Linear Model Regularization (Ridge Regression and the Lasso)

Langtao Chen

Initial: Jan 24, 2019 Update: Mar 13, 2021

Contents

Refer to the book chapter 6.6 for the example of ridge regression and the lasso demonstrated below.

1. Data

As an example, we use the Major League Baseball Data (from the 1986 and 1987 seasons) to demonstrate how to conduct ridge regression and the lasso. The response variable is the salary of major league baseball players.

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.3
```

```r
data(Hitters)
str(Hitters)
```

```
## 'data.frame':    322 obs. of  20 variables:
##  $ AtBat    : int  293 315 479 496 321 594 185 298 323 401 ...
##  $ Hits     : int  66 81 130 141 87 169 37 73 81 92 ...
##  $ HmRun    : int  1 7 18 20 10 4 1 0 6 17 ...
##  $ Runs     : int  30 24 66 65 39 74 23 24 26 49 ...
##  $ RBI      : int  29 38 72 78 42 51 8 24 32 66 ...
##  $ Walks    : int  14 39 76 37 30 35 21 7 8 65 ...
##  $ Years    : int  1 14 3 11 2 11 2 3 2 13 ...
##  $ CAtBat   : int  293 3449 1624 5628 396 4408 214 509 341 5206 ...
##  $ CHits    : int  66 835 457 1575 101 1133 42 108 86 1332 ...
##  $ CHmRun   : int  1 69 63 225 12 19 1 0 6 253 ...
##  $ CRuns    : int  30 321 224 828 48 501 30 41 32 784 ...
##  $ CRBI     : int  29 414 266 838 46 336 9 37 34 890 ...
##  $ CWalks   : int  14 375 263 354 33 194 24 12 8 866 ...
##  $ League   : Factor w/ 2 levels "A","N": 1 2 1 2 2 1 2 1 2 1 ...
##  $ Division : Factor w/ 2 levels "E","W": 1 2 2 1 1 2 1 2 2 1 ...
##  $ PutOuts  : int  446 632 880 200 805 282 76 121 143 0 ...
##  $ Assists  : int  33 43 82 11 40 421 127 283 290 0 ...
##  $ Errors   : int  20 10 14 3 4 25 7 9 19 0 ...
##  $ Salary   : num  NA 475 480 500 91.5 750 70 100 75 1100 ...
##  $ NewLeague: Factor w/ 2 levels "A","N": 1 2 1 2 2 1 1 1 2 1 ...
```

```r
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
```

We can see that the Salary column contains 59 missing values. Let's remove rows with missing values.

```r
# Remove missing values
Hitters <- na.omit(Hitters)
```

```r
hist(Hitters$Salary,
     main = 'Histogram of Salary',
     xlab = 'Salary')
```

2. Data Preparation

2.1. Create Input Matrix

We will use the glmnet package for linear model regularization. The glmnet package fits lasso and elastic-net model paths for regression, logistic and multinomial regression using coordinate descent. The algorithm is extremely fast, and exploits sparsity in the input x matrix where it exists.

The glmnet() function needs an x matrix as input and y as a vector. We can use the model.matrix() method to create the input matrix. The model.matrix() method can automatically transform qualitative variables into dummy variables.

```
# Create input matrix, removing the intercept
x <- model.matrix(Salary ~ ., data = Hitters)[,-1]
colnames(x)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "LeagueN"   "DivisionW"
## [16] "PutOuts"   "Assists"   "Errors"    "NewLeagueN"
```

```
y <- Hitters$Salary
```

2.2. Data Split

Let's split the whole dataset into training (50%) and test (50%) sets.

```
set.seed(1)

train <- sample(1:nrow(x),nrow(x)*0.50)

x_train <- x[train, ]
x_test <- x[-train, ]

dim(x_train)
```

```
## [1] 131  19
```

```
dim(x_test)
```

```
## [1] 132  19
```

```
y_train <- y[train]
y_test <- y[-train]
```

3. Ridge Regression

3.1. Fit Ridge Regression on Training Data

The glmnet() function has an alpha argument that determines what type of model is fit. Set alpha = 0 if a ridge regression model is required. Set alpha = 1 if a lasso model is needed.

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.4
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-1
```

