6.5 Lab 1: Subset Selection Methods

- 6.5.1 Best Subset Selection
- 6.5.2 Forward and Backward Stepwise Selection
- 6.5.3 Choosing Among Models Using the Validation Set Approach and Cross-Validation

6.5.1 Best Subset Selection

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(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.4
```

```
fix(Hitters)
names(Hitters)
```

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

```
dim(Hitters)
```

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

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

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

Hence we see that **Salary** is missing for 59 players. 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
```

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. The syntax is the same as for **lm()**. The **summary()** command outputs the best set of variables for each model size.

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.4.4
```

```
regfit.full = regsubsets(Salary ~ ., Hitters)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., 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
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "  
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
## 6  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "  
## 7  ( 1 ) " "   "*"  " "   " "  " " "*"   " "   "*"    "*"   "*"    " "  
## 8  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   "*"    "*"  
##          CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "       
## 2  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "       
## 3  ( 1 ) "*"  " "    " "     " "       "*"     " "     " "    " "       
## 4  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "       
## 5  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "       
## 6  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "       
## 7  ( 1 ) " "  " "    " "     "*"       "*"     " "     " "    " "       
## 8  ( 1 ) " "  "*"    " "     "*"       "*"     " "     " "    " "       
```

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**. 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"
```

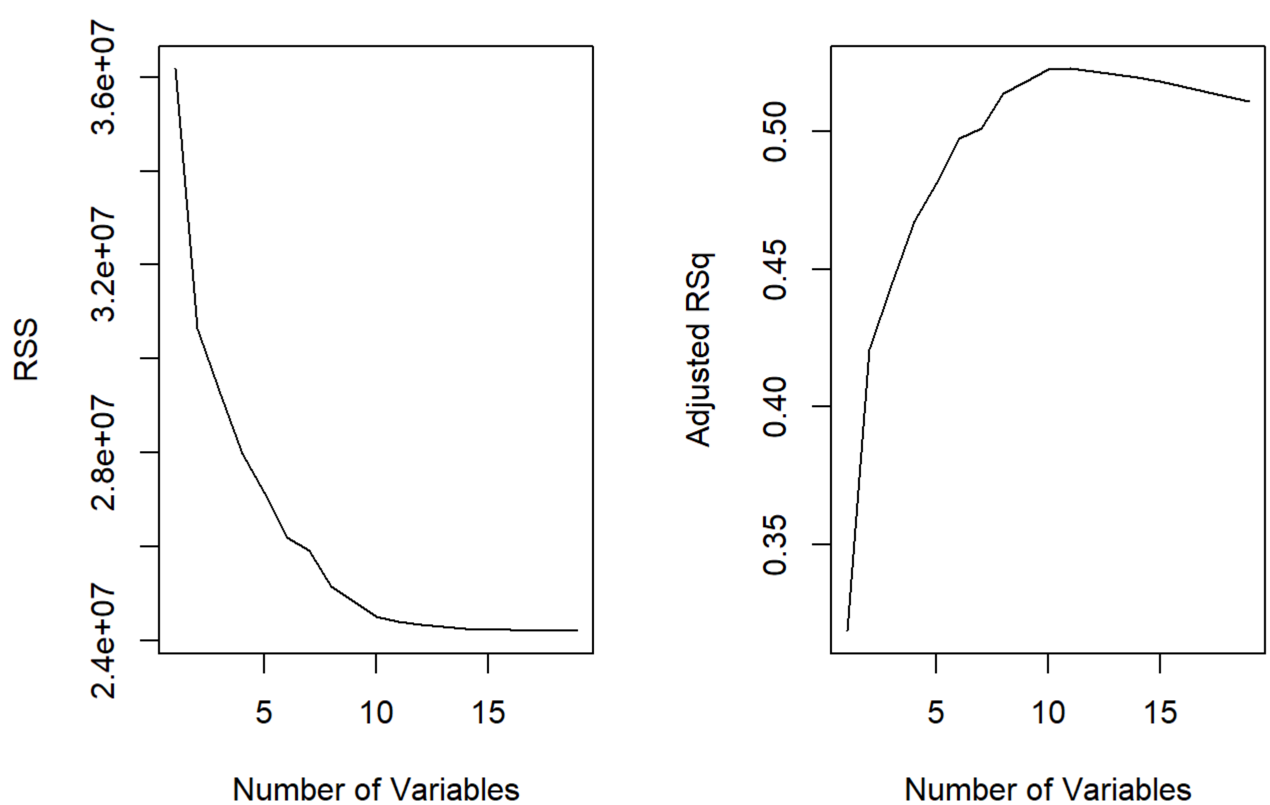
For instance, we see that the R^2 statistic increases from 32%, when only one variable is included in the model, to almost 55%, when all variables are included. As expected, the R^2 statistic increases monotonically as more variables are included.

```
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
```

