STAT 1361 - Final Project

Gordon Lu

4/16/2021

R Markdown

```r
#### Libraries ####
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.6.3
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```r
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.6.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.3
```

```r
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```r
# library(regclass) #for VIF
require("car")
```

```
## Loading required package: car

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

## Loading required package: carData

## Warning: package 'carData' was built under R version 3.6.3
```

```r
library(olsrr) # to get partial correlations and VIF
```

```
## Warning: package 'olsrr' was built under R version 3.6.3

##
## Attaching package: 'olsrr'

## The following object is masked from 'package:MASS':
##
##     cement

## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
library(glmnet) # For lasso and ridge regression
```

```
## Warning: package 'glmnet' was built under R version 3.6.3

## Loading required package: Matrix

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

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-1
```

```r
library(randomForest) # For Random Forests
```

```
## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

```r
library(tree) # For trees
```

```
## Warning: package 'tree' was built under R version 3.6.3
```

```r
library(gbm) # For gradient boosted trees
```

```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## Loaded gbm 2.1.8
```

```r
library(analogue) # For PCR
```

```
## Warning: package 'analogue' was built under R version 3.6.3
```

```
## Loading required package: vegan
```

```
## Warning: package 'vegan' was built under R version 3.6.3
```

```
## Loading required package: permute
```

```
## Warning: package 'permute' was built under R version 3.6.3
```

```
## This is vegan 2.5-7
```

```
##
## Attaching package: 'vegan'
```

```
## The following object is masked from 'package:caret':
##
##     tolerance
```

```
## Registered S3 methods overwritten by 'analogue':
##   method    from
##   plot.roc  pROC
##   print.roc pROC
```

```
## analogue version 0.17-5
```

```r
library(pls) # For PLS
```

```
## Warning: package 'pls' was built under R version 3.6.3
```

```
##
## Attaching package: 'pls'
```

```
## The following objects are masked from 'package:analogue':
##
##     crossval, pcr, RMSEP
```

```
## The following object is masked from 'package:vegan':
##
##     scores
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
library(gam) # For GAM
```

```
## Warning: package 'gam' was built under R version 3.6.3
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.6.3
```

```
## Loaded gam 1.16.1
```

```r
set.seed(100)
# Column Indices to Variable Mappings:
####################################
        # 1  -> id
        # 2  -> price
        # 3  -> desc
        # 4  -> numstories
        # 5  -> yearbuilt
        # 6  -> exteriorfinish
        # 7  -> rooftype
        # 8  -> basement
        # 9  -> totalrooms
        # 10 -> bedrooms
        # 11 -> bathrooms
        # 12 -> fireplaces
        # 13 -> sqft
        # 14 -> lotarea
        # 15 -> state
        # 16 -> zipcode
        # 17 -> AvgIncome
####################################
df.train <- read.table("train-1.csv", sep=",", header=T)
# ID doesn't help with regression, it's just to identify the house.
# AvgIncome and State are enough to cover what ZipCode is trying to...
remove_cols <- c(1,16) #try with zipcode: 16
newdf = df.train[, -(remove_cols)]
# After removing ID and Zipcode...
```

```r
####################################
        # 1  -> price
        # 2  -> desc
        # 3  -> numstories
        # 4  -> yearbuilt
        # 5  -> exteriorfinish
        # 6  -> rooftype
        # 7  -> basement
        # 8  -> totalrooms
        # 9  -> bedrooms
        # 10 -> bathrooms
        # 11 -> fireplaces
        # 12 -> sqft
        # 13 -> lotarea
        # 14 -> state
        # 15 -> AvgIncome
####################################
# Try a 60/40 Train-Test Split
train_size <- .60*nrow(newdf)
test_size <- .40*nrow(newdf)
# Shuffle around train and test...
train <- sample(1:nrow(newdf), train_size)
trainX <- newdf[train, -1]
trainY <- newdf[train, 1]
trainSet <- data.frame(trainY, trainX)
testX <- newdf[-train, -1]
testY <- newdf[-train, 1]
testSet <- data.frame(testY, testX)
```

```r
# Rename the response to be Price for both the training set and test set
colnames(trainSet)[1] <- "price"
colnames(testSet)[1] <- "price"

# Fireplaces has several NAs... either get rid of fire places all together, or replace it with the mean
fire_mean <- mean(trainSet$fireplaces, na.rm = TRUE)
trainSet$fireplaces <- replace_na(trainSet$fireplaces, fire_mean)
fire_mean <- mean(testSet$fireplaces, na.rm = TRUE)
testSet$fireplaces <- replace_na(testSet$fireplaces, fire_mean)

#Baseline is mean difference between train and test price....
mean((trainSet$price - testSet$price)^2)
```

```
## Warning in trainSet$price - testSet$price: longer object length is not a
## multiple of shorter object length
```

```
## [1] 198916733473
```

```r
# Predict using a Multiple Linear Regression model
mod <- lm(price ~., data = trainSet)
pred.lm <- predict(mod, testSet)
mse <- mean((pred.lm-testY)^2)
mse
```

```
## [1] 13399903739
```

```r
set.seed(1)
# Stepwise regression model
step.model <- stepAIC(mod, direction = "both",
                      trace = FALSE)
step.model$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## price ~ desc + numstories + yearbuilt + exteriorfinish + rooftype +
##     basement + totalrooms + bedrooms + bathrooms + fireplaces +
##     sqft + lotarea + state + AvgIncome
##
## Final Model:
## price ~ desc + numstories + yearbuilt + exteriorfinish + rooftype +
##     basement + bedrooms + bathrooms + fireplaces + sqft + lotarea +
##     state + AvgIncome
##
##
##           Step Df    Deviance Resid. Df   Resid. Dev      AIC
## 1                                    816 1.721611e+13 19992.51
## 2 - totalrooms  1 10595652850       817 1.722671e+13 19991.03
```

```r
# Look at VIFs and Partial Correlations
ols_coll_diag(step.model)
```

```
## Tolerance and Variance Inflation Factor
## ---------------------------------------
##                 Variables Tolerance      VIF
## 1        descMOBILE HOME 0.3889138 2.571264
## 2       descMULTI-FAMILY 0.4881520 2.048542
## 3          descROWHOUSE 0.7751328 1.290101
## 4      descSINGLE FAMILY 0.4076945 2.452817
## 5             numstories 0.5957254 1.678626
## 6              yearbuilt 0.5202260 1.922241
## 7  exteriorfinishConcrete 0.9518300 1.050608
## 8     exteriorfinishFrame 0.7679800 1.302117
## 9       exteriorfinishLog 0.8927988 1.120073
## 10    exteriorfinishStone 0.7869838 1.270674
## 11   exteriorfinishStucco 0.8931801 1.119595
## 12          rooftypeROLL 0.6600335 1.515075
## 13       rooftypeSHINGLE 0.2343396 4.267310
## 14         rooftypeSLATE 0.4440873 2.251809
## 15              basement 0.3118322 3.206853
## 16              bedrooms 0.4163402 2.401882
## 17             bathrooms 0.2635066 3.794971
## 18            fireplaces 0.7210970 1.386776
## 19                  sqft 0.2502466 3.996059
## 20               lotarea 0.3609273 2.770641
## 21               stateVA 0.2132536 4.689252
```

```
## 22              AvgIncome 0.4928508 2.029012
##
##
## Eigenvalue and Condition Index
## ------------------------------
##       Eigenvalue Condition Index    intercept descMOBILE HOME descMULTI-FAMILY
## 1  1.036368e+01        1.000000 1.336057e-06    1.036899e-05     1.395424e-04
## 2  1.838825e+00        2.374033 2.849285e-08    5.036784e-02     2.170469e-03
## 3  1.601789e+00        2.543632 1.181939e-07    5.785820e-02     3.014437e-03
## 4  1.247112e+00        2.882732 5.082580e-07    3.683633e-03     2.080953e-02
## 5  1.133434e+00        3.023841 1.364981e-07    1.151837e-02     1.677418e-01
## 6  1.068364e+00        3.114565 4.789729e-08    6.477791e-05     7.274470e-03
## 7  1.001877e+00        3.216250 2.429176e-09    1.745455e-05     5.643452e-04
## 8  9.051781e-01        3.383685 1.906752e-09    7.728782e-03     1.420721e-01
## 9  8.422255e-01        3.507864 3.565244e-08    2.714816e-03     3.797414e-02
## 10 7.674216e-01        3.674853 1.640063e-07    6.455474e-04     6.501010e-02
## 11 5.503982e-01        4.339288 5.463770e-08    3.434414e-03     5.613118e-04
## 12 4.522001e-01        4.787311 5.660602e-06    7.824423e-03     6.243265e-03
## 13 3.786209e-01        5.231843 8.313319e-06    3.130410e-02     1.381986e-03
## 14 2.490078e-01        6.451349 4.508489e-06    3.633479e-02     3.385482e-02
## 15 2.042566e-01        7.123098 2.311525e-06    7.073872e-01     9.448689e-03
## 16 1.335080e-01        8.810558 8.258042e-07    1.992647e-02     4.192981e-05
## 17 7.966732e-02       11.405567 4.144391e-07    1.400903e-04     1.684225e-02
## 18 6.093034e-02       13.041879 1.926289e-05    1.616083e-03     3.094579e-03
## 19 3.859573e-02       16.386544 3.835480e-10    2.306997e-02     7.398925e-03
## 20 3.663847e-02       16.818541 3.749874e-05    1.937153e-02     1.346027e-01
## 21 2.723037e-02       19.508786 1.994343e-04    9.414851e-03     1.788031e-01
## 22 1.896780e-02       23.374830 2.358245e-03    4.839628e-03     1.185953e-01
## 23 7.385153e-05      374.608162 9.973611e-01    7.266753e-04     4.236018e-02
##    descROWHOUSE descSINGLE FAMILY   numstories    yearbuilt
## 1  4.567750e-05      2.940714e-04 3.946568e-04 1.310140e-06
## 2  1.601994e-03      7.989819e-05 6.589586e-05 1.744609e-08
## 3  9.630928e-03      8.438894e-07 7.833653e-06 1.269109e-07
## 4  4.083552e-02      5.152050e-05 2.255160e-04 4.704102e-07
## 5  4.409558e-03      5.382621e-04 3.495645e-06 1.385904e-07
## 6  1.638303e-01      1.821942e-04 9.839170e-05 4.121411e-08
## 7  2.772010e-01      5.416448e-05 4.505241e-05 6.234538e-09
## 8  8.917571e-02      2.410526e-04 2.164024e-05 3.717847e-09
## 9  1.515141e-01      1.494117e-04 2.999226e-04 1.968366e-08
## 10 2.623819e-02      2.951954e-04 1.372249e-05 1.588924e-07
## 11 1.399673e-04      1.869112e-08 7.974998e-05 3.908494e-08
## 12 3.828195e-03      9.749912e-04 2.234917e-04 5.108322e-06
## 13 8.311688e-03      2.166251e-03 2.498066e-04 8.255895e-06
## 14 1.232378e-04      8.870443e-04 9.797684e-04 4.054660e-06
## 15 2.612294e-04      1.168435e-03 9.058370e-04 2.408069e-06
## 16 9.606203e-05      4.778620e-03 2.643986e-02 1.043216e-07
## 17 1.399955e-02      3.355636e-05 8.195341e-02 3.435375e-08
## 18 3.120008e-05      2.013243e-02 1.584099e-01 1.656557e-05
## 19 4.234926e-03      1.183363e-02 8.762413e-04 4.239996e-08
## 20 1.015334e-01      3.232087e-01 3.901646e-01 4.117428e-05
## 21 3.526823e-02      3.504092e-01 3.000274e-03 1.760467e-04
## 22 3.947912e-02      2.543822e-01 3.203227e-01 2.274230e-03
## 23 2.821017e-02      2.813827e-02 1.521822e-02 9.974696e-01
##    exteriorfinishConcrete exteriorfinishFrame exteriorfinishLog
```

