Model selection

In this notebook we're going to analyse different techniques for model selection and afterwards we're going to discuss their shortcomings.

Selection criteria

First of all, we're going to look at different criteria to compare models based on their performance and complexity.

```
require(ISLR)
```

```
## Loading required package: ISLR

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

```
head(Hitters)
```

```
##                    AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## -Andy Allanson       293   66     1   30  29    14     1    293    66      1
## -Alan Ashby          315   81     7   24  38    39    14   3449   835     69
## -Alvin Davis         479  130    18   66  72    76     3   1624   457     63
## -Andre Dawson        496  141    20   65  78    37    11   5628  1575    225
## -Andres Galarraga    321   87    10   39  42    30     2    396   101     12
## -Alfredo Griffin     594  169     4   74  51    35    11   4408  1133     19
##                    CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Andy Allanson        30   29     14      A        E     446      33     20
## -Alan Ashby          321  414    375      N        W     632      43     10
## -Alvin Davis         224  266    263      A        W     880      82     14
## -Andre Dawson        828  838    354      N        E     200      11      3
## -Andres Galarraga     48   46     33      N        E     805      40      4
## -Alfredo Griffin     501  336    194      A        W     282     421     25
##                    Salary NewLeague
## -Andy Allanson         NA         A
## -Alan Ashby         475.0         N
## -Alvin Davis        480.0         A
## -Andre Dawson       500.0         N
## -Andres Galarraga    91.5         N
## -Alfredo Griffin    750.0         A
```

```
summary(Hitters)
```

```
##      AtBat            Hits         HmRun            Runs       
##  Min.   : 16.0   Min.   :  1   Min.   : 0.00   Min.   :  0.00  
##  1st Qu.:255.2   1st Qu.: 64   1st Qu.: 4.00   1st Qu.: 30.25  
##  Median :379.5   Median : 96   Median : 8.00   Median : 48.00  
##  Mean   :380.9   Mean   :101   Mean   :10.77   Mean   : 50.91  
##  3rd Qu.:512.0   3rd Qu.:137   3rd Qu.:16.00   3rd Qu.: 69.00  
##  Max.   :687.0   Max.   :238   Max.   :40.00   Max.   :130.00  
##                                                                 
##       RBI             Walks            Years            CAtBat       
##  Min.   :  0.00   Min.   :  0.00   Min.   : 1.000   Min.   :   19.0
##  1st Qu.: 28.00   1st Qu.: 22.00   1st Qu.: 4.000   1st Qu.:  816.8
##  Median : 44.00   Median : 35.00   Median : 6.000   Median : 1928.0
##  Mean   : 48.03   Mean   : 38.74   Mean   : 7.444   Mean   : 2648.7
##  3rd Qu.: 64.75   3rd Qu.: 53.00   3rd Qu.:11.000   3rd Qu.: 3924.2
##  Max.   :121.00   Max.   :105.00   Max.   :24.000   Max.   :14053.0
##
##      CHits            CHmRun           CRuns             CRBI
##  Min.   :   4.0   Min.   :  0.00   Min.   :   1.0   Min.   :   0.00
##  1st Qu.: 209.0   1st Qu.: 14.00   1st Qu.: 100.2   1st Qu.:  88.75
##  Median : 508.0   Median : 37.50   Median : 247.0   Median : 220.50
##  Mean   : 717.6   Mean   : 69.49   Mean   : 358.8   Mean   : 330.12
##  3rd Qu.:1059.2   3rd Qu.: 90.00   3rd Qu.: 526.2   3rd Qu.: 426.25
##  Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.00
##
##      CWalks          League  Division    PutOuts          Assists
##  Min.   :   0.00   A:175   E:157    Min.   :   0.0   Min.   :  0.0
##  1st Qu.:  67.25   N:147   W:165    1st Qu.: 109.2   1st Qu.:  7.0
##  Median : 170.50                    Median : 212.0   Median : 39.5
##  Mean   : 260.24                    Mean   : 288.9   Mean   :106.9
##  3rd Qu.: 339.25                    3rd Qu.: 325.0   3rd Qu.:166.0
##  Max.   :1566.00                    Max.   :1378.0   Max.   :492.0
##
##      Errors          Salary       NewLeague
##  Min.   : 0.00   Min.   :  67.5   A:176
##  1st Qu.: 3.00   1st Qu.: 190.0   N:146
##  Median : 6.00   Median : 425.0
##  Mean   : 8.04   Mean   : 535.9
##  3rd Qu.:11.00   3rd Qu.: 750.0
##  Max.   :32.00   Max.   :2460.0
##                  NA's   :59
```

```r
# removing the NA
dim(Hitters)
```

```
## [1] 322  20
```

```r
Hitters<- na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

We're going to use cross-validation to compare the results from different selection criteria.

```r
nfolds <- 10
n <- dim(Hitters)[1]
folds <- cut(1:n, nfolds, labels = F)
# a bit of shuffling
indices <- sample(1:n, size=n, replace=F)
```

```r
library(leaps)
```

