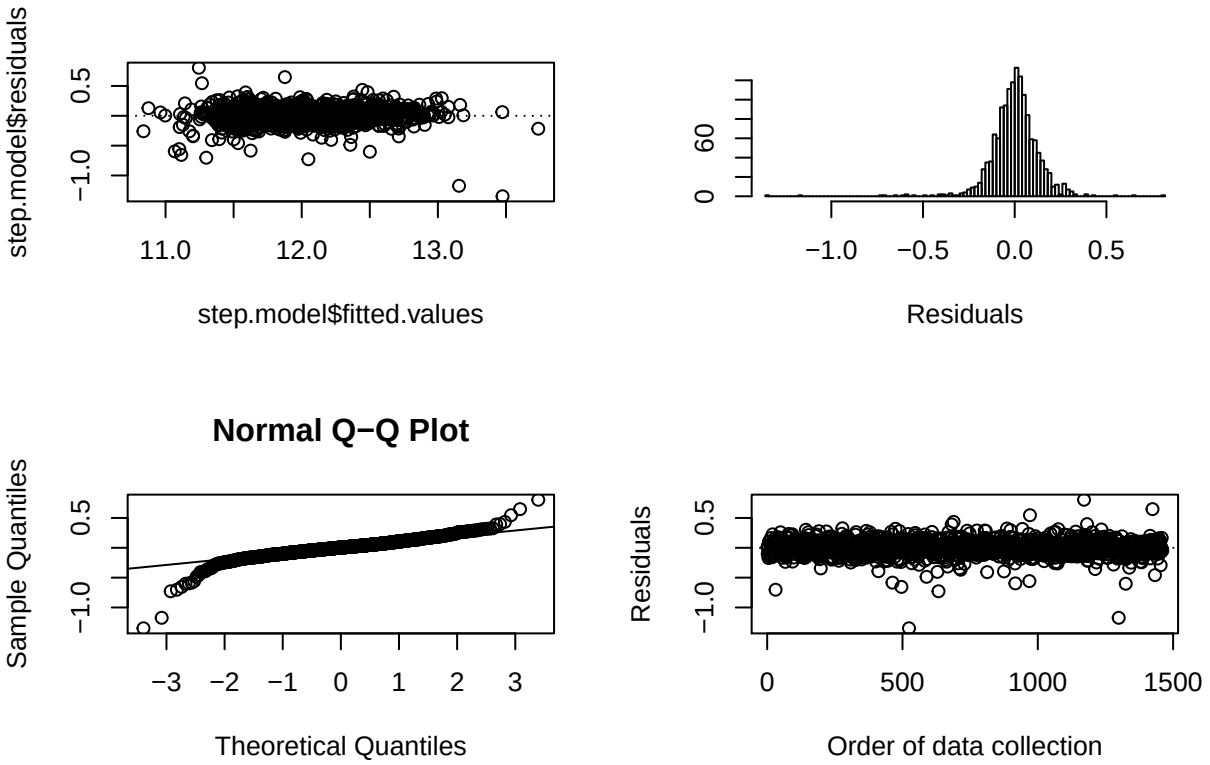
Training Model

```
full.model <- lm(SalePrice ~., data = train)

step.model <- stepAIC(full.model, direction = "forward",
                      trace = FALSE)
m = summary(step.model)

m$adj.r.squared
```

```
## [1] 0.8868161
```

```
par(mfrow=c(2,2))

# residuals plot ------------------------------------------------

plot(step.model$residuals ~ step.model$fitted.values)
abline(h = 0, lty = 3)

# residuals histogram -------------------------------------------

hist(step.model$residuals,
     xlab = "Residuals", ylab = "", main = "", breaks = 85,
     xlim = c(min(step.model$residuals), max(step.model$residuals)))

# normal probability plot of residuals --------------------------

qqnorm(step.model$residuals)
qqline(step.model$residuals)

# order of residuals ------------------------------------------===
```

```
plot(step.model$residuals,
     xlab = "Order of data collection", ylab = "Residuals", main = "")
abline(h = 0, lty = 3)
```




```
predictions <- predict(step.model, test, na.action=na.pass)

predictions <- exp(predictions)
```

Kaggle

Kaggle Name: Will Outcault Kaggle Score: 0.14318