Linear Regression - Regularization

Suzana Iacob

23/09/2019

This analysis looks at profitability of resuturants based on restaurant characteristics.

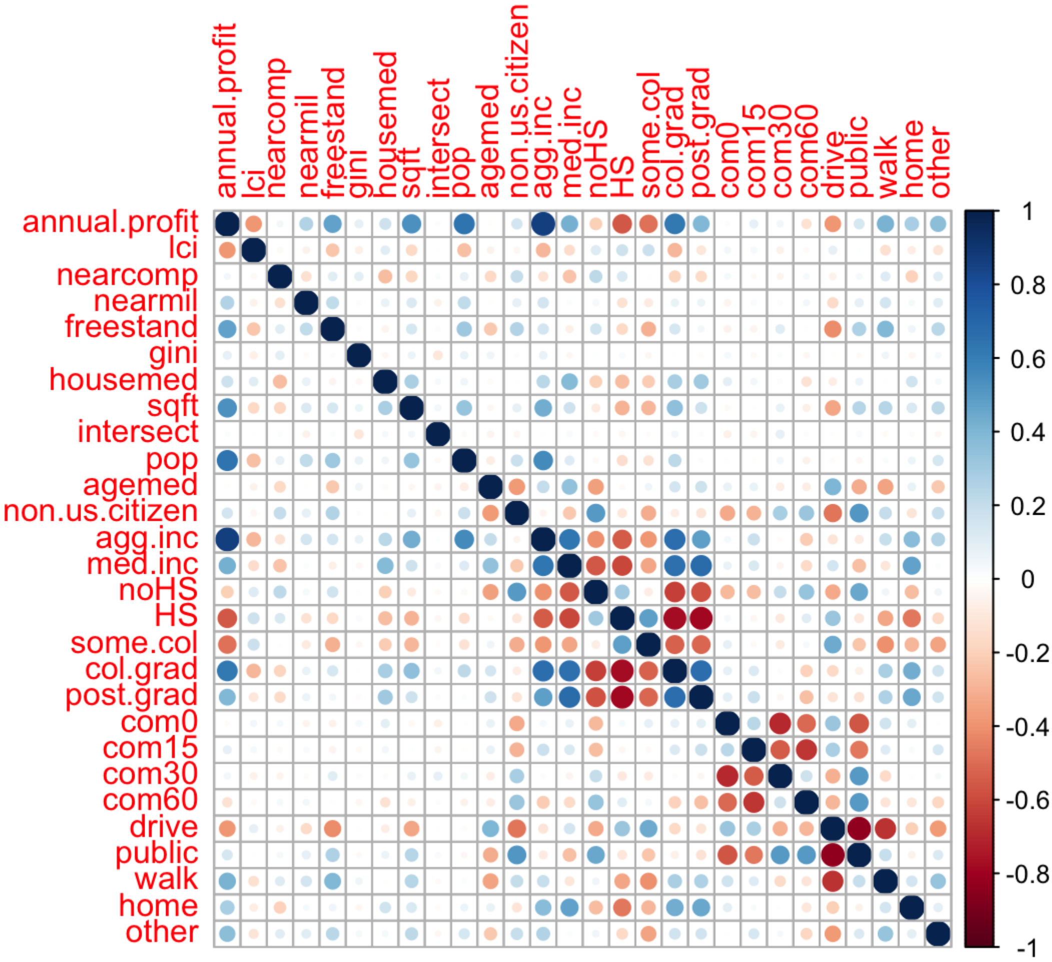
```
train = read.csv("train_data.csv")
test = read.csv("test_data.csv")
sites = rbind(train,test)
sites.const = read.csv("site_const_data.csv")
head(sites)
```

```
##   store.number annual.profit      lci nearcomp nearmil freestand   gini
## 1            1      414343.2 5.989973        2     5.3         0 0.3889
## 2            2      514644.0 8.057567        6    13.1         0 0.2434
## 3            3      443096.4 6.267259        0    30.2         0 0.3179
## 4            4      495031.1 8.566326        0    29.4         0 0.4132
## 5            5      962170.0 4.077841        6    10.1         0 0.4911
## 6            6      817722.2 7.847131        3    27.0         0 0.5250
##    housemed state sqft intersect  pop agemed non.us.citizen   agg.inc
## 1  951.0618    AZ  292         1 1951   48.5     0.13025608  42556953
## 2  778.0847    AZ  399         1 3369   29.0     0.35522451  71256942
## 3  844.5556    AZ  666         1 3301   37.5     0.06144594  97667500
## 4 1420.1387    AZ  862         1 1797   33.5     0.27073658  55558839
## 5 1164.4268    AZ  724         1 4178   39.0     0.37564235 133634294
## 6 1654.7472    AZ  678         0 1576   47.0     0.15015384  85933836
##   med.inc        noHS        HS  some.col   col.grad  post.grad       com0
## 1 51470.4 0.153041961 0.2676903 0.1715621 0.23547606 0.17222964 0.11960543
## 2 34083.2 0.203410991 0.3711291 0.3554073 0.03956185 0.03049071 0.03968254
## 3 33772.8 0.118452457 0.3338397 0.2138245 0.21576706 0.11811622 0.28871549
## 4 59000.0 0.140203538 0.2760033 0.3267813 0.13661874 0.12039312 0.09296482
## 5 38687.2 0.147936366 0.3334286 0.1460000 0.25977792 0.11285714 0.03206191
## 6 77612.8 0.004224841 0.1315093 0.1059135 0.33835239 0.42000000 0.09007634
##       com15     com30      com60     drive     public        walk
## 1 0.2688039 0.3267571 0.28483354 0.7280912 0.17346939 0.038415366
## 2 0.3022487 0.4616402 0.19642857 0.7595223 0.19399613 0.007101356
## 3 0.3931573 0.2545018 0.06362545 0.8492837 0.07106017 0.020057307
## 4 0.1557789 0.3605528 0.39070352 0.6257745 0.32713755 0.013630731
## 5 0.2194583 0.3211719 0.42730790 0.6317957 0.32646897 0.023064250
## 6 0.1893130 0.5679389 0.15267176 0.5931507 0.19726027 0.106849315
##          home      other
## 1 0.026410564 0.03361344
## 2 0.023886378 0.01549387
## 3 0.045272206 0.01432665
## 4 0.013630731 0.01982652
## 5 0.006589786 0.01208127
## 6 0.102739726 0.00000000
```

