Home_Prices4

Shailaja_Kotagiri

June 16, 2017

```
knitr::opts_chunk$set(error = TRUE)
library(dplyr)
```

```
##
## 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(ggplot2)
library(tidyr)
library(rpart)
library(rpart.plot)
library(poLCA)
```

```
## Loading required package: scatterplot3d

## Loading required package: MASS

##
## Attaching package: 'MASS'

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

```
library(AER)
```

```
## Loading required package: car

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
## Loading required package: survival
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```r
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:survival':
##
##     cluster
```

```r
# install.packages('ALS', dependencies = T)
library(ALS)
```

```
## Loading required package: nnls
```

```
## Loading required package: Iso
```

```
## Iso 0.0-17
```

```r
library(Matrix)
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```r
suppressWarnings(library(relaimpo))
```

```
## Loading required package: boot
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
## The following object is masked from 'package:survival':
##
##     aml
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
## Loading required package: survey
```

```
## Loading required package: grid
```

```
##
## Attaching package: 'survey'
```

```
## The following object is masked from 'package:graphics':
##
##     dotchart
```

```
## Loading required package: mitools
```

```
## This is the global version of package relaimpo.
```

```
## If you are a non-US user, a version with the interesting additional metric pmvd is available
```

```
## from Ulrike Groempings web site at prof.beuth-hochschule.de/groemping.
```

This R-markdown page describes my approach to predicting home prices from home features. This house prices dataset is from this (https://www.kaggle.com/c/house-prices-advanced-regression-techniques) Kaggle competition.

The objective of my analysis is to get to model building stage starting with messy data. This dataset contains 1460 rows with 1/3 of rows with missing values. Another problem with the dataset is that it has too many predictors, most of them categorical and insignificant. The focus of this analysis is to systematically reduce the number of insignificant parameters with keeping as many observations as possible within the dataset.

This project is still in-progress.

The analysis has the following steps: 1. Determine the distribution of the dependent variable. 2. Cleaning the missing data as appropriate. 3. Try out multiple predictive models.

Distribution of the Dependent Variable

The dependent variable is SalePrice. Plot the histogram of SalePrice to identify the distribution.

```
trainHouse <- read.csv("C:/GitHub_Local/Home_Prices/train.csv", header = T)
testHouse <- read.csv("C:/GitHub_Local/Home_Prices/test.csv", header = T)

glimpse(trainHouse)
```

```
## Observations: 1,460
## Variables: 81
## $ Id            <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
## $ MSSubClass    <int> 60, 20, 60, 70, 60, 50, 20, 60, 50, 190, 20, 60,...
## $ MSZoning      <fctr> RL, RL, RL, RL, RL, RL, RL, RL, RM, RL, RL, RL,...
## $ LotFrontage   <int> 65, 80, 68, 60, 84, 85, 75, NA, 51, 50, 70, 85, ...
## $ LotArea       <int> 8450, 9600, 11250, 9550, 14260, 14115, 10084, 10...
## $ Street        <fctr> Pave, Pave, Pave, Pave, Pave, Pave, Pave, Pave,...
## $ Alley         <fctr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ LotShape      <fctr> Reg, Reg, IR1, IR1, IR1, IR1, Reg, IR1, Reg, Re...
## $ LandContour   <fctr> Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lvl, Lv...
## $ Utilities     <fctr> AllPub, AllPub, AllPub, AllPub, AllPub, AllPub,...
## $ LotConfig     <fctr> Inside, FR2, Inside, Corner, FR2, Inside, Insid...
## $ LandSlope     <fctr> Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gtl, Gt...
```

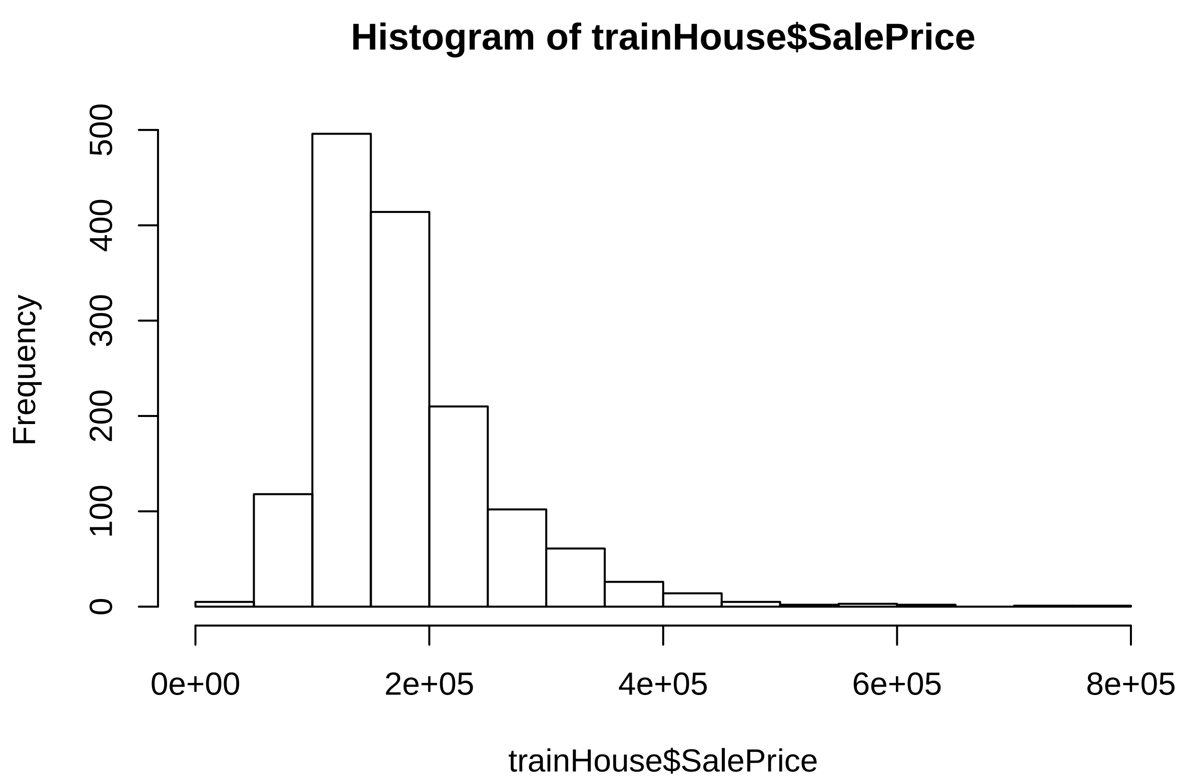
```
## $ Neighborhood  <fctr> CollgCr, Veenker, CollgCr, Crawfor, NoRidge, Mi...
## $ Condition1    <fctr> Norm, Feedr, Norm, Norm, Norm, Norm, Norm, PosN...
## $ Condition2    <fctr> Norm, Norm, Norm, Norm, Norm, Norm, Norm, Norm,...
## $ BldgType      <fctr> 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam, 1Fam,...
## $ HouseStyle    <fctr> 2Story, 1Story, 2Story, 2Story, 2Story, 1.5Fin,...
## $ OverallQual   <int> 7, 6, 7, 7, 8, 5, 8, 7, 7, 5, 5, 9, 5, 7, 6, 7, ...
## $ OverallCond   <int> 5, 8, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 8, ...
## $ YearBuilt     <int> 2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, ...
## $ YearRemodAdd  <int> 2003, 1976, 2002, 1970, 2000, 1995, 2005, 1973, ...
## $ RoofStyle     <fctr> Gable, Gable, Gable, Gable, Gable, Gable, Gable...
## $ RoofMatl      <fctr> CompShg, CompShg, CompShg, CompShg, CompShg, Co...
## $ Exterior1st   <fctr> VinylSd, MetalSd, VinylSd, Wd Sdng, VinylSd, Vi...
## $ Exterior2nd   <fctr> VinylSd, MetalSd, VinylSd, Wd Shng, VinylSd, Vi...
## $ MasVnrType    <fctr> BrkFace, None, BrkFace, None, BrkFace, None, St...
## $ MasVnrArea    <int> 196, 0, 162, 0, 350, 0, 186, 240, 0, 0, 0, 286, ...
## $ ExterQual     <fctr> Gd, TA, Gd, TA, Gd, TA, Gd, TA, TA, TA, TA, Ex,...
## $ ExterCond     <fctr> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA,...
## $ Foundation    <fctr> PConc, CBlock, PConc, BrkTil, PConc, Wood, PCon...
## $ BsmtQual      <fctr> Gd, Gd, Gd, TA, Gd, Gd, Ex, Gd, TA, TA, TA, Ex,...
## $ BsmtCond      <fctr> TA, TA, TA, Gd, TA, TA, TA, TA, TA, TA, TA, TA,...
## $ BsmtExposure  <fctr> No, Gd, Mn, No, Av, No, Av, Mn, No, No, No, No,...
## $ BsmtFinType1  <fctr> GLQ, ALQ, GLQ, ALQ, GLQ, GLQ, GLQ, ALQ, Unf, GL...
## $ BsmtFinSF1    <int> 706, 978, 486, 216, 655, 732, 1369, 859, 0, 851,...
## $ BsmtFinType2  <fctr> Unf, Unf, Unf, Unf, Unf, Unf, Unf, BLQ, Unf, Un...
## $ BsmtFinSF2    <int> 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 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,...
## $ Heating       <fctr> GasA, GasA, GasA, GasA, GasA, GasA, GasA, GasA,...
## $ HeatingQC     <fctr> Ex, Ex, Ex, Gd, Ex, Ex, Ex, Ex, Gd, Ex, Ex, Ex,...
## $ CentralAir    <fctr> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y,...
## $ Electrical    <fctr> SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr, SBrkr...
## $ X1stFlrSF     <int> 856, 1262, 920, 961, 1145, 796, 1694, 1107, 1022...
## $ X2ndFlrSF     <int> 854, 0, 866, 756, 1053, 566, 0, 983, 752, 0, 0, ...
## $ LowQualFinSF  <int> 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, ...
## $ BsmtFullBath  <int> 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, ...
## $ BsmtHalfBath  <int> 0, 1, 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, ...
## $ HalfBath      <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ BedroomAbvGr  <int> 3, 3, 3, 3, 4, 1, 3, 3, 2, 2, 3, 4, 2, 3, 2, 2, ...
## $ KitchenAbvGr  <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, ...
## $ KitchenQual   <fctr> Gd, TA, Gd, Gd, Gd, TA, Gd, TA, TA, TA, TA, Ex,...
## $ TotRmsAbvGrd  <int> 8, 6, 6, 7, 9, 5, 7, 7, 8, 5, 5, 11, 4, 7, 5, 5,...
## $ Functional    <fctr> Typ, Typ, Typ, Typ, Typ, Typ, Typ, Typ, Min1, T...
## $ Fireplaces    <int> 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 0, 2, 0, 1, 1, 0, ...
## $ FireplaceQu   <fctr> NA, TA, TA, Gd, TA, NA, Gd, TA, TA, TA, NA, Gd,...
## $ GarageType    <fctr> Attchd, Attchd, Attchd, Detchd, Attchd, Attchd,...
## $ GarageYrBlt   <int> 2003, 1976, 2001, 1998, 2000, 1993, 2004, 1973, ...
## $ GarageFinish  <fctr> RFn, RFn, RFn, Unf, RFn, Unf, RFn, RFn, Unf, RF...
## $ GarageCars    <int> 2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 1, 3, 1, 2, ...
## $ GarageArea    <int> 548, 460, 608, 642, 836, 480, 636, 484, 468, 205...
## $ GarageQual    <fctr> TA, TA, TA, TA, TA, TA, TA, TA, Fa, Gd, TA, TA,...
## $ GarageCond    <fctr> TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA, TA,...
## $ PavedDrive    <fctr> 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, ...
## $ OpenPorchSF   <int> 61, 0, 42, 35, 84, 30, 57, 204, 0, 4, 0, 21, 0, ...
## $ EnclosedPorch <int> 0, 0, 0, 272, 0, 0, 0, 228, 205, 0, 0, 0, 0, 0, ...
## $ X3SsnPorch    <int> 0, 0, 0, 0, 0, 320, 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...
## $ PoolArea      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ PoolQC        <fctr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ Fence         <fctr> NA, NA, NA, NA, NA, MnPrv, NA, NA, NA, NA, NA, ...
## $ MiscFeature   <fctr> NA, NA, NA, NA, NA, Shed, NA, Shed, NA, NA, NA,...
## $ MiscVal       <int> 0, 0, 0, 0, 0, 700, 0, 350, 0, 0, 0, 0, 0, 0, 0,...
## $ MoSold        <int> 2, 5, 9, 2, 12, 10, 8, 11, 4, 1, 2, 7, 9, 8, 5, ...
## $ YrSold        <int> 2008, 2007, 2008, 2006, 2008, 2009, 2007, 2009, ...
## $ SaleType      <fctr> WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, WD, New...
## $ SaleCondition <fctr> Normal, Normal, Normal, Abnorml, Normal, Normal...
## $ SalePrice     <int> 208500, 181500, 223500, 140000, 250000, 143000, ...
```

