Linear Model Selection and Regularization Exercises

Simone Collier

1 Getting Started

Start by loading the packages that have the data we need. If you need to install the packages first then run `install.packages("PACKAGENAME")` in your console before running the code chunk.

```
library(ISLR2)
library(leaps)
library(glmnet)
```

We will be making use of the `Hitters` data set in the `ISLR` package. It contains baseball player's salaries along with other information about their performance in the previous year.

```
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

We can use `is.na()` to see whether there is any missing data in the set. Using the `sum()` function in combination on the `Salary` column will tell us how many player's salaries we are missing.

```
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

The `na.omit()` function allows us to create a new data set from `Hitters` that removes rows with no entries.

```
Hitters.noNA <- na.omit(Hitters)
```

We want to predict a baseball player's salary based on the other predictors in the data set using linear regression.

2 Subset Selection

2.1 Best Subset Selection

We can use the `regsubsets()` function from the `leaps` package to perform best subset selection. It identifies the best model (smallest RSS) among those with the same number of predictors.

```r
best.subset.fit <- regsubsets(Salary ~ ., data = Hitters.noNA)
summary(best.subset.fit)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters.noNA)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*" 
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*" 
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*" 
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*" 
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*" 
## 6  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "   "*" 
## 7  ( 1 ) " "   "*"  " "   " "  " " "*"   " "   "*"    "*"   "*"    " "   " " 
## 8  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   "*"    "*"   " " 
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "       
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "       
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "       
## 4  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "       
## 5  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "       
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "       
## 7  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "       
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "       
```

The `summary()` function displays the best combination of predictors for models with 1-8 predictors. The asteriks indicate that the variable is included in the model. The `nvmax` argument in the function `regsubsets()` can be increased to return models with more than 8 predictors.

```r
# Perform best subset selection with all predictors (all columns except Salary)
best.subset.fit <- regsubsets(Salary ~ ., data = Hitters.noNA, nvmax = ncol(Hitters) - 1)
summary.best.subset <- summary(best.subset.fit)
```

2.2 Stepwise Selection

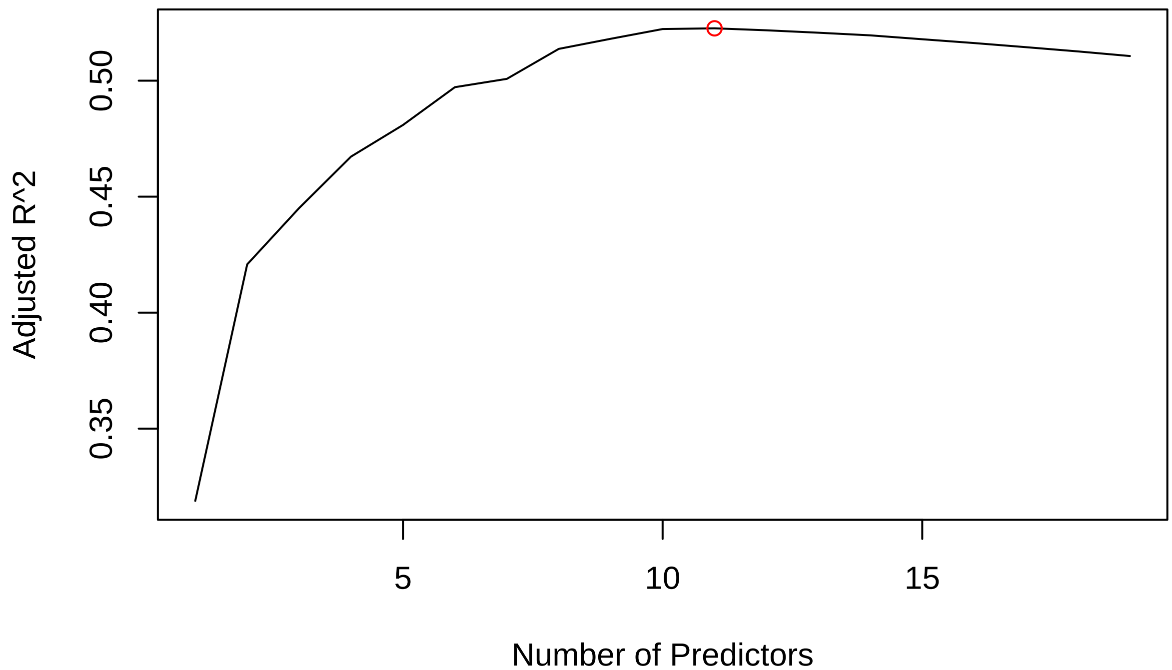
We can use `regsubsets()` to perform forward and backward stepwise selection with the argument `method = "forward` or `method = "backward"`.

```r
fwd.fit <- regsubsets(Salary ~ ., data = Hitters.noNA,
                      nvmax = ncol(Hitters) - 1, method = "forward")
bwd.fit <- regsubsets(Salary ~ ., data = Hitters.noNA,
                      nvmax = ncol(Hitters) - 1, method = "backward")
summary.fwd <- summary(fwd.fit)
summary.bwd <- summary(bwd.fit)
```