Plotting RSS, adjusted R^2, C_p, and BIC for all of the models at once will help us decided which model to select. Note the **type="1"** options tells R to connect the plotted points with lines.

```
par(mfrow = c(1,2))
plot(reg.summary$rss, xlab = 'Number of Variables', ylab = 'RSS', type = 'l')
plot(reg.summary$adjr2, xlab = 'Number of Variables', ylab = 'Adjusted RSq', type = 'l')
```

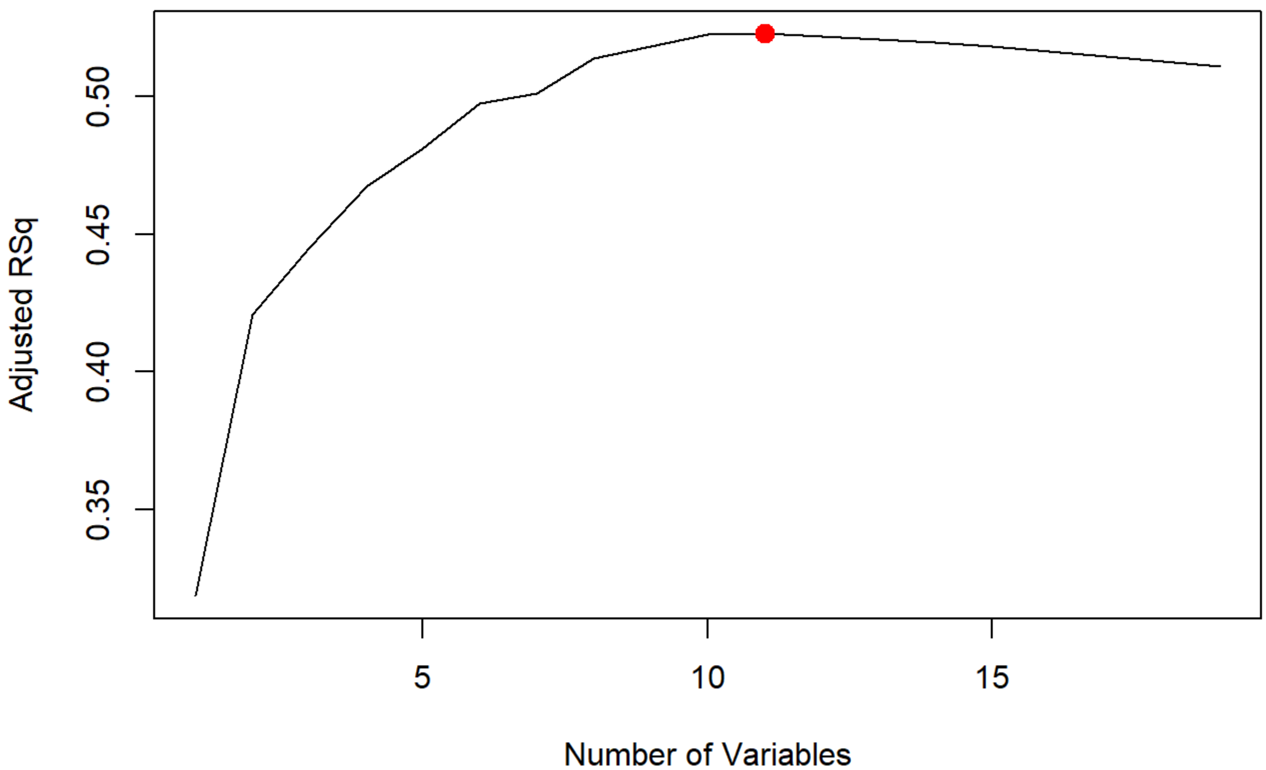


The **points()** command works like the **plot()** command, except that it puts points on a plot that has already been created, instead of creating a new plot. The **which.max()** function can be used to identify the location of the maximum point of a vector. We will now plot a red dot to indicate the model with the largest adjusted R^2 statistic.

```
which.max(reg.summary$adjr2)
```

```
## [1] 11
```

```
plot(reg.summary$adjr2, xlab = 'Number of Variables', ylab = 'Adjusted RSq', type = 'l')
points(11, reg.summary$adjr2[11], col = 'red', cex = 2, pch = 20)
```