```
# summary(trainHouse)

# Remove Id variable
trainHouse <- trainHouse[,!colnames(trainHouse) %in% c("Id")]
dim(trainHouse)
```

```
## [1] 1460   80
```

```
hist(trainHouse$SalePrice)
```



The house prices are skewed to the right side. Let us try fitting a set of skewed distributions to SalePrice and determine if the fit is appropriate using kolmogorov-smirnov test.

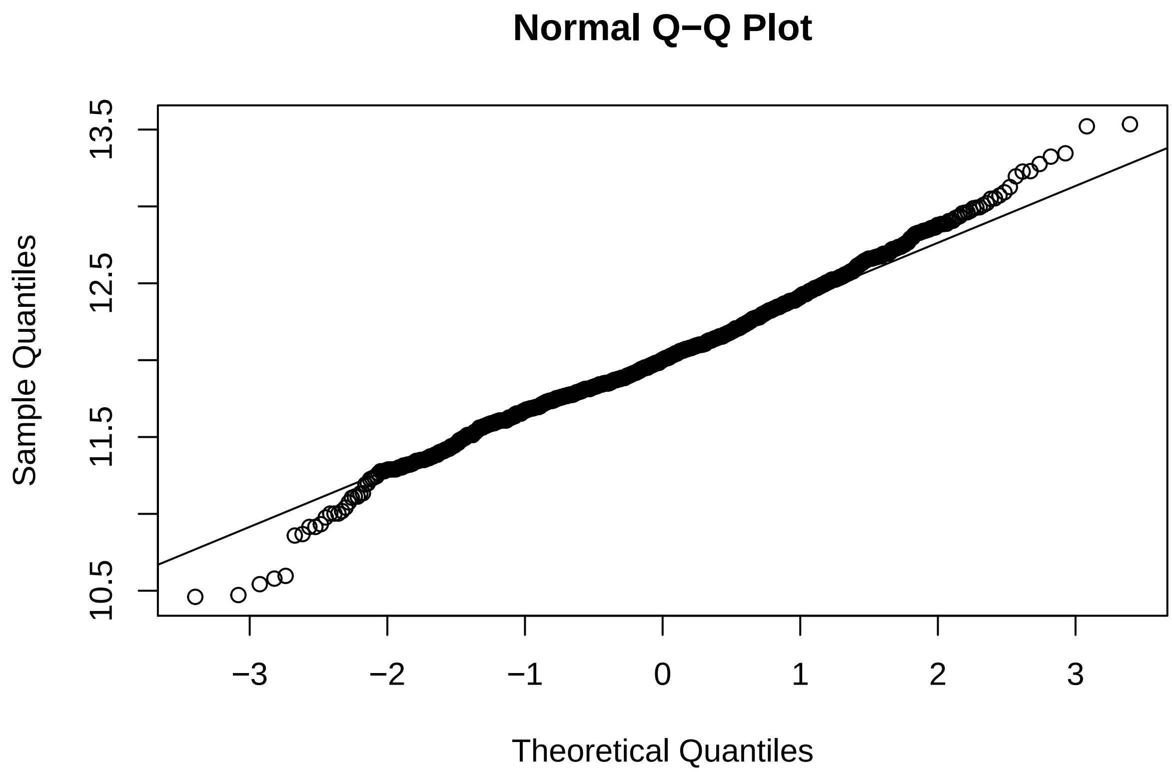
1. Log-normal: Price is a real valued variable. Log-normal is a skewed distribution, typically applied to prices. The following function performs a 1000 ks.tests for the given data vector with the given distribution. Since the test depends on random number generation, the ks.tests are performed multiple times, instead of performing just once.

```
set.seed(0)

# 1. Log-normal
# A function to run 1000 ks.tests.
fitDist1000 <- function(vec,fun,params){
  counter = 0
  size = length(vec)
  listofParams <- lapply(c(size,params), function(x){x})
  for(i in c(1:1000)){

  res <- ks.test(vec, do.call(match.fun(fun),listofParams))

  if (res$p.value > 0.05){
    counter = counter+1
    }
  }
  return(counter)
}

fit.lognorm.Params <- fitdistr(trainHouse$SalePrice, "lognormal")
(fitDist1000(trainHouse$SalePrice,"rlnorm",fit.lognorm.Params$estimate))
```

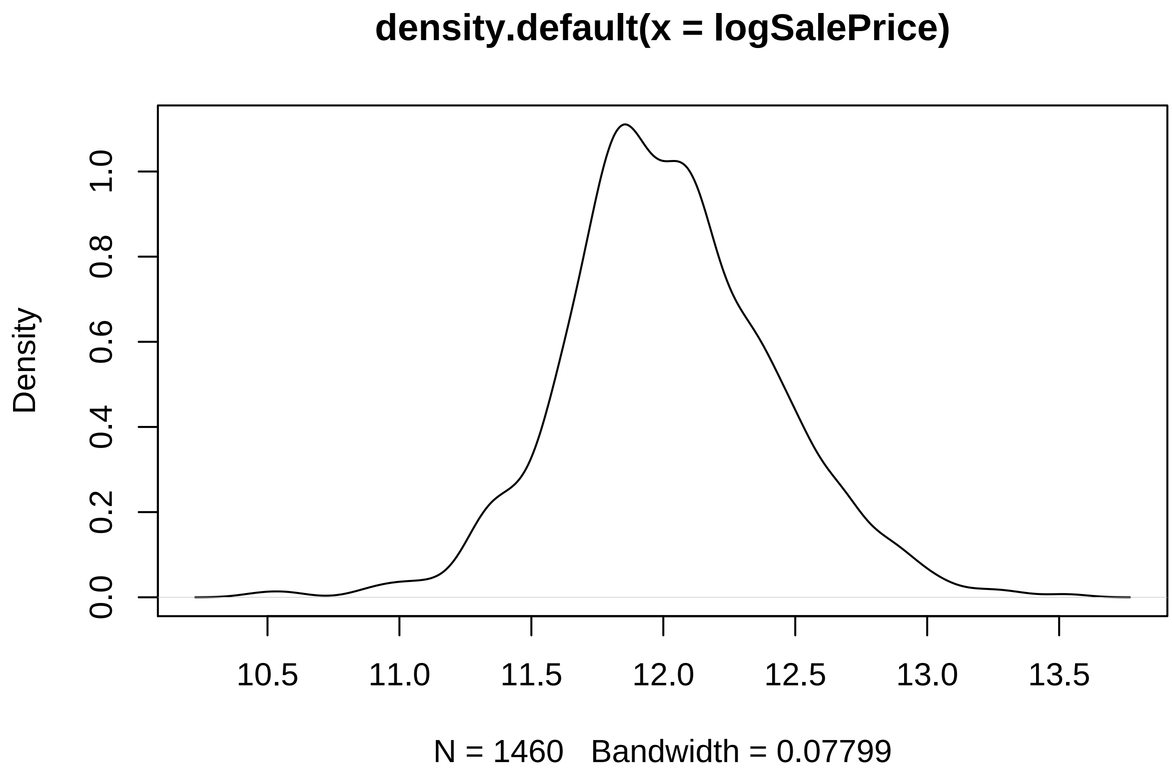
```
## [1] 629
```

Null hypothesis of the ks.test is that the two input vectors have the same distribution. But at 95% confidence level, the null hypothesis is not rejected 63% times. Let us make sure visually that SalePrice distribution looks like log-normal:

```
logSalePrice <- log(trainHouse$SalePrice)
qqnorm(logSalePrice)
qqline(logSalePrice)
```




```
plot(density(logSalePrice))
```



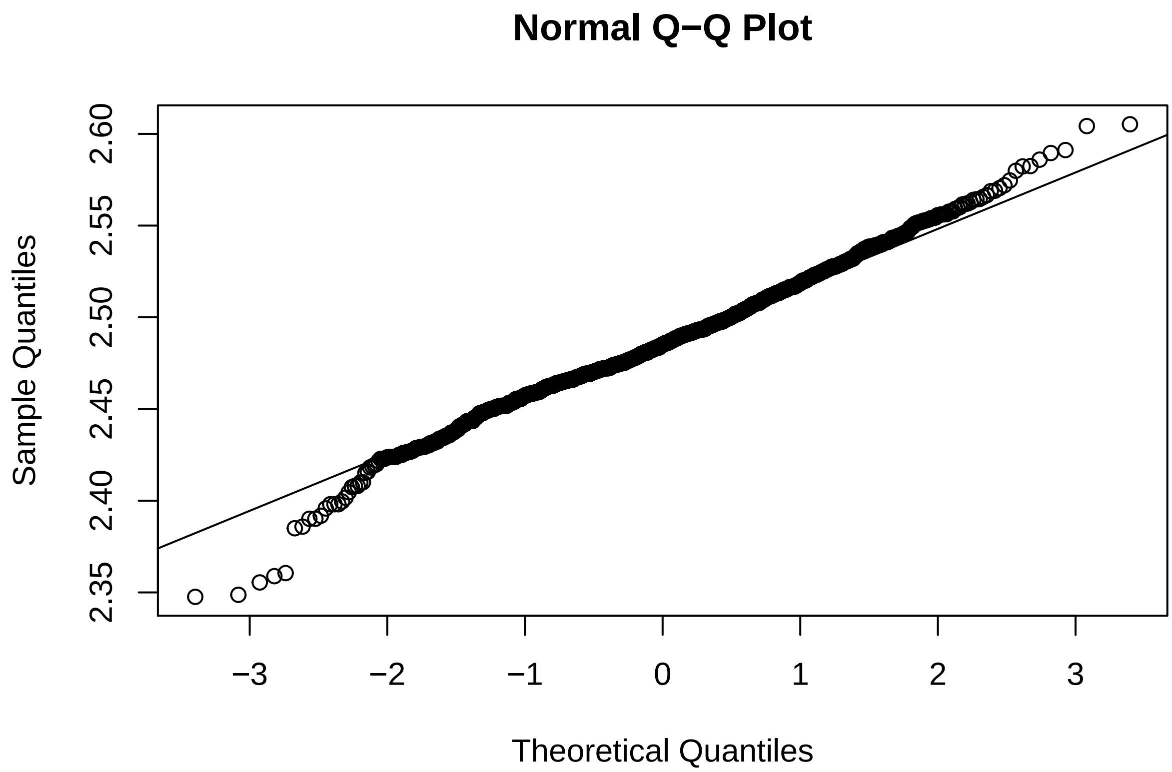
QQ-plot shows that log(SalesPrice) has fatter tails compared to the normal distribution. The distribution is still skewed to the right even after taking log.

2. Loglog transformation: Let us try taking log twice:

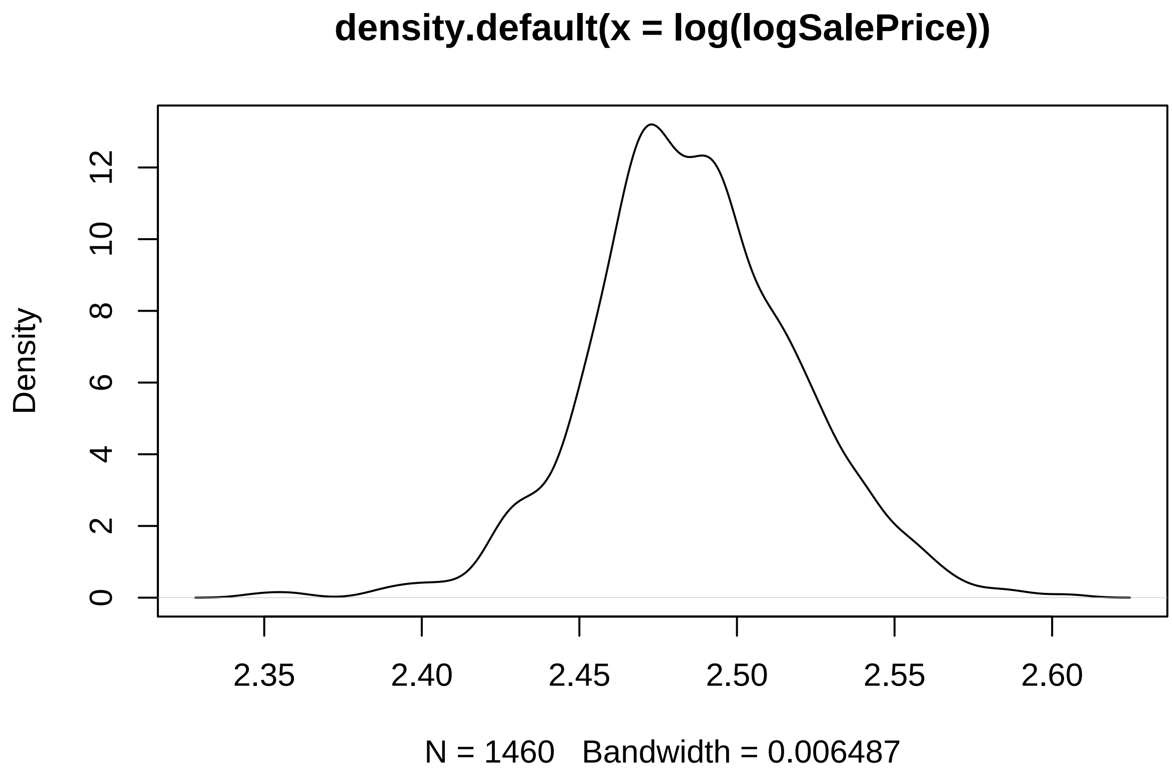
```
# loglog <- log(log(trainHouse$SalePrice))
# hist(loglog)
logSalePrice <- log(trainHouse$SalePrice)

set.seed(0)
fit.loglognorm.Params <- fitdistr(logSalePrice, "lognormal")
(fitDist1000(logSalePrice,"rlnorm",fit.loglognorm.Params$estimate))
```