2.3 Deciding Between Models using Indirect Error Estimation

The `summary()` function for `regsubsets()` output gives the R^2 (`rsq`), RSS (`rss`), adjusted R^2 (`adjr2`), C_p (`cp`), and BIC (`bic`). We can make a plot of the adjusted R^2 statistic of the best subset selection outputs to help us decide which model to select. We can use the `which.max()` function to find at what index the maximum adjusted R^2 value occurs.

```r
# type = "l" connects the points of the plot with lines
plot(summary.best.subset$adjr2, xlab = "Number of Predictors",
     ylab = "Adjusted R^2", type = "l")
# find the index of the maximum adjusted R^2
max.adjr2 <- which.max(summary.best.subset$adjr2)
# add point at maximum adjusted R^2
points(max.adjr2, summary.best.subset$adjr2[max.adjr2], col = "red")
```



Recall that in the case of the C_p and BIC statistics, the best model will have the smallest C_p or BIC value.

Plot the C_p and BIC statistics and include the minimum points. Based on the three plots, which model size is the best? Justify your answer and list the predictors that are included in the model.

Choose which model size you think is the best for both the forward and backward selection. How do the three models we have chosen compare?

2.4 Deciding Between Models using Direct Error Estimation

2.4.1 Validation Set Approach

We just saw how we can indirectly estimate the test error of our models. We will now look at directly estimate the test error using the validation set approach and cross-validation.

We start by splitting the observations into a training set and a test set. We can do this by creating a random sample of `TRUE` and `FALSE` that is the same size as our data set. `TRUE` will signify that the element is in the training set and `FALSE` will signify that the element is in the test set.

```
set.seed(1)
train <- sample(c(TRUE, FALSE), nrow(Hitters.noNA), replace = TRUE)
test <- (!train)
```

Now we can perform best subset selection on the training set.

```
best.subset.fit2 <- regsubsets(Salary ~ ., data = Hitters.noNA[train, ], nvmax = ncol(Hitters.noNA) - 1)
```

Now that we have our fitted models we need to estimate the test error by predicting the response of the observations in our test set. This is a little tricky since the function `regsubsets()` which we used to fit our models does not work with the `predict()` function we have been using in previous sections. This means we will have to make our own function called `predict.regsubsets()`. Before we make the function let's go through the steps to understand what our function will need to do.

We know that `best.subset.fit2` contains 19 linear regression models. What information do we need to extract from `best.subset.fit2` in order to be able to use each of the 19 models to predict the responses on the test set?... The fitted coefficients! Recall from the linear regression module that responses for linear models are found using:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$$

So we need to write a function that extracts the fitted coefficients for each model size and then multiplies the corresponding predictors for each test observations. Let's write out each step first...

To start let's get the test data on its own. Instead of using a data frame for this we will use a matrix. This will allow us to multiply our fitted coefficients and the test observations in the matrix directly. The `model.matrix()` function is commonly uded for build a predictor matrix from data in a regression context.

```
test.mat <- model.matrix(Salary ~ ., data = Hitters.noNA[test, ])
```

Run `test.mat` see what it looks like.

Now we need to extract the coefficients from each of the 19 fitted models in `best.subset.fit2` and multiply them with the corresponding test predictors. Then we can compute the test error. We can write a `for` loop for this.

```
# Make empty vector to be filled with the computed test errors
val.errors <- numeric()
for (i in 1:19) {
  # Extract the coefficients of the i-th model
  coefs <- coef(best.subset.fit2, id = i)
  # Extract the predictors from the test matrix that are relevant to the i-th model
  obs <- test.mat[, names(coefs)]
  # Predict the response on the test observations
  pred <- obs %*% coefs
  # Compute the test MSE for the i-th model
  mse <- mean((Hitters.noNA$Salary[test] - pred)^2)
  # Add the test error to the list
  val.errors <- c(val.errors, mse)
}

val.errors
```

```
##  [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0 136191.4
##  [9] 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2 141508.2 142164.4
## [17] 141767.4 142339.6 142238.2
```