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

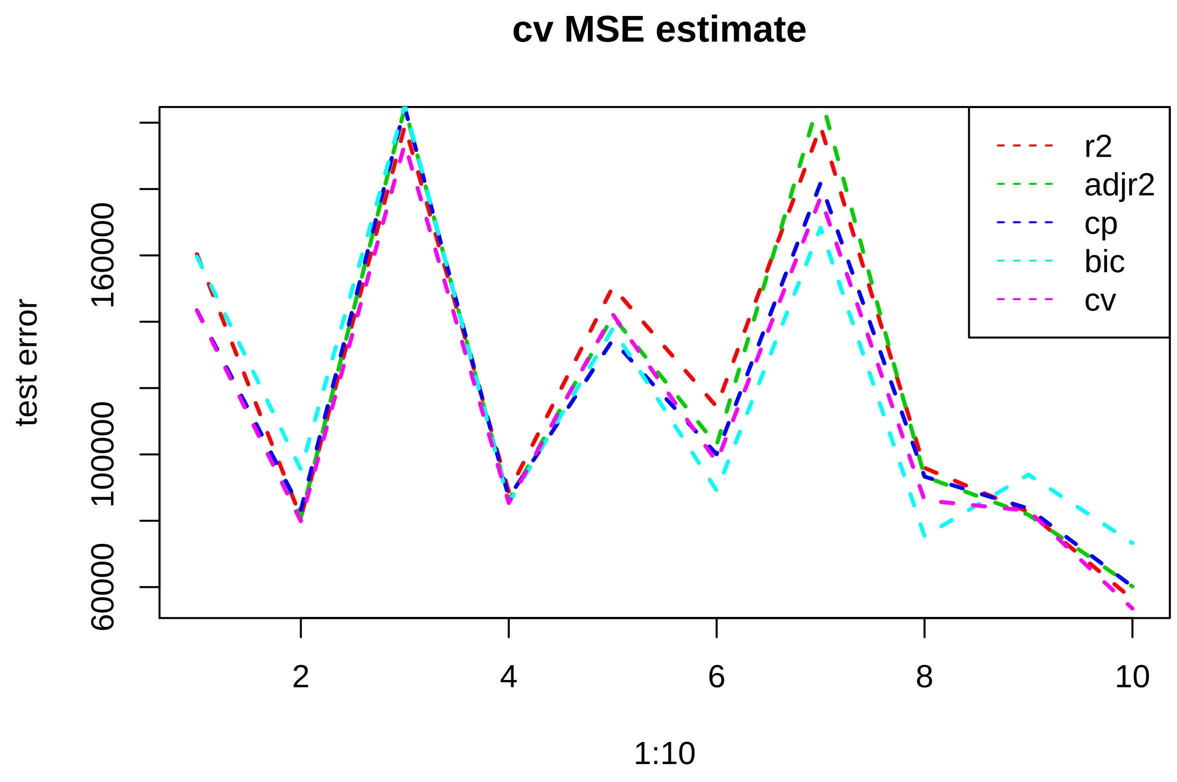
```r
get.bss.test.error<- function(train, test, cv.best){
  # estimates the error on the test dataset for the best model
  # according to each criteria
  all.best<- regsubsets(x=Salary~.,data=train,nbest=1,
                        nvmax=dim(train)[2]-1, # using all variables
                        method="forward" )
  s <- summary(all.best)
  r2 <- coef(all.best, id=which.max(s$rsq))
  adjr2 <- coef(all.best, id=which.max(s$adjr2))
  cp <- coef(all.best, id=which.min(s$cp))
  bic <- coef(all.best, id=which.min(s$bic))
  cv.coefs <- coef(all.best, id=cv.best)
  # test predictions
  r2.pred <- model.matrix(Salary~.,test)[,names(r2)]%*%r2
  adjr2.pred <- model.matrix(Salary~.,test)[,names(adjr2)]%*%adjr2
  cp.pred <- model.matrix(Salary~.,test)[,names(cp)]%*%cp
  bic.pred <- model.matrix(Salary~.,test)[,names(bic)]%*%bic
  cv.pred <- model.matrix(Salary~.,test)[,names(cv.coefs)]%*%cv.coefs
  # test errors
  errors <- mean((r2.pred - test$Salary)**2)
  errors <- c(errors,mean((adjr2.pred - test$Salary)**2))
  errors <- c(errors,mean((cp.pred - test$Salary)**2))
  errors <- c(errors,mean((bic.pred - test$Salary)**2))
  errors <- c(errors,mean((cv.pred - test$Salary)**2))
  return(errors)
}


get.cv.error <- function(ncv, nmodels, data){
  # evaluates the mean cross-validation error of the linear model
  # with the selected coefficients
  n.cv <- dim(data)[1]
  folds.cv <- cut(1:n.cv, ncv, labels=F)
  cv.errors <- matrix(nrow = ncv, ncol = nmodels)
  indices.cv <- 1:n.cv
  for(j in 1:ncv){
    test.indices.cv <- indices.cv[folds.cv==j]
    test.cv <- data[test.indices.cv,]
    train.cv <- data[-test.indices.cv,]
    cv.all.best<- regsubsets(x=Salary~.,data=train.cv,
                             nbest=1,nvmax=nmodels, # using all variables
                             method="forward" )
     for(m in 1:nmodels){
       cv.coefs <- coef(cv.all.best, id=m)
       cv.preds <- model.matrix(Salary~.,test)[,names(cv.coefs)]%*%cv.coefs
       # test errors
       cv.errors[j,m] <- mean((cv.preds - test$Salary)**2)
    }
  }
  # selecting the model with the least mean error
  # expected test MSE estimated by CV for each model
  return(which.min(colMeans(cv.errors)))
}

test.errors <- matrix(nrow=nfolds, ncol=5)

for(i in 1:nfolds){
  test.indices <- indices[folds==i]
  test <- Hitters[test.indices,]
  train <- Hitters[-test.indices,]
  # Now we'll use BSS on the train dataset
  # And we'll record the error on the test set
  # get best cv model
  cv.best <- get.cv.error(ncv=5, nmodels=(dim(Hitters)[2]-1),data = train)
  test.errors[i,] <- get.bss.test.error(train=train, test=test, cv.best=cv.best)

}
```

Let's look at the results.

```
test.errors <- data.frame(test.errors)
names(test.errors) <- c("r2","adjr2","cp","bic","cv")
test.errors
```

```
##            r2     adjr2        cp       bic        cv
## 1  160378.72 143422.40 143422.40 159610.15 143422.40
## 2   80984.61  80957.27  83288.87  95476.85  79866.42
## 3  199024.56 204565.73 204565.73 205284.86 193625.86
## 4   88793.36  86079.44  86995.54  85359.33  85359.33
## 5  150460.62 141612.89 134504.21 137837.55 142262.61
## 6  114399.12 102855.15 100042.77  89121.79  97775.38
## 7  198900.94 208226.35 181812.81 168333.41 177456.26
## 8   95930.56  93306.29  93306.29  75335.05  86151.64
## 9   82647.53  81797.56  83679.26  93913.87  83031.22
## 10  56376.12  60188.58  60188.58  73283.04  53540.29
```

```
plot(1:10, test.errors$r2, type="l", lty="dashed", col=2, ylab="test error", main="cv MSE estimate ", lw
lines(1:10, test.errors$adjr2, type="l", lty="dashed", col=3, lwd=2)
lines(1:10, test.errors$cp, type="l", lty="dashed", col=4, lwd=2)
lines(1:10, test.errors$bic, type="l", lty="dashed", col=5, lwd=2)
lines(1:10, test.errors$cv, type="l", lty="dashed", col=6, lwd=2)
legend("topright", legend = c("r2", "adjr2","cp","bic","cv"), col=c(2,3,4,5,6), lty="dashed")
```



```r
colMeans(test.errors)
```

```
##       r2    adjr2       cp      bic       cv
## 122789.6 120301.2 117180.6 118355.6 114249.1
```

```r
which.min(colMeans(test.errors))
```

```
## cv
##  5
```