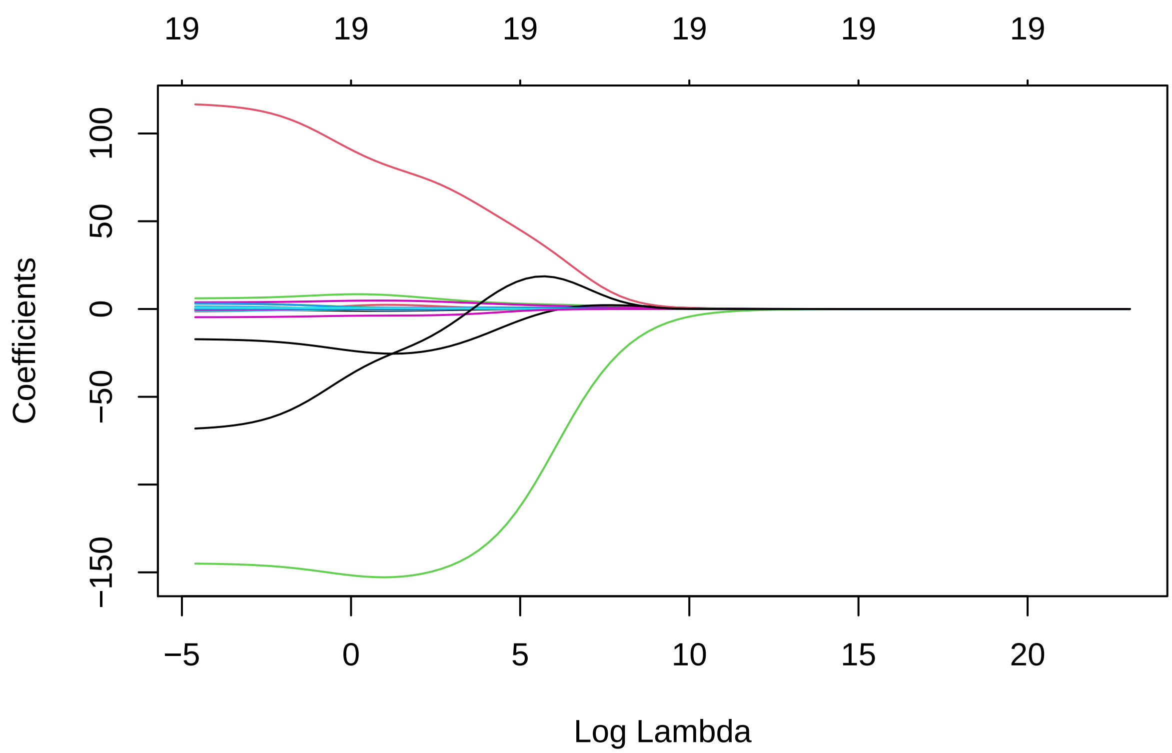
```
# Set the range of lambda as from 10^10 to 10^-2.
grid <- 10^seq(10, -2, length=100)

ridge_mod <- glmnet(x = x_train, y=y_train, alpha = 0, lambda = grid)

dim(coef(ridge_mod))
```

```
## [1]  20 100
```

```r
# Plot the coefficients
plot(ridge_mod, xvar = 'lambda')
```



From the above plot, we can see coefficients can be very close to zero when a large λ is chosen.

The ridge regression minimizes

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

. For example, below are the L2 norm when $\lambda = 11497.57$ and $\lambda = 42.288$.

```r
# lamda = 11497.57
ridge_mod$lambda[50]
```

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

```r
# Coefficients when lambda = 11497.57
coef(ridge_mod)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 413.274464634   0.028460876   0.096884551   0.567798321   0.195484863
##           RBI         Walks         Years        CAtBat         CHits
##   0.205057361   0.289854562   1.038901902   0.003935824   0.014914466
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.120260508   0.029692335   0.031357933   0.038028311   1.295514899
```

```
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
##  -7.832334416   0.011781120   0.002692320  -0.006206931   0.446577683
```

```r
# l2 norm when lambda = 11497.57
sqrt(sum(coef(ridge_mod)[-1,50]^2))
```

```
## [1] 8.05094
```

```r
# lamda = 43.288
ridge_mod$lambda[70]
```

```
## [1] 43.28761
```

```r
# Coefficients when lambda = 43.288
coef(ridge_mod)[,70]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  1.419246e+02 -2.803498e-01  6.837057e-01  4.127127e+00  2.094963e-01
##           RBI         Walks         Years        CAtBat         CHits
##  4.194287e-01  3.324401e+00 -1.581536e+01  3.450917e-03  1.481460e-01
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  9.181912e-01  1.539008e-01  2.922149e-01  1.211254e-01  5.943583e+01
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -1.375331e+02  1.476678e-01  2.946059e-01 -2.567031e+00  2.554818e+00
```