```
## 1            2.096909e-05        1.474256e-03      3.390364e-05
## 2            7.538664e-07        1.840822e-02      1.162812e-02
## 3            5.663730e-04        3.157883e-02      2.118252e-03
## 4            3.713421e-02        3.349099e-03      3.560415e-03
## 5            1.312798e-04        1.817057e-02      3.181531e-01
## 6            1.603783e-01        9.563747e-03      2.584312e-02
## 7            5.814136e-01        2.000974e-05      3.600515e-04
## 8            5.071633e-02        3.619022e-03      4.024399e-01
## 9            7.190507e-02        2.520414e-03      2.815012e-02
## 10           3.978969e-02        8.078948e-03      5.646622e-02
## 11           1.047446e-02        5.213435e-01      3.338392e-02
## 12           2.943588e-04        3.448954e-03      2.404865e-03
## 13           4.162737e-03        2.969494e-01      4.338392e-03
## 14           2.223353e-03        1.370217e-03      3.314637e-04
## 15           3.506662e-05        9.434603e-04      9.314041e-02
## 16           8.958122e-03        1.994490e-03      5.714829e-05
## 17           2.354324e-05        1.741310e-03      8.112087e-03
## 18           1.361398e-03        5.340218e-02      5.079556e-04
## 19           1.750926e-03        2.692634e-04      5.081824e-03
## 20           6.125554e-03        1.790807e-02      2.255193e-03
## 21           1.200281e-04        3.614216e-03      7.867674e-04
## 22           2.023693e-02        1.564027e-05      6.242573e-04
## 23           2.176924e-03        2.161304e-04      2.224980e-04
##    exteriorfinishStone exteriorfinishStucco rooftypeROLL rooftypeSHINGLE
## 1         4.052259e-04         5.629894e-04 1.753676e-04    0.0005244259
## 2         9.022088e-05         2.698370e-02 1.404284e-03    0.0049814831
## 3         5.056650e-03         2.849059e-02 2.572717e-05    0.0075010348
## 4         1.662690e-01         5.225392e-02 1.607893e-01    0.0000343581
## 5         3.140366e-02         5.620998e-03 3.009088e-03    0.0009887602
## 6         2.073190e-01         9.488756e-03 9.136327e-02    0.0008543732
## 7         2.524240e-03         5.037632e-03 1.307313e-03    0.0003438385
## 8         1.643157e-02         2.304660e-02 4.759869e-02    0.0012639262
## 9         7.386054e-02         2.241276e-01 1.585896e-01    0.0014467138
## 10        1.048165e-02         5.369536e-01 1.491769e-01    0.0048141497
## 11        1.035872e-01         2.041505e-02 3.654632e-03    0.0062969463
## 12        2.928968e-01         1.121649e-02 2.286608e-02    0.0095749990
## 13        7.230306e-04         8.408419e-03 1.264843e-02    0.0091100543
## 14        6.081258e-03         5.549774e-03 4.369930e-03    0.0050325673
## 15        7.735551e-04         1.324045e-03 2.684813e-03    0.0097627656
## 16        1.208519e-03         8.469113e-03 4.776088e-02    0.2862678774
## 17        1.121828e-02         7.155570e-03 2.314397e-02    0.1320150868
## 18        4.284402e-03         1.846106e-02 4.444789e-02    0.1357306589
## 19        6.003585e-05         7.550583e-06 4.062841e-04    0.0045873521
## 20        1.412707e-02         1.124422e-03 1.132589e-03    0.0297335634
## 21        1.710020e-02         1.932391e-03 4.360145e-03    0.0083854542
## 22        3.196530e-02         3.298747e-03 1.810586e-01    0.2286829881
## 23        2.132632e-03         7.092378e-05 3.802621e-02    0.1120666232
##    rooftypeSLATE     basement     bedrooms    bathrooms   fireplaces
## 1   9.409826e-04 7.622740e-04 3.310392e-04 4.925620e-04 1.979419e-03
## 2   1.620696e-02 3.483675e-03 4.810835e-05 5.403779e-05 8.203006e-04
## 3   2.150394e-02 4.350902e-03 3.368588e-06 7.986665e-05 2.221745e-03
## 4   6.119724e-03 2.030439e-03 1.691598e-05 1.531122e-04 7.130289e-03
## 5   1.328594e-03 5.569686e-05 1.599335e-05 2.967861e-04 7.465624e-04
## 6   4.909280e-04 7.253503e-04 3.760843e-05 4.688260e-05 2.555952e-03
```

```
## 7   1.213714e-03 3.827400e-05 1.995915e-07 2.317415e-05 4.415818e-05
## 8   7.510268e-05 1.044534e-05 1.068728e-04 1.811228e-04 1.112931e-03
## 9   2.524154e-02 8.316400e-04 1.322960e-04 3.354337e-04 1.392860e-03
## 10  1.122495e-03 1.751303e-03 1.051976e-06 9.152880e-05 3.915392e-04
## 11  8.907127e-02 1.282665e-03 2.742566e-04 1.821549e-03 7.185513e-02
## 12  1.724580e-01 3.443296e-02 9.722398e-05 1.795844e-03 1.048888e-02
## 13  6.088598e-02 3.763809e-02 9.916675e-07 4.009550e-03 3.014272e-01
## 14  4.987771e-02 3.457117e-03 3.202989e-03 4.669649e-02 5.480328e-01
## 15  3.290478e-02 2.217826e-03 1.364411e-03 1.501231e-02 1.617669e-02
## 16  2.531864e-01 3.245148e-01 2.116548e-05 3.855131e-03 5.510781e-04
## 17  1.008913e-01 3.738108e-04 1.190908e-02 4.565984e-02 5.052674e-03
## 18  6.870777e-02 3.661253e-01 8.849468e-03 1.354842e-01 3.665352e-03
## 19  7.155954e-05 8.714392e-02 2.944508e-01 2.817253e-01 2.393907e-03
## 20  5.942753e-03 6.307663e-03 1.319679e-01 1.398717e-01 1.336059e-02
## 21  5.268963e-04 1.148980e-01 5.236400e-01 2.831697e-01 1.916709e-05
## 22  5.356016e-02 4.142175e-03 2.344397e-02 2.843847e-02 1.438004e-03
## 23  3.767135e-02 3.425624e-03 8.431688e-05 1.070545e-02 7.142746e-03
##            sqft      lotarea      stateVA    AvgIncome
## 1  5.374996e-04 2.054667e-04 4.396579e-04 5.327671e-04
## 2  1.994525e-05 4.834773e-02 6.026386e-03 1.604024e-04
## 3  1.310039e-04 4.733053e-02 4.199087e-03 1.771949e-04
## 4  2.757249e-04 3.850315e-04 3.439474e-03 1.142839e-04
## 5  4.034848e-04 5.020880e-05 5.240925e-04 1.483050e-05
## 6  1.611645e-04 7.635217e-05 3.085487e-03 6.401737e-06
## 7  1.005787e-05 9.782632e-06 1.926377e-05 2.340294e-05
## 8  5.867560e-05 1.588433e-04 8.429481e-05 1.100083e-04
## 9  8.011610e-05 8.874935e-06 5.203513e-03 2.389505e-04
## 10 1.144232e-05 1.063834e-04 4.650866e-03 2.018680e-08
## 11 2.688977e-03 4.022211e-03 4.770759e-04 1.845393e-05
## 12 4.093524e-03 2.687250e-03 6.676376e-02 5.120555e-04
## 13 1.179099e-02 2.059283e-02 2.180484e-04 2.996473e-03
## 14 6.419175e-02 2.830915e-02 5.605378e-04 6.933438e-04
## 15 6.655780e-03 7.536596e-01 3.533985e-03 1.093500e-03
## 16 1.200565e-02 3.269988e-02 1.726546e-02 2.041680e-02
## 17 3.062396e-02 9.463079e-04 1.781165e-02 4.721368e-01
## 18 9.169029e-03 3.579877e-04 3.682097e-01 1.133499e-01
## 19 6.793856e-01 4.285738e-02 1.801382e-01 4.023759e-02
## 20 1.373320e-01 1.292301e-02 7.874114e-04 1.513539e-02
## 21 1.208373e-03 7.072878e-04 1.896326e-01 1.461496e-01
## 22 3.601615e-02 1.148390e-05 1.263223e-01 8.605457e-02
## 23 3.149137e-03 3.546366e-03 6.071340e-04 9.982719e-02
```

```r
ols_correlations(step.model)
```

```
##                        Correlations
## ----------------------------------------------------------
## Variable                  Zero Order    Partial     Part
## ----------------------------------------------------------
## descMOBILE HOME               -0.003     -0.076    -0.032
## descMULTI-FAMILY              -0.034     -0.115    -0.049
## descROWHOUSE                  -0.067      0.031     0.013
## descSINGLE FAMILY              0.104     -0.030    -0.013
## numstories                     0.312     -0.081    -0.035
## yearbuilt                      0.045     -0.084    -0.036
```