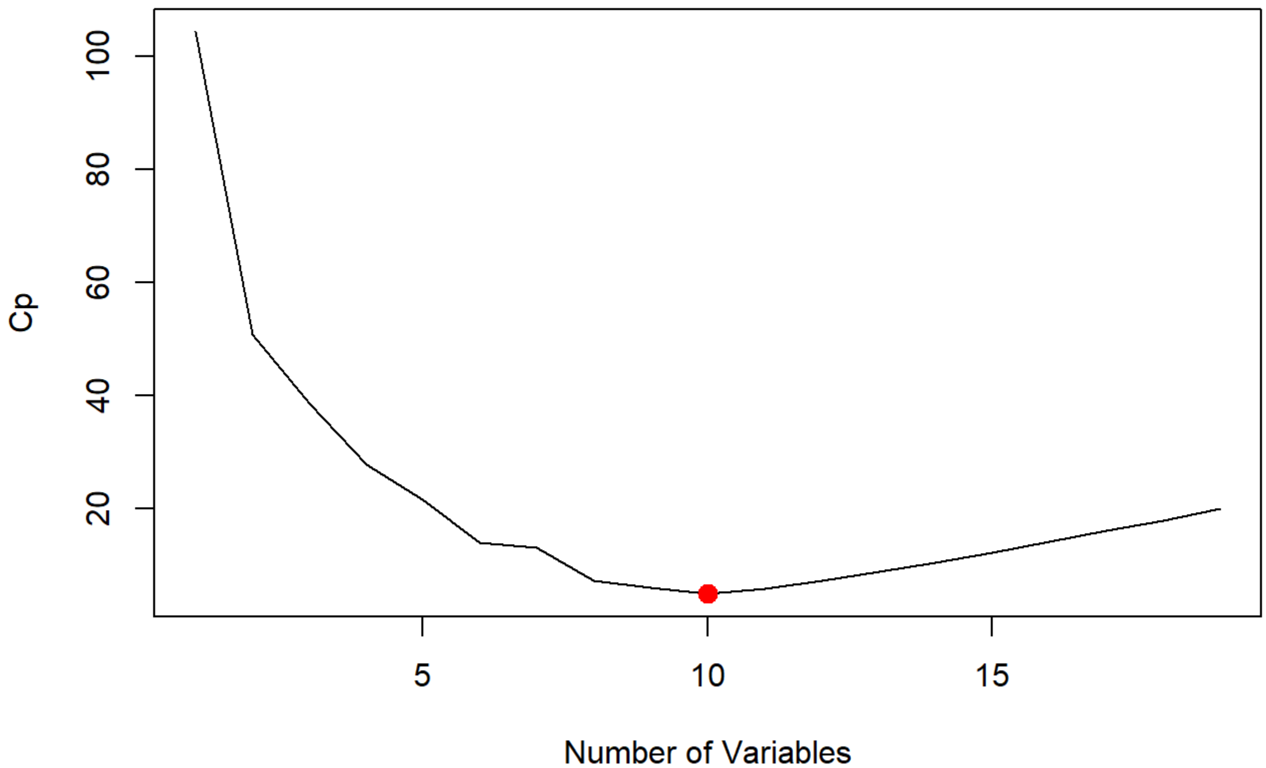


In a similar fashion we can plot the C_p and BIC statistics, and indicate the models with the smallest statistic using **which.min()**.