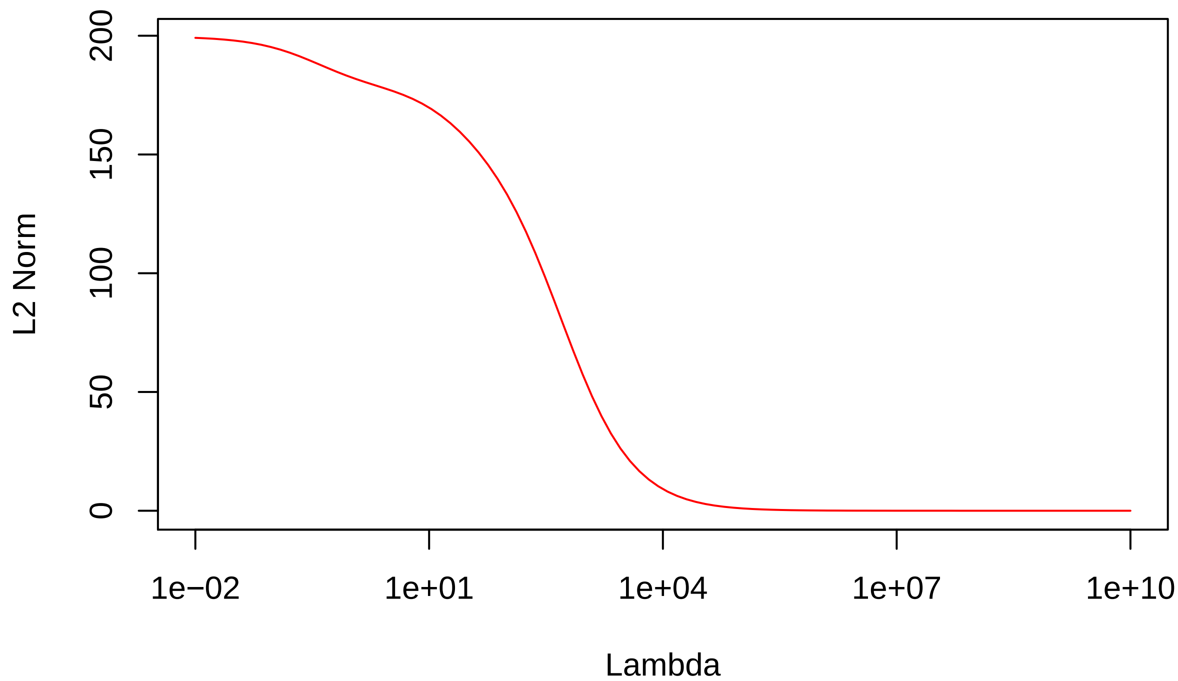
```r
# L2 norm when lambda = 43.288
sqrt(sum(coef(ridge_mod)[-1,70]^2))
```

```
## [1] 150.8017
```

```r
# Calculate all L2
L2 <- NULL

for(i in 1:100){
  L2 <- c(L2, sqrt(sum(coef(ridge_mod)[-1,i]^2)))
}

# Plot the relationship between lambda and L2 norm
plot(x = ridge_mod$lambda, y = L2,
     log = 'x', type = 'l', col = 'red',
     xlab = 'Lambda', ylab = 'L2 Norm')
```



We can see from the above plot that generally when a large λ is used, the coefficient estimates to be much smaller, in terms of L2 norm

$$\sum_{j=1}^{p} \beta_j^2$$

.