So the cross validation criteria seems to be the most reliable in model selection. We'll now use this criteria to select the best model fitting it on the whole data.

```r
best.cv <- get.cv.error(ncv=10, nmodels=(dim(Hitters)[2]-1), data=Hitters)
best.cv
```

```
## [1] 2
```

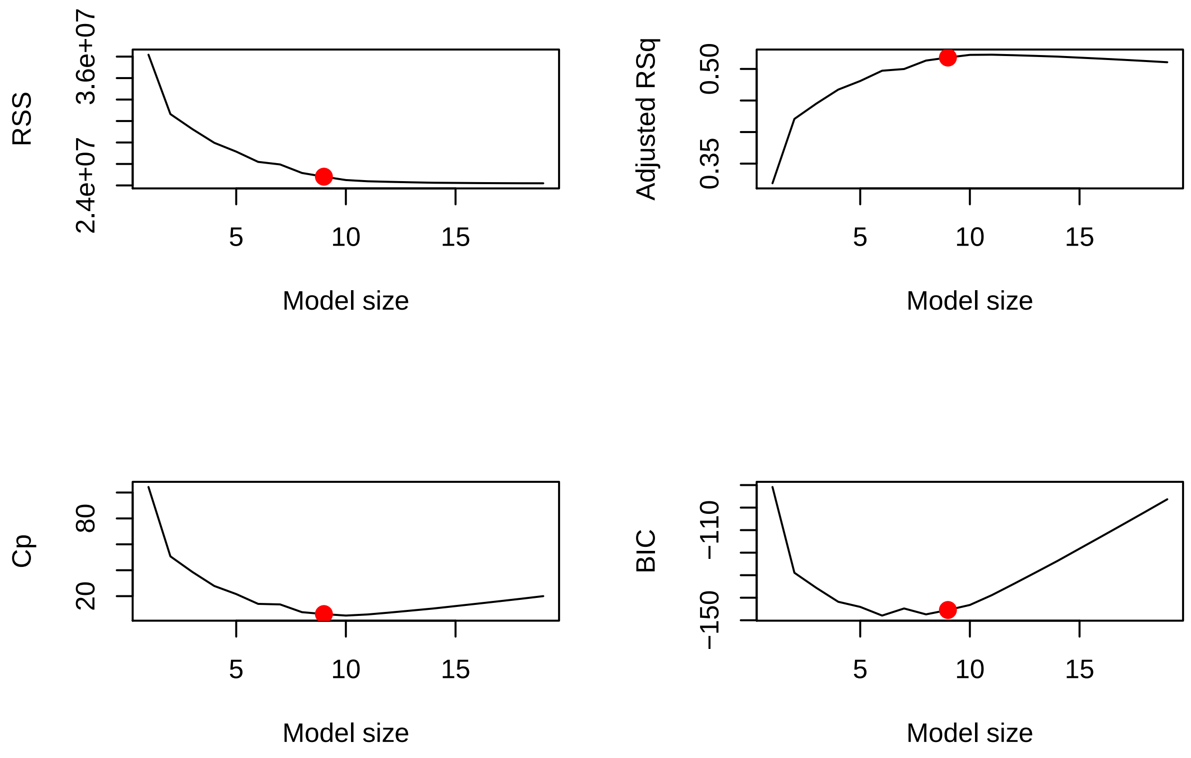
Let's now look at the best model:

```r
nmodels <- 19
all.best <- regsubsets(x=Salary~.,data=Hitters,
                       nbest=1,nvmax=nmodels, # using all variables
                       method="forward")
coef(all.best, id=9)
```

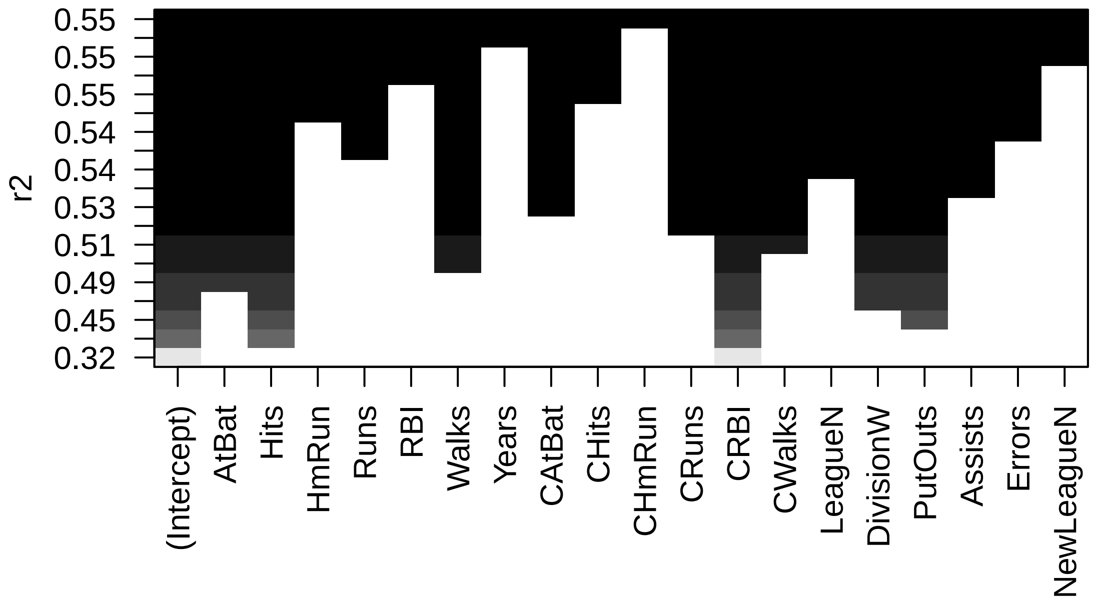
```
##   (Intercept)         AtBat          Hits         Walks        CAtBat
##  146.24960033   -1.93676754    6.65672102    5.55204413   -0.09953904
##         CRuns          CRBI        CWalks     DivisionW       PutOuts
##    1.25067124    0.66176849   -0.77798498 -115.34950146    0.27773062
```

Let's look at some plots to see how the best model is seen according to other criteria.