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

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

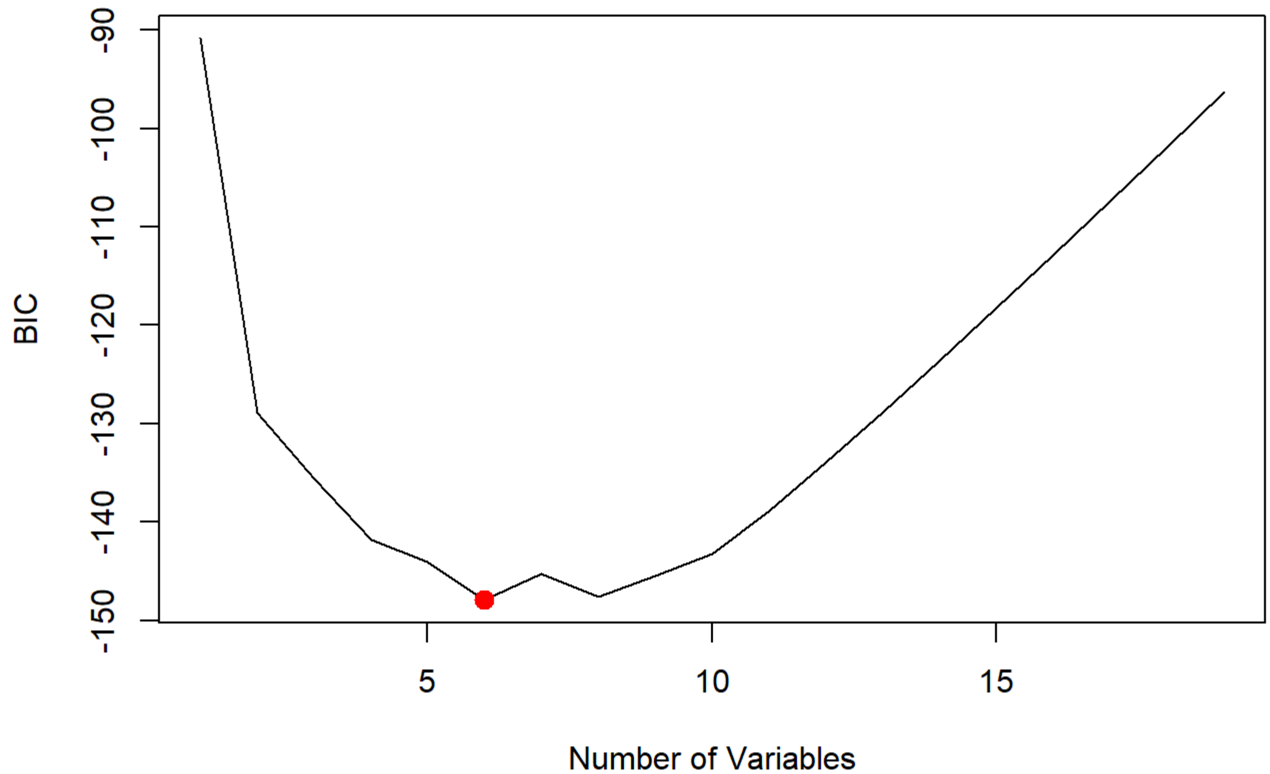
```
plot(reg.summary$cp, xlab = 'Number of Variables', ylab = 'Cp', type = 'l')
points(10, reg.summary$cp[10], col = 'red', cex = 2, pch = 20)
```




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

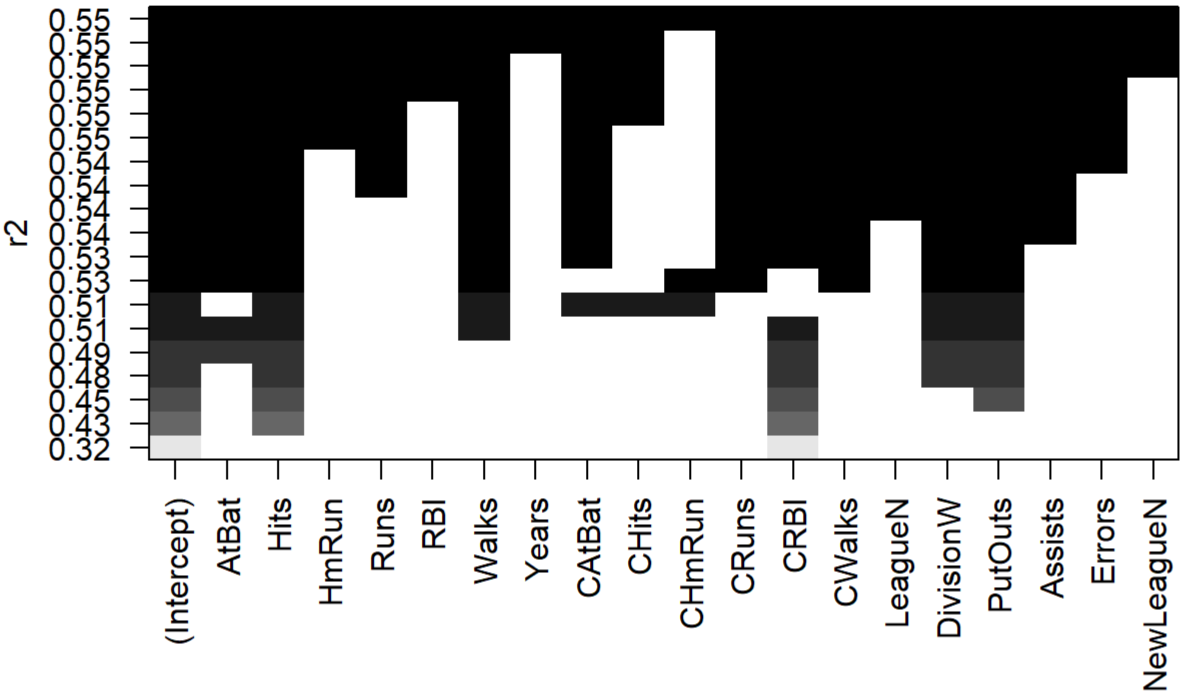
```
## [1] 6
```

```
plot(reg.summary$bic, xlab = 'Number of Variables', ylab = 'BIC', type = 'l')
points(6, reg.summary$bic[6], col = 'red', cex = 2, pch = 20)
```