3.2. Test Ridge Regression on Test Data

Now we test the performance of ridge regression on the test set, arbitrarily choosing $\lambda = 4$.

```
pred_ridge <- predict(ridge_mod, s = 4, newx = x_test)

library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
postResample(pred = pred_ridge, obs = y_test)
```

```
##        RMSE    Rsquared         MAE
## 377.1292931   0.4500816 276.1626238
```

Calculate the performance of OLS, which is simply ridge regression with $\lambda = 0$.

```
pred_ridge_ols <- predict(ridge_mod, s = 0, newx = x_test,
                          exact = TRUE, x = x_train, y = y_train)

postResample(pred = pred_ridge_ols, obs = y_test)
```

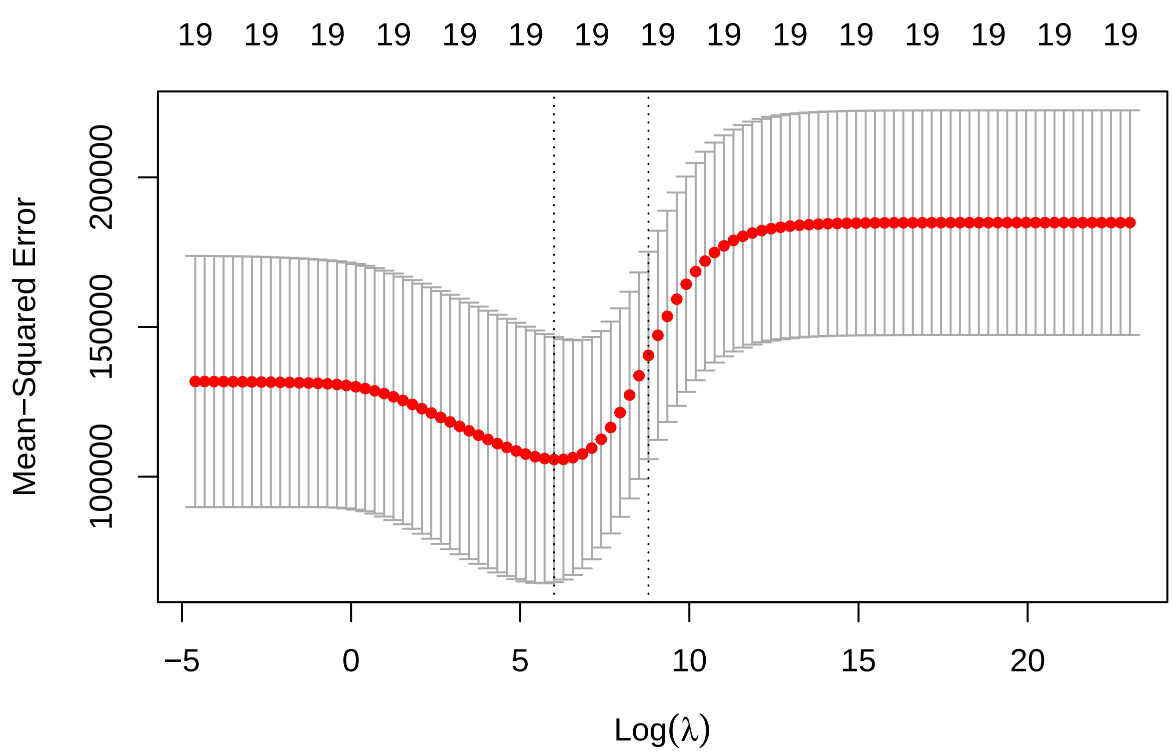
```
##        RMSE    Rsquared         MAE
## 408.6786633   0.4383238 296.2532303
```

We can find that ridge regression has lower RMSE and MAE and higher R^2 than OLS.

3.3. Use K-Fold Cross-Validation to Tune Pamarameter Lambda

In general, it's better to use cross-validation to choose the tuning parameter λ. We can use the built-in cross-validation function cv.glmnet() to fine tune the lambda .

```
set.seed(1)
cv_out <- cv.glmnet(x_train, y_train, alpha = 0, lambda = grid, nfolds = 5)
plot(cv_out)
```




```
best_lambda <- cv_out$lambda.min
best_lambda
```

```
## [1] 403.7017
```

```r
pred_ridge2 <- predict(ridge_mod, s = best_lambda, newx = x_test)

postResample(pred = pred_ridge2, obs = y_test)
```

```
##        RMSE    Rsquared         MAE
## 372.8542334   0.4181121 265.2810532
```

Compared with the ridge regression with an arbitrary $\lambda = 4$, the best parameter λ tuned by the 5-fold cross-validation can further improve the performance of the ridge regression on the test dataset.