```
## [1] 745
```

```
qqnorm(log(logSalePrice))
qqline(log(logSalePrice))
```

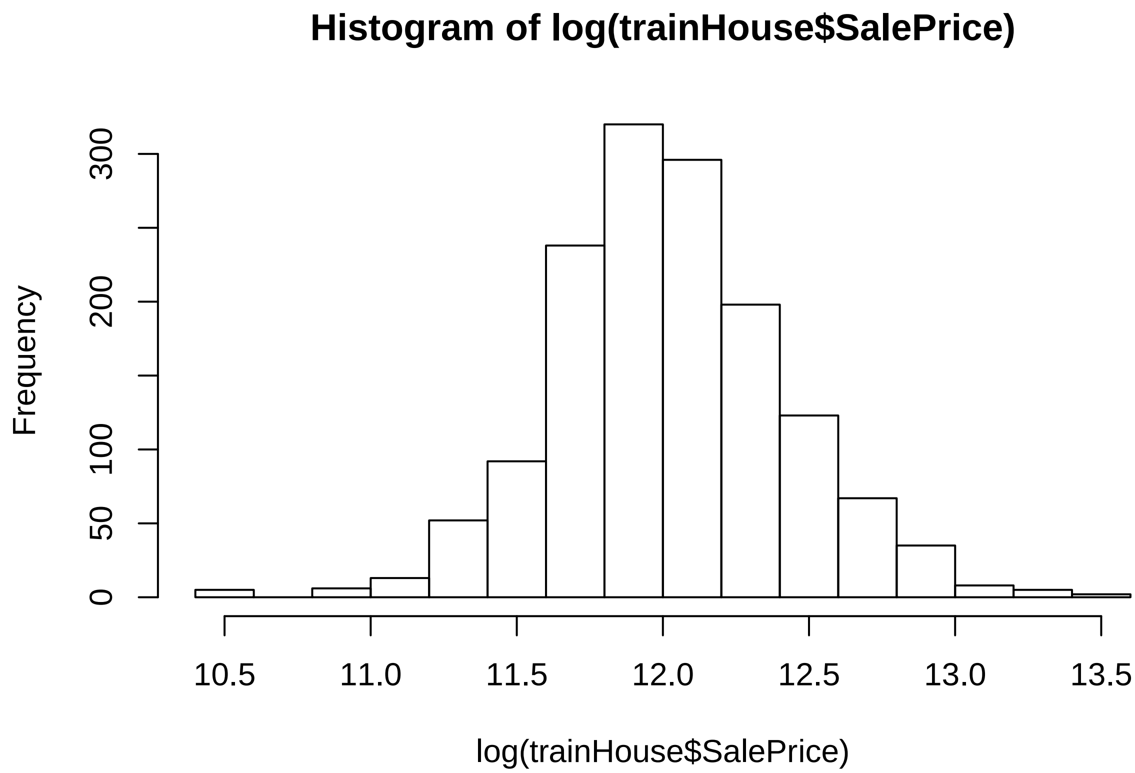



```
plot(density(log(logSalePrice)))
```



Though the density plot and qqplot look similar to those of logSalePrice, loglog transformation seem to fit lognormal distribution better. ks.test could not reject the null hypothesis 74.5% of the time.

```
hist(log(trainHouse$SalePrice))
```




```r
hist(log(log(trainHouse$SalePrice)))
```

Both log and loglog transformed values have almost symmetric distribution with fat tails. Let us try t-distribution.

4. t- distribution

```
set.seed(0)
logSalePrice <- log(trainHouse$SalePrice)
fit.pois.Params <- fitdistr(logSalePrice, "t")
(fitDist1000(logSalePrice,"rt",(fit.pois.Params$estimate)[3]))
```

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

```
set.seed(0)
loglogSalePrice <- log(log(trainHouse$SalePrice))
fit.pois.Params <- fitdistr(loglogSalePrice, "t")
(fitDist1000(loglogSalePrice,"rt",(fit.pois.Params$estimate)[3]))
```

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

The log and log-log transformations do not fit t-distribution.

3. Poisson: Even though Poisson distribution is not appropriate for real valued variables, I would like to try fitting Poisson distribution to sales price:

```
set.seed(0)
fit.pois.Params <- fitdistr(trainHouse$SalePrice, "Poisson")
(fitDist1000(trainHouse$SalePrice,"rpois",fit.pois.Params$estimate))
```

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

4. Negative binomial distribution

```r
# set.seed(0)
# fit.pois.Params <- fitdistr(trainHouse$SalePrice, "negative binomial")
# (fitDist1000(trainHouse$SalePrice,"rnbinom",fit.pois.Params$estimate))
#
# scaledSalePrice <- scale(trainHouse$SalePrice)
# hist(scaledSalePrice)
```

SalePrice could not fit negative binomial distribution. The system became singular. Scaling the variable resulted in negative values, but negative binomial expects positive values, so can't fit negative binomial to scaled values.

```r
set.seed(0)
fit.pois.Params <- fitdistr(trainHouse$SalePrice, "gamma")
```

```
## Error in stats::optim(x = c(208500L, 181500L, 223500L, 140000L, 250000L, : non-finite finite-differen
```

```r
(fitDist1000(trainHouse$SalePrice,"rgamma",fit.pois.Params$estimate))
```

```
## Error in (function (n, shape, rate = 1, scale = 1/rate) : unused argument (lambda = 180921.195890411)
```

Gamma distribution did not fit, either.

---

2. Cleaning the missing data as appropriate.

```r
# Data prep
logtrainHouse <- trainHouse
# Add logSalePrice column to the dataset
logtrainHouse$logSalePrice <- log(trainHouse$SalePrice)

# Remove SalePrice column
logtrainHouse <- logtrainHouse[, !colnames(logtrainHouse) %in% c("SalePrice")]

# Make the logSalePrice column to be the first column.
logtrainHouse <- logtrainHouse[,c(80,c(1:79))]
#colnames(logtrainHouse)

# Count the missing values in all columns
missingValuesinColumns <- apply(logtrainHouse,2, function(x){sum(is.na(x))})
missingValuesinColumns
```

```
##  logSalePrice    MSSubClass      MSZoning   LotFrontage       LotArea
##             0             0             0           259             0
##        Street         Alley      LotShape   LandContour     Utilities
##             0          1369             0             0             0
##     LotConfig     LandSlope  Neighborhood    Condition1    Condition2
##             0             0             0             0             0
##      BldgType    HouseStyle   OverallQual   OverallCond     YearBuilt
##             0             0             0             0             0
##  YearRemodAdd     RoofStyle      RoofMatl   Exterior1st   Exterior2nd
##             0             0             0             0             0
##    MasVnrType    MasVnrArea     ExterQual     ExterCond    Foundation
##             8             8             0             0             0
##      BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1    BsmtFinSF1
##            37            37            38            37             0
```

```
##   BsmtFinType2    BsmtFinSF2     BsmtUnfSF   TotalBsmtSF       Heating
##            38             0             0             0             0
##     HeatingQC    CentralAir    Electrical     X1stFlrSF     X2ndFlrSF
##             0             0             1             0             0
##   LowQualFinSF     GrLivArea  BsmtFullBath  BsmtHalfBath      FullBath
##             0             0             0             0             0
##      HalfBath  BedroomAbvGr  KitchenAbvGr   KitchenQual  TotRmsAbvGrd
##             0             0             0             0             0
##    Functional    Fireplaces   FireplaceQu    GarageType   GarageYrBlt
##             0             0           690            81            81
##   GarageFinish    GarageCars    GarageArea    GarageQual    GarageCond
##            81             0             0            81            81
##    PavedDrive    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch
##             0             0             0             0             0
##   ScreenPorch      PoolArea        PoolQC         Fence   MiscFeature
##             0             0          1453          1179          1406
##       MiscVal        MoSold        YrSold      SaleType SaleCondition
##             0             0             0             0             0
```

Alley, PoolQC, Fence, MiscFeature - These variables have more than 80% values missing. Imputing them from the available values would be unrealistic. Therefore, deleting these columns.

```r
logtrainHouse <- logtrainHouse[,c(c(1:6),c(8:57),c(59:72),c(76:80))]
#colnames(logtrainHouse)[4]
```

Garage related fields do not seem to be missing at random. All garage related fields have (almost) equal number of values missing. Let us investigate further:

```r
# colnames(logtrainHouse)

# Give 1 to each cell of the df with a missing value
missingDF <- as.data.frame(abs(is.na(logtrainHouse)))

# Extract columns with missing values. sapply applies mean function to each column and returns
# 0 or a positive value indicating no nulls and nulls, respectively.
onlyMissingDF <- missingDF[sapply(missingDF, mean) > 0 ] %>% dplyr::select(contains('Garage'))
head(onlyMissingDF,2)
```

```
##   GarageType GarageYrBlt GarageFinish GarageQual GarageCond
## 1          0           0            0          0          0
## 2          0           0            0          0          0
```

```r
# Check the relationship of these variables:
cor(onlyMissingDF)
```

```
##              GarageType GarageYrBlt GarageFinish GarageQual GarageCond
## GarageType            1           1            1          1          1
## GarageYrBlt           1           1            1          1          1
## GarageFinish          1           1            1          1          1
## GarageQual            1           1            1          1          1
## GarageCond            1           1            1          1          1
```

```r
head(logtrainHouse[logtrainHouse$GarageArea==0,]%>% dplyr::select(contains('Garage')),2)
```

```
##    GarageType GarageYrBlt GarageFinish GarageCars GarageArea GarageQual
## 40       <NA>          NA         <NA>          0          0       <NA>
## 49       <NA>          NA         <NA>          0          0       <NA>
```

```
##    GarageCond
## 40       <NA>
## 49       <NA>
```

The correlation value 1 shows that Garage related fields are not missing at random at all! The Garage related attributes, such as finish and yearbuilt are missing because there is no garage in these houses.

But instead of removing 81 rows with missing garage related attributes, only the variable indicating garage presence, which has no null values - GarageArea - can be included.

```
head(logtrainHouse[is.na(logtrainHouse$GarageType),] %>% dplyr::select(contains("Garage")),2)
```

```
##    GarageType GarageYrBlt GarageFinish GarageCars GarageArea GarageQual
## 40       <NA>          NA         <NA>          0          0       <NA>
## 49       <NA>          NA         <NA>          0          0       <NA>
##    GarageCond
## 40       <NA>
## 49       <NA>
```

```
# Exclude all garage related fileds except GarageArea
logtrainHouse <- logtrainHouse %>% dplyr::select(-starts_with('Garage'), GarageArea)

# colnames(logtrainHouse)
```

Check the missingess of Basement related fields:

```
head(logtrainHouse[is.na(logtrainHouse$BsmtQual),] %>% dplyr::select(contains("Bsmt")),2)
```

```
##    BsmtQual BsmtCond BsmtExposure BsmtFinType1 BsmtFinSF1 BsmtFinType2
## 18     <NA>     <NA>         <NA>         <NA>          0         <NA>
## 40     <NA>     <NA>         <NA>         <NA>          0         <NA>
##    BsmtFinSF2 BsmtUnfSF TotalBsmtSF BsmtFullBath BsmtHalfBath
## 18          0         0           0            0            0
## 40          0         0           0            0            0
```

```
logtrainHouse <- logtrainHouse %>% dplyr::select(-starts_with('Bsmt'),BsmtFinSF1,BsmtFinSF2)

# colnames(logtrainHouse)
```

Basement related fields are also not missing at random. Excluding all additional parameteres related to basement except BasementFinSF.

LotForntage has 259 values missing. Investigate the nature of missingness.