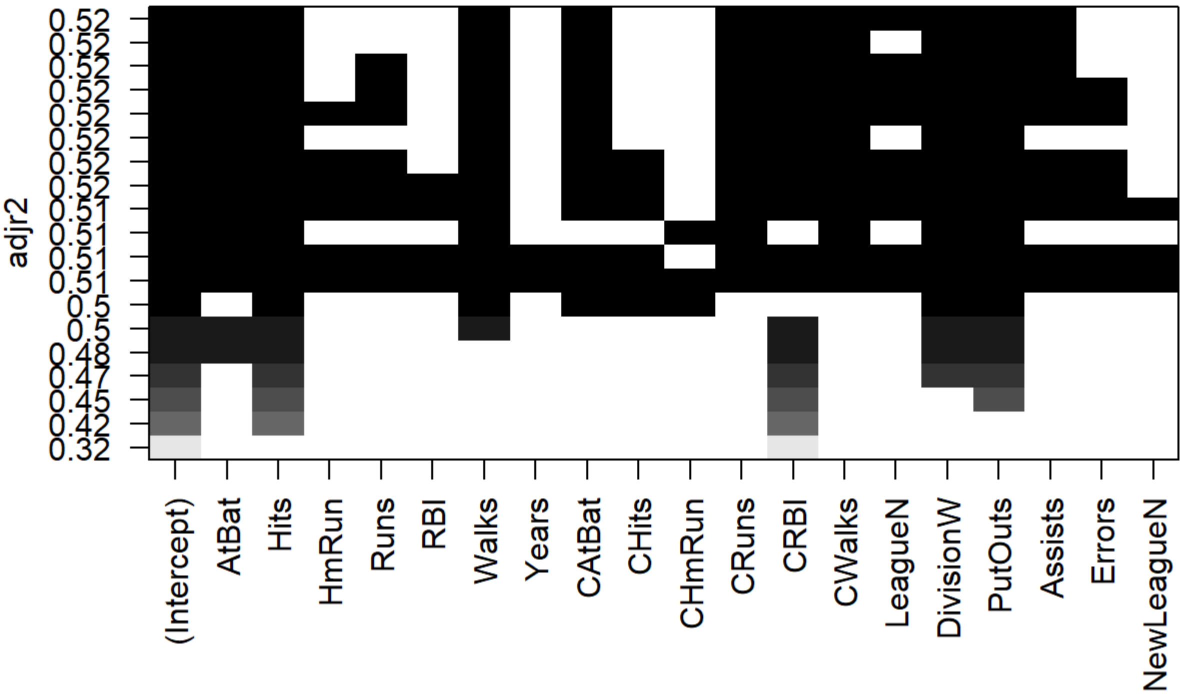


The **regsubsets()** function has a built-in **plot()** command which can be used to display the selected variables for the best model with a given number of predictors, ranked according to the BIC, C_p, adjusted R^2, or AIC. To find out more about this function, type **?plot.regsubsets**.