```
## exteriorfinishConcrete         0.002      0.008     0.003
## exteriorfinishFrame           -0.190     -0.019    -0.008
## exteriorfinishLog              0.017     -0.037    -0.016
## exteriorfinishStone            0.112     -0.106    -0.045
## exteriorfinishStucco          -0.024     -0.153    -0.066
## rooftypeROLL                  -0.063      0.026     0.011
## rooftypeSHINGLE               -0.347     -0.053    -0.023
## rooftypeSLATE                  0.368      0.134     0.057
## basement                      -0.080      0.067     0.028
## bedrooms                       0.554     -0.104    -0.044
## bathrooms                      0.793      0.374     0.171
## fireplaces                     0.321     -0.177    -0.076
## sqft                           0.814      0.596     0.314
## lotarea                        0.155      0.114     0.049
## stateVA                        0.222      0.306     0.136
## AvgIncome                      0.154      0.119     0.051
## ---------------------------------------------------------
```

```r
car::vif(step.model)
```

```
##                    GVIF Df GVIF^(1/(2*Df))
## desc           4.164719  4        1.195221
## numstories     1.678626  1        1.295618
## yearbuilt      1.922241  1        1.386449
## exteriorfinish 2.008384  5        1.072222
## rooftype       3.215085  3        1.214876
## basement       3.206853  1        1.790769
## bedrooms       2.401882  1        1.549801
## bathrooms      3.794971  1        1.948069
## fireplaces     1.386776  1        1.177614
## sqft           3.996059  1        1.999014
## lotarea        2.770641  1        1.664524
## state          4.689252  1        2.165468
## AvgIncome      2.029012  1        1.424434
```

```r
# Stepwise aside, if we look at the correlations of the desc variable, yearbuilt, exteriorfinish and ba
# Predict with stepwise regression model
pred.steplm <- predict(step.model, testSet)
step.mse <- mean((pred.steplm-testY)^2)
# As we can see from the anova table, Stepwise regression tells us that totalrooms is the only detected
step.mse # The MSE isn't much better...
```

```
## [1] 13404266546
```

```r
# Backward regression model
backward.model <- stepAIC(mod, direction = "backward",
                      trace = FALSE)
backward.model$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
```

```
## Initial Model:
## price ~ desc + numstories + yearbuilt + exteriorfinish + rooftype +
##     basement + totalrooms + bedrooms + bathrooms + fireplaces +
##     sqft + lotarea + state + AvgIncome
##
## Final Model:
## price ~ desc + numstories + yearbuilt + exteriorfinish + rooftype +
##     basement + bedrooms + bathrooms + fireplaces + sqft + lotarea +
##     state + AvgIncome
##
##
##           Step Df    Deviance Resid. Df   Resid. Dev      AIC
## 1                                  816 1.721611e+13 19992.51
## 2 - totalrooms  1 10595652850       817 1.722671e+13 19991.03
```

```r
# Not much change
pred.bckwdlm <- predict(backward.model, testSet)
backward.mse <- mean((pred.bckwdlm-testY)^2)
backward.mse
```

```
## [1] 13404266546
```

```r
# Backward tells us that we also want to get rid of totalrooms!!
```

```r
# Backward regression model
forward.model <- stepAIC(mod, direction = "forward",
                         trace = FALSE)
forward.model$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## price ~ desc + numstories + yearbuilt + exteriorfinish + rooftype +
##     basement + totalrooms + bedrooms + bathrooms + fireplaces +
##     sqft + lotarea + state + AvgIncome
##
## Final Model:
## price ~ desc + numstories + yearbuilt + exteriorfinish + rooftype +
##     basement + totalrooms + bedrooms + bathrooms + fireplaces +
##     sqft + lotarea + state + AvgIncome
##
##
##   Step Df Deviance Resid. Df   Resid. Dev      AIC
## 1                        816 1.721611e+13 19992.51
```

```r
# Not much change
forward.mse <- mean((predict(forward.model)-testY)^2)
```

```
## Warning in predict(forward.model) - testY: longer object length is not a
## multiple of shorter object length
```

```r
forward.mse # Forward says to keep everything!
```

```
## [1] 174596213074
```

```r
# Now, let's try lasso and ridge regression!
train.mat = model.matrix(price ~ ., data = trainSet)
test.mat = model.matrix(price ~ ., data = testSet)
```

```r
# Try Ridge regression first:
grid = 10^seq(4, -2, length=100)
fit.ridge = glmnet(train.mat, trainSet$price, alpha = 0, lambda = grid, thresh = 1e-12)
cv.ridge = cv.glmnet(train.mat, trainSet$price, alpha = 0, lambda = grid, thresh = 1e-12)
bestlam.ridge = cv.ridge$lambda.min

pred.ridge <- predict(fit.ridge, s = bestlam.ridge, newx = test.mat)
mean((pred.ridge - testSet$price)^2)
```

```
## [1] 12822483258
```

```r
# This was a bit better than stepwise!
```

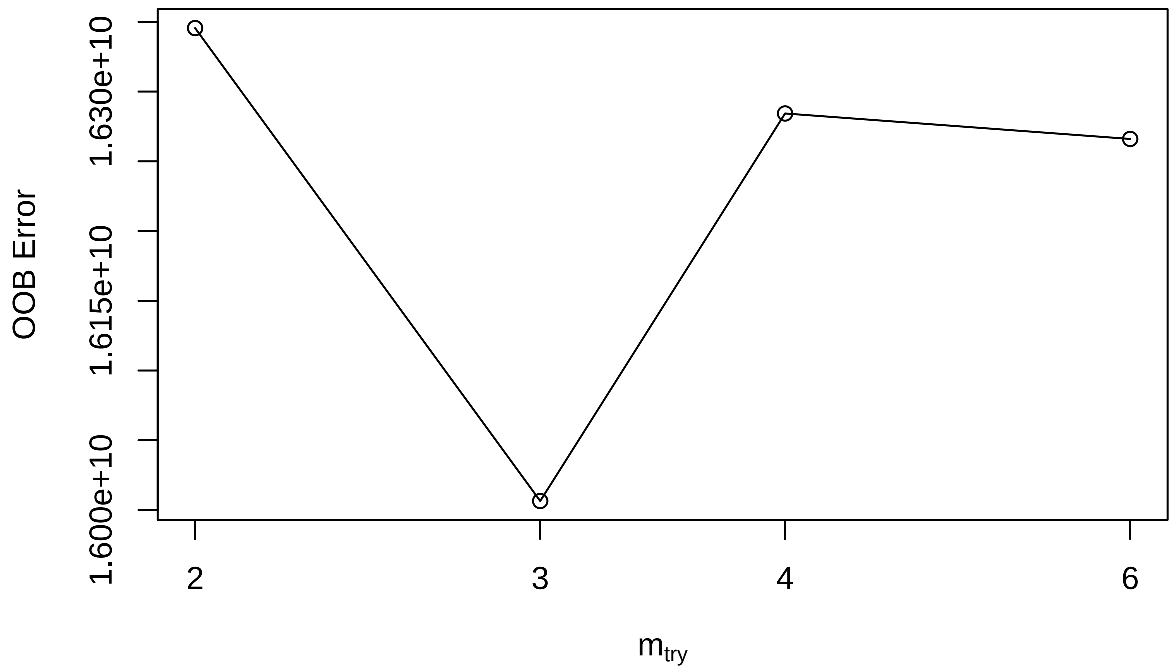
```r
# Now let's try Lasso regression:
set.seed(1)
fit.lasso = glmnet(train.mat, trainSet$price, alpha = 1, lambda = grid, thresh = 1e-12)
cv.lasso = cv.glmnet(train.mat, trainSet$price, alpha = 1, lambda = grid, thresh = 1e-12)
bestlam.lasso = cv.lasso$lambda.min
pred.lasso = predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
lasso.mse <- mean((pred.lasso - testSet$price)^2)
pred.lasso <- predict(fit.lasso, s = bestlam.lasso, type = "coefficients")

pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
mean((pred.lasso - testSet$price)^2)
```

```
## [1] 11301061819
```

```r
# This was a bit better than ridge!
```

```r
# Now, let's try Random Forests:
set.seed(1)
rf <-randomForest(price~.,data=trainSet, ntree=500)
mtry <- tuneRF(trainSet[,-1],trainSet$price, ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 4  OOB error = 16284287614
## Searching left ...
## mtry = 3     OOB error = 16006453500
## 0.01706148 0.01
## mtry = 2     OOB error = 16345506636
## -0.02118228 0.01
## Searching right ...
## mtry = 6     OOB error = 16266032224
## -0.01621713 0.01
```




```r
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]

bag.price <-randomForest(price~.,data=trainSet, mtry=best.m, importance=TRUE,ntree=500,proximity=TRUE)
yhat.bag <- predict(bag.price, newdata = testSet)
mean((yhat.bag - testSet$price)^2)
```

```
## [1] 8047775013
```

```r
#This is the best thus far!
# # lots of improvement here!!
importance(bag.price)
```

```
##                  %IncMSE IncNodePurity
## desc            5.018202  2.929155e+11
## numstories      4.068617  2.025330e+12
## yearbuilt       9.526390  2.662106e+12
## exteriorfinish  7.806638  1.262046e+12
## rooftype       19.875569  4.842395e+12
## basement        8.381222  8.783871e+11
## totalrooms     10.522858  1.162692e+13
## bedrooms        6.456804  8.051154e+12
## bathrooms      22.225454  2.029640e+13
## fireplaces     12.017086  2.425487e+12
## sqft           27.930543  2.498816e+13
## lotarea        13.731483  8.299823e+12
```

```
## state          20.280200  2.503993e+12
## AvgIncome      14.142734  3.441822e+12
```