We can find the model with the smallest validation set error and choose this as our model.

```
which.min(val.errors)
```

```
## [1] 7
```

Now that we have the method layed out, we can write a generic function that we can use to predict responses for models from `regsubsets()`.

```
# object is the `regsubsets()` output (i.e. best.subset.fit2)
predict.regsubsets <- function(object, newdata, id, ...) {
  # get the formula used in `regsubsets()`
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefs <- coef(object, id = id)
  vars <- names(coefs)
  mat[, vars] %*% coefs
}
```

Let's try the function out.

```
predict.regsubsets(best.subset.fit2, Hitters.noNA[test, ], 7)
```

```
##                           [,1]
## -Alvin Davis        727.39222
## -Alfredo Griffin    673.80948
## -Andre Thornton     628.54831
## -Alan Trammell      952.23602
## -Buddy Biancalana    58.65845
## -Bruce Bochy        101.17418
## -Barry Bonds        456.73992
## -Bobby Bonilla      407.28753
## -Billy Hatcher      115.22671
## -Bill Madlock       730.38011
## -Bob Melvin          54.36360
## -BillyJo Robidoux   302.31423
## -Chris Bando        303.06348
## -Chili Davis        794.88821
## -Curt Ford          210.03316
## -Carney Lansford    712.22213
## -Chet Lemon         794.89166
## -Candy Maldonado     66.70965
## -Craig Reynolds     314.13616
## -Cal Ripken         985.00289
## -Cory Snyder        206.10648
## -Chris Speier       396.98031
## -Curt Wilkerson      51.96348
## -Daryl Boston       139.57096
## -Dave Concepcion    702.98418
## -Doug DeCinces      646.27730
## -Damaso Garcia      632.80611
## -Dan Gladden        393.65465
## -Dave Henderson     398.26524
## -Don Mattingly     1350.43868
```

```
## -Dale Murphy        951.93960
## -Dwayne Murphy      531.52823
## -Dave Parker       1126.79635
## -Dan Pasqua         481.40267
## -Darrell Porter     320.23183
## -Don Slaught        251.27139
## -Darryl Strawberry  688.91728
## -Dale Sveum         220.36953
## -Danny Tartabull    400.92118
## -Eddie Milner       375.77744
## -Eddie Murray      1464.82918
## -George Bell        750.67771
## -Glenn Braggs        95.95374
## -George Brett      1322.80475
## -Gary Gaetti        580.35839
## -George Hendrick    809.32355
## -Garth Iorg         336.80881
## -Gary Pettis        582.39550
## -Gary Redus         523.01393
## -Glenn Wilson       562.22245
## -Harold Baines      684.10692
## -Howard Johnson     310.02633
## -Hal McRae          613.96246
## -Harold Reynolds     41.81559
## -John Cangelosi     398.01440
## -Jim Dwyer          312.45512
## -Julio Franco       660.16835
## -Jim Gantner        599.30176
## -Johnny Grubb       549.95017
## -John Kruk          347.76920
## -Jeffrey Leonard    371.20901
## -Jim Morrison       578.52033
## -John Moses         201.13809
## -Johnny Ray         856.90827
## -Jerry Royster      465.45299
## -John Russell       290.42612
## -Jim Sundberg       480.91708
## -Joel Youngblood    308.26960
## -Kal Daniels        190.77503
## -Kirk Gibson        825.60821
## -Ken Griffey       1095.55005
## -Kent Hrbek         917.51361
## -Kevin Mitchell     340.34197
## -Ken Oberkfell      671.24863
## -Ken Phelps         547.38188
## -Kirby Puckett      686.02908
## -Len Dykstra        644.62235
## -Lance Parrish      821.74381
## -Larry Parrish      813.01797
## -Larry Sheets       224.19213
## -Lou Whitaker       912.59432
## -Mike Aldrete       216.38045
## -Mariano Duncan     106.77883
## -Mike Heath         422.13467
```

```
## -Mike Marshall      208.79210
## -Mike Schmidt        55.02747
## -Milt Thompson      266.45271
## -Mitch Webster      642.69038
## -Mookie Wilson      757.21771
## -Marvell Wynne      187.84572
## -Ozzie Guillen       38.78130
## -Pete Incaviglia    276.33167
## -Pete Rose         1563.41556
## -Rafael Belliard    177.67184
## -Randy Bush         320.64594
## -Rick Cerone        381.64192
## -Ron Cey            747.79334
## -Rick Dempsey       527.71147
## -Reggie Jackson    1256.48726
## -Ron Kittle         164.21179
## -Rick Leach         248.33912
## -Rick Manning       584.50829
## -Rance Mulliniks    456.03845
## -Rafael Ramirez     243.50198
## -Rick Schu          246.50534
## -Rob Wilfong        157.03932
## -Robin Yount       1529.43379
## -Scott Bradley      136.45261
## -Sid Bream          790.53638
## -Steve Buechele     109.03581
## -Shawon Dunston     185.99881
## -Scott Fletcher     490.36808
## -Steve Jeltz        382.78993
## -Steve Sax          925.56961
## -Tom Foley          287.71981
## -Tony Gwynn         832.23122
## -Terry Harper       175.26040
## -Tommy Herr         758.38409
## -Tim Hulett         -50.06588
## -Terry Kennedy      120.53833
## -Tito Landrum       194.56136
## -Tom Paciorek       391.51903
## -Terry Pendleton    215.25127
## -Terry Puhl         392.53430
## -Tim Teufel         336.13853
## -Vince Coleman      472.54412
## -Will Clark         459.86191
## -Willie Randolph   1187.96388
## -Wayne Tolleson     390.03303
```