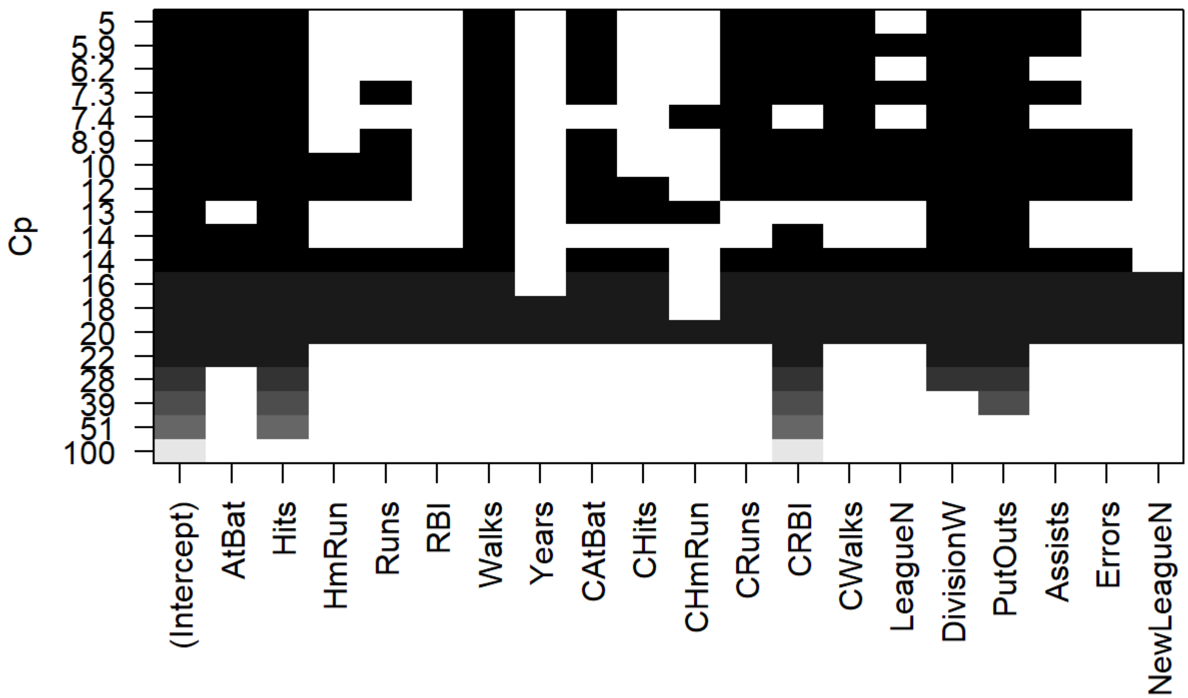
```
plot(regfit.full, scale = 'r2')
```



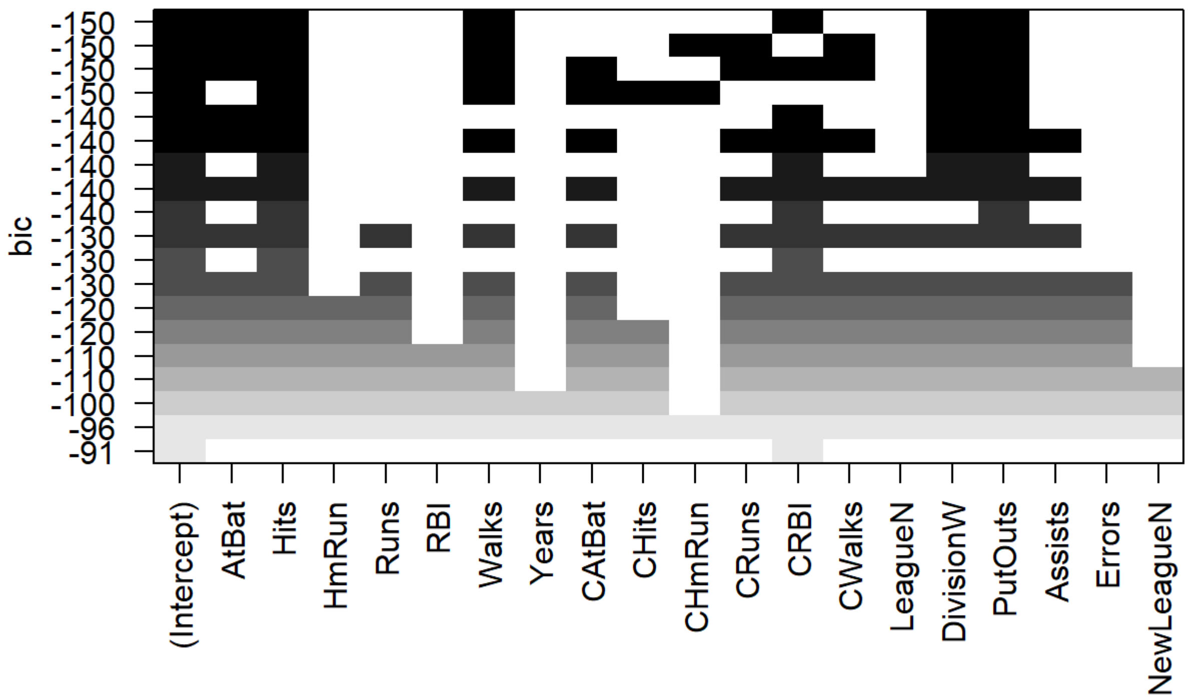

```
plot(regfit.full, scale = 'adjr2')
```




```
plot(regfit.full, scale = 'Cp')
```




```
plot(regfit.full, scale = 'bic')
```



The top row of each plot contains a black square for each variable selected according to the optimal model associated with that statistic. For instance, we see that several models share a BIC close to -150. However, the model with the lowest BIC is the six-variable model that contains only **AtBat**, **Hits**, **Walks**, **CRBI**, **DivisionW**, and **PutOuts**. We can use the **coef()** function to see the coefficient estimates associated with this model.

```
coef(regfit.full, 6)
```

```
##  (Intercept)        AtBat         Hits        Walks         CRBI
##   91.5117981   -1.8685892    7.6043976    3.6976468    0.6430169
##    DivisionW      PutOuts
## -122.9515338    0.2643076
```