```r
varImp(bag.price)
```

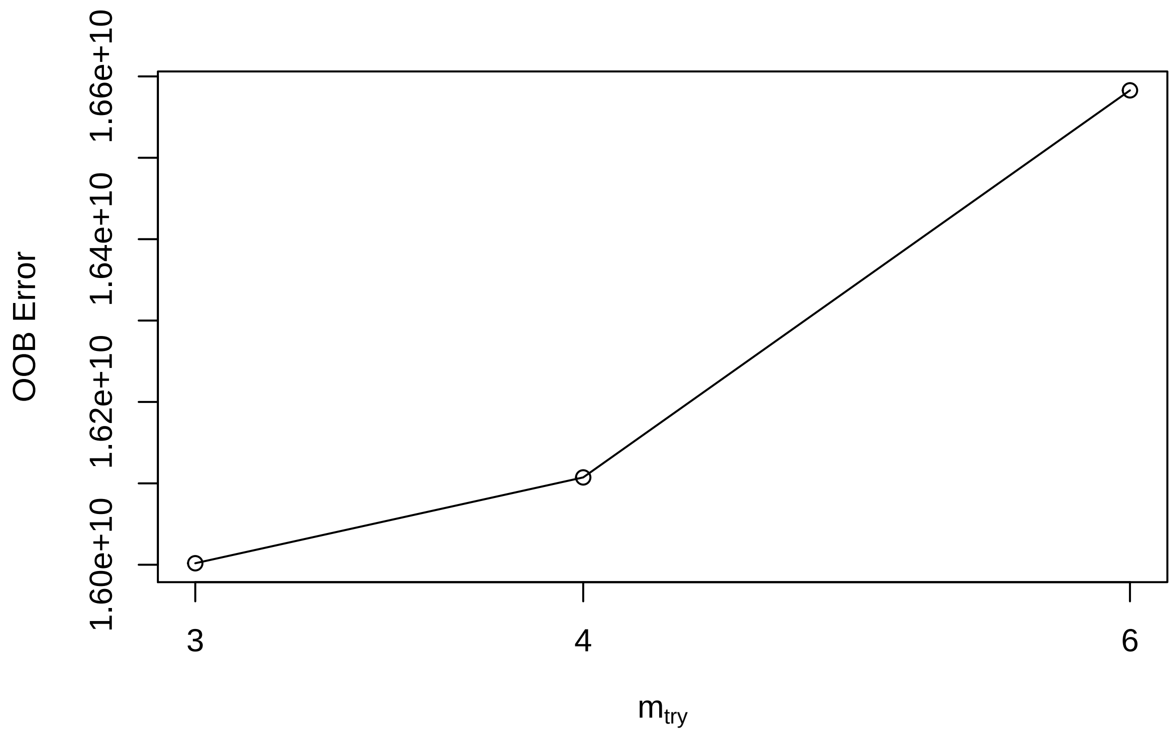
```
##                  Overall
## desc            5.018202
## numstories      4.068617
## yearbuilt       9.526390
## exteriorfinish  7.806638
## rooftype       19.875569
## basement        8.381222
## totalrooms     10.522858
## bedrooms        6.456804
## bathrooms      22.225454
## fireplaces     12.017086
## sqft           27.930543
## lotarea        13.731483
## state          20.280200
## AvgIncome      14.142734
```

```r
# Sort the variable importance...
imp.all <- as.data.frame(sort(importance(bag.price)[,1],decreasing = TRUE),optional = T)
names(imp.all) <- "% Inc MSE"
imp.all
```

```
##                % Inc MSE
## sqft           27.930543
## bathrooms      22.225454
## state          20.280200
## rooftype       19.875569
## AvgIncome      14.142734
## lotarea        13.731483
## fireplaces     12.017086
## totalrooms     10.522858
## yearbuilt       9.526390
## basement        8.381222
## exteriorfinish  7.806638
## bedrooms        6.456804
## desc            5.018202
## numstories      4.068617
```

```r
rf <-randomForest(price~.,data=trainSet, ntree=500)
mtry <- tuneRF(trainSet[,-1],trainSet$price, ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 4  OOB error = 16107305419
## Searching left ...
## mtry = 3     OOB error = 16001842291
## 0.006547534 0.01
## Searching right ...
## mtry = 6     OOB error = 16582840486
## -0.02952294 0.01
```




```r
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]

bag.price <-randomForest(price~.,data=trainSet, mtry=(ncol(trainSet)-1), importance=TRUE,ntree=500)
yhat.bag <- predict(bag.price, newdata = testSet)
mean((yhat.bag - testSet$price)^2)
```

```
## [1] 8273064878
```

```r
# # lots of improvement here!!
# Sort the variable importance...
imp.all <- as.data.frame(sort(importance(bag.price)[,1],decreasing = TRUE),optional = T)
names(imp.all) <- "% Inc MSE"
imp.all
```

```
##                 % Inc MSE
## sqft            41.440442
## state           26.503675
## rooftype        25.082366
## exteriorfinish 19.900989
## AvgIncome       18.785925
## bathrooms       18.476018
## lotarea         13.824175
## desc            10.827428
## yearbuilt        9.317115
## basement         8.834042
```

```
## fireplaces      7.566721
## numstories      3.090908
## bedrooms       -1.531089
## totalrooms     -3.908243
```

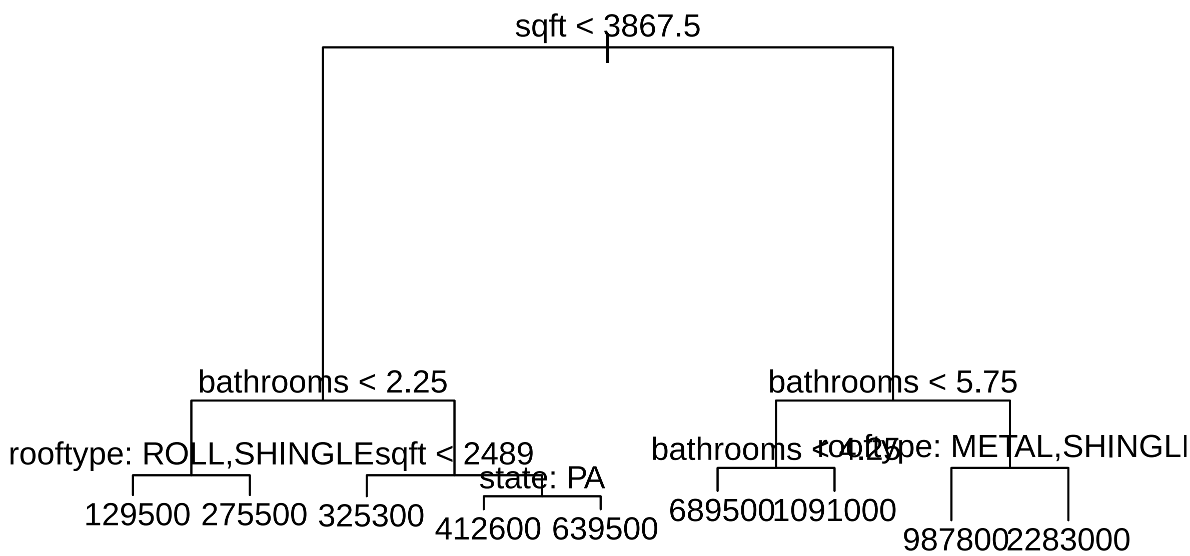
```r
varImp(bag.price)
```

```
##                  Overall
## desc           10.827428
## numstories      3.090908
## yearbuilt       9.317115
## exteriorfinish 19.900989
## rooftype       25.082366
## basement        8.834042
## totalrooms     -3.908243
## bedrooms       -1.531089
## bathrooms      18.476018
## fireplaces      7.566721
## sqft           41.440442
## lotarea        13.824175
## state          26.503675
## AvgIncome      18.785925
```

```r
# Now try a tree based approach:
tree.price <- tree(price ~ ., data = trainSet)
summary(tree.price)
```

```
##
## Regression tree:
## tree(formula = price ~ ., data = trainSet)
## Variables actually used in tree construction:
## [1] "sqft"      "bathrooms" "rooftype"  "state"
## Number of terminal nodes:  9
## Residual mean deviance:  2.565e+10 = 2.132e+13 / 831
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1252000   -71670   -10760        0    65190  1708000
```

```r
plot(tree.price)
text(tree.price, pretty = 0)
```




```r
yhat <- predict(tree.price, data = testSet)
mean((yhat - testSet$price)^2)
```

```
## Warning in yhat - testSet$price: longer object length is not a multiple of
## shorter object length
```

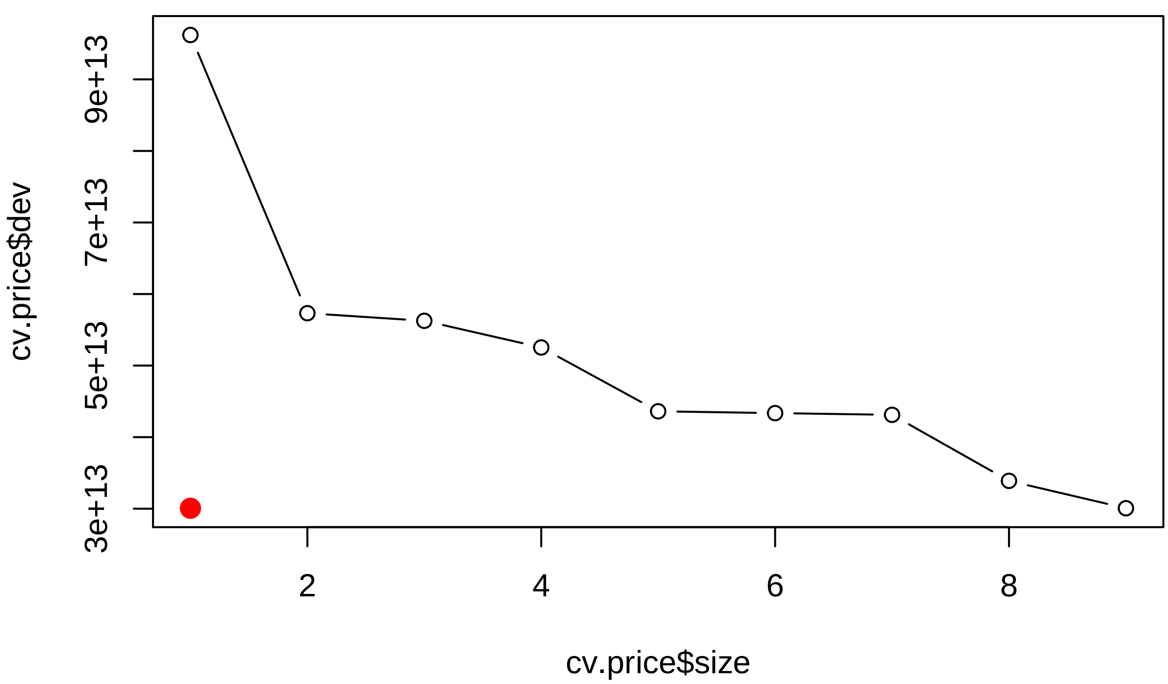
```
## [1] 171475158117
```

