Subset Selection, Ridge Regression and LASSO

Here we apply the best subset selection approach to the `Hitters` data. We wish to predict a baseball player's `Salary` on the basis of various statistics associated with performance in the previous year.

First of all, we note that the `Salary` variable is missing for some of the players. The `is.na()` function can be used to identify the missing observations. It returns a vector of the same length as the input vector, with a `TRUE` for any elements that are missing, and a `FALSE` for non-missing elements. The `sum()` function can then be used to count all of the missing elements.

```
library(ISLR2)
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"
```

```
dim(Hitters)
```

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

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

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

The `na.omit()` function removes all of the rows that have missing values in any variable.

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

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

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

```
## [1] 0
```

Best subset selection and stepwise selection

The `regsubsets()` function (part of the `leaps` library) performs best subset selection by identifying the best model that contains a given number of predictors, where best is quantified using RSS. An asterisk indicates that a given variable is included in the corresponding model. For instance, this output indicates that the best two-variable model contains only `Hits` and `CRBI`.

```
library(leaps)
regfit.full <- regsubsets(Salary ~ ., data = Hitters)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters)
## 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 ) "*"    " "     "*"       "*"     " "     " "    " "
```

By default, `regsubsets()` only reports results up to the best eight-variable model. But the `nvmax` option can be used in order to return as many variables as are desired. Here we fit up to a 19-variable model.

```
regfit.full <- regsubsets(Salary ~ ., data = Hitters,
    nvmax = 19)
reg.summary <- summary(regfit.full)
```

The `summary()` function also returns R^2, RSS, adjusted R^2, C_p, and BIC. We can examine these to try to select the best overall model.

```
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

The R^2 statistic increases monotonically as more variables are included:

```
reg.summary$rsq
```

```
##  [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227
```

```
##  [8] 0.5285569 0.5346124 0.5404950 0.5426153 0.5436302 0.5444570 0.5452164
## [15] 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```

The C_p statistics:

```
reg.summary$cp
```

```
##  [1] 104.281319  50.723090  38.693127  27.856220  21.613011  14.023870
##  [7]  13.128474   7.400719   6.158685   5.009317   5.874113   7.330766
## [13]   8.888112  10.481576  12.346193  14.187546  16.087831  18.011425
## [19]  20.000000
```

```
which.min(reg.summary$cp)
```

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

The BIC:

```
reg.summary$bic
```

```
##  [1]  -90.84637 -128.92622 -135.62693 -141.80892 -144.07143 -147.91690
##  [7] -145.25594 -147.61525 -145.44316 -143.21651 -138.86077 -133.87283
## [13] -128.77759 -123.64420 -118.21832 -112.81768 -107.35339 -101.86391
## [19]  -96.30412
```

```
which.min(reg.summary$bic)
```

```
## [1] 6
```

We can also use the `regsubsets()` function to perform forward stepwise or backward stepwise selection, using the argument `method = "forward"` or `method = "backward"`.

```
regfit.fwd <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")
## 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 19
```

```
## Selection Algorithm: forward
##           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 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    "*"   "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```r
regfit.bwd <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19, method = "backward")
summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "backward")
## 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 19
## Selection Algorithm: backward
##           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 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    "*"   "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 5  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 6  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

For this data, the best one-variable through six-variable models are each identical for best subset and forward selection. However, the best seven-variable models identified by forward stepwise selection, backward stepwise selection, and best subset selection are different.

```r
coef(regfit.full, 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
```

```r
coef(regfit.fwd, 7)
```

```
##  (Intercept)        AtBat         Hits        Walks         CRBI       CWalks
##  109.7873062   -1.9588851    7.4498772    4.9131401    0.8537622   -0.3053070
##    DivisionW      PutOuts
## -127.1223928    0.2533404
```

```r
coef(regfit.bwd, 7)
```

```
##  (Intercept)        AtBat         Hits        Walks        CRuns       CWalks
##  105.6487488   -1.9762838    6.7574914    6.0558691    1.1293095   -0.7163346
##    DivisionW      PutOuts
## -116.1692169    0.3028847
```

We just saw that it is possible to choose among a set of models of different sizes using C_p, BIC, and adjusted R^2. We will now consider how to do this using the validation set and cross-validation approaches.

In order to use the validation set approach, we begin by splitting the observations into a training set and a test set.

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

We apply `regsubsets()` to the training set in order to perform best subset selection.