Initial analysis

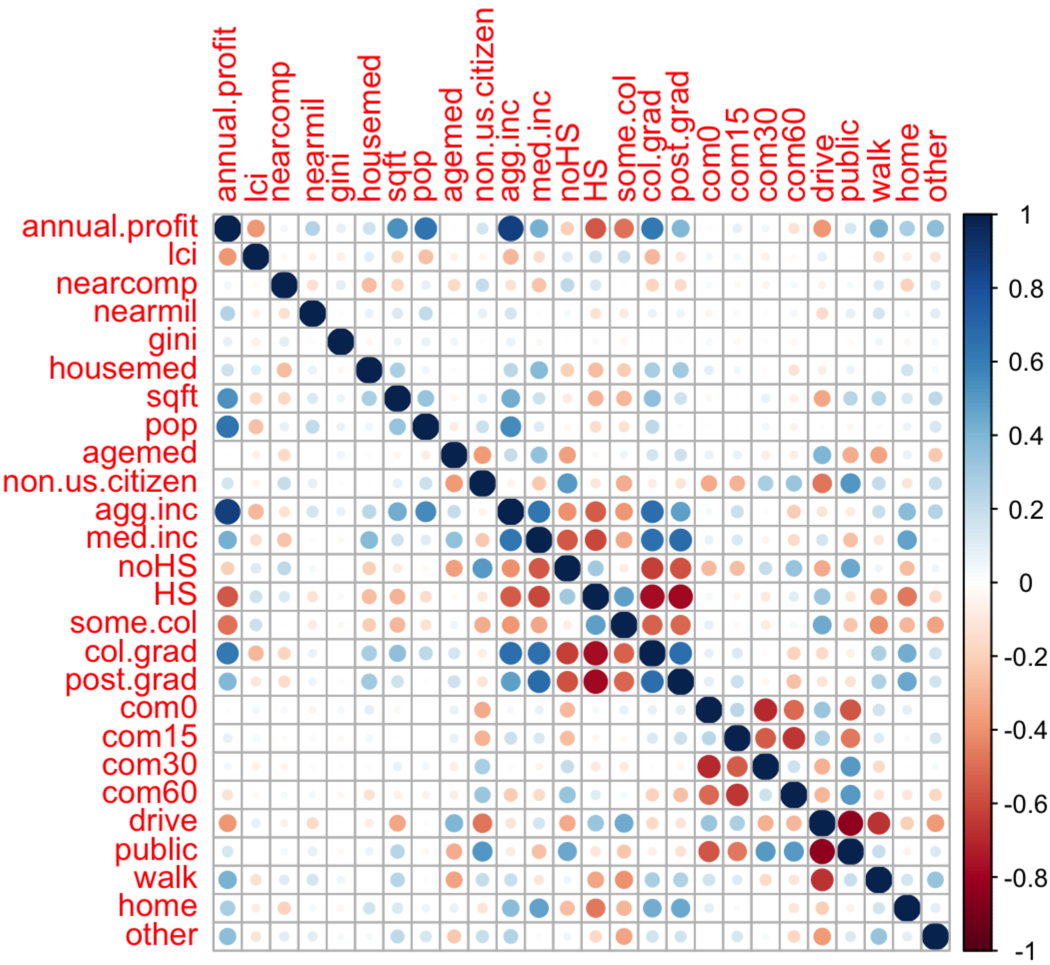
We now explore the correlation matrix of all variables. We remove store number since this is not a predictor but an index. We remove state (character variable).

```
sites_numeric = sites
sites_numeric = subset(sites_numeric, select = -c(store.number, state) )
library(corrplot)
corrplot(cor(sites_numeric), method = "circle")
```



We can further remove intersect and freestand (categorical variables).

```
sites_numeric = sites
sites_numeric = subset(sites_numeric, select = -c(store.number, state, intersect, fre
estand))
corrplot(cor(sites_numeric), method = "circle")
```



Pairs of highly correlated: - drive and public -0.8368219 - drive and walk -0.6649011 - median income and post grad 0.6714688 - pop and agg.inc 0.554165 - correlated and we should only pick one so agg.inc

Model Building

We will fit a model having all available predictors, as well as a new model. At each step **we will use the datasets as follows:**

- The **training set** will be used to train the model, inspect the coefficients and guide model selection.
- The **testing set** will be used to evaluate the model and assess its predictive performance.
- The **dataset with sites under construction** will be used to obtain a valuation for the new sites. For this we will employ the model to generate the valuation figure. Since the model was built on the training set, we will **additionally re-train using both the training and testing set**, and then compute the valuation, in order to obtain a more accurate figure.

Original Model Evaluation

```
model.original = lm(annual.profit ~ agg.inc + sqft + col.grad + com60, data=train)
summary(model.original)
```

```
##
## Call:
## lm(formula = annual.profit ~ agg.inc + sqft + col.grad + com60,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -502374 -134458  -33205  103566  917238
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.360e+04  4.108e+04   2.035  0.04258 *
## agg.inc     2.807e-03  1.384e-04  20.288  < 2e-16 ***
## sqft        3.834e+02  6.123e+01   6.261 1.06e-09 ***
## col.grad    3.468e+05  1.130e+05   3.069  0.00231 **
## com60       2.183e+05  9.821e+04   2.222  0.02687 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207800 on 369 degrees of freedom
## Multiple R-squared:  0.7861, Adjusted R-squared:  0.7838
## F-statistic: 339.1 on 4 and 369 DF,  p-value: < 2.2e-16
```

All of the variables meet the significance threshold. We note that the values correspond to the reported figures.

```
model.original.r2 = summary(model.original)$r.squared

sst.test = sum((mean(train$annual.profit) - test$annual.profit )^2)
model.original.predict = predict(model.original, newdata=test)
model.original.sse = sum((model.original.predict - test$annual.profit)^2)
model.original.osr2 = 1 - model.original.sse/sst.test

print(model.original.r2)
```

```
## [1] 0.7861431
```

```
print(model.original.osr2)
```

```
## [1] 0.7201434
```

When predicting on the 48 stores under construction we obtain the reported $40.02 million.

```
newpred.original = predict(model.original,newdata=sites.const)
value.original1 = sum(newpred.original)
print(value.original1)
```

```
## [1] 40016174
```

```
model.original = lm(annual.profit ~ agg.inc + sqft + col.grad + com60, data=sites)
newpred.original = predict(model.original,newdata=sites.const)
value.original2 = sum(newpred.original)
print(value.original2)
```

```
## [1] 40199576
```