```
# colnames(onlyMissingDF)

# Give 1 to each cell of the df with a missing value
missingDF <- as.data.frame(abs(is.na(logtrainHouse)))

# Extract columns with missing values. sapply applies mean function to each column
# and returns 0 or a positive value indicating no nulls and nulls, respectively.
onlyMissingDF <- missingDF[sapply(missingDF, mean) > 0 ]
head(onlyMissingDF,2)
```

```
##   LotFrontage MasVnrType MasVnrArea Electrical
## 1           0          0          0          0
## 2           0          0          0          0
```

```
# Missingmess in LotFrontage is not coinciding with missingness in any other column.
# This may be missing at random. Remove 259 rows from the dataset.
```

```r
cor(onlyMissingDF)
```

```
##             LotFrontage   MasVnrType   MasVnrArea   Electrical
## LotFrontage  1.00000000  0.014107374  0.014107374 -0.012157681
## MasVnrType   0.01410737  1.000000000  1.000000000 -0.001943274
## MasVnrArea   0.01410737  1.000000000  1.000000000 -0.001943274
## Electrical  -0.01215768 -0.001943274 -0.001943274  1.000000000
```

```r
# dim(logtrainHouse)
logtrainHouse <- logtrainHouse[!is.na(logtrainHouse$LotFrontage),]
```

Missingness of LotFrontage does not seem to coincide with others. This could be missing at random. Therefore, removing the rows with missing values in LotFrontage column.

Eventhough missingness in MasVnrArea and MasVnrType are coinciding, the number of rows with missing values are small. Removing those rows may not impact the solution much.

Check the missingness of the rest of the data frame. Since the number of rows with missing data is small, remove the rows.

```r
missingValuesinColumns <- apply(logtrainHouse,2, function(x){sum(is.na(x))})
missingValuesinColumns
```

```
##  logSalePrice    MSSubClass      MSZoning   LotFrontage       LotArea
##             0             0             0             0             0
##        Street      LotShape   LandContour     Utilities     LotConfig
##             0             0             0             0             0
##     LandSlope  Neighborhood    Condition1    Condition2      BldgType
##             0             0             0             0             0
##    HouseStyle   OverallQual   OverallCond     YearBuilt  YearRemodAdd
##             0             0             0             0             0
##     RoofStyle      RoofMatl   Exterior1st   Exterior2nd    MasVnrType
##             0             0             0             0             6
##    MasVnrArea     ExterQual     ExterCond    Foundation   TotalBsmtSF
##             6             0             0             0             0
##       Heating     HeatingQC    CentralAir    Electrical     X1stFlrSF
##             0             0             0             1             0
##     X2ndFlrSF  LowQualFinSF     GrLivArea      FullBath      HalfBath
##             0             0             0             0             0
##  BedroomAbvGr  KitchenAbvGr   KitchenQual  TotRmsAbvGrd    Functional
##             0             0             0             0             0
##    Fireplaces    PavedDrive    WoodDeckSF   OpenPorchSF EnclosedPorch
##             0             0             0             0             0
##    X3SsnPorch   ScreenPorch      PoolArea       MiscVal        MoSold
##             0             0             0             0             0
##        YrSold      SaleType SaleCondition    GarageArea    BsmtFinSF1
##             0             0             0             0             0
##    BsmtFinSF2
##             0
```

```r
logtrainHouse <- logtrainHouse[apply(logtrainHouse,1, function(x){sum(is.na(x))==0}),]
# dim(logtrainHouse)

# sum(is.na(logtrainHouse))
```

All missing values have been eliminated.

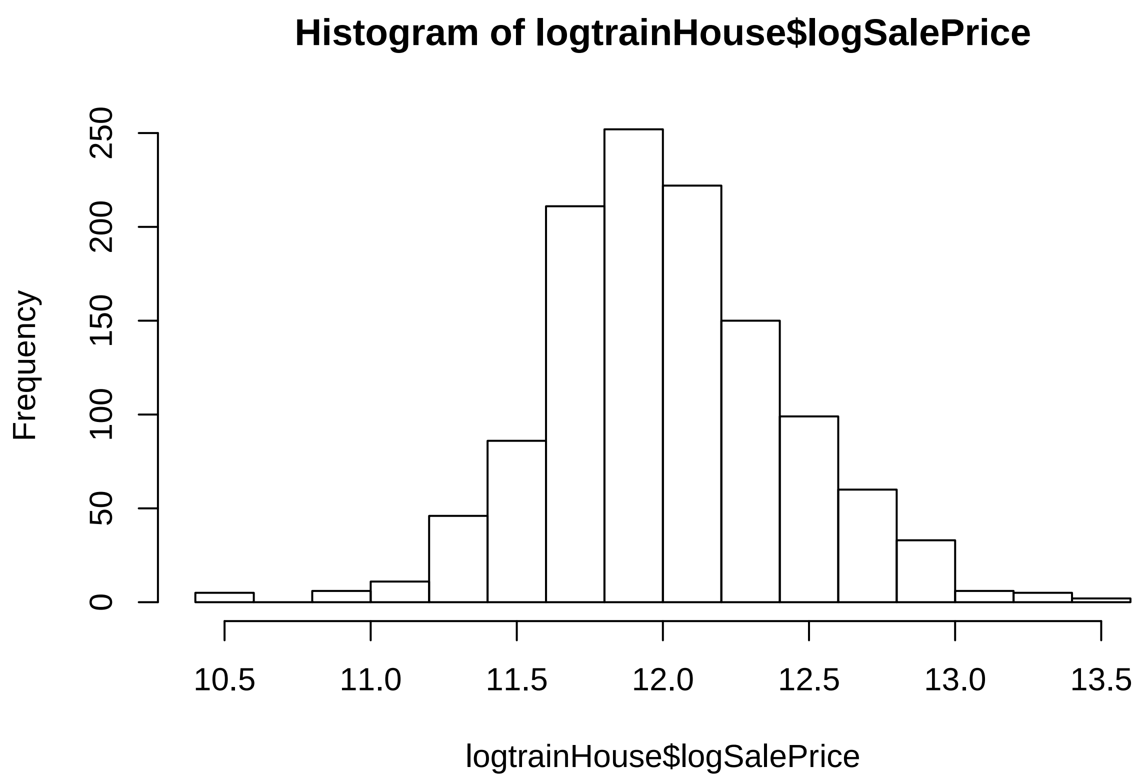
Some of the variables are 'Year' vaules, converting them to duration would be appropriate.

```
# Replace year column with age, w.r.t 2017.

yearToAge <- (logtrainHouse %>% dplyr::select(contains('Year'),YrSold) %>%
                mutate(BuiltBefore = (2017 - YearBuilt),
                       RemodBefore = (2017 - YearBuilt),
                       SoldBefore = (2017 - YrSold)) %>%
                dplyr::select(BuiltBefore, RemodBefore,SoldBefore))

logtrainHouse <- logtrainHouse %>% dplyr::select(-contains('Year'))
logtrainHouse <- logtrainHouse %>% dplyr::select(-YrSold)

logtrainHouse$BuiltBefore <- yearToAge$BuiltBefore
logtrainHouse$RemodBefore <-  yearToAge$RemodBefore
logtrainHouse$SoldBefore <- yearToAge$SoldBefore
logtrainHouse <- logtrainHouse[,!colnames(logtrainHouse) %in% c("MoSold")]
```

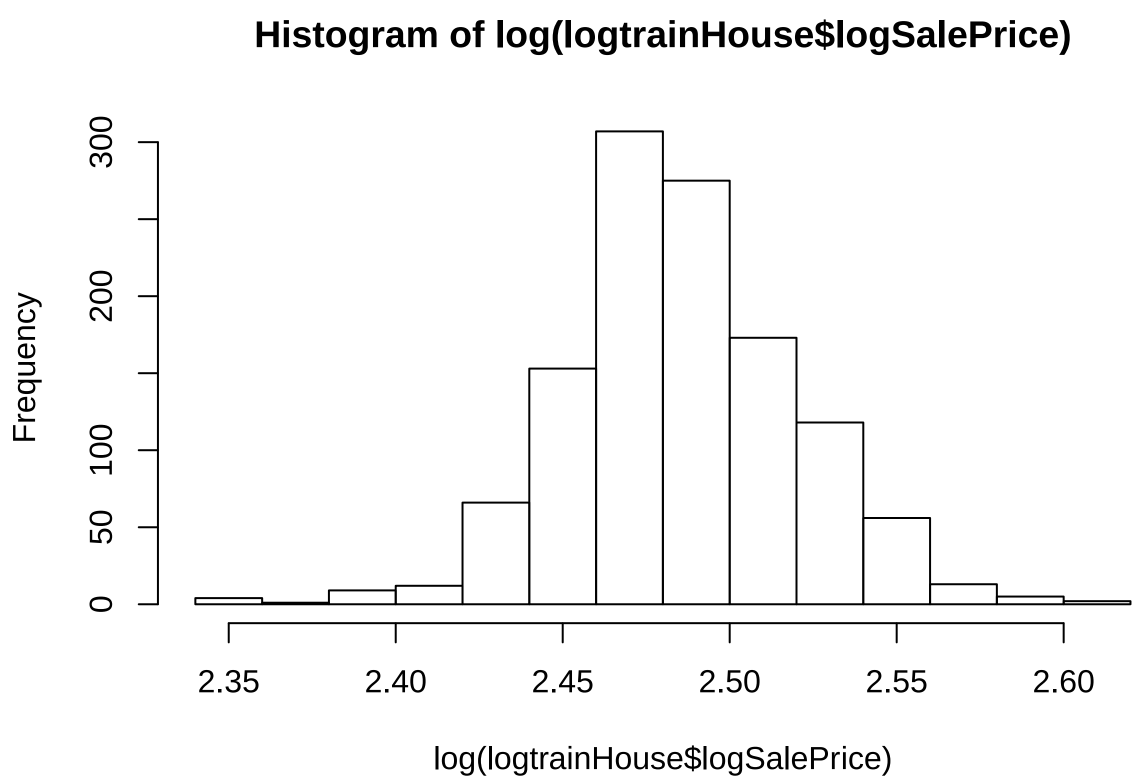
The above code also removes 'Month sold' column, since 'Year Sold' variable is already present in the dataset, this variable do not add much value.

Let us check the distribution of the dependent variable again.

```
hist(logtrainHouse$logSalePrice)
```



```r
hist(log(logtrainHouse$logSalePrice))
```



Check if the cleansed data fits t-distribution.

```r
set.seed(0)
logSalePrice <- logtrainHouse$logSalePrice
fit.pois.Params <- fitdistr(logSalePrice, "t")
(fitDist1000(logSalePrice,"rt",(fit.pois.Params$estimate)[3]))
```

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

```r
set.seed(0)
loglogSalePrice <- log(logSalePrice)
fit.pois.Params <- fitdistr(loglogSalePrice, "t")
(fitDist1000(loglogSalePrice,"rt",(fit.pois.Params$estimate)[3]))
```

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

```r
# colnames(logtrainHouse)
```

t-distribution does not fit the data.

Check again what transformation fits the response variable better with lognormal distribution.

```r
set.seed(0)
logSalePrice <- logtrainHouse$logSalePrice
fit.pois.Params <- fitdistr(logSalePrice, "normal")
(fitDist1000(logSalePrice,"rnorm",fit.pois.Params$estimate))
```

```
## [1] 590
```

```
set.seed(0)
loglogSalePrice <- log(logSalePrice)
fit.pois.Params <- fitdistr(loglogSalePrice, "normal")
(fitDist1000(loglogSalePrice,"rnorm",fit.pois.Params$estimate))
```

```
## [1] 740
```

```
# colnames(logtrainHouse)
```

Log-log transformation seems to work well even with the truncated data.

3. Model fitting

1. GLM

```
model2 <- glm(log(logSalePrice)~. , data = logtrainHouse)
```

```
## Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]): contrasts can be applied only to f
```

```
summary(model2)
```

```
## Error in summary(model2): object 'model2' not found
```

But glm is failing because 'Utilities' column is categorical and has a single level. Since all the values are the same, it doesn't explain any variance in logSalePrice column. Therefore, removing the column.