Finally, we refit the ridge regression model on the full dataset, using the best value of λ and examine the coefficient estimates.

```r
ridge_full <- glmnet(x, y, alpha = 0)
predict(ridge_full, type="coefficients", s= best_lambda)[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##  22.19435611   0.09207479   0.79700606   0.80005914   1.03400401   0.87729136
##        Walks        Years       CAtBat        CHits       CHmRun        CRuns
##   1.53964596   1.81673011   0.01130983   0.05422674   0.38631591   0.10831375
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##   0.11420837   0.06104007  19.67900736 -72.35198384   0.15301348   0.02456779
##       Errors   NewLeagueN
##  -1.15675717   9.44112309
```

None of the coefficients are zero. That is to say, ridge regression does not perform variable selection!

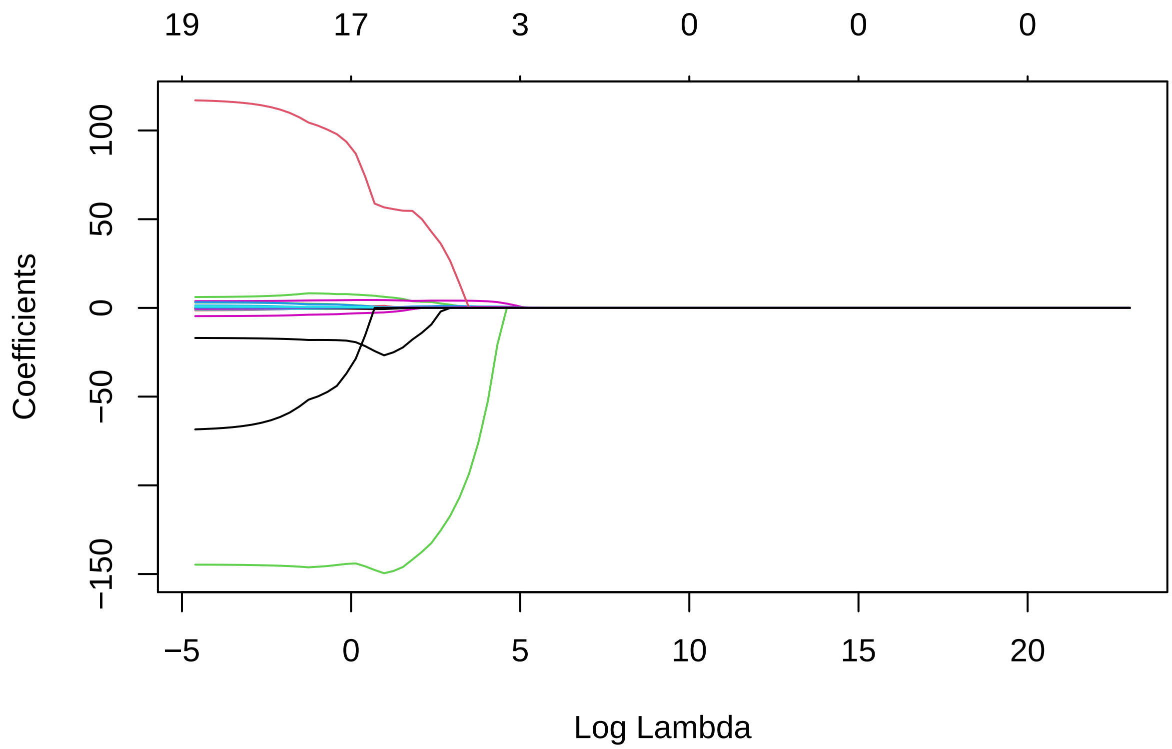
4. The Lasso

4.1. Fit Lasso Model on Training Data

To fit a lasso model, we can simply set the alpha parameter as 1.

```r
# Fit a lasso model on the training dataset
lasso_mod <- glmnet(x = x_train, y = y_train, alpha = 1, lambda = grid)

# Plot the coefficients
plot(lasso_mod, xvar = 'lambda')
```