All-Variables Model Evaluation

We refit the model with all variables only on the training set.

```
model.all = lm(annual.profit ~. - store.number, data=train)
summary(model.all)
```

```
##
## Call:
## lm(formula = annual.profit ~ . - store.number, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -350687  -56648      20   65519  470387
##
## Coefficients: (3 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.553e+05  2.942e+05   1.887 0.059958 .
## lci             -1.519e+04  4.103e+03  -3.703 0.000249 ***
## nearcomp         1.816e+04  3.025e+03   6.003 4.94e-09 ***
## nearmil          1.485e+03  4.783e+02   3.106 0.002058 **
## freestand        2.171e+05  2.080e+04  10.440  < 2e-16 ***
## gini             6.689e+03  4.333e+04   0.154 0.877419
## housemed         3.096e+01  1.794e+01   1.726 0.085212 .
## stateCA         -6.087e+03  2.413e+04  -0.252 0.800995
## stateKS          1.472e+04  3.100e+04   0.475 0.635136
## stateNM          1.995e+04  2.397e+04   0.832 0.405844
## stateNV          9.503e+03  2.454e+04   0.387 0.698822
## stateOK          9.301e+03  3.655e+04   0.254 0.799282
## stateTX          1.654e+04  2.637e+04   0.627 0.531079
## stateUT          4.299e+04  3.381e+04   1.272 0.204411
## stateWY          3.724e+04  4.851e+04   0.768 0.443113
## sqft             1.520e+02  3.486e+01   4.360 1.72e-05 ***
## intersect        6.163e+03  1.277e+04   0.483 0.629671
## pop              5.723e+01  5.738e+00   9.974  < 2e-16 ***
## agemed           2.112e+03  1.054e+03   2.004 0.045832 *
## non.us.citizen -6.539e+04  4.802e+04  -1.362 0.174189
## agg.inc          2.144e-03  1.142e-04  18.772  < 2e-16 ***
## med.inc          1.033e-01  4.873e-01   0.212 0.832216
## noHS             1.246e+05  1.096e+05   1.138 0.256067
## HS              -3.283e+04  1.136e+05  -0.289 0.772699
## some.col         1.464e+05  1.350e+05   1.084 0.279076
## col.grad         4.340e+05  1.381e+05   3.143 0.001819 **
## post.grad               NA         NA      NA       NA
## com0             1.148e+05  1.019e+05   1.126 0.260803
## com15            4.944e+04  8.393e+04   0.589 0.556231
## com30            5.865e+04  8.524e+04   0.688 0.491855
## com60                   NA         NA      NA       NA
## drive           -7.627e+05  2.580e+05  -2.956 0.003337 **
## public          -2.519e+05  2.801e+05  -0.899 0.369211
## walk            -1.240e+05  2.762e+05  -0.449 0.653713
## home            -1.390e+06  3.404e+05  -4.084 5.52e-05 ***
## other                   NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 106200 on 341 degrees of freedom
## Multiple R-squared:  0.9484, Adjusted R-squared:  0.9436
## F-statistic: 195.8 on 32 and 341 DF,  p-value: < 2.2e-16
```

```
model.all.r2 = summary(model.all)$r.squared
model.all.predict = predict(model.all, newdata=test)
model.all.sse = sum((model.all.predict - test$annual.profit)^2)
model.all.osr2 = 1 - model.all.sse/sst.test
print(model.all.r2)
```

```
## [1] 0.9483931
```

```
print(model.all.osr2)
```

```
## [1] 0.8047443
```

```
set.seed(123)
```

```
newpred.all = predict(model.all,newdata=sites.const)
value.all1 = sum(newpred.all)
print(value.all1)
```

```
## [1] 33257851
```

```
model.all = lm(annual.profit ~. - store.number, data=sites)
newpred.all = predict(model.all,newdata=sites.const)
value.all2 = sum(newpred.all)
print(value.all2)
```

```
## [1] 34051269
```

New Model

We start by adding all of the new variables into a model, in addition to the original variables.

```
model.new = lm(annual.profit ~ agg.inc + sqft + col.grad + com60 +
                      lci + nearcomp + nearmil + freestand, data=train)
summary(model.new )
```

```
##
## Call:
## lm(formula = annual.profit ~ agg.inc + sqft + col.grad + com60 +
##     lci + nearcomp + nearmil + freestand, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -457859  -70744  -11132   93100  365010
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.256e+05  5.349e+04   2.348 0.019397 *
## agg.inc      2.673e-03  8.739e-05  30.581  < 2e-16 ***
## sqft         2.994e+02  3.876e+01   7.726 1.08e-13 ***
## col.grad     3.130e+05  7.218e+04   4.337 1.87e-05 ***
## com60        1.783e+05  6.164e+04   2.893 0.004051 **
## lci         -1.763e+04  4.866e+03  -3.624 0.000332 ***
## nearcomp     3.167e+04  3.327e+03   9.519  < 2e-16 ***
## nearmil      2.648e+03  5.575e+02   4.749 2.94e-06 ***
## freestand    3.673e+05  2.049e+04  17.927  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 129600 on 365 degrees of freedom
## Multiple R-squared:  0.9177, Adjusted R-squared:  0.9159
## F-statistic: 508.7 on 8 and 365 DF,  p-value: < 2.2e-16
```

```
model.new.r2 = summary(model.new)$r.squared
model.new.predict = predict(model.new, newdata=test)
model.new.sse = sum((model.new.predict - test$annual.profit)^2)
model.new.osr2 = 1 - model.new.sse/sst.test
print(model.new.r2)
```

```
## [1] 0.9176995
```

```
print(model.new.osr2)
```

```
## [1] 0.8419712
```

```
newpred.new = predict(model.new,newdata=sites.const)
value.new1 = sum(newpred.new)
print(value.new1)
```

```
## [1] 36292492
```

```
model.new = lm(annual.profit ~ agg.inc + sqft + col.grad + com60 +
                      lci + nearcomp + nearmil + freestand, data=sites)
newpred.new = predict(model.new,newdata=sites.const)
value.new2 = sum(newpred.new)
print(value.new2)
```

```
## [1] 36612889
```