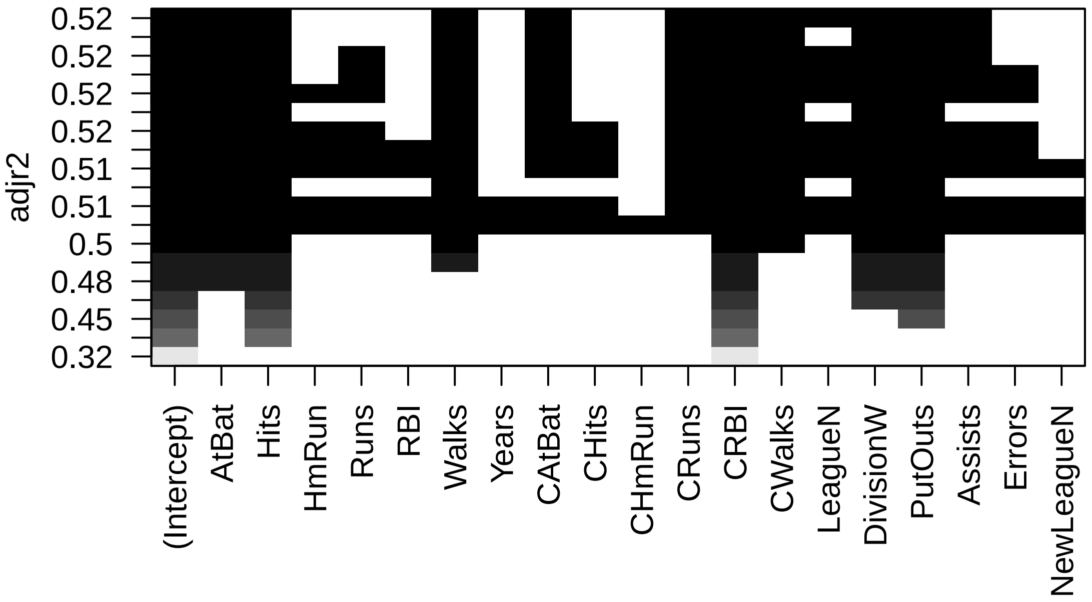
```r
par(mfrow=c(2,2))
s <- summary(all.best)
# rss
plot(s$rss,xlab="Model size",ylab="RSS",type="l")
points(9, s$rss[9], col="red",cex=2,pch=20)
# adjr2
plot(s$adjr2,xlab="Model size",ylab="Adjusted RSq",type="l")
points(9, s$adjr2[9], col="red",cex=2,pch=20)
# Cp
plot(s$cp,xlab="Model size",ylab="Cp",type='l')
points(9, s$cp[9], col="red", cex=2, pch=20)
# BIC
plot(s$bic,xlab="Model size",ylab="BIC",type='l')
points(9, s$bic[9], col="red", cex=2, pch=20)
```



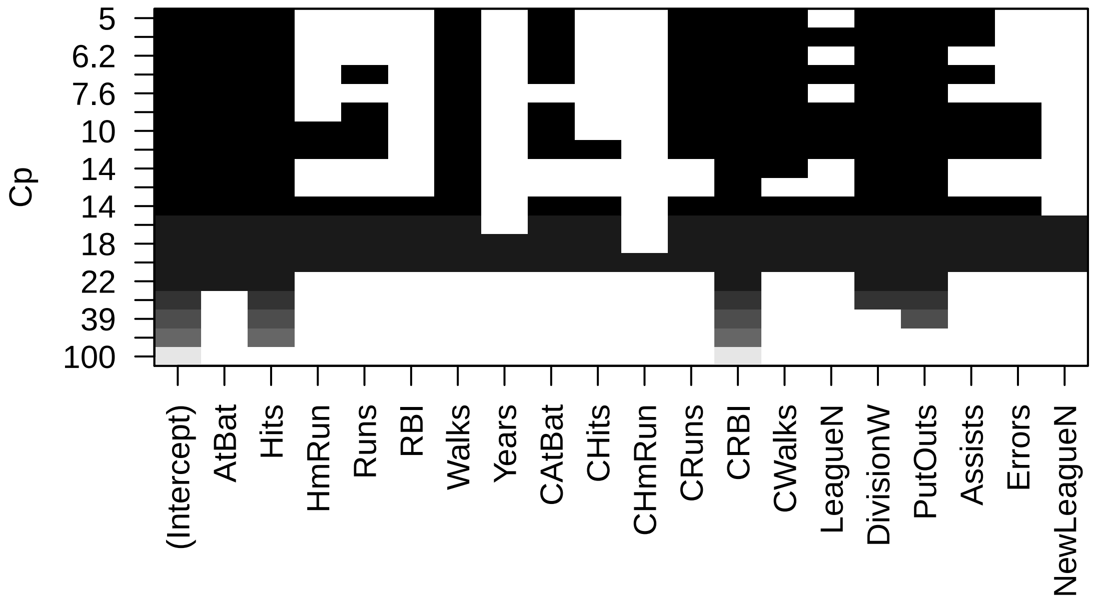
```r
par(mfrow=c(1,1))
plot(all.best, scale="r2")
```