```r
regfit.best <- regsubsets(Salary ~ ., data = Hitters[train, ], nvmax = 19)
```

We now compute the validation set error for the best model of each model size. We first make a model matrix from the test data. The `model.matrix()` function is used in many regression packages for building an "X'' matrix from data.

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

Now we run a loop,and for each size `i`, we extract the coefficients from `regfit.best` for the best model of that size, multiply them into the appropriate columns of the test model matrix to form the predictions, and compute the test MSE.

```r
val.errors <- rep(NA, 19)
for (i in 1:19) {
 coefi <- coef(regfit.best, id = i)
 pred <- test.mat[, names(coefi)] %*% coefi
 val.errors[i] <- mean((Hitters$Salary[test] - pred)^2)
}
```

We find that the best model is the one that contains seven variables.

```
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
```

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

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

```
coef(regfit.best, 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
```

Finally, we perform best subset selection on the full data set, and select the best seven-variable model. Note that we perform best subset selection on the full data set and select the best seven-variable model, rather than simply using the variables that were obtained from the training set, because the best seven-variable model on the full data set may differ from the corresponding model on the training set.

```
regfit.best <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19)
coef(regfit.best, 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
```

We write a function for the `predict()` method for `regsubsets()`.

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

We now try to choose among the models of different sizes using cross-validation. We must perform best subset selection within each of the k training sets. First, we create a vector that allocates each observation to one of $k = 10$ folds, and we create a matrix in which we will store the results.

```
k <- 10
n <- nrow(Hitters)
set.seed(1)
folds <- sample(rep(1:k, length = n))
cv.errors <- matrix(NA, k, 19,)
```

In the jth fold, the elements of `folds` that equal `j` are in the test set, and the remainder are in the training set. We make our predictions for each model size, compute the test errors on the appropriate subset, and store them in the matrix `cv.errors`. This has given us a 10×19 matrix, of which the (j, i)th element corresponds to the test MSE for the jth cross-validation fold for the best i-variable model.

```
for (j in 1:k) {
  best.fit <- regsubsets(Salary ~ ., data = Hitters[folds != j, ], nvmax = 19)
    for (i in 1:19) {
    pred <- predict(best.fit, Hitters[folds == j, ], id = i)
```

```
    cv.errors[j, i] <- mean((Hitters$Salary[folds == j] - pred)^2)
  }
}
```

We use the `apply()` function to average over the columns of this matrix in order to obtain a vector for which the ith element is the cross-validation error for the i-variable model. We see that cross-validation selects a 10-variable model.

```
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
```

We now perform best subset selection on the full data set in order to obtain the 10-variable model.

```
reg.best <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19)
coef(reg.best, 10)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  162.5354420   -2.1686501    6.9180175    5.7732246   -0.1300798    1.4082490
##         CRBI       CWalks    DivisionW      PutOuts      Assists
##    0.7743122   -0.8308264 -112.3800575    0.2973726    0.2831680
```

Ridge regression and the LASSO

We will use the function `glmnet()` in the `glmnet` package. We will now perform ridge regression and the lasso in order to predict `Salary` on the `Hitters` data. The `model.matrix()` function is useful for creating `x`; not only does it produce a matrix corresponding to the 19 predictors but it also automatically transforms any qualitative variables into dummy variables. `glmnet()` can only take numerical, quantitative inputs. `[, -1]` removes the intercept.

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

The `glmnet()` function has an `alpha` argument that determines what type of model is fit. If `alpha=0` then a ridge regression model is fit, and if `alpha=1` then a LASSO model is fit.We implement the function over a grid: $\lambda = 10^{10}$ to $\lambda = 10^{-2}$, covering the null model containing only the intercept, to the least squares fit. By default, the `glmnet()` function standardizes the variables so that they are on the same scale.