We note that the predictive accuracy has improved but the above brings us to a valuation of 36 million. After trial-and-error we observe that the below model with **lci (retail store labor cost)** and **nearcomp (compeating nearby businesses)** preserves the valuation to over 40 million.

```
model.new = lm(annual.profit ~ agg.inc + sqft + col.grad + com60 +
                      lci + nearcomp, data=train)
model.new.r2 = summary(model.new)$r.squared
model.new.predict = predict(model.new, newdata=test)
model.new.sse = sum((model.new.predict - test$annual.profit)^2)
model.new.osr2 = 1 - model.new.sse/sst.test
print(model.new.r2)
```

```
## [1] 0.8289329
```

```
print(model.new.osr2)
```

```
## [1] 0.7686991
```

```
newpred.new = predict(model.new,newdata=sites.const)
value.new1 = sum(newpred.new)
print(value.new1)
```

```
## [1] 40054717
```

```
model.new = lm(annual.profit ~ agg.inc + sqft + col.grad + com60 +
                      lci + nearcomp, data=sites)
newpred.new = predict(model.new,newdata=sites.const)
value.new2 = sum(newpred.new)
print(value.new2)
```

```
## [1] 40576369
```

```
summary <- data.frame(ModelName = c("Original Model", "All Predictors", "New Model"),
                          In_Sample_R2 = c(model.original.r2, model.all.r2, model.ne
w.r2),
                          Out_Of_Sample_R2 = c(model.original.osr2, model.all.osr2, m
odel.new.osr2),
                          Valuation_train =c(value.original1, value.all1, value.new
1),
                          Valuation_all =c(value.original2, value.all2, value.new2))
summary
```

```
##        ModelName In_Sample_R2 Out_Of_Sample_R2 Valuation_train
## 1 Original Model    0.7861431        0.7201434        40016174
## 2 All Predictors    0.9483931        0.8047443        33257851
## 3      New Model    0.8289329        0.7686991        40054717
##   Valuation_all
## 1      40199576
## 2      34051269
## 3      40576369
```

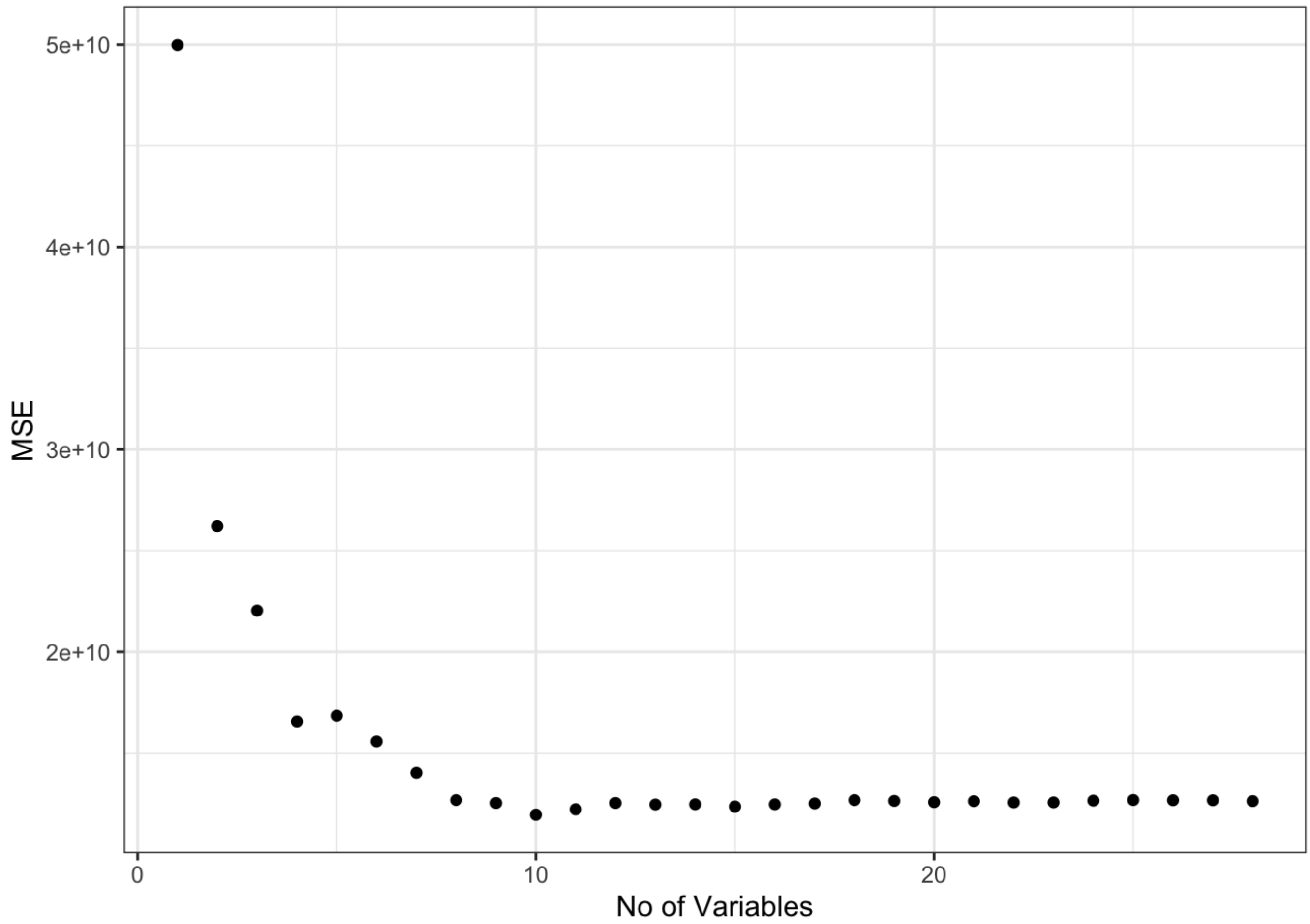
Best subset selection

Forward step selection

```
library(leaps)
n.predictors <- ncol(train)-2
forward.subset <- regsubsets(annual.profit~.- store.number,train,nvmax=n.predictors,m
ethod="forward")
```

Cross-validation

```
set.seed(144)
predict.regsubsets = function(object, newdata, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id = id)
  mat[, names(coefi)] %*% coefi
}

folds <- sample(1:10,nrow(train),replace=TRUE)

MSE.forward.subset <- matrix(NA,10,n.predictors)
for (i in 1:10){
  forward.subset <- regsubsets(annual.profit~.- store.number,train[folds!=i,],nvmax=
n.predictors,method="forward")
  for (j in 1:n.predictors){
    prediction.forward.subset <- predict.regsubsets(forward.subset,train[folds==i,],i
d=j)
    MSE.forward.subset[i,j] = sum((prediction.forward.subset - train[folds==i,]$annua
l.profit)^2) /nrow(train[folds==i,])
  }
}

MSE.average = rep(NA, n.predictors)
for (j in 1:n.predictors){
  MSE.average[j] = mean(MSE.forward.subset[, j])
}
```



Best subset model evaluation

We see that the MSE drops and then remains constant. We pick the minimum value which leads to the lowest Mean Squared Error.

```
best.no = which.min(MSE.average)
print(best.no )
```

```
## [1] 10
```

```
coef(forward.subset,best.no)
```

```
##   (Intercept)           lci      nearcomp       nearmil     freestand
##  4.981576e+05 -1.541868e+04  1.925204e+04  1.727756e+03  2.182301e+05
##          sqft           pop       agg.inc      col.grad         drive
##  1.532351e+02  4.960588e+01  2.309402e-03  4.425625e+05 -4.634572e+05
##          home
## -9.501210e+05
```