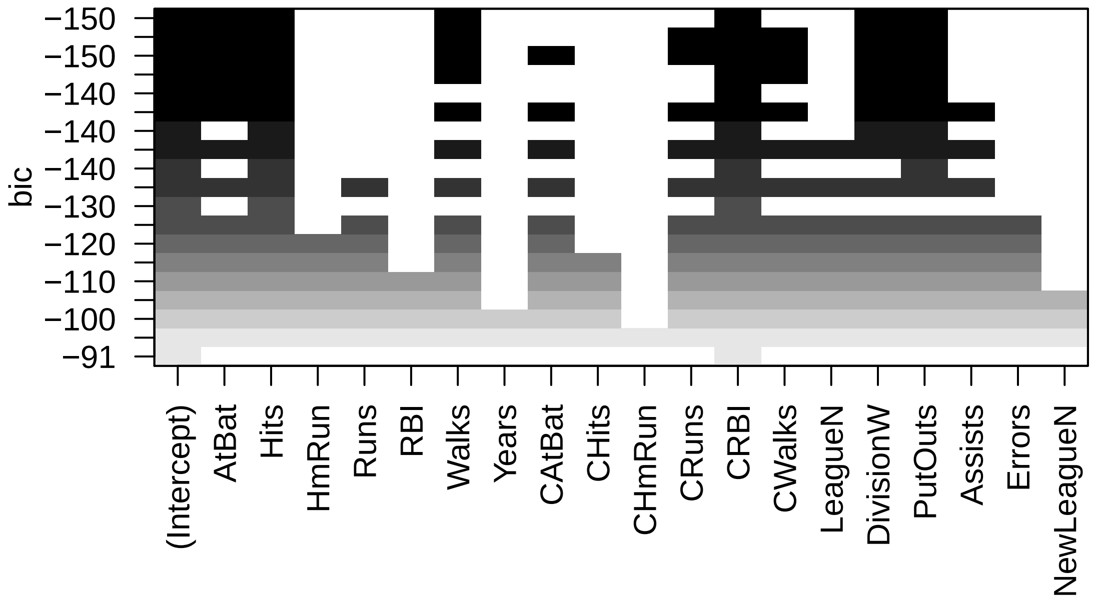
```
plot(all.best, scale="adjr2")
```



```r
plot(all.best, scale="Cp")
```



```
plot(all.best, scale="bic")
```



## Shrinkage methods

We'll now loook at a different set of selection tools: shrinkage methods like LASSO and RIDGE. We'll actually use the ElasticNet model, of which Lasso and RIdge are special cases.

```
require(hdi)
```

```
## Loading required package: hdi

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

## Loading required package: scalreg

## Loading required package: lars

## Loaded lars 1.2
```

```
data("riboflavin")
```

The riboflavin dataset records the riboflavin production by Bacillus subtilis together with the gene expressions. Each row refers to one gene, storing in x its expression level.

```
require(glmnet)
```

```
## Loading required package: glmnet

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

## Loading required package: Matrix

## Loaded glmnet 3.0-2
```

```r
attach(riboflavin)
lambda.grid <- grid <- 10^seq(10,-2, length = 100)
lasso <- glmnet(x = x, y=y, alpha = 1, lambda = lambda.grid)
ridge <- glmnet(x = x, y=y, alpha = 0, lambda = lambda.grid)
```

Exploration of the output:

```r
dim(coef(lasso))
```

```
## [1] 4089  100
```

```r
dim(coef(ridge))
```

```
## [1] 4089  100
```

```r
lasso$lambda[50]
```

```
## [1] 11497.57
```

```r
round(coef(lasso)[,50],2)[coef(lasso)[,50]!=0]
```

```
## (Intercept)
##       -7.16
```

```r
ridge$lambda[50]
```

```
## [1] 11497.57
```

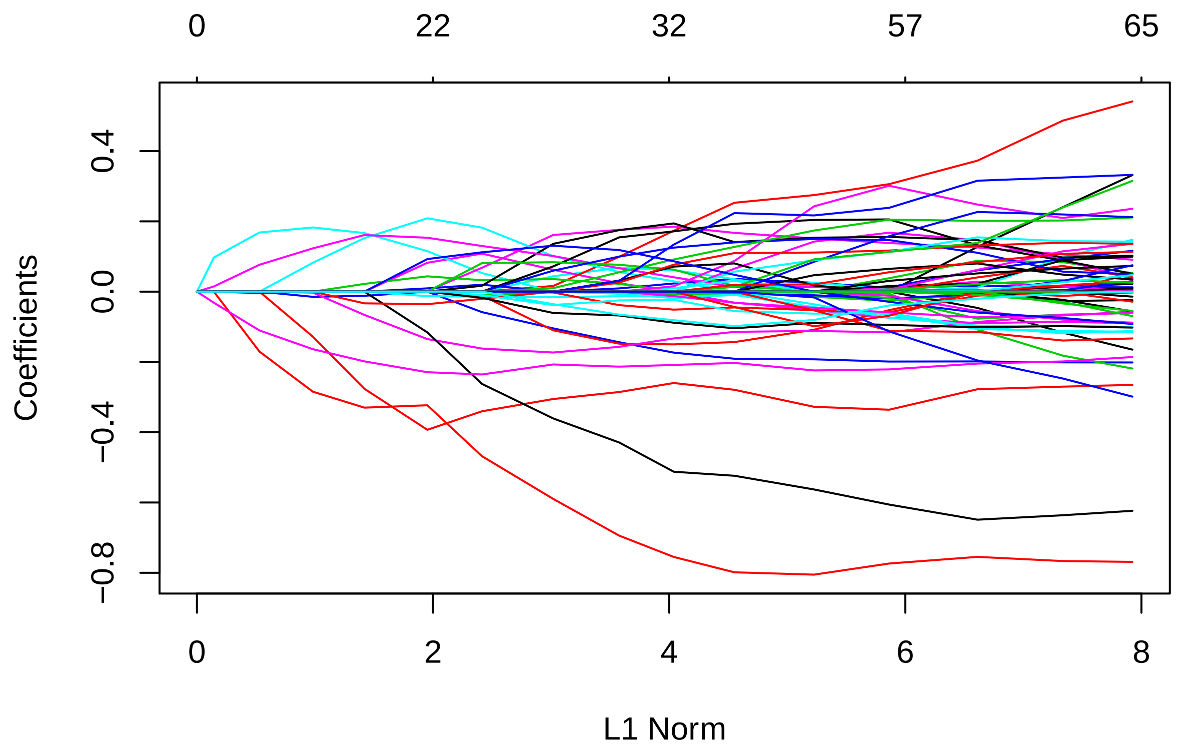
```r
round(coef(ridge)[,50],2)[round(coef(ridge)[,50],2)!=0]
```

```
## (Intercept)
##       -7.15
```