```
factorsWith1Level <- function(x){
  if(is.factor(x)){
    return(length(unique(x)) == 1)
  }
  else{
    return(FALSE)
  }
}

names(which(sapply(logtrainHouse, factorsWith1Level)))
```

```
## [1] "Utilities"
```

```
logtrainHouse <- logtrainHouse[,-which(sapply(logtrainHouse, factorsWith1Level))]
```

glm with log and loglog SalePrice as response variable:

```
# Test if glm fails again by trying log model
loglinearModel <- glm(logSalePrice~.,data = logtrainHouse)

# log-log model
logloglinearModel <- glm(log(logSalePrice)~.,data = logtrainHouse)


(c(logModelAIC=loglinearModel$aic, loglogModelAIC=logloglinearModel$aic))
```

```
##    logModelAIC loglogModelAIC
##       -1727.37       -7614.23
```

loglog transformation of the dependent variable results in lower AIC for the given dataset. Therefore, I am pursuing loglog model further.

```r
summary(logloglinearModel)
```

```
##
## Call:
## glm(formula = log(logSalePrice) ~ ., data = logtrainHouse)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.061828  -0.003914   0.000133   0.004698   0.056312
##
## Coefficients: (3 not defined because of singularities)
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.077e+00  2.333e-02  89.046  < 2e-16 ***
## MSSubClass           -2.285e-05  4.183e-05  -0.546 0.585053
## MSZoningFV            4.072e-02  4.939e-03   8.243 5.17e-16 ***
## MSZoningRH            3.721e-02  5.022e-03   7.408 2.69e-13 ***
## MSZoningRL            3.694e-02  4.213e-03   8.769  < 2e-16 ***
## MSZoningRM            3.500e-02  3.936e-03   8.893  < 2e-16 ***
## LotFrontage           4.021e-05  2.008e-05   2.003 0.045497 *
## LotArea               2.513e-07  5.653e-08   4.446 9.73e-06 ***
## StreetPave            7.244e-03  5.332e-03   1.359 0.174582
## LotShapeIR2           2.445e-03  2.121e-03   1.153 0.249385
## LotShapeIR3           4.727e-03  4.603e-03   1.027 0.304691
## LotShapeReg           7.432e-04  7.718e-04   0.963 0.335818
## LandContourHLS        4.849e-03  2.237e-03   2.168 0.030398 *
## LandContourLow       -2.117e-03  3.281e-03  -0.645 0.518837
## LandContourLvl        3.578e-03  1.641e-03   2.181 0.029450 *
## LotConfigCulDSac      2.926e-03  1.914e-03   1.529 0.126602
## LotConfigFR2         -2.576e-03  1.956e-03  -1.317 0.188147
## LotConfigFR3         -8.581e-03  5.149e-03  -1.666 0.095926 .
## LotConfigInside      -1.054e-03  8.307e-04  -1.269 0.204890
## LandSlopeMod          3.217e-03  1.782e-03   1.805 0.071368 .
## LandSlopeSev         -1.758e-02  6.016e-03  -2.923 0.003548 **
## NeighborhoodBlueste  -1.707e-03  7.740e-03  -0.221 0.825518
## NeighborhoodBrDale   -5.186e-03  4.707e-03  -1.102 0.270777
## NeighborhoodBrkSide   9.667e-04  4.145e-03   0.233 0.815626
## NeighborhoodClearCr   3.386e-03  4.422e-03   0.766 0.444022
## NeighborhoodCollgCr   4.550e-05  3.178e-03   0.014 0.988578
## NeighborhoodCrawfor   1.091e-02  3.770e-03   2.894 0.003887 **
## NeighborhoodEdwards  -5.475e-03  3.472e-03  -1.577 0.115142
## NeighborhoodGilbert   3.686e-04  3.479e-03   0.106 0.915645
## NeighborhoodIDOTRR   -1.500e-03  4.639e-03  -0.323 0.746421
## NeighborhoodMeadowV  -1.418e-02  4.879e-03  -2.906 0.003739 **
## NeighborhoodMitchel  -2.995e-03  3.634e-03  -0.824 0.410058
## NeighborhoodNAmes    -1.089e-03  3.406e-03  -0.320 0.749254
## NeighborhoodNoRidge   3.505e-03  3.723e-03   0.941 0.346722
## NeighborhoodNPkVill  -1.162e-03  7.519e-03  -0.155 0.877173
## NeighborhoodNridgHt   6.975e-03  3.196e-03   2.182 0.029336 *
## NeighborhoodNWAmes   -2.421e-03  3.616e-03  -0.670 0.503268
## NeighborhoodOldTown  -3.358e-03  4.167e-03  -0.806 0.420518
## NeighborhoodSawyer   -3.202e-04  3.622e-03  -0.088 0.929569
## NeighborhoodSawyerW   3.166e-04  3.410e-03   0.093 0.926036
## NeighborhoodSomerst   1.836e-03  3.853e-03   0.477 0.633716
## NeighborhoodStoneBr   1.206e-02  3.697e-03   3.263 0.001140 **
```

```
## NeighborhoodSWISU      1.675e-03  4.166e-03   0.402 0.687762
## NeighborhoodTimber     1.121e-03  3.608e-03   0.311 0.756183
## NeighborhoodVeenker    7.412e-03  5.001e-03   1.482 0.138629
## Condition1Feedr        1.099e-03  2.110e-03   0.521 0.602596
## Condition1Norm         6.041e-03  1.694e-03   3.566 0.000380 ***
## Condition1PosA         2.532e-03  6.213e-03   0.408 0.683715
## Condition1PosN         4.801e-03  4.220e-03   1.138 0.255524
## Condition1RRAe        -1.937e-03  4.067e-03  -0.476 0.634007
## Condition1RRAn         4.617e-03  2.877e-03   1.605 0.108808
## Condition1RRNe         7.744e-03  9.711e-03   0.797 0.425400
## Condition1RRNn         8.270e-03  5.909e-03   1.400 0.161932
## Condition2Feedr        8.064e-03  9.123e-03   0.884 0.376977
## Condition2Norm         1.952e-03  7.747e-03   0.252 0.801121
## Condition2PosA         1.292e-02  1.489e-02   0.867 0.386009
## Condition2PosN        -6.991e-02  1.124e-02  -6.219 7.31e-10 ***
## Condition2RRNn        -6.021e-04  1.067e-02  -0.056 0.955003
## BldgType2fmCon         4.865e-03  6.009e-03   0.810 0.418310
## BldgTypeDuplex        -1.213e-05  3.283e-03  -0.004 0.997053
## BldgTypeTwnhs         -4.020e-03  4.898e-03  -0.821 0.411969
## BldgTypeTwnhsE         2.169e-04  4.493e-03   0.048 0.961503
## HouseStyle1.5Unf      -2.785e-03  3.199e-03  -0.870 0.384274
## HouseStyle1Story      -3.654e-03  1.921e-03  -1.902 0.057498 .
## HouseStyle2.5Fin      -4.130e-03  5.043e-03  -0.819 0.413060
## HouseStyle2.5Unf       2.305e-03  3.934e-03   0.586 0.557998
## HouseStyle2Story      -1.705e-03  1.574e-03  -1.083 0.278885
## HouseStyleSFoyer       2.439e-04  2.716e-03   0.090 0.928443
## HouseStyleSLvl        -6.553e-04  2.570e-03  -0.255 0.798817
## OverallQual            3.762e-03  4.505e-04   8.350 2.22e-16 ***
## OverallCond            3.568e-03  3.695e-04   9.657  < 2e-16 ***
## RoofStyleGable         1.292e-02  1.073e-02   1.203 0.229092
## RoofStyleGambrel       1.393e-02  1.122e-02   1.242 0.214668
## RoofStyleHip           1.287e-02  1.077e-02   1.196 0.232158
## RoofStyleMansard       1.725e-02  1.175e-02   1.468 0.142507
## RoofMatlCompShg        2.359e-01  1.341e-02  17.591  < 2e-16 ***
## RoofMatlMembran        2.931e-01  2.125e-02  13.794  < 2e-16 ***
## RoofMatlRoll           2.402e-01  1.664e-02  14.435  < 2e-16 ***
## RoofMatlTar&Grv        2.456e-01  1.641e-02  14.966  < 2e-16 ***
## RoofMatlWdShake        2.263e-01  1.657e-02  13.657  < 2e-16 ***
## RoofMatlWdShngl        2.445e-01  1.392e-02  17.559  < 2e-16 ***
## Exterior1stAsphShn     9.552e-04  1.351e-02   0.071 0.943634
## Exterior1stBrkComm    -1.690e-02  1.246e-02  -1.356 0.175346
## Exterior1stBrkFace     6.545e-03  5.220e-03   1.254 0.210146
## Exterior1stCBlock     -6.818e-03  1.086e-02  -0.628 0.530148
## Exterior1stCemntBd    -1.182e-02  8.983e-03  -1.315 0.188707
## Exterior1stHdBoard     2.090e-03  5.303e-03   0.394 0.693590
## Exterior1stImStucc    -3.463e-03  1.127e-02  -0.307 0.758721
## Exterior1stMetalSd     8.236e-03  5.993e-03   1.374 0.169657
## Exterior1stPlywood     4.427e-05  5.258e-03   0.008 0.993285
## Exterior1stStone       4.998e-03  1.258e-02   0.397 0.691227
## Exterior1stStucco      3.206e-03  5.843e-03   0.549 0.583357
## Exterior1stVinylSd     2.881e-03  5.433e-03   0.530 0.596038
## Exterior1stWd Sdng    -3.664e-04  4.992e-03  -0.073 0.941508
## Exterior1stWdShing     2.294e-03  5.362e-03   0.428 0.668865
## Exterior2ndAsphShn     7.808e-04  9.021e-03   0.087 0.931041
```

```
## Exterior2ndBrk Cmn    7.147e-03  9.489e-03   0.753 0.451524
## Exterior2ndBrkFace   -1.906e-03  5.422e-03  -0.352 0.725235
## Exterior2ndCBlock            NA         NA      NA       NA
## Exterior2ndCmentBd    1.701e-02  8.849e-03   1.922 0.054886 .
## Exterior2ndHdBoard    5.976e-04  5.124e-03   0.117 0.907172
## Exterior2ndImStucc    4.035e-03  5.850e-03   0.690 0.490478
## Exterior2ndMetalSd   -2.829e-03  5.836e-03  -0.485 0.628010
## Exterior2ndOther     -6.343e-03  1.111e-02  -0.571 0.568007
## Exterior2ndPlywood    2.817e-03  4.909e-03   0.574 0.566254
## Exterior2ndStone      1.645e-03  7.745e-03   0.212 0.831883
## Exterior2ndStucco     1.761e-03  5.621e-03   0.313 0.754137
## Exterior2ndVinylSd    1.819e-03  5.218e-03   0.349 0.727522
## Exterior2ndWd Sdng    4.079e-03  4.800e-03   0.850 0.395624
## Exterior2ndWd Shng    1.446e-04  4.975e-03   0.029 0.976818
## MasVnrTypeBrkFace     3.144e-03  3.417e-03   0.920 0.357780
## MasVnrTypeNone        2.697e-03  3.417e-03   0.789 0.430167
## MasVnrTypeStone       4.694e-03  3.553e-03   1.321 0.186767
## MasVnrArea           -7.580e-07  2.533e-06  -0.299 0.764782
## ExterQualFa           1.484e-03  4.620e-03   0.321 0.748182
## ExterQualGd          -1.070e-04  2.084e-03  -0.051 0.959050
## ExterQualTA          -1.469e-04  2.353e-03  -0.062 0.950211
## ExterCondFa          -7.571e-03  7.378e-03  -1.026 0.305031
## ExterCondGd          -5.251e-03  7.000e-03  -0.750 0.453312
## ExterCondPo          -9.583e-03  1.297e-02  -0.739 0.460017
## ExterCondTA          -4.343e-03  6.980e-03  -0.622 0.533941
## FoundationCBlock      1.120e-03  1.334e-03   0.839 0.401403
## FoundationPConc       2.954e-03  1.472e-03   2.007 0.044985 *
## FoundationSlab       -3.424e-03  3.449e-03  -0.993 0.321101
## FoundationStone       1.276e-02  4.441e-03   2.873 0.004147 **
## FoundationWood       -1.495e-02  7.338e-03  -2.037 0.041876 *
## TotalBsmtSF           5.736e-06  1.735e-06   3.305 0.000983 ***
## HeatingGasW           6.577e-03  2.924e-03   2.250 0.024691 *
## HeatingGrav          -1.907e-02  4.703e-03  -4.056 5.38e-05 ***
## HeatingOthW           1.829e-03  7.541e-03   0.243 0.808403
## HeatingWall           9.925e-03  6.793e-03   1.461 0.144293
## HeatingQCFa          -2.888e-03  2.143e-03  -1.347 0.178175
## HeatingQCGd          -2.792e-03  9.335e-04  -2.990 0.002853 **
## HeatingQCPo          -7.425e-03  1.090e-02  -0.681 0.496012
## HeatingQCTA          -3.844e-03  9.384e-04  -4.097 4.53e-05 ***
## CentralAirY           6.221e-03  1.644e-03   3.784 0.000164 ***
## ElectricalFuseF      -1.030e-03  2.505e-03  -0.411 0.680978
## ElectricalFuseP      -6.699e-03  7.011e-03  -0.956 0.339513
## ElectricalMix         8.138e-03  1.139e-02   0.715 0.474980
## ElectricalSBrkr      -4.151e-04  1.282e-03  -0.324 0.746157
## X1stFlrSF             2.082e-05  2.332e-06   8.927  < 2e-16 ***
## X2ndFlrSF             1.531e-05  2.328e-06   6.576 7.75e-11 ***
## LowQualFinSF          1.267e-05  7.985e-06   1.586 0.112979
## GrLivArea                    NA         NA      NA       NA
## FullBath              2.261e-03  9.705e-04   2.330 0.020028 *
## HalfBath              2.788e-03  9.302e-04   2.997 0.002790 **
## BedroomAbvGr          2.036e-04  5.961e-04   0.342 0.732733
## KitchenAbvGr         -3.353e-03  2.401e-03  -1.397 0.162770
## KitchenQualFa        -4.869e-03  2.658e-03  -1.831 0.067326 .
## KitchenQualGd        -4.469e-03  1.462e-03  -3.057 0.002292 **
```