We see that the best model picked via subset selection has 10 variables. Let us fit a linear model with these coefficients.

```
model.fwd.selection = lm(annual.profit ~ lci + nearcomp + nearmil + freestand + sqft
+ pop + agg.inc  +
                           col.grad + drive + home, data=train)
model.fwd.selection.r2 = summary(model.fwd.selection)$r.squared
model.fwd.selection.predict = predict(model.fwd.selection, newdata=test)
model.fwd.selection.sse = sum((model.fwd.selection.predict - test$annual.profit)^2)
model.fwd.selection.osr2 = 1 - model.fwd.selection.sse/sst.test
print(model.fwd.selection.r2)
```

```
## [1] 0.9450826
```

```
print(model.fwd.selection.osr2)
```

```
## [1] 0.8447191
```

```
newpred.fwd.selection = predict(model.fwd.selection,newdata=sites.const)
value.fwd.selection1 = sum(newpred.fwd.selection)
print(value.fwd.selection1)
```

```
## [1] 33634405
```

```
model.fwd.selection = lm(annual.profit ~ lci + nearcomp + nearmil + freestand + sqft
+ pop + agg.inc  +
                           col.grad + drive + +public + home, data=sites)
newpred.fwd.selection = predict(model.fwd.selection,newdata=sites.const)
value.fwd.selection2 = sum(newpred.fwd.selection)
print(value.fwd.selection2)
```

```
## [1] 33194053
```

LASSO Regularization

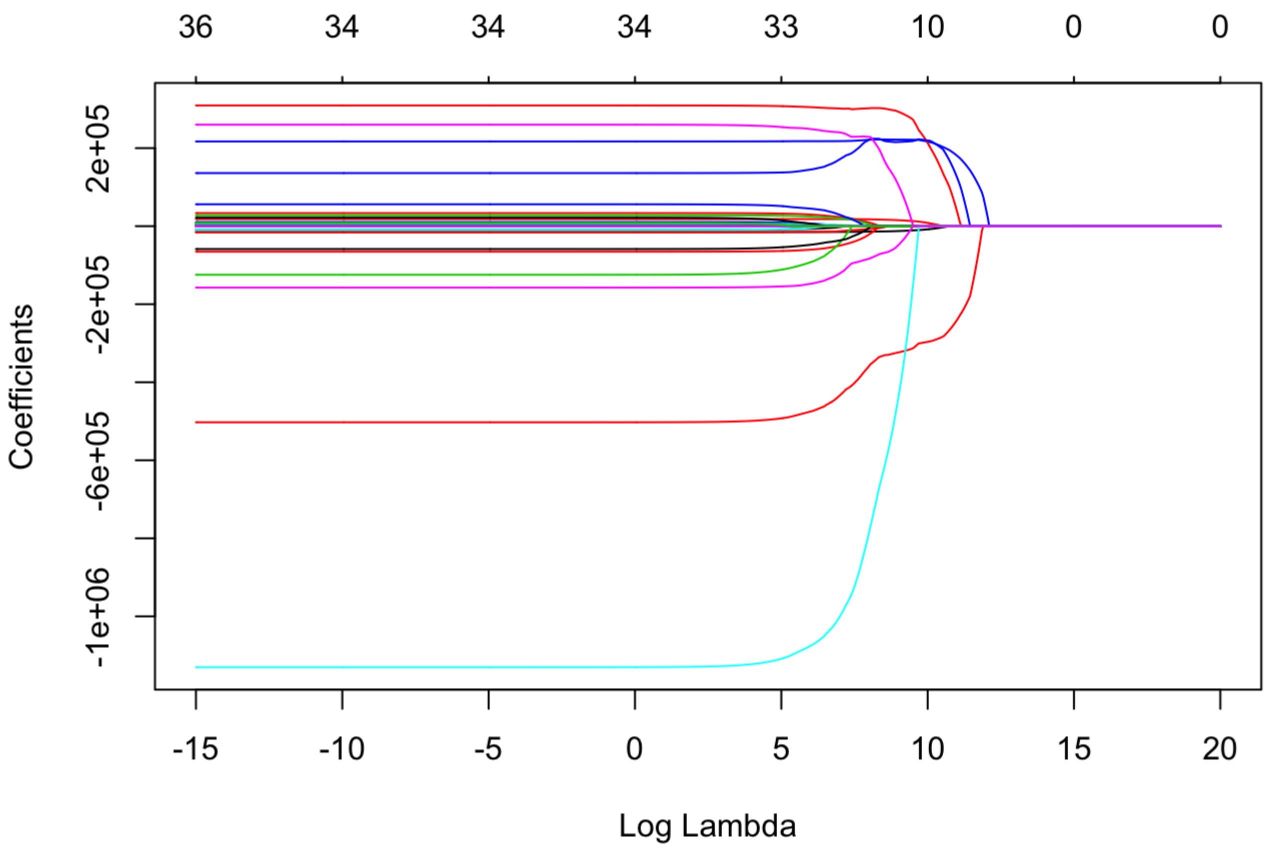
We note that none of the new stroes are located in Ocklahoma, hence we need to add a new column to the matrix having only 0 values.

```
x.train=model.matrix(annual.profit~.-store.number-1,data=train)
y.train=train$annual.profit
x.test=model.matrix(annual.profit~.-store.number-1,data=test)
y.test=test$annual.profit
x.sites=model.matrix(annual.profit~.-store.number-1,data=sites)
y.sites=sites$annual.profit
x.newsites=model.matrix(Kathleen.Previous.Prediction~.-store.number-1,data=sites.cons
t)
x.newsites = cbind(x.newsites, stateOK = rep(0, 48))
order <- c(1:11, 36, 12:35)
x.newsites <- x.newsites[, order]
```