Now that we have decided to go with the 7 variable model we need to refit the model using ALL the observations. We did this previously so we can go ahead and grab the corresponding model.

```
coef(best.subset.fit, id = 7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits       CHmRun
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073    1.4420538
```

```
##    DivisionW      PutOuts
## -129.9866432    0.2366813
```

Note this model is a little different than the one that was fit with only the training observations… It even has different variables.

```
coef(best.subset.fit2, id = 7)
```

```
##  (Intercept)        AtBat         Hits        Walks        CRuns       CWalks
##   67.1085369   -2.1462987    7.0149547    8.0716640    1.2425113   -0.8337844
##    DivisionW      PutOuts
## -118.4364998    0.2526925
```

2.4.2 Cross-validation

Let's try out the cross-validation technique for estimating test error. We will using $k = 10$ folds.

```
k <- 10
n <- nrow(Hitters.noNA)
set.seed(1)
folds <- sample(rep(1:k, length = n))
```

We can make a matrix for storing the computed test errors, one for each fold and model size.

```
cv.errors <- matrix(nrow = k, ncol = 19)
```

Now we can go about computing the errors by first predicting the responses for each fold using each of the 19 models with our function `predict.regsubsets()` and then computing the MSE.

```
for (i in 1:k) {
  # Fit the linear regression by best subset selection using all of the observations
  # except those in the i-th fold
  fold.fit <- regsubsets(Salary ~ ., data = Hitters.noNA[folds != i, ], nvmax = 19)

  for (j in 1:19) {
    # Predict the response for the observations in the i-th fold using the fitted model of size j
    pred <- predict.regsubsets(fold.fit, Hitters.noNA[folds == i, ], id = j)
    # Compute the MSE using the predictions and add it to the matrix
    cv.errors[i, j] <- mean((Hitters.noNA$Salary[folds == i] - pred)^2)
  }
}
```

Now we have the cross-validation error for each fold and each model size. If we take the mean of each column in the matrix, we will have the cross-validation error for each model size.

```
mean.cv.errors <- apply(cv.errors, 2, mean)
mean.cv.errors
```

```
##  [1] 143439.8 126817.0 134214.2 131782.9 130765.6 120382.9 121443.1 114363.7
##  [9] 115163.1 109366.0 112738.5 113616.5 115557.6 115853.3 115630.6 116050.0
## [17] 116117.0 116419.3 116299.1
```

Make a plot of the cross-validation errors and identify the model with the lowest error error with a point. Refit the models with best subset selection using the complete data set. Return the coefficients of the model which was identified as the one with the lowest CV error.

3 Ridge Regression

We will use the glmnet() function from the glmnet package to perform ridge regression. Instead of using a data frame, the glmnet() function requires that the predictors be in the form of a matrix and the response be in the form of a vector.

```
x <- model.matrix(Salary ~., Hitters.noNA)[, -1]
y <- Hitters.noNA$Salary
```

The model.matrix() automatically transforms qualitative variables into dummy variables.