```
## KitchenQualTA        -5.495e-03  1.675e-03  -3.281 0.001072 **
## TotRmsAbvGrd          5.697e-04  4.223e-04   1.349 0.177568
## FunctionalMaj2       -2.093e-02  5.907e-03  -3.542 0.000415 ***
## FunctionalMin1        2.357e-03  3.687e-03   0.639 0.522848
## FunctionalMin2       -5.215e-04  3.627e-03  -0.144 0.885705
## FunctionalMod        -4.665e-03  4.532e-03  -1.029 0.303536
## FunctionalTyp         4.878e-03  3.107e-03   1.570 0.116674
## Fireplaces            2.074e-03  6.203e-04   3.344 0.000858 ***
## PavedDriveP          -1.780e-03  2.384e-03  -0.747 0.455410
## PavedDriveY           8.634e-04  1.433e-03   0.602 0.547034
## WoodDeckSF            8.915e-06  2.687e-06   3.318 0.000938 ***
## OpenPorchSF           4.857e-06  5.228e-06   0.929 0.353051
## EnclosedPorch         1.382e-05  5.602e-06   2.467 0.013774 *
## X3SsnPorch            2.031e-05  9.956e-06   2.040 0.041575 *
## ScreenPorch           2.414e-05  5.554e-06   4.346 1.53e-05 ***
## PoolArea              4.290e-06  8.292e-06   0.517 0.604959
## MiscVal              -3.148e-06  1.684e-06  -1.869 0.061918 .
## SaleTypeCon           6.515e-03  7.309e-03   0.891 0.372984
## SaleTypeConLD         1.219e-02  4.282e-03   2.847 0.004507 **
## SaleTypeConLI        -4.281e-03  5.261e-03  -0.814 0.416034
## SaleTypeConLw         1.920e-03  5.007e-03   0.384 0.701406
## SaleTypeCWD           4.877e-03  5.336e-03   0.914 0.360875
## SaleTypeNew           3.955e-03  6.557e-03   0.603 0.546498
## SaleTypeOth           7.723e-03  5.925e-03   1.303 0.192718
## SaleTypeWD           -1.803e-03  1.923e-03  -0.938 0.348637
## SaleConditionAdjLand  1.317e-02  5.996e-03   2.197 0.028268 *
## SaleConditionAlloca   9.189e-03  3.802e-03   2.417 0.015840 *
## SaleConditionFamily   4.243e-04  2.645e-03   0.160 0.872579
## SaleConditionNormal   7.056e-03  1.277e-03   5.523 4.23e-08 ***
## SaleConditionPartial  5.028e-03  6.315e-03   0.796 0.426088
## GarageArea            1.561e-05  1.936e-06   8.064 2.08e-15 ***
## BsmtFinSF1            8.220e-06  8.497e-07   9.674  < 2e-16 ***
## BsmtFinSF2            2.900e-06  2.047e-06   1.417 0.156890
## BuiltBefore          -1.813e-04  3.075e-05  -5.897 5.04e-09 ***
## RemodBefore                  NA         NA      NA       NA
## SoldBefore            7.364e-05  2.281e-04   0.323 0.746919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 8.637668e-05)
##
##     Null deviance: 1.427347  on 1193  degrees of freedom
## Residual deviance: 0.087327  on 1011  degrees of freedom
## AIC: -7614.2
##
## Number of Fisher Scoring iterations: 2
```

GLM resulted in many insignificant parameters. Most of the categorical factors are insignificant. a. Try factorizing the categorical variables using alternating least squares. This will reduce the number of variables. Since the new variables will be a linear combination of the original variables, fewer columns could represent the information in all the categorical columns.Then significant ones can be found out of those reduced number of columns.

b. Use PCA for continuous variables and find signifcant factors. Use both sets of significant factors in finding the house prices.

The following code does PCA on continuous variables and extracts factors.

```
# Extract continuous predictors
factorDataCont <- logtrainHouse[,sapply(logtrainHouse,function(x){!is.factor(x)} )]

# Scale them
scaledContinuousData <- scale(factorDataCont[,-1])

# Generate factors and loadings
pcaCont <- princomp(scaledContinuousData)

# Read factors into continuousFactors
continuousFactors <- as.data.frame(pcaCont$scores)
```

The following code tries to reduce the categorical columns into fewer columns using alternate least squares approach. It does a grid search of number of factors to break the data into. It tries to break categorical values into 5 to 30 factors with an increment of 5 factors each time.

In each iteration it combines the factors obtained by als method with the PCA factors and builds a linear model.

Finally, the best model will be the one which results in the least cross-validated RMSE with fewer parameters.

```
# Categorical columns
factorData <- logtrainHouse[,sapply(logtrainHouse,is.factor)]
dim(factorData)
```

```
## [1] 1194   28
```

```
# Convert Categorical columns data into a data matrix
factorData.matrix <- as.matrix(sparse.model.matrix(~.-1,factorData))

diDat <- dim(factorData.matrix)


numFactors <- seq(5, 30, by=5)
modelResults <- data.frame(parameter=0,RMSE=0,Rsquared=0,RMSESD=0,RsquaredSD=0)
for(i in numFactors){

  set.seed(0)
  ## Row-wise elements with two components with random uniform priors.
  lInitFactors <- list(cbind(sapply(c(1:i), function(x){runif(diDat[1])})))

  ## Column-wise elements
  SInit<-matrix(1,nrow=diDat[2],ncol=i)

  # Split the sparse matrix into factors and loadings
  suppressMessages(  alsFactors <- als(CList=lInitFactors,
                    S=SInit,
                    PsiList=list(factorData.matrix)))

  # Get the factors
  requiredFactors <- alsFactors$CList[[1]]

  # Combine the response variable, continuous factors and categorical factors
  allFactorsData <- as.data.frame(cbind(logSalePrice=logtrainHouse$logSalePrice,continuousFactors, requi
```

```
  # Fit a 10-fold cv glm
  trCtrl <- trainControl(method = "cv", number = 10)
  model <- train(log(logSalePrice)~.,
                 data = allFactorsData,
                 method = "glm",
                 trControl = trCtrl
                 )
  modelResults <- rbind(modelResults,model$results)

}
```

```
## Initial RSS 1179028
## Iteration (opt. S): 1, RSS: 11209.57, RD: 0.9904925
## Iteration (opt. C): 2, RSS: 8766.643, RD: 0.2179321
## Iteration (opt. S): 3, RSS: 7621.618, RD: 0.1306116
## Iteration (opt. C): 4, RSS: 7360.846, RD: 0.03421471
## Iteration (opt. S): 5, RSS: 7218.221, RD: 0.01937624
## Iteration (opt. C): 6, RSS: 7139.678, RD: 0.01088114
## Iteration (opt. S): 7, RSS: 7079.505, RD: 0.008428001
## Iteration (opt. C): 8, RSS: 7045.568, RD: 0.004793714
## Iteration (opt. S): 9, RSS: 7022.746, RD: 0.003239241
## Iteration (opt. C): 10, RSS: 7005.209, RD: 0.002497159
## Iteration (opt. S): 11, RSS: 6991.049, RD: 0.00202138
## Iteration (opt. C): 12, RSS: 6980.066, RD: 0.001570873
## Iteration (opt. S): 13, RSS: 6971.042, RD: 0.001292928
## Iteration (opt. C): 14, RSS: 6963.806, RD: 0.001037994
## Iteration (opt. S): 15, RSS: 6957.526, RD: 0.0009018247
## Initial RSS / Final RSS = 1179028 / 6957.526 = 169.4608
## Initial RSS 4749294
## Iteration (opt. S): 1, RSS: 10611.46, RD: 0.9977657
## Iteration (opt. C): 2, RSS: 7865.615, RD: 0.258762
## Iteration (opt. S): 3, RSS: 6826.676, RD: 0.1320861
## Iteration (opt. C): 4, RSS: 6462.252, RD: 0.0533824
## Iteration (opt. S): 5, RSS: 6208.589, RD: 0.03925301
## Iteration (opt. C): 6, RSS: 6067.144, RD: 0.0227822
## Iteration (opt. S): 7, RSS: 5973.187, RD: 0.01548622
## Iteration (opt. C): 8, RSS: 5905.413, RD: 0.01134635
## Iteration (opt. S): 9, RSS: 5847.524, RD: 0.009802698
## Iteration (opt. C): 10, RSS: 5803.205, RD: 0.007579104
## Iteration (opt. S): 11, RSS: 5770.35, RD: 0.005661503
## Iteration (opt. C): 12, RSS: 5747.079, RD: 0.004032823
## Iteration (opt. S): 13, RSS: 5730.127, RD: 0.002949756
## Iteration (opt. C): 14, RSS: 5718.05, RD: 0.002107633
## Iteration (opt. S): 15, RSS: 5708.742, RD: 0.001627751
## Iteration (opt. C): 16, RSS: 5701.513, RD: 0.001266368
## Iteration (opt. S): 17, RSS: 5695.46, RD: 0.001061572
## Iteration (opt. C): 18, RSS: 5690.328, RD: 0.0009010271
## Initial RSS / Final RSS = 4749294 / 5690.328 = 834.6255
## Initial RSS 10690469
## Iteration (opt. S): 1, RSS: 10389.34, RD: 0.9990282
## Iteration (opt. C): 2, RSS: 7213.242, RD: 0.3057075
## Iteration (opt. S): 3, RSS: 6262.607, RD: 0.1317902
## Iteration (opt. C): 4, RSS: 5841.152, RD: 0.06729707
## Iteration (opt. S): 5, RSS: 5574.779, RD: 0.04560279
```