```r
# Now try a pruned tree:
cv.price <- cv.tree(tree.price)
cv.price
```

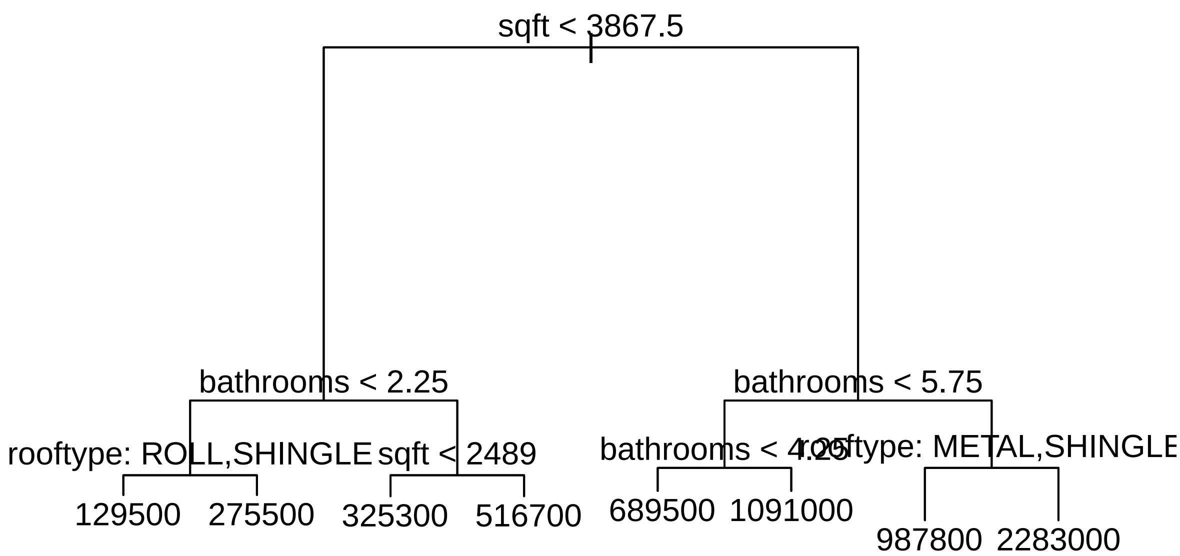
```
## $size
## [1] 9 8 7 6 5 4 3 2 1
##
## $dev
## [1] 3.006418e+13 3.389127e+13 4.311833e+13 4.335818e+13 4.361139e+13
## [6] 5.253330e+13 5.625827e+13 5.732133e+13 9.620283e+13
##
## $k
## [1]          -Inf 1.559278e+12 2.373678e+12 2.514685e+12 2.755800e+12
## [6] 6.264637e+12 8.056231e+12 8.938202e+12 4.225026e+13
##
## $method
```

```
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
plot(cv.price$size, cv.price$dev, type = "b")
tree.min <- which.min(cv.price$dev)
points(tree.min, cv.price$dev[tree.min], col = "red", cex = 2, pch = 20)
```




```r
prune.price <- prune.tree(tree.price, best = 8)
plot(prune.price)
text(prune.price, pretty = 0)
```




```
yhat <- predict(prune.price, data = testSet)
mean((yhat - testSet$price)^2)
```

```
## Warning in yhat - testSet$price: longer object length is not a multiple of
## shorter object length

## [1] 168708295040
```

```
gam.fit <- gam(price ~ ., data= trainSet)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
res <- predict(gam.fit, testSet)
mean((res-testSet$price))^2
```

```
## [1] 40324401
```

```
set.seed(1)
pows <- seq(-10, -0.2, by = 0.1)
lambdas <- 10^pows
train.err <- rep(NA, length(lambdas))
for (i in 1:length(lambdas)) {
```

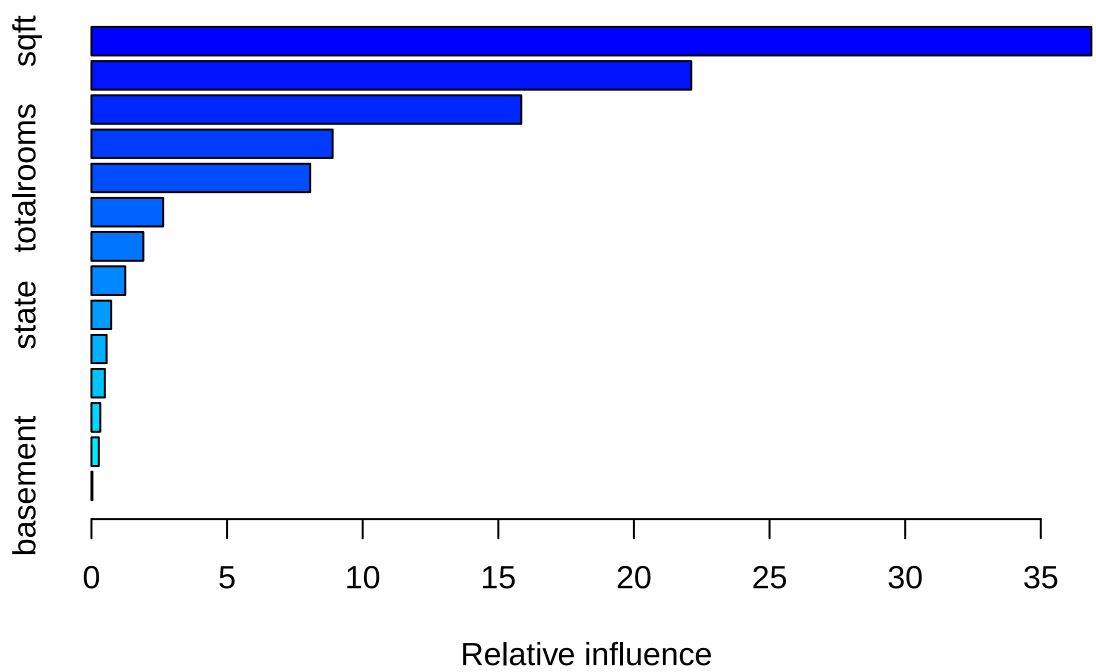
```r
    boost.price <- gbm(price ~ ., data = trainSet, distribution = "gaussian", n.trees = 1000, shrinkage
    pred.train <- predict(boost.price, trainSet, n.trees = 1000)
    train.err[i] <- mean((pred.train - trainSet$price)^2)
}
boost.price <- gbm(price ~ ., data = trainSet, distribution = "gaussian", n.trees = 1000, shrinkage = la
mean((predict(boost.price,data=testSet)-testSet$price)^2)
```

```
## Using 1000 trees...
```

```
## Warning in predict(boost.price, data = testSet) - testSet$price: longer object
## length is not a multiple of shorter object length
```

```
## [1] 184701844359
```

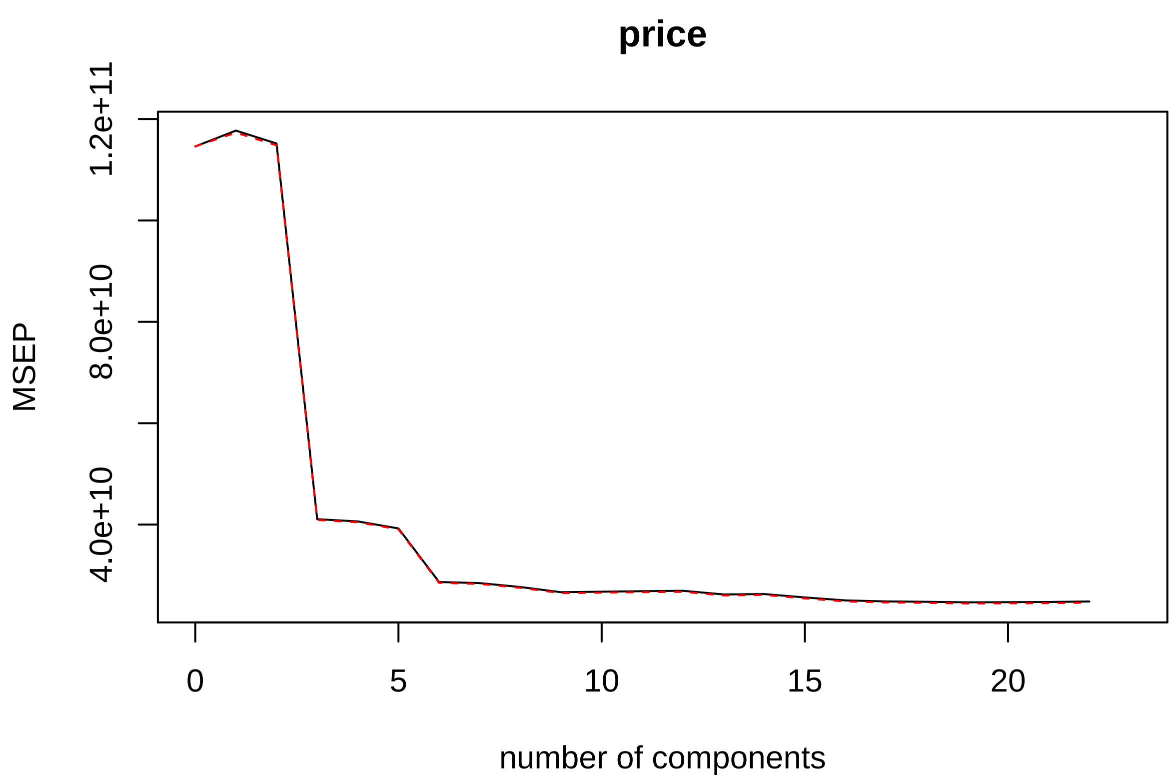
```r
summary(boost.price)
```




```
##                                var     rel.inf
## sqft                          sqft 36.86406174
## bathrooms                bathrooms 22.11344018
## lotarea                    lotarea 15.84634201
## bedrooms                  bedrooms  8.89252415
## totalrooms              totalrooms  8.06339895
## numstories              numstories  2.64387589
```