The crucial step for ridge regression and the lasso is selecting the tuning parameter. We will start by fitting with many lambdas (this is the default for the function) and then use cross-validation after the fact to choose the lambda with the smallest MSE. Let's first split our data into a training and testing set so that we can compute the test error after the fact.

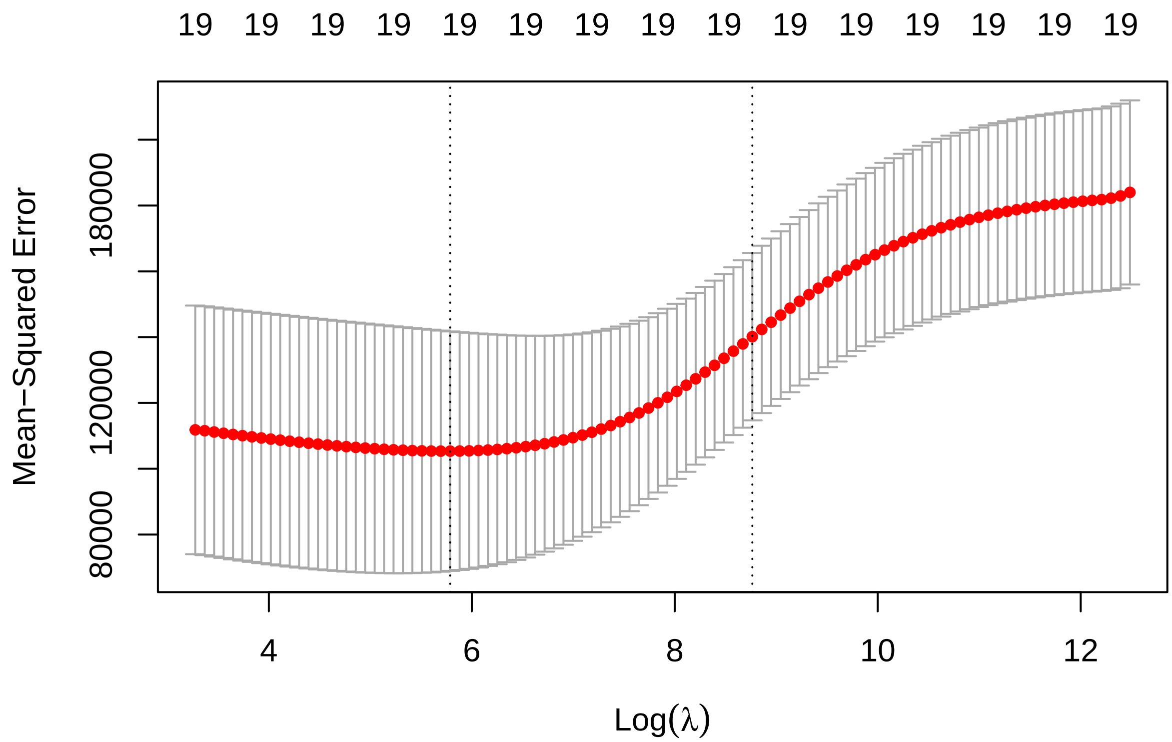
```
set.seed(1)
train <- sample (1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

We choose alpha = 0 in the glmnet() function perform ridge regression. The function standardizes the variables automatically so we need not worry about that.

```
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0)
```

Now we will use the function cv.glmnet(). This function performs cross-validation with a bunch of λ values with a default of nfolds = 10.

```
set.seed(1)
cv.ridge <- cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.ridge)
```



We can find the λ with the smallest cross-validation error and make predictions for the test set using the model in `ridge.mod` with the corresponding λ.

```
best.lambda <- cv.ridge$lambda.min
ridge.pred <- predict(ridge.mod, s = best.lambda, newx = x[test, ])
```

Compute the test MSE.

Now we can refit the ridge regression using the complete data set and return the coefficients.