```
library(glmnet)
```

```
##     Matrix
```

```
## Loaded glmnet 4.1-2
```

```
grid <- 10^seq(10, -2, length = 100)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

Associated with each value of λ is a vector of ridge regression coefficients, stored in a matrix that can be accessed by `coef()`. In this case, it is a 20×100 matrix, with 20 rows (one for each predictor, plus an intercept) and 100 columns (one for each value of λ).

```
dim(coef(ridge.mod))
```

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

We can use the `predict()` function for a number of purposes. For instance, we can obtain the ridge regression coefficients for a new value of λ, say 50:

```
predict(ridge.mod, s = 50, type = "coefficients")[1:20, ]
```

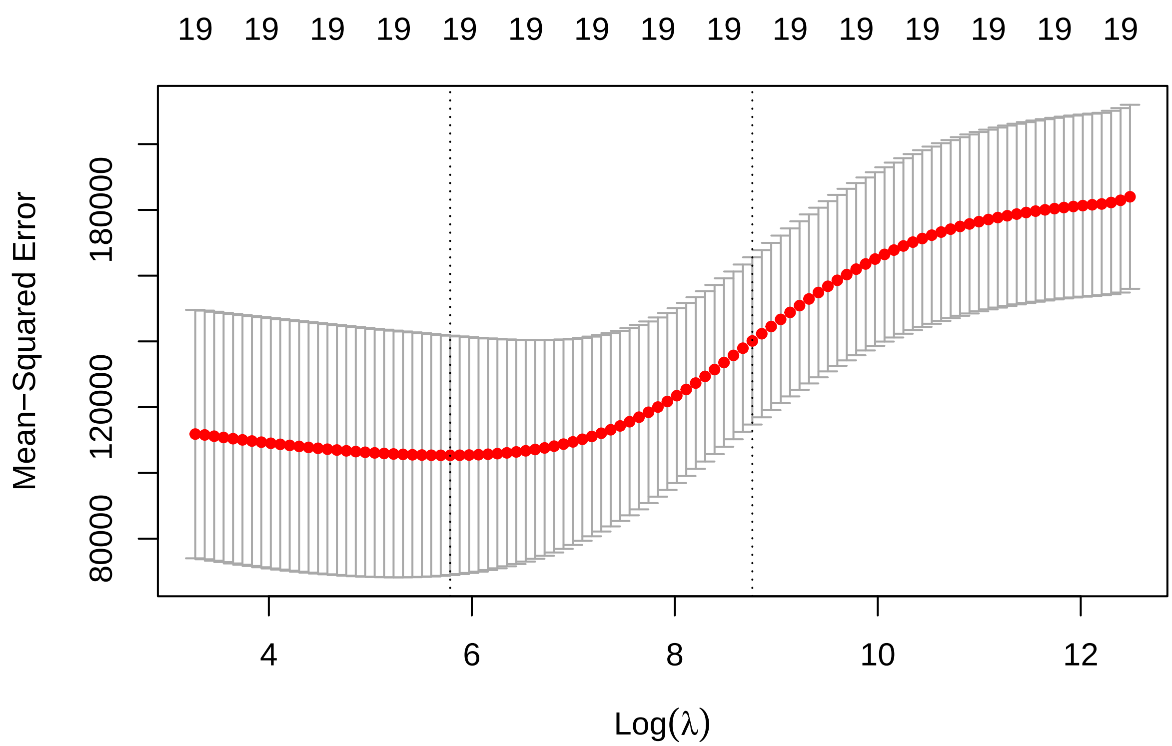
```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  4.876610e+01 -3.580999e-01  1.969359e+00 -1.278248e+00  1.145892e+00
##           RBI         Walks         Years        CAtBat         CHits
##  8.038292e-01  2.716186e+00 -6.218319e+00  5.447837e-03  1.064895e-01
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  6.244860e-01  2.214985e-01  2.186914e-01 -1.500245e-01  4.592589e+01
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -1.182011e+02  2.502322e-01  1.215665e-01 -3.278600e+00 -9.496680e+00
```

We now split the samples into a training set and a test set in order to estimate the test error of ridge regression and the LASSO We randomly choose a subset of numbers between 1 and n as the indices for the training observations.

```
set.seed(1)
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0, lambda = grid)
```

We use cross-validation to choose the tuning parameter λ. We can do this using the built-in cross-validation function, `cv.glmnet()`. By default, the function performs ten-fold cross-validation, though this can be changed using the argument `nfolds`.