```
## rooftype               rooftype  1.91422300
## AvgIncome             AvgIncome  1.24706756
## state                     state  0.72666796
## fireplaces           fireplaces  0.55947896
## exteriorfinish exteriorfinish  0.49486852
## yearbuilt             yearbuilt  0.32614425
## desc                       desc  0.27217301
## basement               basement  0.03573381
```

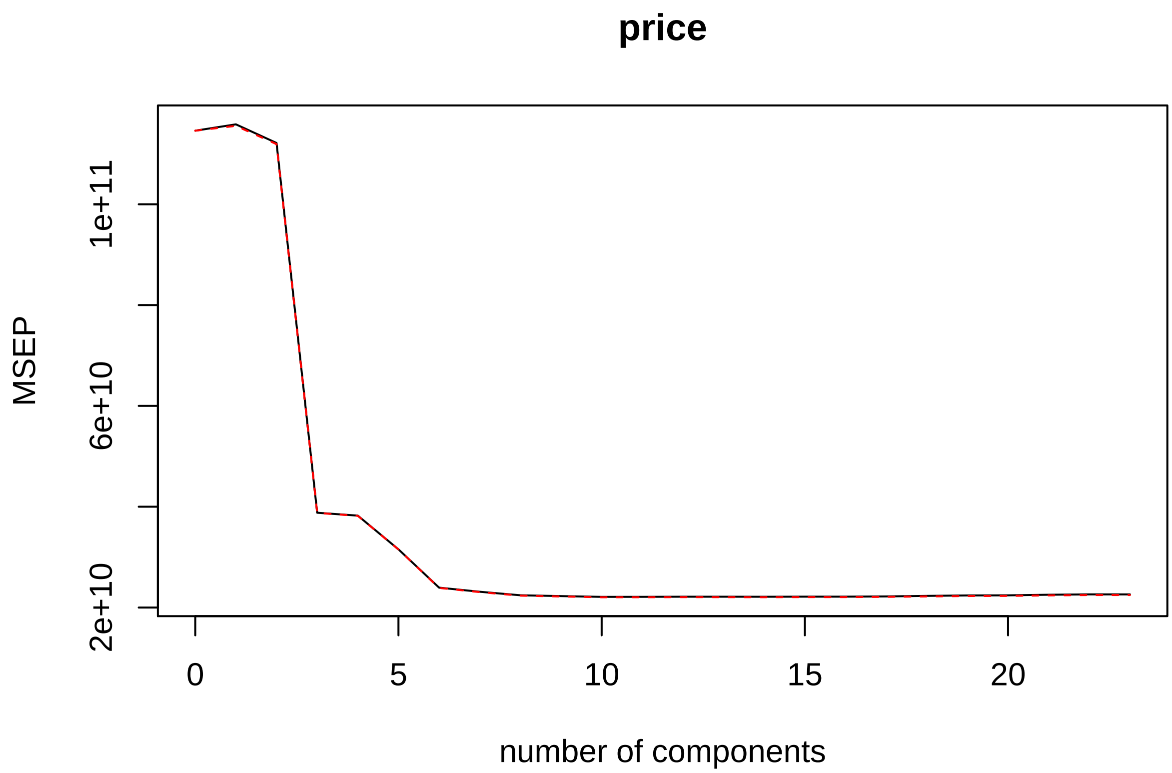
```r
fit.pcr = pcr(price ~ ., data = trainSet, scale = FALSE, validation = "CV")
validationplot(fit.pcr, val.type = "MSEP")
```




```r
pred.pcr <- predict(fit.pcr, testSet, ncomp = ncol(trainSet))
pcr.mse <- mean((pred.pcr - testSet$price)^2)
cat(pcr.mse)
```

```
## 12484792499
```

```r
fit.pls = plsr(price ~ ., data = trainSet, scale = FALSE, validation = "CV")
validationplot(fit.pls, val.type = "MSEP")
```




```
pred.pls = predict(fit.pls, testSet, ncomp = ncol(trainSet))
pls.mse <- mean((pred.pls - testSet$price)^2)
pls.mse
```

```
## [1] 12574252434
```

```
library(olsrr)
set.seed(1)
####################################
        # 1  -> price
        # 2  -> desc
        # 3  -> numstories
        # 4  -> yearbuilt
        # 5  -> exteriorfinish
        # 6  -> rooftype
        # 7  -> basement
        # 8  -> totalrooms
        # 9  -> bedrooms
        # 10 -> bathrooms
        # 11 -> fireplaces
        # 12 -> sqft
        # 13 -> lotarea
        # 14 -> state
        # 15 -> AvgIncome
####################################
```

```r
#desc variable, yearbuilt, exteriorfinish and basement variable, they're all pretty low. Numstories, de
# So let's see how Stepwise fails when we get rid of those variable along with total rooms
elim_cols <- c(2,3,4,5,7,8,11)
trainSet <- trainSet[,-elim_cols]
# Let's run a bunch of models first, then apply the changes!
# Fit the full model
mod <- lm(price ~., data = trainSet)
# Stepwise regression model
step.model <- stepAIC(mod, direction = "both",
                      trace = FALSE)
step.model$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## price ~ rooftype + bedrooms + bathrooms + sqft + lotarea + state +
##     AvgIncome
##
## Final Model:
## price ~ rooftype + bedrooms + bathrooms + sqft + lotarea + state +
##     AvgIncome
##
##
##   Step Df Deviance Resid. Df   Resid. Dev      AIC
## 1                        830 1.904557e+13 20049.34
```

```r
# MSE improves!!
pred.steplm <- predict(step.model, testSet)

step.mse <- mean((pred.steplm-testY)^2)
step.mse
```

```
## [1] 12153231879
```

```r
elim_cols <- c(2,3,4,5,7,8,11)
testSet <- testSet[,-elim_cols]
```

```r
# Now, let's try lasso and ridge regression!
train.mat = model.matrix(price ~ ., data = trainSet)
test.mat = model.matrix(price ~ ., data = testSet)
```

```r
# Try Ridge regression first:
grid = 10^seq(4, -2, length=100)
fit.ridge = glmnet(train.mat, trainSet$price, alpha = 0, lambda = grid, thresh = 1e-12)
cv.ridge = cv.glmnet(train.mat, trainSet$price, alpha = 0, lambda = grid, thresh = 1e-12)
bestlam.ridge = cv.ridge$lambda.min

pred.ridge <- predict(fit.ridge, s = bestlam.ridge, newx = test.mat)
mean((pred.ridge - testSet$price)^2)
```

```
## [1] 12153231362
```

```r
# This was a bit better than stepwise!
```

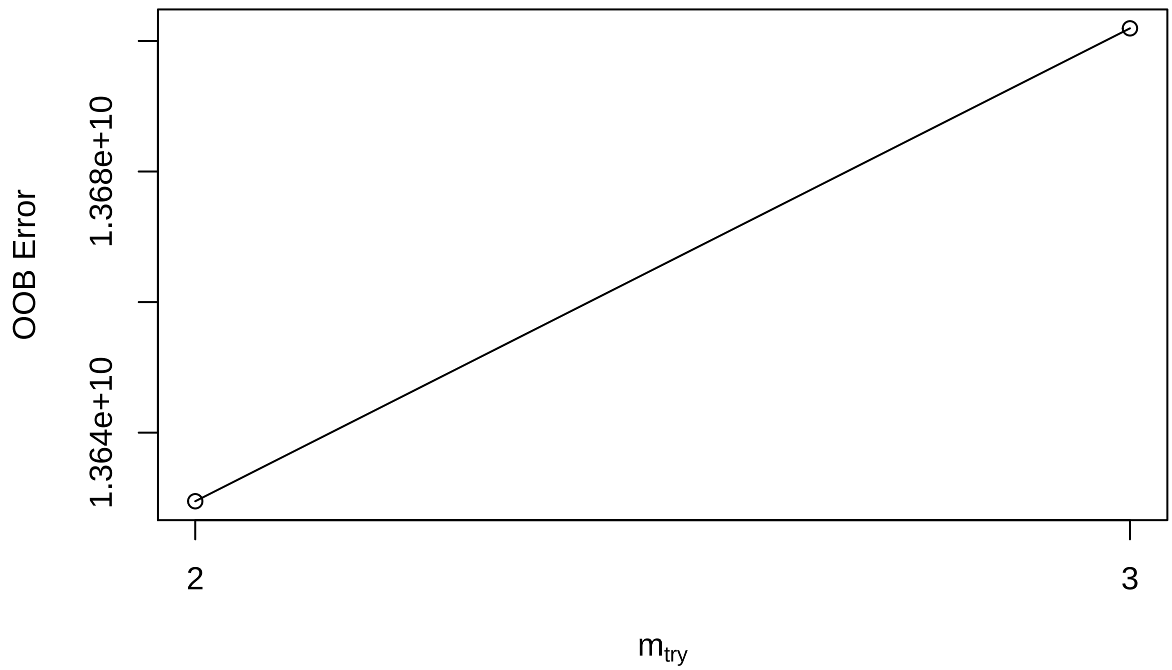
```r
# Now let's try Lasso regression:
set.seed(1)
fit.lasso = glmnet(train.mat, trainSet$price, alpha = 1, lambda = grid, thresh = 1e-12)
cv.lasso = cv.glmnet(train.mat, trainSet$price, alpha = 1, lambda = grid, thresh = 1e-12)
bestlam.lasso = cv.lasso$lambda.min
pred.lasso = predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
lasso.mse <- mean((pred.lasso - testSet$price)^2)
pred.lasso <- predict(fit.lasso, s = bestlam.lasso, type = "coefficients")

pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
mean((pred.lasso - testSet$price)^2)
```

```
## [1] 12086226260
```

```r
# This was a bit better than ridge!
```

```r
# Now, let's try Random Forests:
set.seed(1)
rf <-randomForest(price~.,data=trainSet, ntree=500)
mtry <- tuneRF(trainSet[,-1],trainSet$price, ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 2  OOB error = 13629495620
## Searching left ...
## Searching right ...
## mtry = 3     OOB error = 13701930965
## -0.005314602 0.01
```




```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]

bag.price <-randomForest(price~.,data=trainSet, mtry=best.m, importance=TRUE,ntree=500,proximity=TRUE)
yhat.bag <- predict(bag.price, newdata = testSet)
mean((yhat.bag - testSet$price)^2)
```

```
## [1] 7400264174
```

```
#This is the best thus far!
# # lots of improvement here!!
importance(bag.price)
```

```
##              %IncMSE IncNodePurity
## rooftype   23.941198  6.944983e+12
## bedrooms    7.736634  9.114482e+12
## bathrooms 24.959218  2.568260e+13
## sqft      31.604485  3.246260e+13
## lotarea   15.563270  9.912060e+12
## state     27.549769  3.979692e+12
## AvgIncome 19.163314  3.955849e+12
```