```
ridge <- glmnet(x, y, alpha = 0)
predict(ridge, type = "coefficients", s = best.lambda)[1:20,]
```

```
##   (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##   15.44383120   0.07715547   0.85911582   0.60103106   1.06369007   0.87936105
##         Walks        Years       CAtBat        CHits       CHmRun        CRuns
##    1.62444617   1.35254778   0.01134999   0.05746654   0.40680157   0.11456224
##          CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##    0.12116504   0.05299202  22.09143197 -79.04032656   0.16619903   0.02941950
##        Errors   NewLeagueN
##   -1.36092945   9.12487765
```

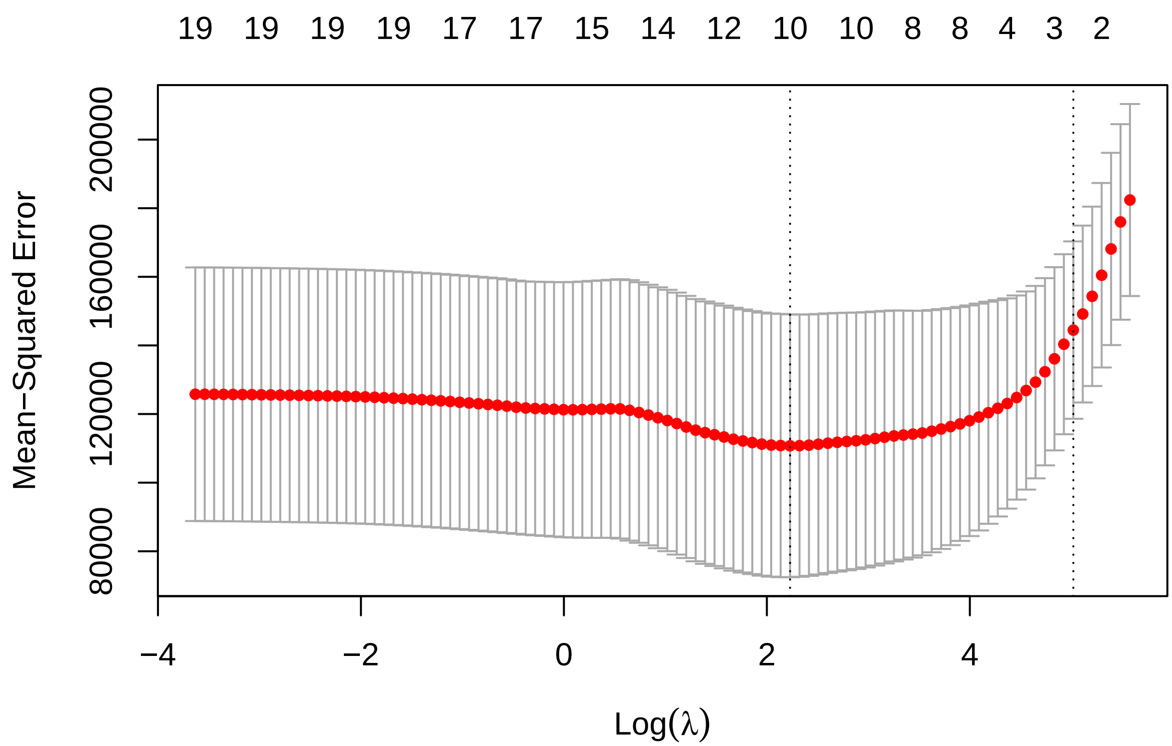
4 The Lasso

We will perform the lasso the exact same way as ridge regression except we have `alpha = 1` in the `glmnet()` function to specify the lasso method is to be used.

```r
lasso.mod <- glmnet(x[train, ], y[train], alpha = 1)
```

Now we can perform cross-validation and compute the MSE.

```r
set.seed(1)
cv.lasso <- cv.glmnet(x[train, ], y[train], alpha = 1)
plot(cv.lasso)
```




```r
best.lambda2 <- cv.lasso$lambda.min
lasso.pred <- predict(lasso.mod, s = best.lambda2, newx = x[test, ])
mse <- mean((lasso.pred - y.test)^2)
mse
```

```
## [1] 143668.8
```

```r
lasso <- glmnet(x, y, alpha = 1)
predict(lasso, type = "coefficients", s = best.lambda2)[1:20, ]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##   -3.04787656    0.00000000    2.02551572    0.00000000    0.00000000
##           RBI         Walks         Years        CAtBat         CHits
##    0.00000000    2.26853781    0.00000000    0.00000000    0.00000000
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
```

```
##    0.01647106     0.21177390     0.41944632     0.00000000    20.48456551
##     DivisionW        PutOuts        Assists         Errors     NewLeagueN
## -116.59062083     0.23718459     0.00000000    -0.94739923     0.00000000
```

The resulting model has 10 coefficients that are exactly equal to zero so it performed variable selection!

Fit a linear regression model to the data. Compute the test MSE for the linear regression model. Compare the results from the linear regression, ridge regression, and the lasso. Which is the best model and why?

These exercises were adapted from : James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R, 2nd ed., Springer, 2021.