Subset Selection Methods

Laura Cline

24/10/2021

Contents


```
library(ISLR)
library(leaps)
```

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 the missing elements.

```
# Load the hitters data and do a quick inspection of it
colnames(Hitters) <- tolower(colnames(Hitters))
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
```

```
# Salary is missing for some observations, so find our how many and drop them
sum(is.na(Hitters$salary))
```

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

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

Hence, we see that `Salary` is missing for 59 players. The `na.omit()` function removes all the rows that have missing values in any variables.

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.

```
# Using the regsubsets() function, perform best subset selection
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 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 ) "*"    " "     "*"       "*"     " "     " "    " "
```

An asterisk indicates that a given variable is included in then corresponding model. For instance, the best model contains three variables will have `hits`, `crbi` and `putouts`.

As asterisk indicates that a given variable is included in the corresponding model. For instance, this output indicates that the best two-variable model contains `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.

```
# By default, we only see the first 8 models, but with the nvmax option the function can return results
regfit.fuller <- regsubsets(salary ~ ., Hitters, nvmax=19)
reg.summary <- summary(regfit.fuller)
```

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.

```
# summary() will also return R^2, RSS, adj R^2, C_p and BIC of the model; let's check those out
names(reg.summary)
```

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

```
reg.summary$rsq # Observe that monotonic relationship between R^2 and the number of included variables
```

```
##  [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
```

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.

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

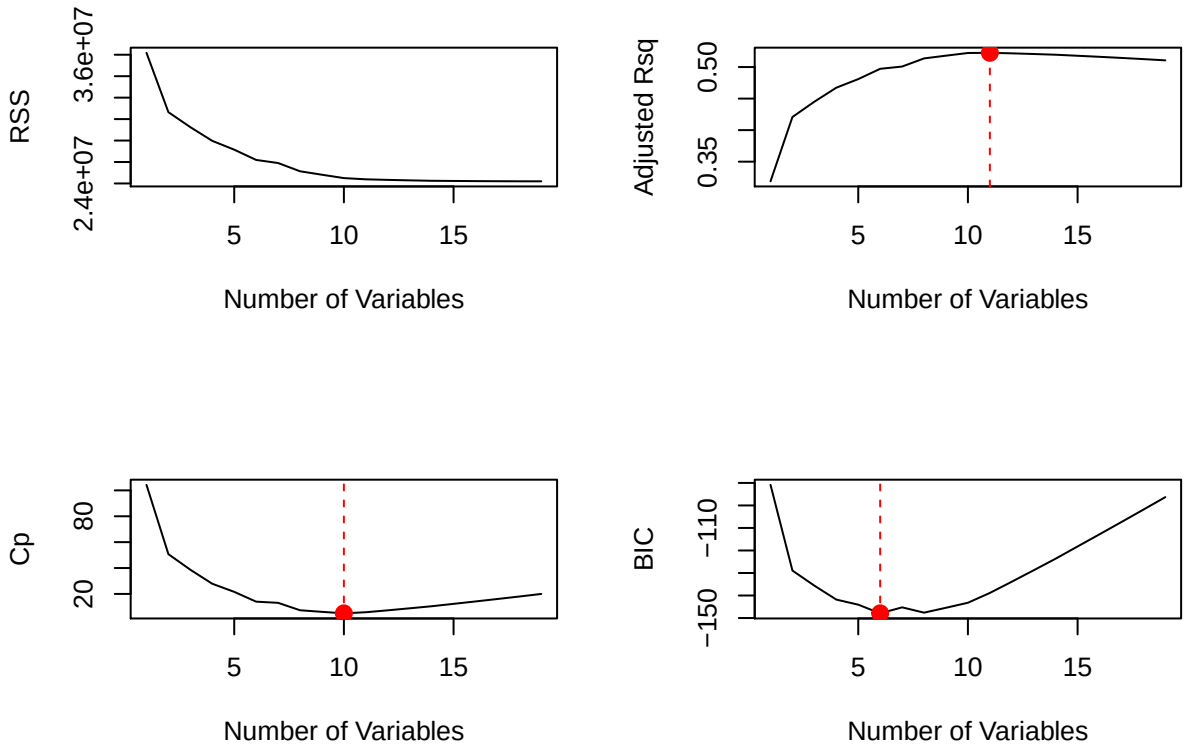
```
# Plot the RSS, adj R^2, C_p, and BIC for all of the models at once to do some visual judgement
par(mfrow=c(2,2))

plot(reg.summary$rss, xlab="Number of Variables", ylab="RSS", type="l")

plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted Rsq", type = "l")
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)], col="red", cex=2,
abline(v = which.max(reg.summary$adjr2), col="red", lty=2) # Drop a dashed line to the x-axis (lty = 2

plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)], col="red", cex=2, pch=20)
abline(v = which.min(reg.summary$cp), col="red", lty=2)

plot(reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)], col="red", cex=2, pch=2
abline(v = which.min(reg.summary$bic), col="red", lty=2)
```