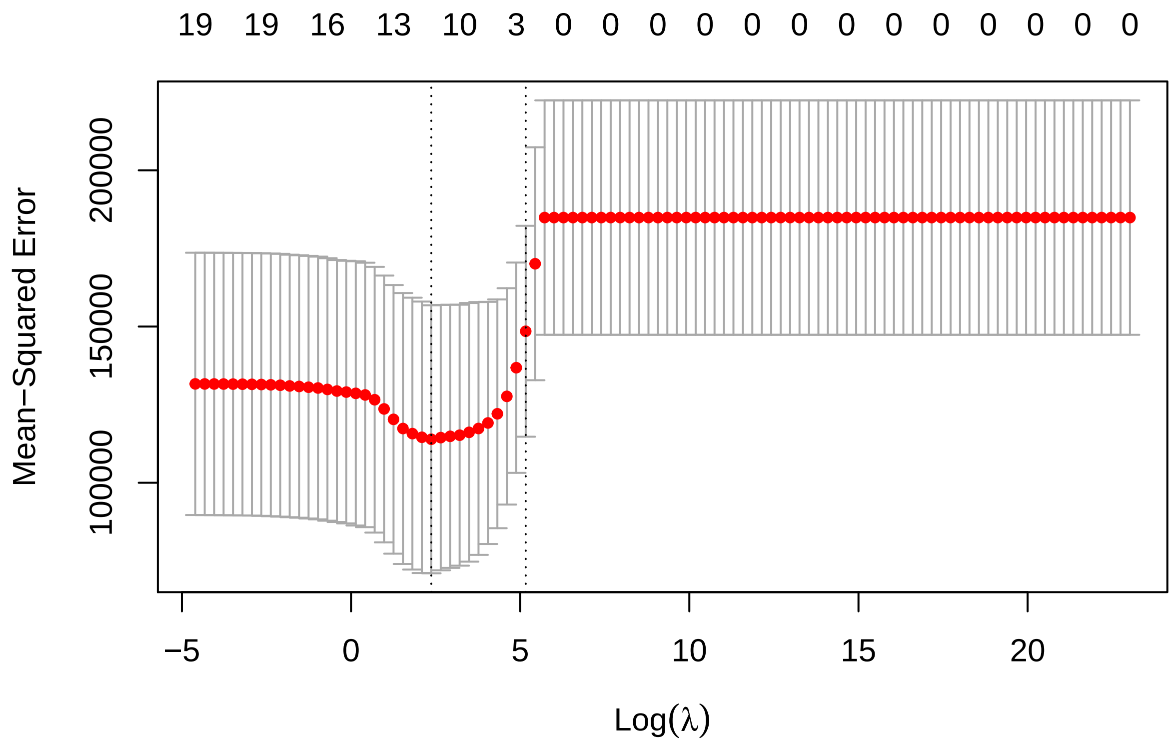


From the above plot, we can see some coefficients can be exactly equal to zero.

4.2. Use K-Fold Cross-Validation to Tune Parameter Lambda

Like what we did in section 3.2, we can use cross-validation to choose the tuning parameter λ.

```
set.seed(1)
cv_out2 <- cv.glmnet(x_train, y_train, alpha = 1, lambda = grid, nfolds = 5)
plot(cv_out2)
```




```
# Get the best lambda tuned by cross-validation
best_lambda2 <- cv_out2$lambda.min
best_lambda2
```

```
## [1] 10.72267
```

```
pred_lasso <- predict(lasso_mod, s = best_lambda2, newx = x_test)

postResample(pred = pred_lasso, obs = y_test)
```

```
##        RMSE    Rsquared         MAE
## 378.8504497   0.4183905 273.5205246
```

The above lasso has a similar performance as the ridge regression with the best lambda paramter.

Finally, we refit the lasso model on the full dataset, using the best value of λ and examine the coefficient estimates.

```
lasso_full <- glmnet(x, y, alpha = 1)
predict(lasso_full, type="coefficients", s= best_lambda2)[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  8.156262e-01  0.000000e+00  1.987578e+00  0.000000e+00  0.000000e+00
##           RBI         Walks         Years        CAtBat         CHits
```

```
##  0.000000e+00  2.264841e+00  0.000000e+00  0.000000e+00  0.000000e+00
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  7.287687e-03  2.105772e-01  4.203213e-01  0.000000e+00  1.659838e+01
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -1.141332e+02  2.340370e-01  0.000000e+00 -6.386012e-01  0.000000e+00
```

An advantage of the lasso over ridge regression is that the lasso can do variable selection. From the above result, we can see that 12 of the 19 coefficients are exactly zero. So the lasso model with the best λ tuned by cross-validation only contains seven variables.