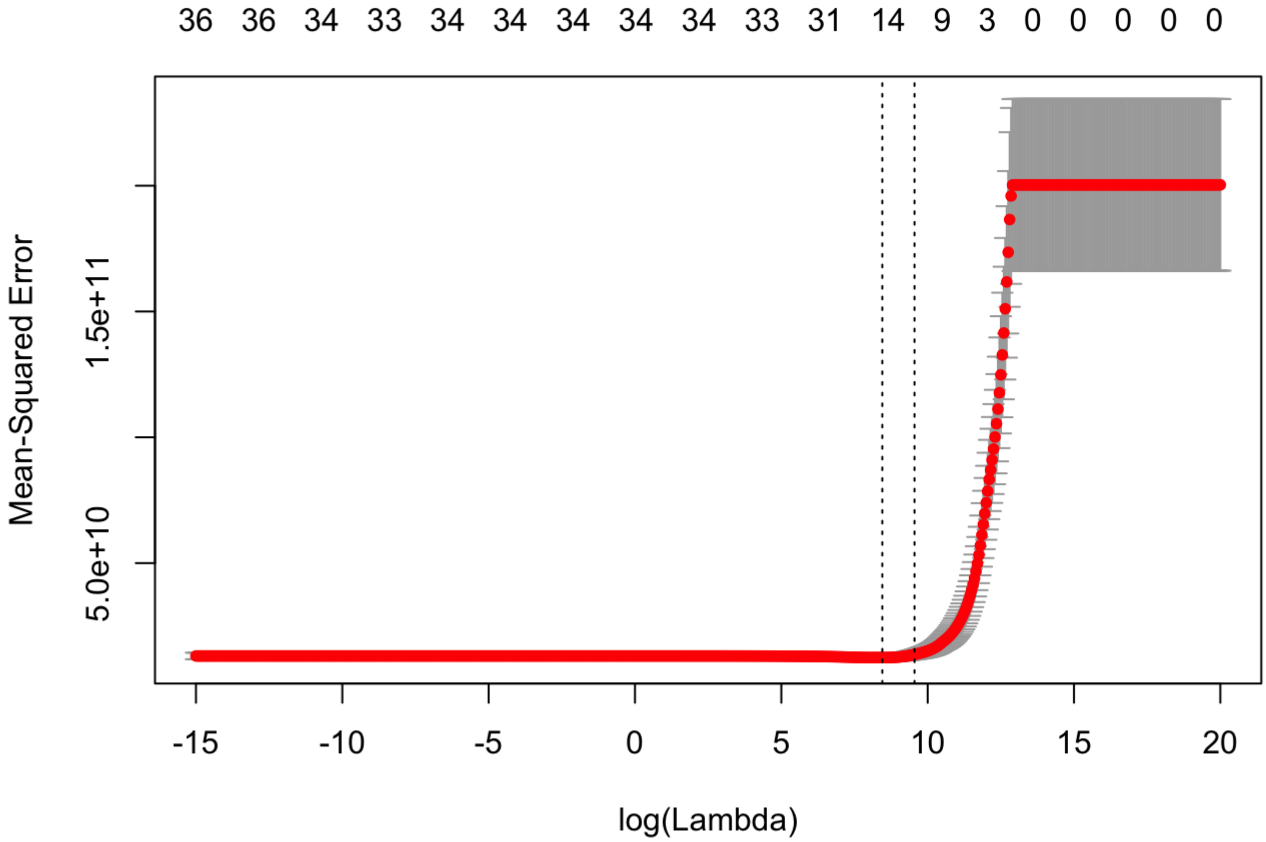
LASSO model

We now create the LASSO model and plot the coefficients as a function of Log Lambda. We choose the sequence -15 to 20 as we observed it yelds lammbda values which lead to models will all variables (36 variables to 0 variables), exemplifying LASSO's subset selection properties.

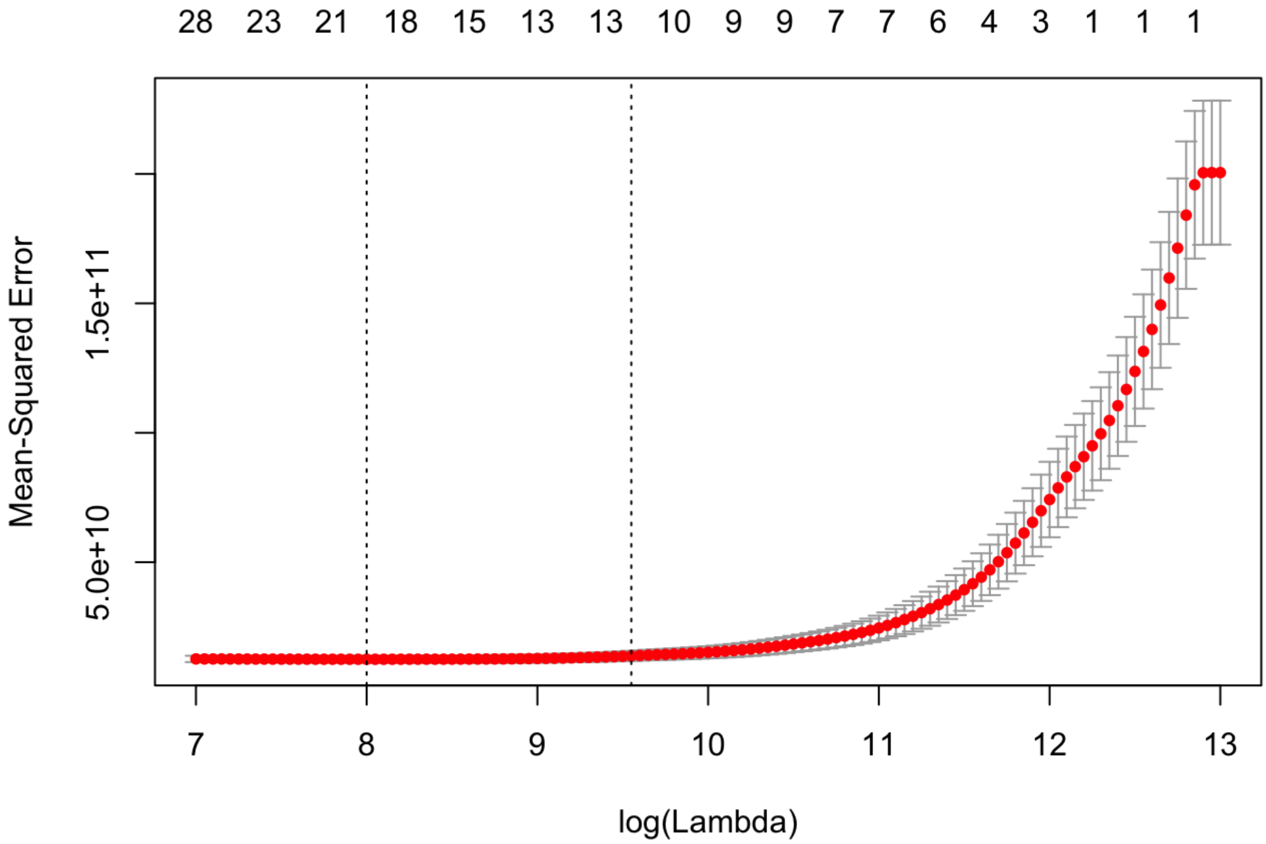
```
library(glmnet)
set.seed(144)
lambdas.lasso = exp(seq(20, -15, -0.05))
model.lasso = glmnet(x.train,y.train,alpha=1,lambda=lambdas.lasso)
plot(model.lasso ,"lambda")
```




```
cv.lasso <- cv.glmnet(x.train,y.train,lambda=lambdas.lasso,alpha=1,nfolds=10)
plot(cv.lasso)
```