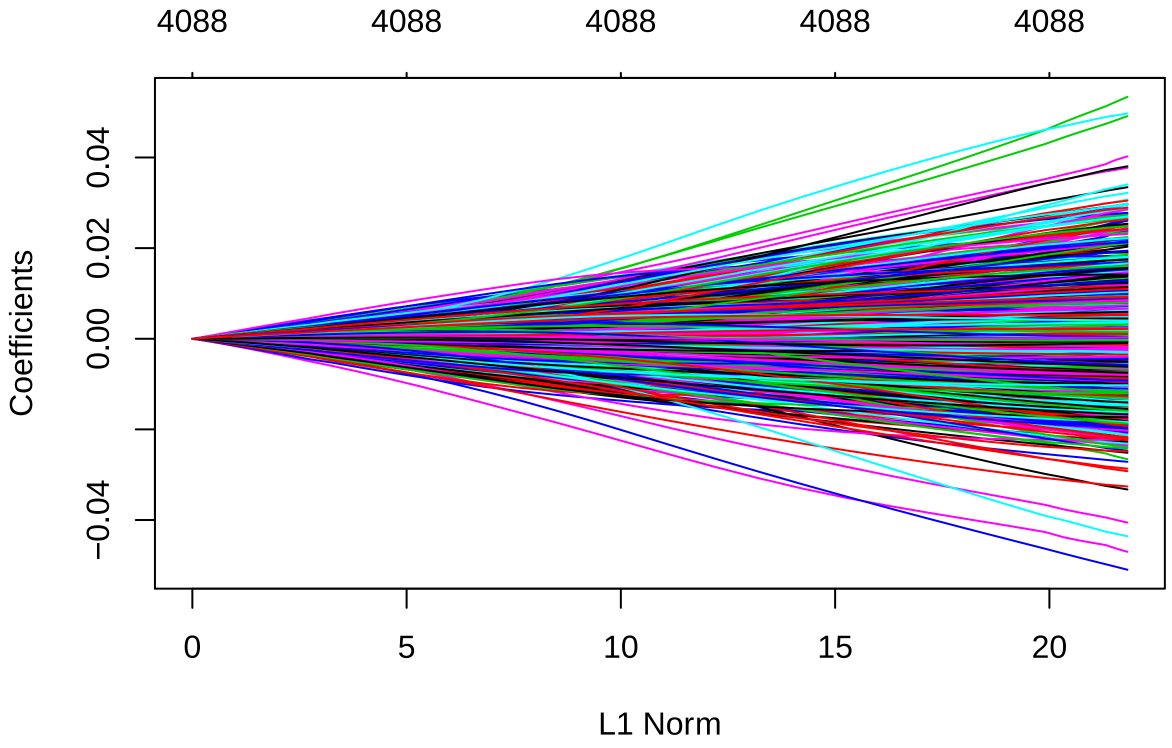
The above lambda is too high to allow any value to be different from zero. Let's have a broader look at the results with a plot:

```r
plot(lasso)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to unique
## 'x' values
```

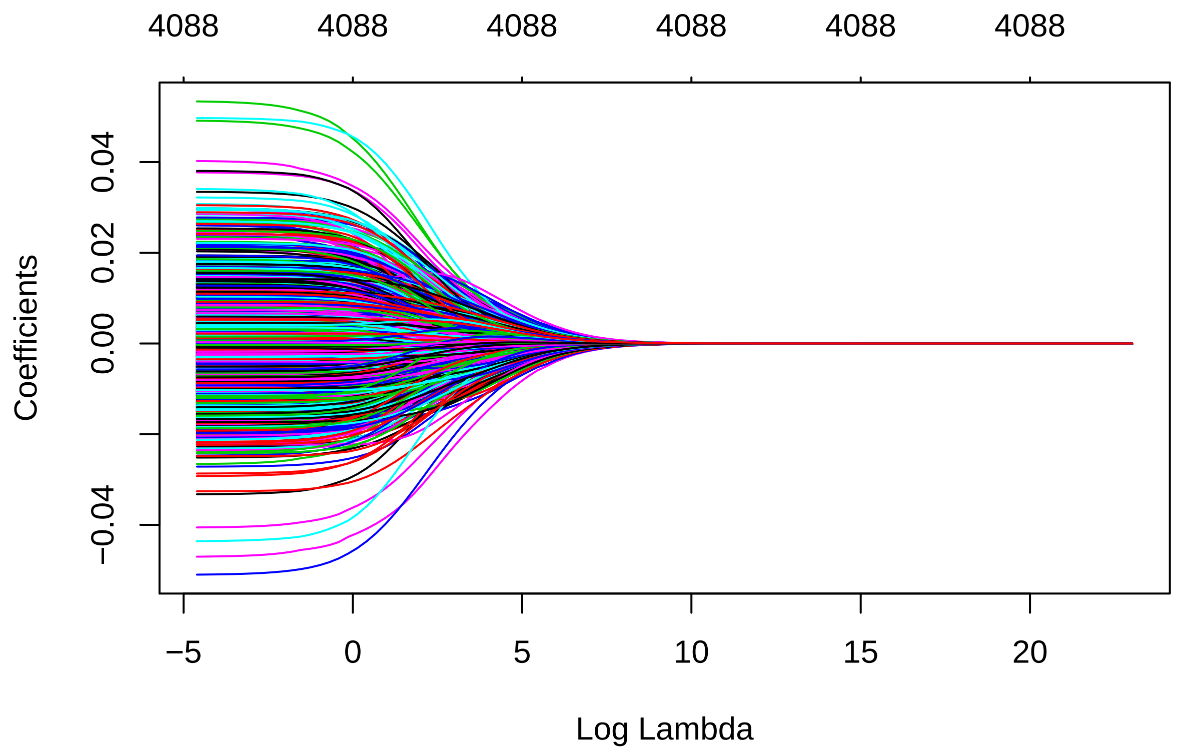


```r
plot(ridge)
```



The above plots are a perfect synthesis of the differences between ridge and lasso: while ridge performs a "soft thresholding", slowly shrinking all variables to 0 as lambda increases, lasso performs a "hard thresholding", cutting off variables as they reach a certain threshold (determined by lambda).

```r
plot(ridge, xvar="lambda")
```