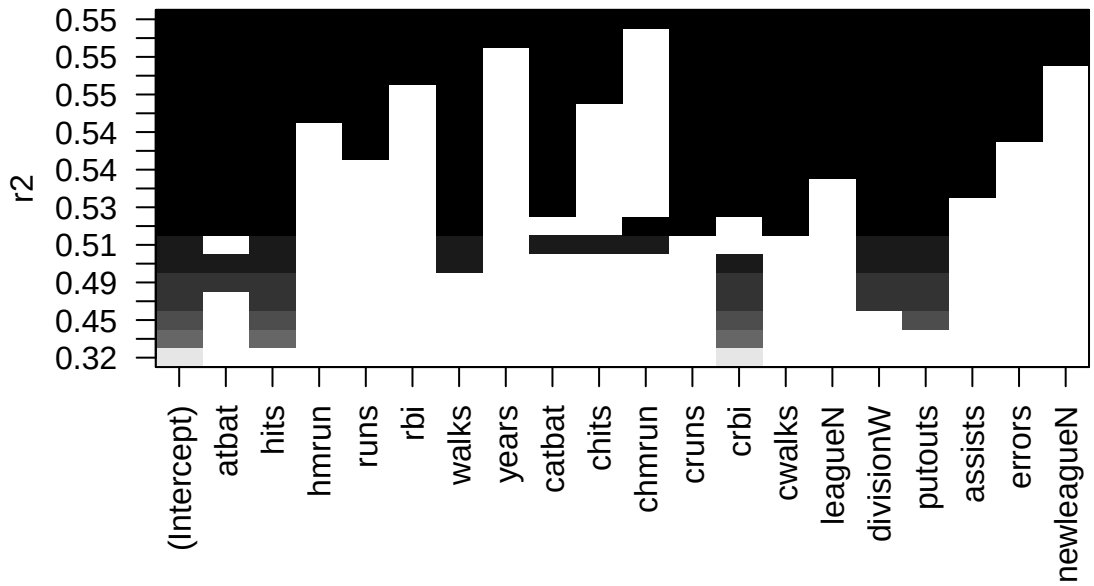


The `points()` command works like the `plot()` command, except that it puts points on the plot that has already been created, instead of creating a new plot. The `which.max()` function can be used to identify the location of the maximum point of a vector. We will plot a red dot to indicate the model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