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



```r
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 326.0828
```

The test MSE associated with this value of λ:

```r
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test, ])
mean((ridge.pred - y.test)^2)
```

```
## [1] 139833.6
```

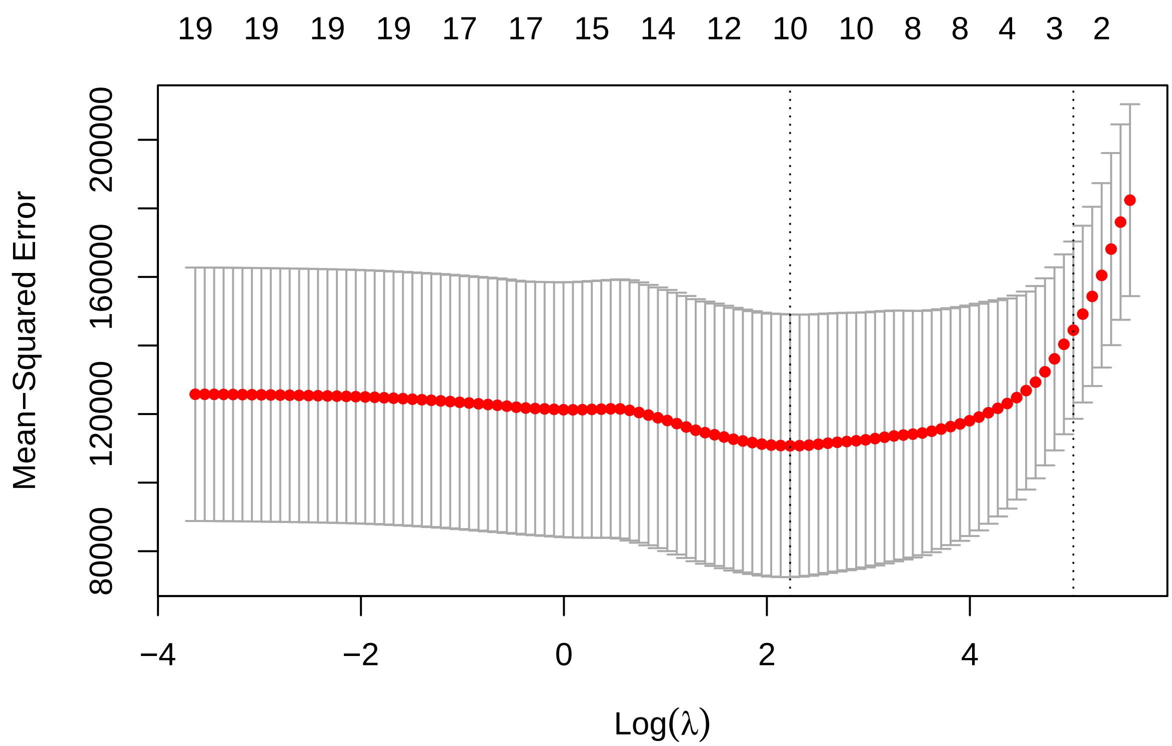
We examine the coefficient estimates. None of the coefficients are zero: ridge regression does not perform variable selection.

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

```
##  (Intercept)        AtBat         Hits        HmRun         Runs          RBI 
##  15.44383135   0.07715547   0.85911581   0.60103107   1.06369007   0.87936105 
##        Walks        Years       CAtBat        CHits       CHmRun        CRuns 
##   1.62444616   1.35254780   0.01134999   0.05746654   0.40680157   0.11456224 
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists 
##   0.12116504   0.05299202  22.09143189 -79.04032637   0.16619903   0.02941950 
##       Errors   NewLeagueN 
##  -1.36092945   9.12487767
```

In order to fit a LASSO model, we once again use the `glmnet()` function with `alpha=1`. We perform cross-validation and compute the associated test error.

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



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

```
## [1] 143673.6
```

However, the lasso has a substantial advantage over ridge regression in that the resulting coefficient estimates are sparse.

```r
out <- glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:20, ]
lasso.coef
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##    1.27479059   -0.05497143    2.18034583    0.00000000    0.00000000
##           RBI         Walks         Years        CAtBat         CHits
##    0.00000000    2.29192406   -0.33806109    0.00000000    0.00000000
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##    0.02825013    0.21628385    0.41712537    0.00000000   20.28615023
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -116.16755870    0.23752385    0.00000000   -0.85629148    0.00000000
```