We can further zoom into the above graph for a bettwe depiction of the MSE.



```
lasso.lambda.cv <- cv.lasso$lambda.min
lasso.lambda.1SE.cv <- cv.lasso$lambda.1se
print(lasso.lambda.cv)
```

```
## [1] 2980.958
```

```
print(lasso.lambda.1SE.cv)
```

```
## [1] 14044.69
```

```
print(log(lasso.lambda.cv))
```

```
## [1] 8
```

```
print(log(lasso.lambda.1SE.cv))
```

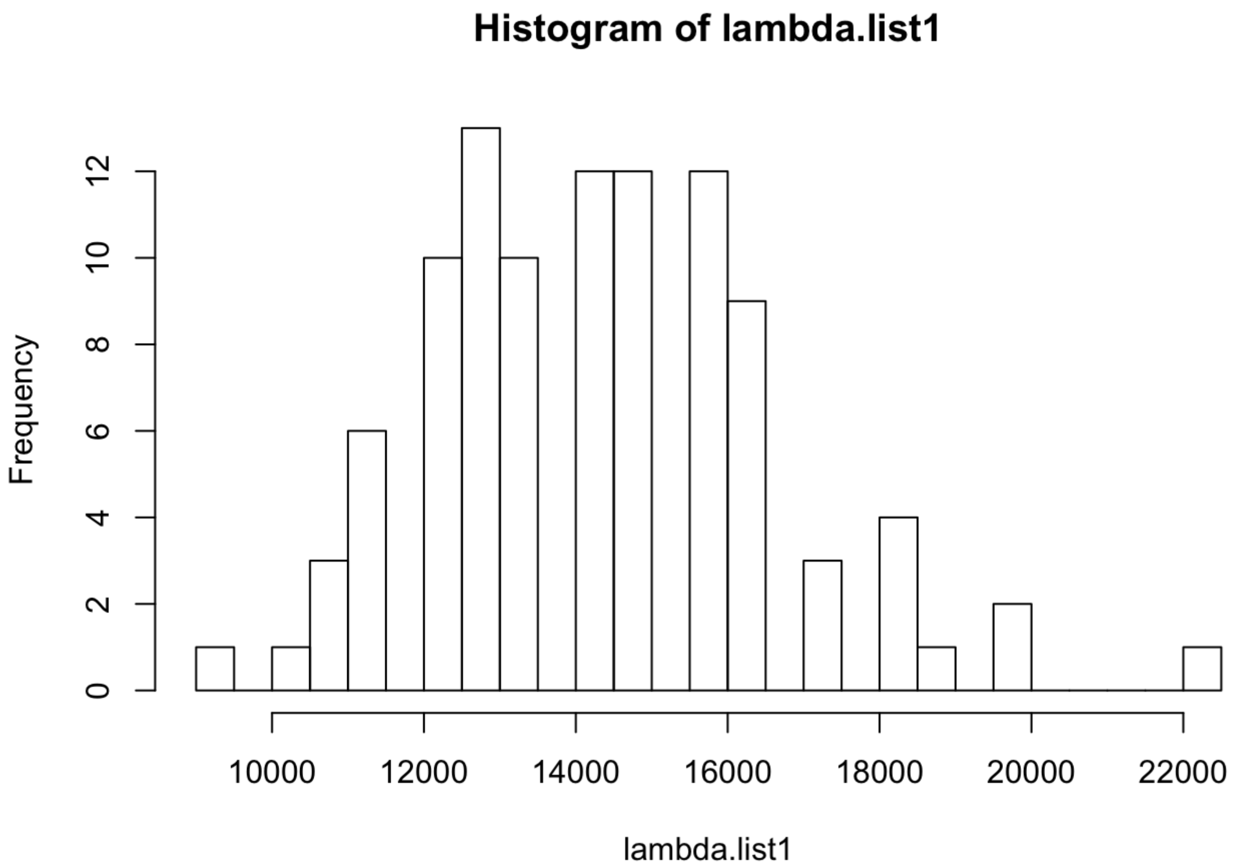
```
## [1] 9.55
```

We see the values of lambda and the log(lamda) which we can also track on the graph. We choose the **lambda + 1 standard error** because, in this case we prefer interpretability over minimizing the MSE. The model must drive decison making in a sensitive business scenario (where the store valuation will influence a buyout decision). Hence a smaller number of coefficients leading to increased interpretability is preferable. We also note that the MSE does not drastically change betweent the optimal lambda and 1 standard error above