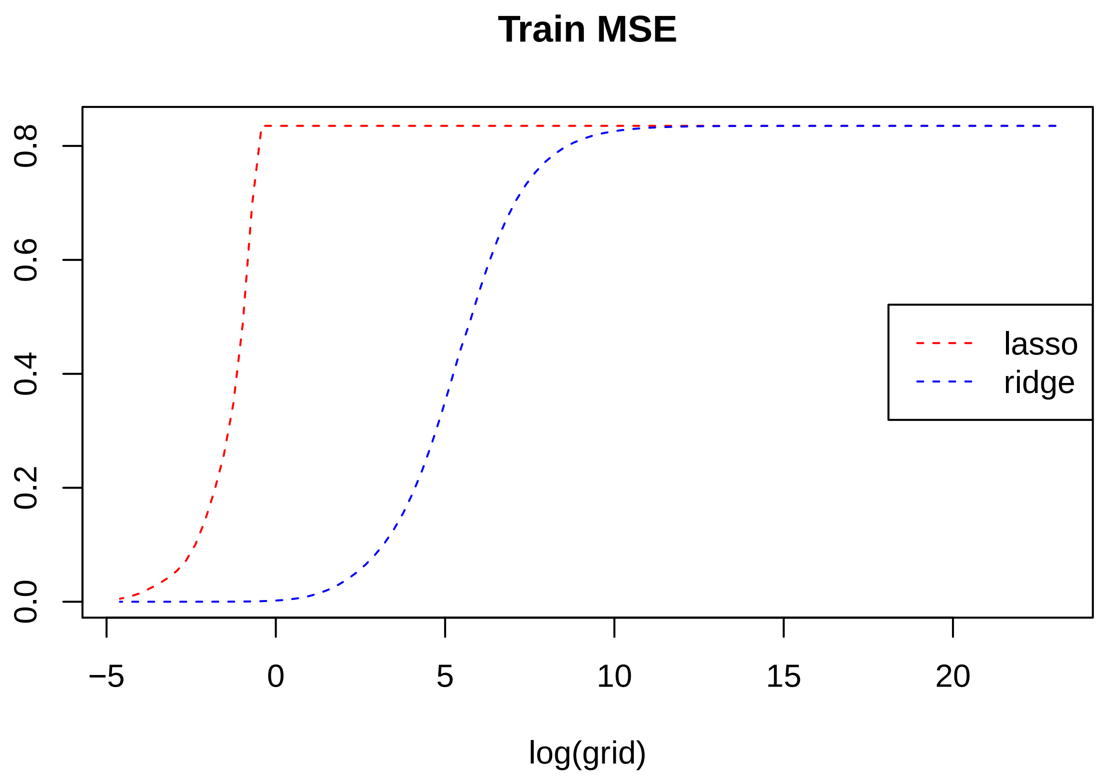


How to use a model for predictions?

```
preds <- predict(lasso, s = 0.01, newx=riboflavin$x)
mse.train <- mean((riboflavin$y - preds)**2)
mse.train
```

```
## [1] 0.005253187
```

```
preds.lasso <- predict(lasso, s = grid, newx=riboflavin$x)
plot(log(grid), colMeans((riboflavin$y - preds.lasso)**2), col="red", main="Train MSE", type="l", ylab =

preds.ridge <- predict(ridge, s = grid, newx=riboflavin$x)
lines(log(grid), colMeans((riboflavin$y - preds.ridge)**2), col="blue", type="l", ylab = "", lty="dashe

legend("right", legend=c("lasso","ridge"), col=c("red","blue"), lty="dashed")
```



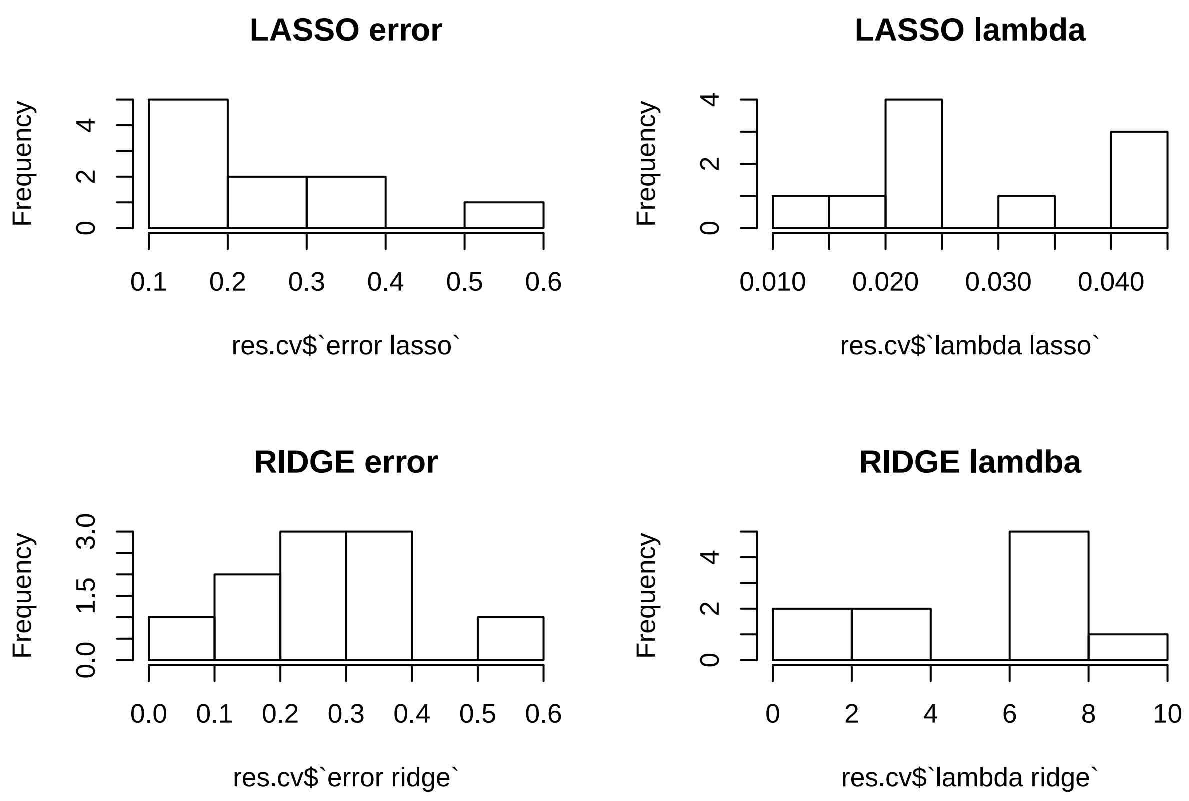
Now let's use cross validation to do both model assessment and model selection: we're going to select the best lasso and ridge models on a train dataset and evaluate them with a second cross-validation against hold-out sets.

```r
nfolds <- 10
n <- dim(riboflavin)[1]
folds <- cut(1:n, breaks = nfolds, labels = F)
#shuffling
indices <- sample(1:n, size=n, replace = F)
res.cv <- matrix(nrow=10, ncol=4)
for(i in 1:nfolds){
  test.indices <- indices[folds==i]
  test <- riboflavin[test.indices, ]
  train <- riboflavin[-test.indices, ]
  ## model selection using cv on the train dataset
  ## LASSO
  lasso.cv <- cv.glmnet(x=train$x, y=train$y,
                        alpha = 1, lambda = lambda.grid, nfolds =10)
  lasso.lambda <- lasso.cv$lambda.min
  res.cv[i,1] <- lasso.lambda
  lasso.fit <- glmnet(train$x, train$y, alpha =1, lambda = lasso.lambda)
  ## RIDGE
  ridge.cv <- cv.glmnet(x=train$x, y=train$y,
                        alpha = 0, lambda = lambda.grid, nfolds = 10)
  ridge.lambda <- ridge.cv$lambda.min
  res.cv[i,3] <- ridge.lambda
  ridge.fit <- glmnet(train$x, train$y, alpha =0, lambda = ridge.lambda)
  ## model assessment on the test dataset
  ## LASSO
  lasso.predict <- predict(lasso.fit, newx=test$x)
  lasso.error<- mean((test$y-lasso.predict)**2)
  res.cv[i,2]<-lasso.error
  ## RIDGE
  ridge.predict <- predict(ridge.fit, newx=test$x)
  ridge.error<- mean((test$y-ridge.predict)**2)
  res.cv[i,4]<-ridge.error
}
res.cv <- data.frame(res.cv)
names(res.cv) <- c("lambda lasso","error lasso","lambda ridge", "error ridge")
```

```
res.cv
```