```
## Iteration (opt. C): 6, RSS: 5423.079, RD: 0.02721189
## Iteration (opt. S): 7, RSS: 5313.317, RD: 0.02023978
## Iteration (opt. C): 8, RSS: 5232.234, RD: 0.01526045
## Iteration (opt. S): 9, RSS: 5164.782, RD: 0.01289164
## Iteration (opt. C): 10, RSS: 5113.195, RD: 0.009988147
## Iteration (opt. S): 11, RSS: 5072.428, RD: 0.007972907
## Iteration (opt. C): 12, RSS: 5039.714, RD: 0.006449445
## Iteration (opt. S): 13, RSS: 5013.97, RD: 0.005108131
## Iteration (opt. C): 14, RSS: 4992.111, RD: 0.00435959
## Iteration (opt. S): 15, RSS: 4971.771, RD: 0.004074437
## Iteration (opt. C): 16, RSS: 4953.459, RD: 0.003683231
## Iteration (opt. S): 17, RSS: 4935.216, RD: 0.003682943
## Iteration (opt. C): 18, RSS: 4917.789, RD: 0.003531043
## Iteration (opt. S): 19, RSS: 4900.165, RD: 0.003583677
## Iteration (opt. C): 20, RSS: 4882.637, RD: 0.003577124
## Iteration (opt. S): 21, RSS: 4863.623, RD: 0.003894251
## Iteration (opt. C): 22, RSS: 4843.822, RD: 0.004071197
## Iteration (opt. S): 23, RSS: 4821.741, RD: 0.004558612
## Iteration (opt. C): 24, RSS: 4799.113, RD: 0.004692783
## Iteration (opt. S): 25, RSS: 4775.992, RD: 0.004817813
## Iteration (opt. C): 26, RSS: 4751.672, RD: 0.005092251
## Iteration (opt. S): 27, RSS: 4725.757, RD: 0.005453793
## Iteration (opt. C): 28, RSS: 4701.364, RD: 0.005161651
## Iteration (opt. S): 29, RSS: 4679.854, RD: 0.004575443
## Iteration (opt. C): 30, RSS: 4661.45, RD: 0.003932461
## Iteration (opt. S): 31, RSS: 4646.603, RD: 0.003185074
## Iteration (opt. C): 32, RSS: 4634.54, RD: 0.002596196
## Iteration (opt. S): 33, RSS: 4623.736, RD: 0.002331046
## Iteration (opt. C): 34, RSS: 4613.976, RD: 0.002110926
## Iteration (opt. S): 35, RSS: 4604.776, RD: 0.00199391
## Iteration (opt. C): 36, RSS: 4596.462, RD: 0.001805533
## Iteration (opt. S): 37, RSS: 4588.551, RD: 0.001721125
## Iteration (opt. C): 38, RSS: 4581.173, RD: 0.001607884
## Iteration (opt. S): 39, RSS: 4574.111, RD: 0.001541429
## Iteration (opt. C): 40, RSS: 4567.534, RD: 0.001437892
## Iteration (opt. S): 41, RSS: 4561.3, RD: 0.001364843
## Iteration (opt. C): 42, RSS: 4555.55, RD: 0.001260644
## Iteration (opt. S): 43, RSS: 4549.946, RD: 0.001230199
## Iteration (opt. C): 44, RSS: 4544.688, RD: 0.001155725
## Iteration (opt. S): 45, RSS: 4540.059, RD: 0.001018413
## Iteration (opt. C): 46, RSS: 4535.955, RD: 0.0009039777
## Initial RSS / Final RSS = 10690469 / 4535.955 = 2356.829
## Initial RSS 19100548
## Iteration (opt. S): 1, RSS: 10278.68, RD: 0.9994619
## Iteration (opt. C): 2, RSS: 6705.078, RD: 0.3476715
## Iteration (opt. S): 3, RSS: 5785.069, RD: 0.1372108
## Iteration (opt. C): 4, RSS: 5371.026, RD: 0.07157097
## Iteration (opt. S): 5, RSS: 5114.362, RD: 0.04778683
## Iteration (opt. C): 6, RSS: 4943.661, RD: 0.03337668
## Iteration (opt. S): 7, RSS: 4798.852, RD: 0.02929197
## Iteration (opt. C): 8, RSS: 4682.807, RD: 0.02418169
## Iteration (opt. S): 9, RSS: 4580.506, RD: 0.02184619
## Iteration (opt. C): 10, RSS: 4504.434, RD: 0.01660769
## Iteration (opt. S): 11, RSS: 4444.795, RD: 0.01324002
```

```
## Iteration (opt. C): 12, RSS: 4401.537, RD: 0.009732303
## Iteration (opt. S): 13, RSS: 4367.993, RD: 0.007621162
## Iteration (opt. C): 14, RSS: 4342.47, RD: 0.005843003
## Iteration (opt. S): 15, RSS: 4319.387, RD: 0.005315778
## Iteration (opt. C): 16, RSS: 4295.812, RD: 0.005457831
## Iteration (opt. S): 17, RSS: 4268.123, RD: 0.006445706
## Iteration (opt. C): 18, RSS: 4237.102, RD: 0.007267965
## Iteration (opt. S): 19, RSS: 4199.998, RD: 0.008756929
## Iteration (opt. C): 20, RSS: 4161.457, RD: 0.009176553
## Iteration (opt. S): 21, RSS: 4120.21, RD: 0.009911655
## Iteration (opt. C): 22, RSS: 4080.132, RD: 0.009727164
## Iteration (opt. S): 23, RSS: 4038.974, RD: 0.01008748
## Iteration (opt. C): 24, RSS: 3999.156, RD: 0.009858354
## Iteration (opt. S): 25, RSS: 3962.269, RD: 0.009223742
## Iteration (opt. C): 26, RSS: 3931.767, RD: 0.007697999
## Iteration (opt. S): 27, RSS: 3905.626, RD: 0.006648666
## Iteration (opt. C): 28, RSS: 3884.113, RD: 0.005508193
## Iteration (opt. S): 29, RSS: 3864.609, RD: 0.00502154
## Iteration (opt. C): 30, RSS: 3847.739, RD: 0.004365183
## Iteration (opt. S): 31, RSS: 3833.487, RD: 0.00370419
## Iteration (opt. C): 32, RSS: 3821.997, RD: 0.002997237
## Iteration (opt. S): 33, RSS: 3812.572, RD: 0.002465914
## Iteration (opt. C): 34, RSS: 3804.917, RD: 0.002007705
## Iteration (opt. S): 35, RSS: 3798.636, RD: 0.001650996
## Iteration (opt. C): 36, RSS: 3793.493, RD: 0.001353693
## Iteration (opt. S): 37, RSS: 3789.16, RD: 0.001142408
## Iteration (opt. C): 38, RSS: 3785.568, RD: 0.0009478109
## Initial RSS / Final RSS = 19100548 / 3785.568 = 5045.622
## Initial RSS 29842526
## Iteration (opt. S): 1, RSS: 10181.64, RD: 0.9996588
## Iteration (opt. C): 2, RSS: 6302.825, RD: 0.3809616
## Iteration (opt. S): 3, RSS: 5360.045, RD: 0.1495805
## Iteration (opt. C): 4, RSS: 4890.547, RD: 0.08759225
## Iteration (opt. S): 5, RSS: 4606.55, RD: 0.05807058
## Iteration (opt. C): 6, RSS: 4414.536, RD: 0.04168282
## Iteration (opt. S): 7, RSS: 4258.959, RD: 0.03524186
## Iteration (opt. C): 8, RSS: 4131.179, RD: 0.03000275
## Iteration (opt. S): 9, RSS: 4014.972, RD: 0.02812932
## Iteration (opt. C): 10, RSS: 3921.967, RD: 0.02316456
## Iteration (opt. S): 11, RSS: 3849.155, RD: 0.01856501
## Iteration (opt. C): 12, RSS: 3795.549, RD: 0.01392674
## Iteration (opt. S): 13, RSS: 3752.044, RD: 0.01146207
## Iteration (opt. C): 14, RSS: 3715.504, RD: 0.009738674
## Iteration (opt. S): 15, RSS: 3681.323, RD: 0.009199568
## Iteration (opt. C): 16, RSS: 3650.097, RD: 0.008482216
## Iteration (opt. S): 17, RSS: 3617.486, RD: 0.008934449
## Iteration (opt. C): 18, RSS: 3583.091, RD: 0.00950785
## Iteration (opt. S): 19, RSS: 3545.377, RD: 0.01052572
## Iteration (opt. C): 20, RSS: 3508.962, RD: 0.01027108
## Iteration (opt. S): 21, RSS: 3473.676, RD: 0.01005601
## Iteration (opt. C): 22, RSS: 3442.394, RD: 0.009005336
## Iteration (opt. S): 23, RSS: 3413.672, RD: 0.00834362
## Iteration (opt. C): 24, RSS: 3389.284, RD: 0.007144332
## Iteration (opt. S): 25, RSS: 3368.226, RD: 0.006213101
```

```
## Iteration (opt. C): 26, RSS: 3350.635, RD: 0.005222468
## Iteration (opt. S): 27, RSS: 3335.107, RD: 0.004634389
## Iteration (opt. C): 28, RSS: 3321.506, RD: 0.004078149
## Iteration (opt. S): 29, RSS: 3307.676, RD: 0.004163713
## Iteration (opt. C): 30, RSS: 3293.835, RD: 0.004184618
## Iteration (opt. S): 31, RSS: 3279.001, RD: 0.004503624
## Iteration (opt. C): 32, RSS: 3264.022, RD: 0.004567996
## Iteration (opt. S): 33, RSS: 3248.849, RD: 0.004648584
## Iteration (opt. C): 34, RSS: 3234.684, RD: 0.004359937
## Iteration (opt. S): 35, RSS: 3220.139, RD: 0.004496583
## Iteration (opt. C): 36, RSS: 3206.642, RD: 0.004191655
## Iteration (opt. S): 37, RSS: 3193.592, RD: 0.004069671
## Iteration (opt. C): 38, RSS: 3181.056, RD: 0.003925233
## Iteration (opt. S): 39, RSS: 3168.282, RD: 0.004015659
## Iteration (opt. C): 40, RSS: 3156.118, RD: 0.003839352
## Iteration (opt. S): 41, RSS: 3143.981, RD: 0.003845417
## Iteration (opt. C): 42, RSS: 3132.858, RD: 0.003537799
## Iteration (opt. S): 43, RSS: 3122.517, RD: 0.003300917
## Iteration (opt. C): 44, RSS: 3113.094, RD: 0.003017712
## Iteration (opt. S): 45, RSS: 3104.108, RD: 0.002886661
## Iteration (opt. C): 46, RSS: 3096.237, RD: 0.002535677
## Iteration (opt. S): 47, RSS: 3089.654, RD: 0.00212596
## Iteration (opt. C): 48, RSS: 3084.354, RD: 0.001715451
## Iteration (opt. S): 49, RSS: 3079.786, RD: 0.001481222
## Iteration (opt. C): 50, RSS: 3076.1, RD: 0.001196683
## Iteration (opt. S): 51, RSS: 3073.002, RD: 0.001007159
## Iteration (opt. C): 52, RSS: 3070.529, RD: 0.0008045821
## Initial RSS / Final RSS = 29842526 / 3070.529 = 9719.016
## Initial RSS 43196738
## Iteration (opt. S): 1, RSS: 10112.14, RD: 0.9997659
## Iteration (opt. C): 2, RSS: 5997.454, RD: 0.4069055
## Iteration (opt. S): 3, RSS: 5008, RD: 0.1649789
## Iteration (opt. C): 4, RSS: 4514.51, RD: 0.09854048
## Iteration (opt. S): 5, RSS: 4217.414, RD: 0.06580912
## Iteration (opt. C): 6, RSS: 4024.773, RD: 0.04567757
## Iteration (opt. S): 7, RSS: 3869.548, RD: 0.03856739
## Iteration (opt. C): 8, RSS: 3745.302, RD: 0.0321085
## Iteration (opt. S): 9, RSS: 3631.808, RD: 0.03030317
## Iteration (opt. C): 10, RSS: 3539.285, RD: 0.02547561
## Iteration (opt. S): 11, RSS: 3458.991, RD: 0.02268645
## Iteration (opt. C): 12, RSS: 3395.141, RD: 0.0184591
## Iteration (opt. S): 13, RSS: 3339.773, RD: 0.0163081
## Iteration (opt. C): 14, RSS: 3292.96, RD: 0.01401699
## Iteration (opt. S): 15, RSS: 3247.776, RD: 0.01372127
## Iteration (opt. C): 16, RSS: 3206.576, RD: 0.01268568
## Iteration (opt. S): 17, RSS: 3164.897, RD: 0.01299778
## Iteration (opt. C): 18, RSS: 3125.417, RD: 0.01247434
## Iteration (opt. S): 19, RSS: 3085.33, RD: 0.01282625
## Iteration (opt. C): 20, RSS: 3048.54, RD: 0.01192422
## Iteration (opt. S): 21, RSS: 3012.228, RD: 0.01191133
## Iteration (opt. C): 22, RSS: 2979.731, RD: 0.01078814
## Iteration (opt. S): 23, RSS: 2948.816, RD: 0.01037523
## Iteration (opt. C): 24, RSS: 2921.022, RD: 0.009425584
## Iteration (opt. S): 25, RSS: 2894.322, RD: 0.00914044
```

```
## Iteration (opt. C): 26, RSS: 2869.652, RD: 0.008523536
## Iteration (opt. S): 27, RSS: 2843.92, RD: 0.008967228
## Iteration (opt. C): 28, RSS: 2820.627, RD: 0.008190447
## Iteration (opt. S): 29, RSS: 2798.113, RD: 0.007981631
## Iteration (opt. C): 30, RSS: 2777.077, RD: 0.007518051
## Iteration (opt. S): 31, RSS: 2754.593, RD: 0.008096145
## Iteration (opt. C): 32, RSS: 2733.388, RD: 0.007698139
## Iteration (opt. S): 33, RSS: 2712.377, RD: 0.007686985
## Iteration (opt. C): 34, RSS: 2694.668, RD: 0.006528708
## Iteration (opt. S): 35, RSS: 2679.277, RD: 0.005711745
## Iteration (opt. C): 36, RSS: 2666.717, RD: 0.004687965
## Iteration (opt. S): 37, RSS: 2654.72, RD: 0.00449879
## Iteration (opt. C): 38, RSS: 2643.586, RD: 0.004193853
## Iteration (opt. S): 39, RSS: 2632.849, RD: 0.004061721
## Iteration (opt. C): 40, RSS: 2623.208, RD: 0.003661792
## Iteration (opt. S): 41, RSS: 2614.24, RD: 0.003418721
## Iteration (opt. C): 42, RSS: 2605.949, RD: 0.003171531
## Iteration (opt. S): 43, RSS: 2598.063, RD: 0.003025973
## Iteration (opt. C): 44, RSS: 2591.085, RD: 0.00268583
## Iteration (opt. S): 45, RSS: 2584.753, RD: 0.002443873
## Iteration (opt. C): 46, RSS: 2579.201, RD: 0.002147873
## Iteration (opt. S): 47, RSS: 2574.123, RD: 0.001969044
## Iteration (opt. C): 48, RSS: 2569.609, RD: 0.001753545
## Iteration (opt. S): 49, RSS: 2565.593, RD: 0.001562596
## Iteration (opt. C): 50, RSS: 2562.054, RD: 0.001379391
## Iteration (opt. S): 51, RSS: 2558.868, RD: 0.001243747
## Iteration (opt. C): 52, RSS: 2556.064, RD: 0.001095673
## Iteration (opt. S): 53, RSS: 2553.754, RD: 0.0009037367
## Initial RSS / Final RSS = 43196738 / 2553.754 = 16914.99
```