```
varImp(bag.price)
```

```
##              Overall
```

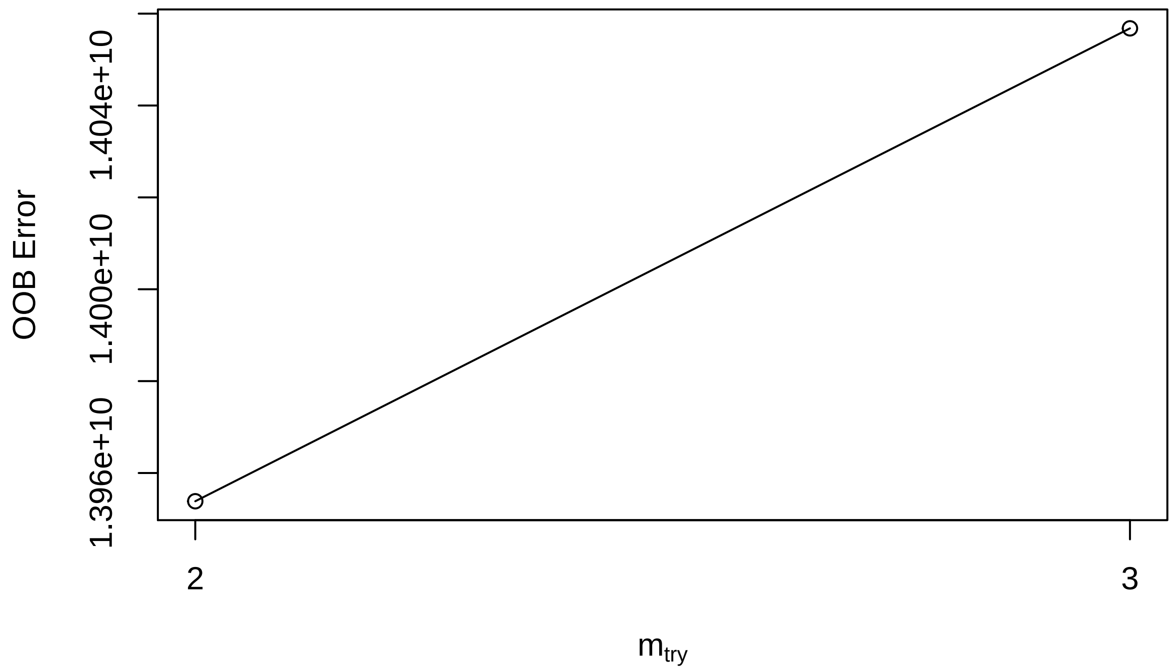
```
## rooftype  23.941198
## bedrooms   7.736634
## bathrooms 24.959218
## sqft      31.604485
## lotarea   15.563270
## state     27.549769
## AvgIncome 19.163314
```

```r
# Sort the variable importance...
imp.all <- as.data.frame(sort(importance(bag.price)[,1],decreasing = TRUE),optional = T)
names(imp.all) <- "% Inc MSE"
imp.all
```

```
##           % Inc MSE
## sqft      31.604485
## state     27.549769
## bathrooms 24.959218
## rooftype  23.941198
## AvgIncome 19.163314
## lotarea   15.563270
## bedrooms   7.736634
```

```r
rf <-randomForest(price~.,data=trainSet, ntree=500)
mtry <- tuneRF(trainSet[,-1],trainSet$price, ntreeTry=500,
               stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 2  OOB error = 13953851663
## Searching left ...
## Searching right ...
## mtry = 3     OOB error = 14056799395
## -0.007377729 0.01
```




```r
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]

bag.price <-randomForest(price~.,data=trainSet, mtry=(ncol(trainSet)-1), importance=TRUE,ntree=500)
yhat.bag <- predict(bag.price, newdata = testSet)
mean((yhat.bag - testSet$price)^2)
```

```
## [1] 8806727426
```

```r
# # lots of improvement here!!
# Sort the variable importance...
imp.all <- as.data.frame(sort(importance(bag.price)[,1],decreasing = TRUE),optional = T)
names(imp.all) <- "% Inc MSE"
imp.all
```

```
##           % Inc MSE
## sqft      46.330371
## state     31.757988
## rooftype  25.588501
## AvgIncome 24.343337
## bathrooms 22.019677
## lotarea   16.328007
## bedrooms   2.426912
```

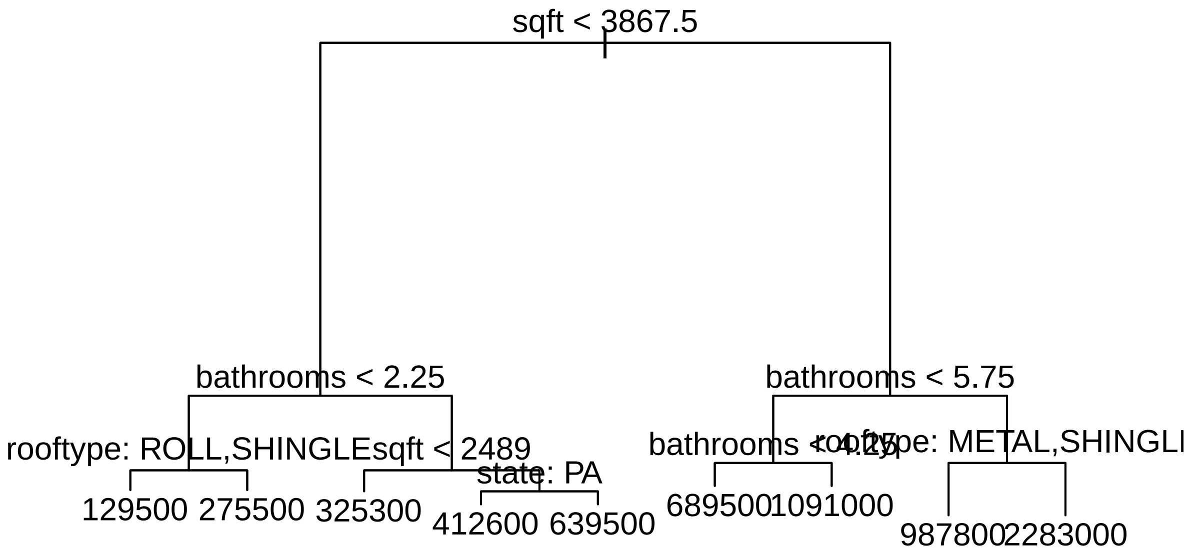
```r
varImp(bag.price)
```

```
##             Overall
## rooftype  25.588501
## bedrooms   2.426912
## bathrooms 22.019677
## sqft      46.330371
## lotarea   16.328007
## state     31.757988
## AvgIncome 24.343337
```

```r
# Now try a tree based approach:
tree.price <- tree(price ~ ., data = trainSet)
summary(tree.price)
```

```
##
## Regression tree:
## tree(formula = price ~ ., data = trainSet)
## Variables actually used in tree construction:
## [1] "sqft"      "bathrooms" "rooftype"  "state"
## Number of terminal nodes:  9
## Residual mean deviance:  2.565e+10 = 2.132e+13 / 831
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1252000   -71670   -10760        0    65190  1708000
```

```r
plot(tree.price)
text(tree.price, pretty = 0)
```

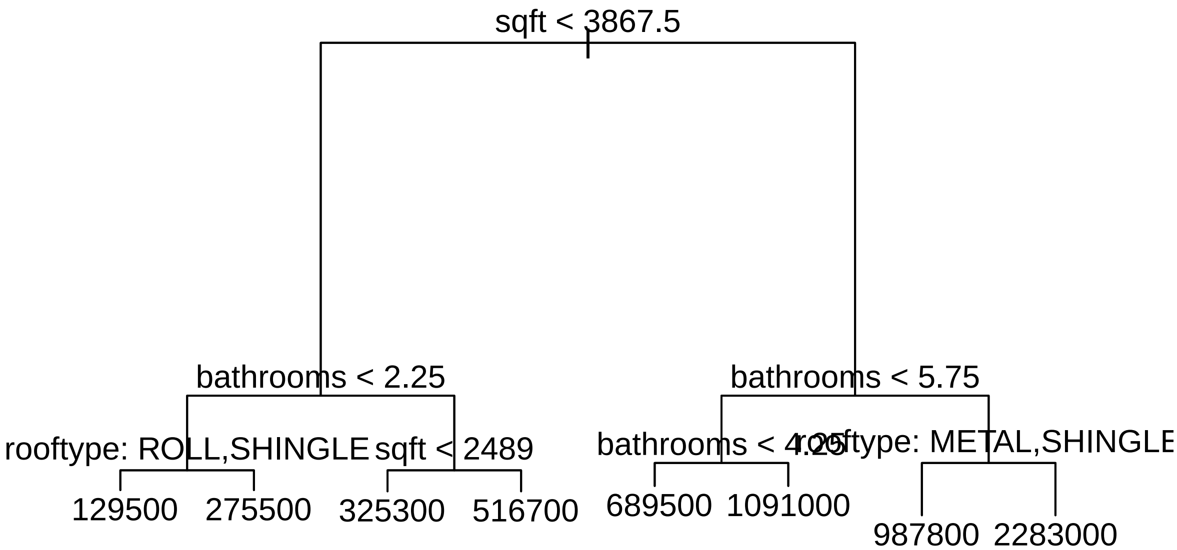



```r
yhat <- predict(tree.price, data = testSet)
mean((yhat - testSet$price)^2)
```

```
## Warning in yhat - testSet$price: longer object length is not a multiple of
## shorter object length

## [1] 171475158117
```

```r
# Now try a pruned tree:
cv.price <- cv.tree(tree.price)
plot(cv.price$size, cv.price$dev, type = "b")
tree.min <- which.min(cv.price$dev)
points(tree.min, cv.price$dev[tree.min], col = "red", cex = 2, pch = 20)
```


```
prune.price <- prune.tree(tree.price, best = 8)
plot(prune.price)
text(prune.price, pretty = 0)
```




```r
yhat <- predict(prune.price, data = testSet)
mean((yhat - testSet$price)^2)
```

```
## Warning in yhat - testSet$price: longer object length is not a multiple of
## shorter object length
```

```
## [1] 168708295040
```

```r
gam.fit <- gam(price ~ ., data= trainSet)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```r
res <- predict(gam.fit, testSet)
mean((res-testSet$price))^2
```