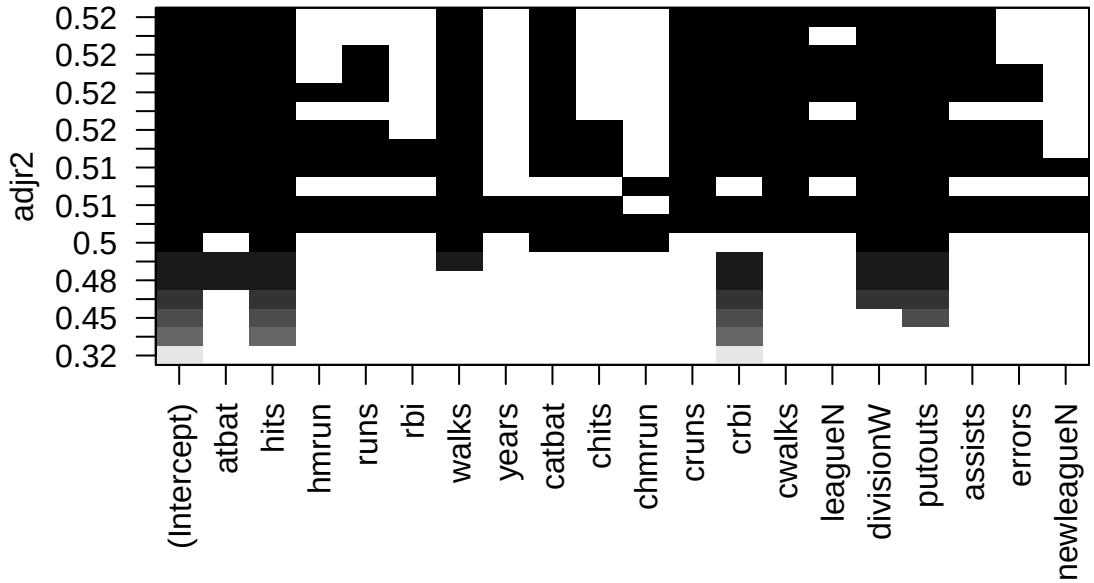
In a similar fashion, we can plot the C_p and BIC statistics, and indicate the models with the smallest statistic using `which.min()`.

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 the function, type `?plot.regsubsets`.

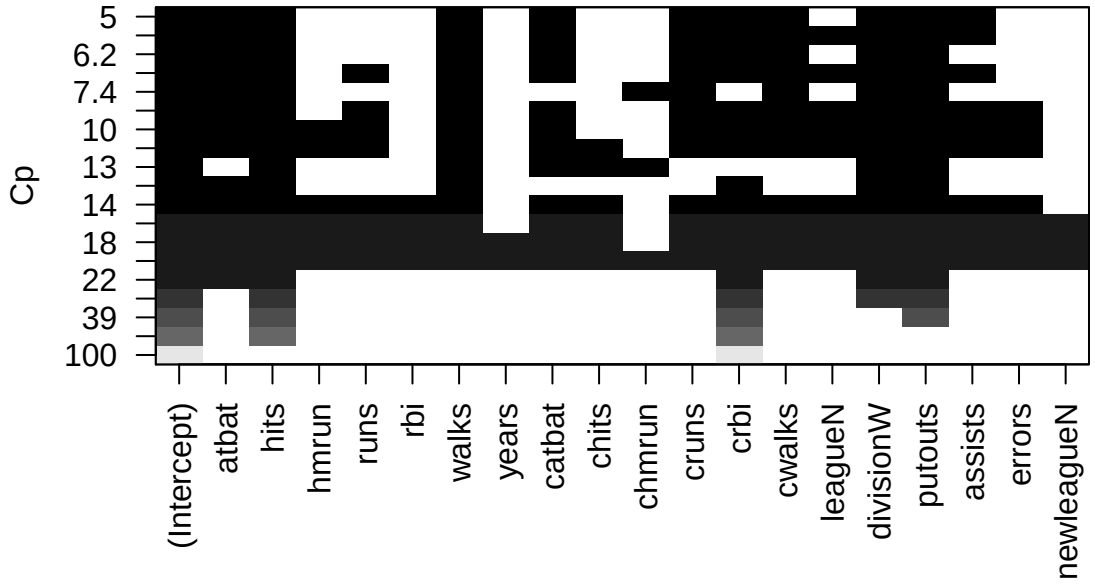
```
# The regsubsets() function has a built-in plot option for doing just this, let's try it out.
# The top row of each plot contains a black square for each variable selected from the optimal model as
plot(regfit.fuller, scale="r2")
```