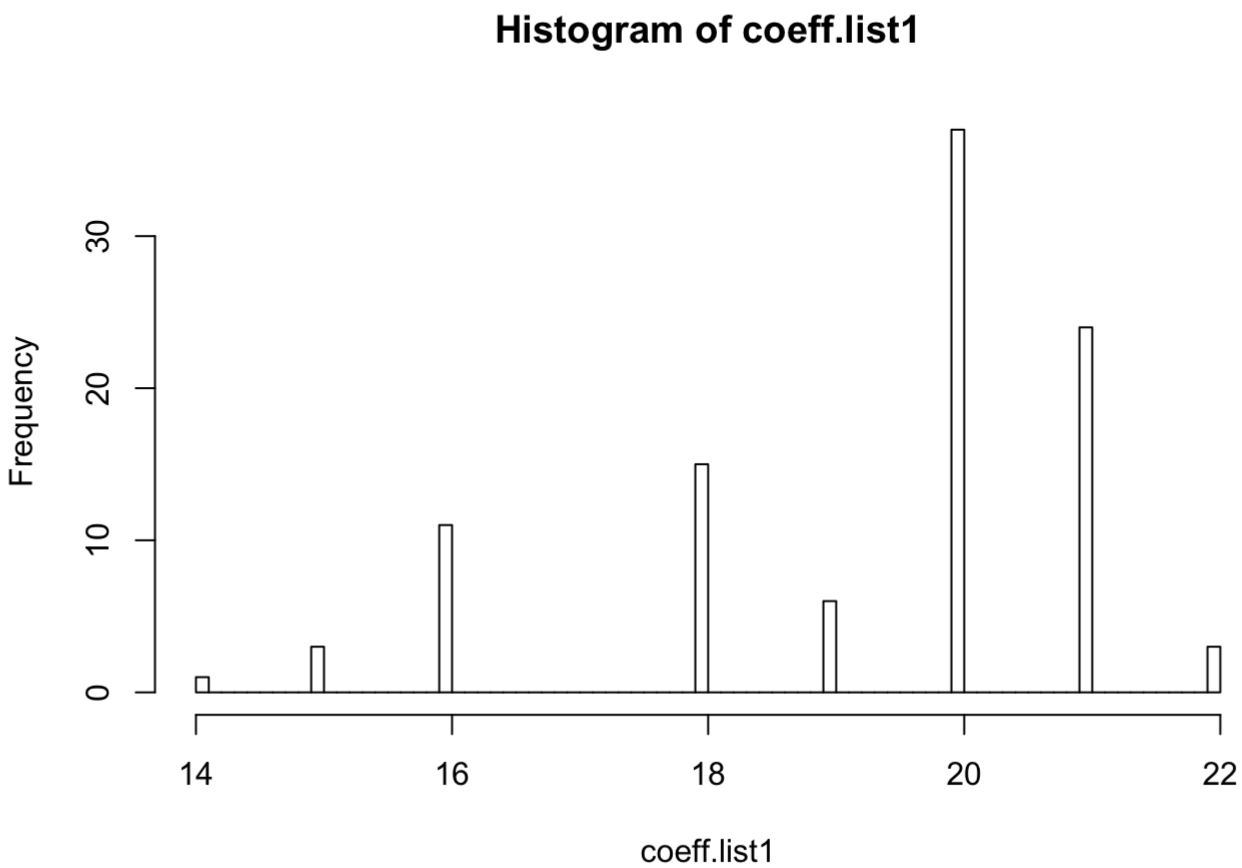
Repeated cross-validation

```
library(coefplot)

lambda.list1 = rep(NA, 100)
coeff.list1 =  rep(NA, 100)

for (i in 1:100){
  lambdas.lasso = exp(seq(13, 7, -0.05))
  model.lasso = glmnet(x.train,y.train,alpha=1,lambda=lambdas.lasso)
  cv.lasso <- cv.glmnet(x.train,y.train,alpha=1,lambda=lambdas.lasso,nfolds=10)
  lasso.lambda.cv <- cv.lasso$lambda.1se
  lambda.list1[i] = lasso.lambda.cv
  coeff.list1[i] = nrow(extract.coef(cv.lasso)) -1
}
```

```
hist(lambda.list1, breaks = 20)
```




```
hist(coeff.list1, breaks = 100)
```



```
median(lambda.list1)
```

```
## [1] 14044.69
```

We note that once more the LASSO fits models with 20 coefficients as the most frequent value. We pick the best lambda value as the median value.

```
best.lambda = median(lambda.list1)
```

##Final LASSO model

```
model.lasso.final = glmnet(x.train,y.train,alpha=1,lambda=best.lambda)
model.lasso.predict = predict(model.lasso.final,x.train)
model.lasso.r2 <- 1-sum((model.lasso.predict-train$annual.profit)^2)/sum((mean(train
$annual.profit)-train$annual.profit)^2)

model.lasso.predict.test = predict(model.lasso.final,x.test)
model.lasso.osr2 <- 1-sum((model.lasso.predict.test-test$annual.profit)^2)/sum((mean
(train$annual.profit)-test$annual.profit)^2)
newpred.lasso = predict(model.lasso.final,x.newsites)
value.lasso1 = sum(newpred.lasso)

model.lasso.final = glmnet(x.sites,y.sites,alpha=1,lambda=best.lambda)
newpred.lasso = predict(model.lasso.final,x.newsites)
value.lasso2 = sum(newpred.lasso)

print(model.lasso.r2)
```

```
## [1] 0.9392624
```

```
print(model.lasso.osr2)
```

```
## [1] 0.8728999
```

```
print(value.lasso1)
```

```
## [1] 33589376
```

```
print(value.lasso2)
```

```
## [1] 33534244
```

We note that 13 of the coefficients are non-zero.

```
library(coefplot)
coef(model.lasso.final)
```

```
## 37 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)     3.347111e+05
## lci            -1.013454e+04
## nearcomp        1.303005e+04
## nearmil         5.923266e+02
## freestand       2.222225e+05
## gini            .
## housemed        .
## stateAZ         .
## stateCA         .
## stateKS         .
## stateNM         .
## stateNV         .
## stateOK         .
## stateTX         .
## stateUT         .
## stateWY         .
## sqft            1.592681e+02
## intersect       .
## pop             4.498665e+01
## agemed          .
## non.us.citizen  .
## agg.inc         2.264830e-03
## med.inc         .
## noHS            .
## HS             -1.062138e+04
## some.col       -3.988171e+04
## col.grad        2.286954e+05
## post.grad       .
## com0            .
## com15           .
## com30           .
## com60           .
## drive          -2.197112e+05
## public          .
## walk            3.074855e+05
## home            .
## other           6.616181e+04
```

Let us finally take a look at all the models.

```
summary <- data.frame(ModelName = c("Original Model", "All Predictors", "New Model",
"Best Subset", "LASSO"),
                          In_Sample_R2 = c(model.original.r2, model.all.r2,
                                           model.new.r2, model.fwd.selection.r2, mode
l.lasso.r2),
                          Out_Of_Sample_R2 = c(model.original.osr2, model.all.osr2,
                                               model.new.osr2, model.fwd.selection.os
r2, model.lasso.osr2),
                          Valuation_train =c(value.original1, value.all1, value.new1,
                                             value.fwd.selection1, value.lasso1),
                          Valuation_all =c(value.original2, value.all2, value.new2,
                                           value.fwd.selection2, value.lasso2))

summary
```

```
##        ModelName In_Sample_R2 Out_Of_Sample_R2 Valuation_train
## 1 Original Model    0.7861431        0.7201434        40016174
## 2 All Predictors    0.9483931        0.8047443        33257851
## 3      New Model    0.8289329        0.7686991        40054717
## 4    Best Subset    0.9450826        0.8447191        33634405
## 5          LASSO    0.9392624        0.8728999        33589376
##   Valuation_all
## 1      40199576
## 2      34051269
## 3      40576369
## 4      33194053
## 5      33534244
```

As a final model we choose **Best Subset Selection, as it achieves the best predictive performance with the and fewer coefficents. As we mentioned, we value interpretability, and a more parsimonious model is more valuable to guide Milagro and Harriman Capital's decision making**.