```
## [1] 33063531
```

```r
set.seed(1)
pows <- seq(-10, -0.2, by = 0.1)
lambdas <- 10^pows
train.err <- rep(NA, length(lambdas))
for (i in 1:length(lambdas)) {
```

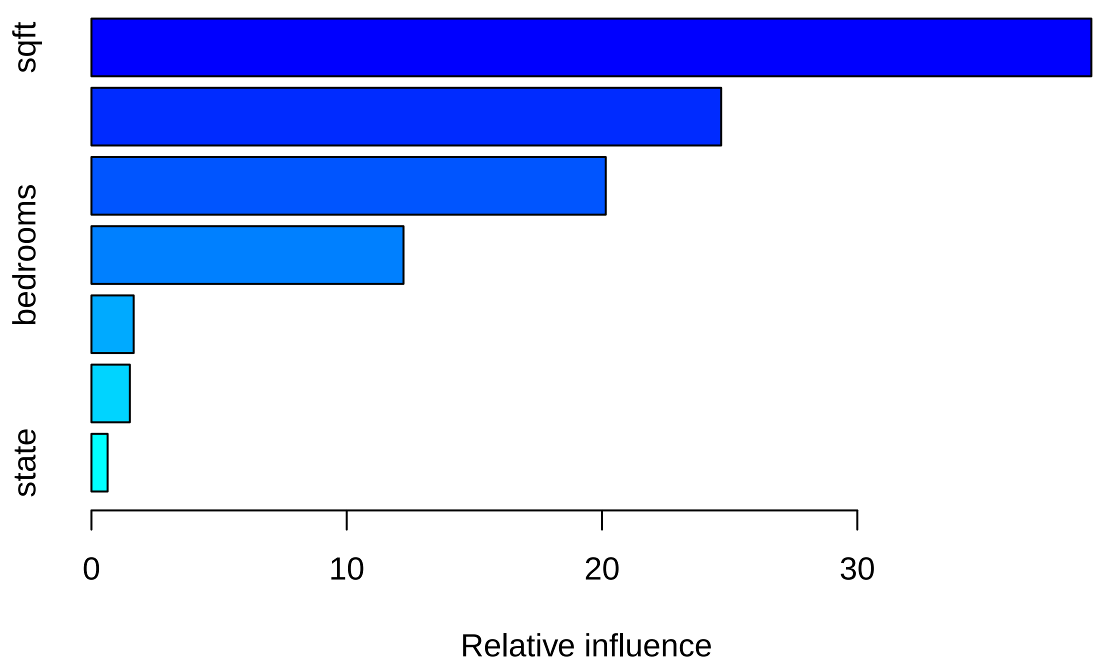
```r
    boost.price <- gbm(price ~ ., data = trainSet, distribution = "gaussian", n.trees = 1000, shrinkage
    pred.train <- predict(boost.price, trainSet, n.trees = 1000)
    train.err[i] <- mean((pred.train - trainSet$price)^2)
}
boost.price <- gbm(price ~ ., data = trainSet, distribution = "gaussian", n.trees = 1000, shrinkage = la
mean((predict(boost.price,data=testSet)-testSet$price)^2)
```

```
## Using 1000 trees...
```

```
## Warning in predict(boost.price, data = testSet) - testSet$price: longer object
## length is not a multiple of shorter object length
```

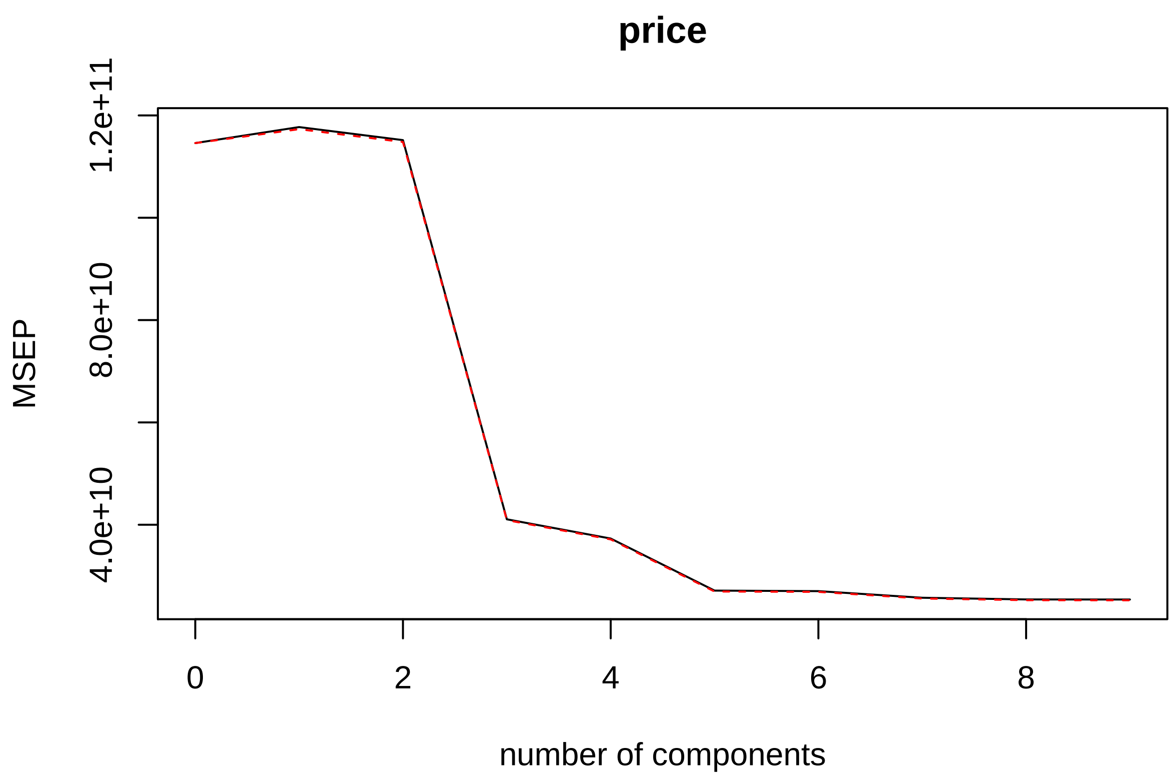
```
## [1] 181808414271
```

```r
summary(boost.price)
```




```
##                 var    rel.inf
## sqft           sqft 39.1679561
## bathrooms bathrooms 24.6688625
## lotarea     lotarea 20.1460664
## bedrooms   bedrooms 12.2243838
## AvgIncome AvgIncome  1.6543263
## rooftype   rooftype  1.5038846
## state         state  0.6345203
```

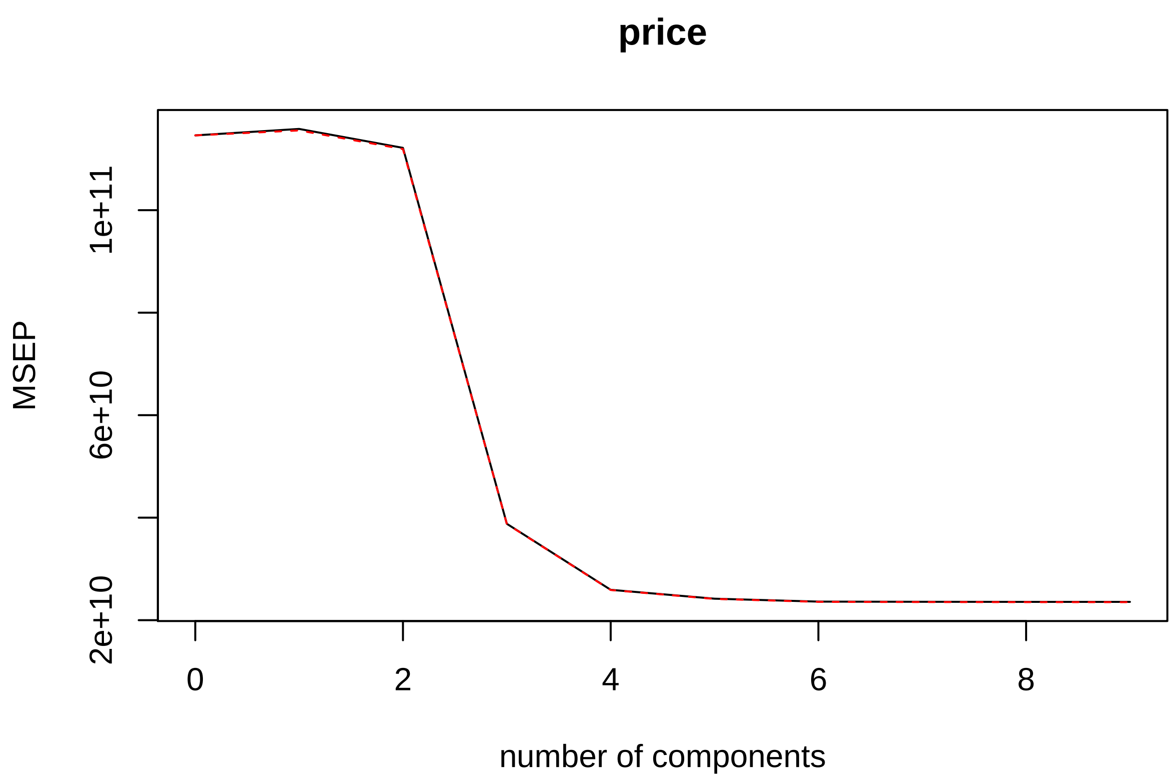
```r
fit.pcr = pcr(price ~ ., data = trainSet, scale = FALSE, validation = "CV")
validationplot(fit.pcr, val.type = "MSEP")
```




```r
pred.pcr <- predict(fit.pcr, testSet, ncomp = ncol(trainSet))
pcr.mse <- mean((pred.pcr - testSet$price)^2)
cat(pcr.mse)
```

```
## 12051286289
```

```r
fit.pls = plsr(price ~ ., data = trainSet, scale = FALSE, validation = "CV")
validationplot(fit.pls, val.type = "MSEP")
```




```
pred.pls = predict(fit.pls, testSet, ncomp = ncol(trainSet))
pls.mse <- mean((pred.pls - testSet$price)^2)
pls.mse
```

```
## [1] 12138078365
```

```
df.orig.test <- read.table("test.csv", sep=",", header=T)
#No fireplace, description, num stories, yearbuilt, exteriorfinish, zipcode, and basement
#desc variable, yearbuilt, exteriorfinish and basement variable, they're all pretty low.
# So let's see how Stepwise fails when we get rid of those variable along with total rooms
bad_cols <- c(1, 3, 4, 5, 6, 8, 12, 16)
df.test <- df.orig.test[,-bad_cols]

yhat.bag <- predict(bag.price, newdata = df.orig.test)
df.orig.test[,2] <- yhat.bag
keep_cols <- c(1,2)
df.output <- df.orig.test[,keep_cols]
df.output$student_id <- rep("4191042", nrow(df.output))
write.csv(df.output,"C:/Users/gordo/Documents/testing_predictions_4191042.csv", row.names = FALSE)
```