```
modelResults
```

```
##   parameter       RMSE  Rsquared      RMSESD RsquaredSD
## 1         0 0.00000000 0.0000000 0.000000000 0.00000000
## 2      none 0.01352852 0.8475676 0.004233161 0.08390702
## 3      none 0.01344673 0.8509370 0.003650187 0.08008821
## 4      none 0.01301645 0.8583387 0.004266357 0.08610211
## 5      none 0.01308068 0.8603089 0.003980264 0.06933103
## 6      none 0.01325231 0.8560137 0.004022252 0.07645518
## 7      none 0.01304481 0.8585001 0.004133533 0.07666487
```

15 categorical factors seem to result in the least RMSE. This reduces the 28 categorical variables to 15.

The following code creates 15 factors from categorical variables, combines the categorical factors with PCA factors and fits the final linear model.

It eliminates the insignificant predictors from the final model and calculates the RMSE of the train predictions.

```
set.seed(0)
## Row-wise elements with two components with random uniform priors.
lInitFactors <- list(cbind(sapply(c(1:15), function(x){runif(diDat[1])})))

## Column-wise elements
SInit<-matrix(1,nrow=diDat[2],ncol=15)

# Split the data matrix into factors and loadings
alsFactors <- als(CList=lInitFactors,
```

```r
                  S=SInit,
                  PsiList=list(factorData.matrix))
```

```
## Initial RSS 10690469
## Iteration (opt. S): 1, RSS: 10389.34, RD: 0.9990282
## Iteration (opt. C): 2, RSS: 7213.242, RD: 0.3057075
## Iteration (opt. S): 3, RSS: 6262.607, RD: 0.1317902
## Iteration (opt. C): 4, RSS: 5841.152, RD: 0.06729707
## Iteration (opt. S): 5, RSS: 5574.779, RD: 0.04560279
## Iteration (opt. C): 6, RSS: 5423.079, RD: 0.02721189
## Iteration (opt. S): 7, RSS: 5313.317, RD: 0.02023978
## Iteration (opt. C): 8, RSS: 5232.234, RD: 0.01526045
## Iteration (opt. S): 9, RSS: 5164.782, RD: 0.01289164
## Iteration (opt. C): 10, RSS: 5113.195, RD: 0.009988147
## Iteration (opt. S): 11, RSS: 5072.428, RD: 0.007972907
## Iteration (opt. C): 12, RSS: 5039.714, RD: 0.006449445
## Iteration (opt. S): 13, RSS: 5013.97, RD: 0.005108131
## Iteration (opt. C): 14, RSS: 4992.111, RD: 0.00435959
## Iteration (opt. S): 15, RSS: 4971.771, RD: 0.004074437
## Iteration (opt. C): 16, RSS: 4953.459, RD: 0.003683231
## Iteration (opt. S): 17, RSS: 4935.216, RD: 0.003682943
## Iteration (opt. C): 18, RSS: 4917.789, RD: 0.003531043
## Iteration (opt. S): 19, RSS: 4900.165, RD: 0.003583677
## Iteration (opt. C): 20, RSS: 4882.637, RD: 0.003577124
## Iteration (opt. S): 21, RSS: 4863.623, RD: 0.003894251
## Iteration (opt. C): 22, RSS: 4843.822, RD: 0.004071197
## Iteration (opt. S): 23, RSS: 4821.741, RD: 0.004558612
## Iteration (opt. C): 24, RSS: 4799.113, RD: 0.004692783
## Iteration (opt. S): 25, RSS: 4775.992, RD: 0.004817813
## Iteration (opt. C): 26, RSS: 4751.672, RD: 0.005092251
## Iteration (opt. S): 27, RSS: 4725.757, RD: 0.005453793
## Iteration (opt. C): 28, RSS: 4701.364, RD: 0.005161651
## Iteration (opt. S): 29, RSS: 4679.854, RD: 0.004575443
## Iteration (opt. C): 30, RSS: 4661.45, RD: 0.003932461
## Iteration (opt. S): 31, RSS: 4646.603, RD: 0.003185074
## Iteration (opt. C): 32, RSS: 4634.54, RD: 0.002596196
## Iteration (opt. S): 33, RSS: 4623.736, RD: 0.002331046
## Iteration (opt. C): 34, RSS: 4613.976, RD: 0.002110926
## Iteration (opt. S): 35, RSS: 4604.776, RD: 0.00199391
## Iteration (opt. C): 36, RSS: 4596.462, RD: 0.001805533
## Iteration (opt. S): 37, RSS: 4588.551, RD: 0.001721125
## Iteration (opt. C): 38, RSS: 4581.173, RD: 0.001607884
## Iteration (opt. S): 39, RSS: 4574.111, RD: 0.001541429
## Iteration (opt. C): 40, RSS: 4567.534, RD: 0.001437892
## Iteration (opt. S): 41, RSS: 4561.3, RD: 0.001364843
## Iteration (opt. C): 42, RSS: 4555.55, RD: 0.001260644
## Iteration (opt. S): 43, RSS: 4549.946, RD: 0.001230199
## Iteration (opt. C): 44, RSS: 4544.688, RD: 0.001155725
## Iteration (opt. S): 45, RSS: 4540.059, RD: 0.001018413
## Iteration (opt. C): 46, RSS: 4535.955, RD: 0.0009039777
## Initial RSS / Final RSS = 10690469 / 4535.955 = 2356.829
```

```r
# Get the factors
requiredFactors <- alsFactors$CList[[1]]
```

```r
# Combine the daata
allFactorsData <- as.data.frame(cbind(logSalePrice=logtrainHouse$logSalePrice,
                                      continuousFactors, requiredFactors))
# Give appropriate column names
colnames(allFactorsData) <- c("logSalePrice",colnames(continuousFactors),
                              sapply(c(1:15),function(x){paste0("Fact.",x)}))

# A function to recursively fit lm and eliminate insignificant predictors.
lmWithSignificantPredictors <- function(data, significantPredictors){
  repeat{
    # print("iter")
    prevNumPredictors <- length(significantPredictors)
    model <- lm(log(logSalePrice)~. ,data = data[,c(significantPredictors,"logSalePrice")])
    model.summary <- summary(model)
    model.coefficients <- model.summary$coefficients
    colnames(model.coefficients) <- c("estimate","stdError","tvalue", "pvalue" )
    numPredictors <- dim(model.coefficients[model.coefficients[,"pvalue"]<0.05,])[1] # 265
    significantPredictors <- (rownames(model.coefficients[model.coefficients[,"pvalue"]<=0.05,]))[-1]
      if (sum(model.coefficients[,"pvalue"]>0.05) <=0){
        break
      }
    }
    return(model)
  }

finalModel <- lmWithSignificantPredictors(allFactorsData,colnames(allFactorsData[,-c(1)]))

lm.summary <- summary(finalModel)
lm.coefficients <- lm.summary$coefficients
colnames(lm.coefficients) <- c("estimate","stdError","tvalue", "pvalue" )

# RMSE of the final GLM
(GLMRMSE = (sum((finalModel$residuals)^2)/nrow(allFactorsData))^0.5)
```

```
## [1] 0.01255046
```

However, the glm model misses interaction terms, which can be captured by a tree based model effectively.

2. Random Forest Since the trees in random forest minimize the sum of squares, it is better to use the transformed variable whose distribution is closer to normal distribution.

```r
# dim(logtrainHouse)
trCtrl <- trainControl(method = "cv",number=10)

# Try multiple mtry
rf1Grid <-  expand.grid(mtry = c(20,25,30,60))

# append the cv results to the below data frame
modelResults <- data.frame(mtry=0,RMSE=0,Rsquared=0,RMSESD=0,RsquaredSD=0)

# Random forest iterations with different number of trees and mtry values
for (i in seq(100,400,by = 50)){
  rf1 <- train(log(logSalePrice)~.,
      data = logtrainHouse,
      method = "rf",
```

```r
      trControl = trCtrl,
      tuneGrid = rf1Grid,
      ntrees = i,
      verbose = F)
  modelResults <- rbind(modelResults,rf1$results)
}

modelResults
```

```r
modelResults[which.min( (modelResults$RMSE)[-1])+1,]
```

```
##   parameter       RMSE  Rsquared      RMSESD RsquaredSD
## 4      none 0.01301645 0.8583387 0.004266357 0.08610211
```

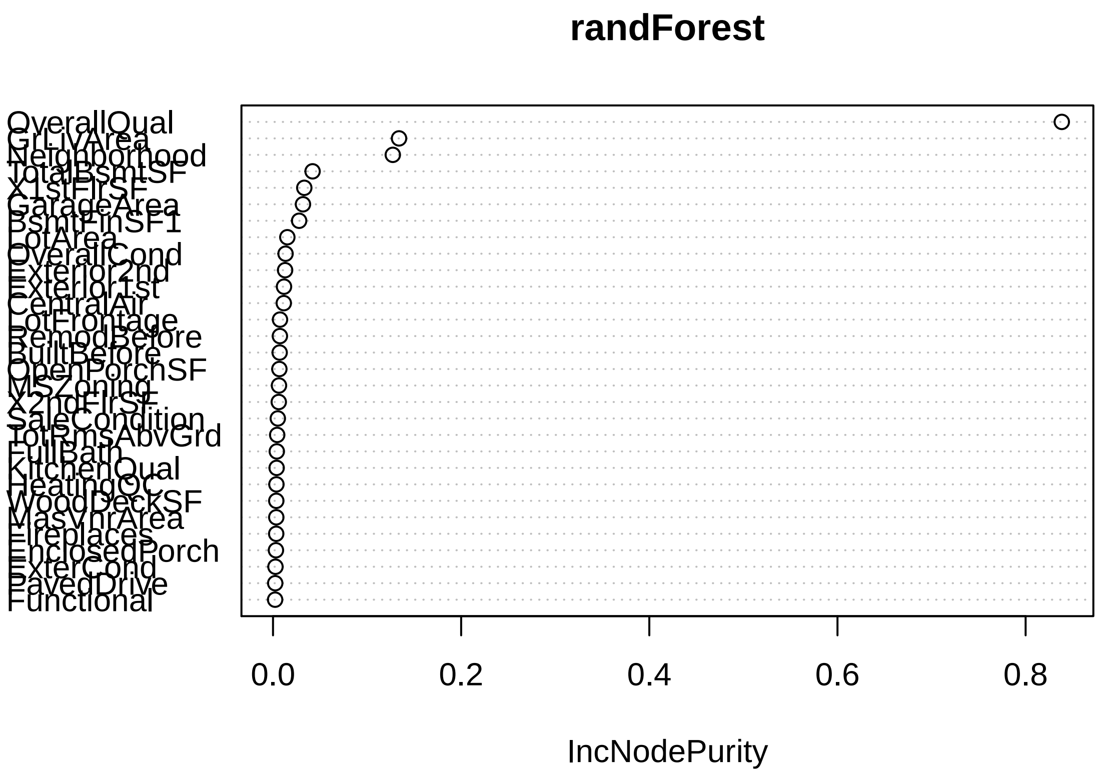
The best test RMSE has occured with mtry = 60 (all the variables) and number of trees = 400. But it can be noted that number of trees made very little difference in reducing test RMSE, but across the iterations, mtry = 60 performed better.

Fit the final random forest with the best parameters.

```r
set.seed(0)
randForest <- randomForest(log(logSalePrice)~.,data = logtrainHouse, mtry = 58, ntree= 400)


(ForestRMSE = (sum((randForest$y - randForest$predicted)^2)/nrow(logtrainHouse))^0.5)
```

```
## [1] 0.01255222
```

```r
varImpPlot(randForest)
```



Random forest and glm resulted in almost similar RMSE.

According to the random forest, Overall Quality, Size of Living Area and Nieghborhood are the most important predictors.