6.5.2 Forward and Backward Stepwise Selection

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
## 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 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"  
##           CRBI 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 ) "*"  "*"    "*"      "*"        "*"      "*"      "*"     "*"
```

```
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
## 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 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"  
##           CRBI 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 instance, we see that using forward stepwise selection, the best one-variable model contains only **CRBI**, and the best two-variable model additionally includes **Hits**. For this data, the best one-variable through six-variable models are each identical for best subset and forward selection. However,k the best seven-variable models identified by forward stepwise selection, backward stepwise selection, and best subset selection are different.

```
coef(regfit.full, 7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073
##        CHmRun    DivisionW      PutOuts
##    1.4420538 -129.9866432    0.2366813
```

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

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

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

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

6.5.3 Choosing Among Models Using the Validation Set Approach and Cross-Validation

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 for these approaches to yield accurate estimates of the test error, we must use *only the training observations* to perform all aspects of model-fitting –including variables selection. Therefore, the determination of which model of a given size is best must be made using *only the training observations*. This point is subtle but important. If the full data set is used to perform the best subset selection step, the validation set errors and cross-validation errors that we obtain will not be accurate estimates of the test error.

In order to use the validation set approach, we begin by splitting the observations into a training set and a test set. We do this by creating a random vector, **train**, of elements equal to **TRUE** if the corresponding observation is in the training set, and **FALSE** otherwise. The vector **test** has a **TRUE** if the observation is in the test set, and a **FALSE** otherwise. Note the **!** in the command to create **test** causes **TRUE**s to be switched to **FALSE**s and vice versa. We also set a random seed so that the user will obtain the same training set/teset set split.

```
set.seed(1)
train = sample(c(TRUE,FALSE), nrow(Hitters), rep = TRUE)
test = !train
```

Now, we apply **regsubsets()** to the training set in order to perform best subset selection.

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

Notice that we subset the **Hitters** data frame directly in the call in order to access only the training subset of the data, using the expression **Hitters[train,]**. 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.

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

The **model.matrix()** function is used in many regression packages for building an “X” matrix from data. 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.

```
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 ten variables.

```
val.errors
```

```
##  [1] 220968.0 169157.1 178518.2 163426.1 168418.1 171270.6 162377.1
##  [8] 157909.3 154055.7 148162.1 151156.4 151742.5 152214.5 157358.7
## [15] 158541.4 158743.3 159972.7 159859.8 160105.6
```

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

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

```
coef(regfit.best, 10)
```

```
## (Intercept)       AtBat        Hits       Walks      CAtBat       CHits
## -80.2751499  -1.4683816   7.1625314   3.6430345  -0.1855698   1.1053238
##      CHmRun      CWalks     LeagueN   DivisionW     PutOuts
##   1.3844863  -0.7483170  84.5576103 -53.0289658   0.2381662
```

This was a little tedious, partly because there is no **predict9)** method for **regsubsets()**. Since we will be using this function again, we can capture our steps above and write our own predict method.

```
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
}
```

Our function pretty much mimics what we did above. The only complex part is how we extracted the formula used in the call to **regsubsets()**. We demonstrate how we use this function below, when we do cross-validation.

Finally, we perform best subset selection on the full data set, and select the best ten-variable model. It is important that we make use of the full data set in order to obtain more accurate coefficient estimates. Note that we perform best subset selection on the full data set and select the best ten-variable model, rather than simply using the variables that were obtained from the training set, because the best ten-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, 10)
```

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

In fact, we see that the best ten-variable model on the full data set has a different set of variables than the best ten-variable model on the training set.