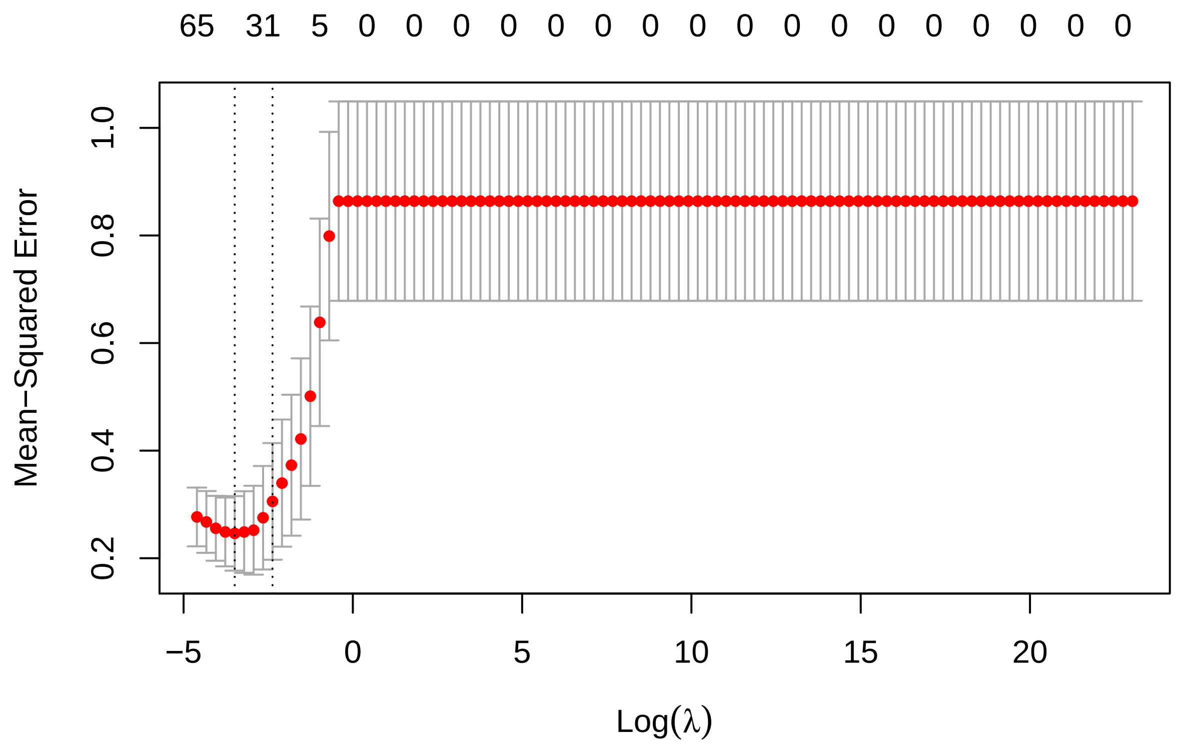
```
##    lambda lasso error lasso lambda ridge error ridge
## 1    0.02310130   0.2283152    6.1359073  0.23587021
## 2    0.04037017   0.5763044    6.1359073  0.55236718
## 3    0.04037017   0.3959881    8.1113083  0.39342692
## 4    0.01000000   0.2913871    1.5199111  0.22009343
## 5    0.01747528   0.1397455    6.1359073  0.07637653
## 6    0.03053856   0.1279141    6.1359073  0.11294895
## 7    0.02310130   0.1406088    0.6579332  0.36587483
## 8    0.02310130   0.3240908    3.5111917  0.31441988
## 9    0.04037017   0.1396851    6.1359073  0.13475449
## 10   0.02310130   0.1864312    3.5111917  0.23379612
```

```
par(mfrow=c(2,2))
hist(res.cv$`error lasso`, main="LASSO error")
hist(res.cv$`lambda lasso`, main="LASSO lambda")
hist(res.cv$`error ridge`, main="RIDGE error")
hist(res.cv$`lambda ridge`, main="RIDGE lamdba")
```



Not only the ridge presents a higher expected error on the test set, but it also has more variability in the lambda values. Therefore we're going to use Lasso, fitting it to the whole dataset.

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



```
best.lambda <- lasso.cv$lambda.min
lasso.fit <- glmnet(x,y, alpha=1, lambda=best.lambda, thresh=1e-12)
sum(coef(lasso.fit)!=0)
```

```
## [1] 41
```

In the final model only 41 of the initial >4000 genes are active.