We now try to choose among the models of different sizes using cross-validation. This approach is somewhat involved, as we must perform best subset selection *within each of the k training sets*. Despite this, we see that with its clever subsetting syntax, R makes this job quite easy. 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
set.seed(1)
folds = sample(1:k, nrow(Hitters), replace = TRUE)
cv.errors = matrix(NA,k,19, dimnames = list(NULL, paste(1:19)))
```

Now we write a loop that performs cross-validation. In the *j*th 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 (using our new **predict()** method), compute the test errors on the appropriate subset, and store them in the appropriate slot in the matrix **cv.errors**.

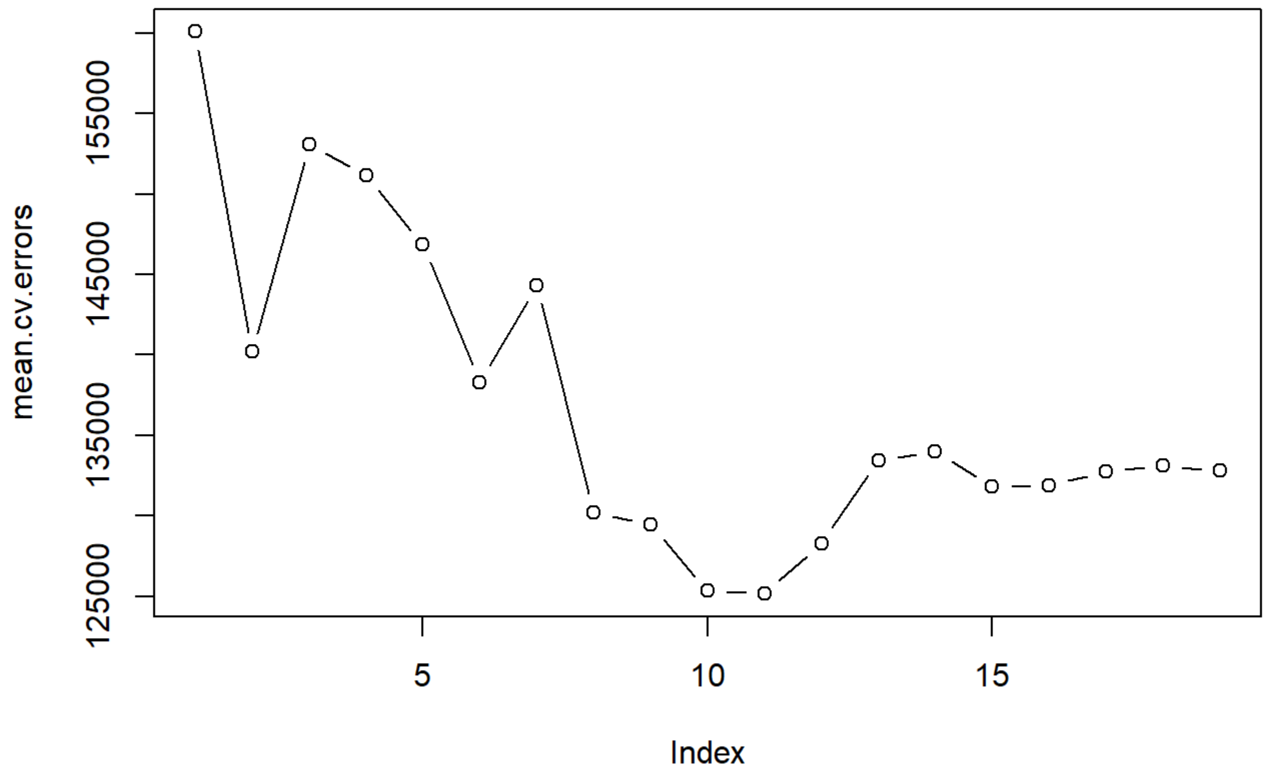
```
for (j in 1:k) {
  best.fit = regsubsets(Salary ~ ., data = Hitters[folds != j,], nvmax = 19)
  for (i in 1:19) {
    pred = predict(best.fit, Hitters[folds == j,], id = i)
    cv.errors[j, i] = mean((Hitters$Salary[folds == j] - pred)^2)
  }
}
```

This has given us a 10x19 matrix, of which the (*i*, *j*)th element corresponds to the test MSE for the *i*th cross-validation fold for the best *j*-variable model. We use the **apply()** function to average over the columns of this matrix in order to obtain a vector for which the *j*th element is the cross-validation error for the *j*-variable model.

```
mean.cv.errors = apply(cv.errors, 2, mean)
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 160093.5 140196.8 153117.0 151159.3 146841.3 138302.6 144346.2 130207.7
##        9       10       11       12       13       14       15       16
## 129459.6 125334.7 125153.8 128273.5 133461.0 133974.6 131825.7 131882.8
##       17       18       19
## 132750.9 133096.2 132804.7
```

```
par(mfrow = c(1,1))
plot(mean.cv.errors, type = 'b')
```



We see that cross-validation select an 11-variable model. We now perform best subset selection on the full data set in order to obtain the 11-variable model.

```
reg.best = regsubsets(Salary ~ ., data = Hitters, nvmax = 19)
coef(reg.best, 11)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat
##  135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914
##        CRuns         CRBI       CWalks      LeagueN    DivisionW
##    1.4553310    0.7852528   -0.8228559   43.1116152 -111.1460252
##      PutOuts      Assists
##    0.2894087    0.2688277
```