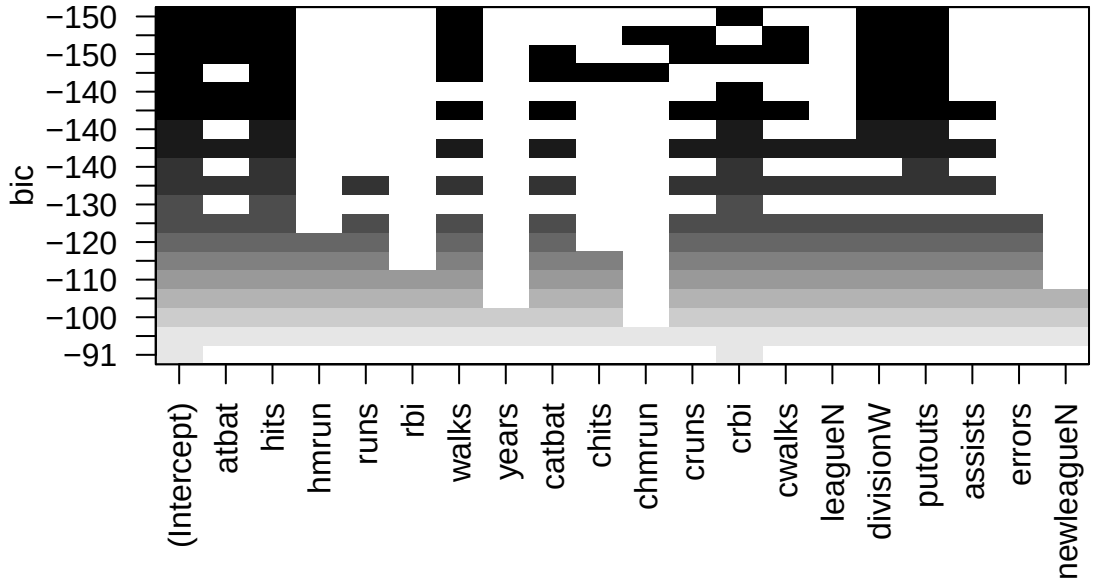

```
plot(regfit.fuller, scale="adjr2")
```




```
plot(regfit.fuller, scale="Cp")
```




```
plot(regfit.fuller, scale="bic")
```



The top row of each plot contains a black dquare 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.

```
# Let's observe the coefficients from the six-variable model
coef(regfit.fuller, 6)
```

```
##  (Intercept)        atbat         hits        walks         crbi    divisionW
##   91.5117981   -1.8685892    7.6043976    3.6976468    0.6430169 -122.9515338
##      putouts
##    0.2643076
```

Forward and Backward Stepwise Selection

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

```
# The regsubsets() function can also perform stepwise selection using the method = "" option
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 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
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 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

Check out the difference in the 1, 2 and 3-variable models. In the forward selection output, `CRBI` is the best variable in the single variable model and `hits` is added along to the best model with two variables. In the backward selection output, `cruns` is the best variable in the single model, and `cruns`/`hits` are the best variables for a model with two variables.

For instance, we see that only 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 models are each identical for the 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.

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 variable 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 dataset 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 that 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/test set split.

```
# Create a logical vector that's the same length as the number of observations in Hitters; this will de
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.

```
# Perform best subset selection on the training set
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.

```
# Goal now is to compute validation set error for each model that was created; we must first make a mod
#This creates a matrix that contains all the data that will be included when the model is run (builds b
test.mat <- model.matrix(salary ~ ., data=Hitters[test,])
```

The `model.matrix()` function is used in many regression packages for building an "X" matrix from the data. Now we run the 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.

```
# Create an empty vector that is 19-long, and loop through each model to the regfit.best object to extr
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.

```
# Which model has the lowest MSE and what are its coefficients?
which.min(val.errors)
```

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

```
coef(regfit.best, which.min(val.errors))
```

```
##  (Intercept)        atbat         hits        walks        cruns       cwalks
##   67.1085369   -2.1462987    7.0149547    8.0716640    1.2425113   -0.8337844
##    divisionW      putouts
## -118.4364998    0.2526925
```

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

```
# Note that we ha to create our own predictions, as regsubsets() does not contain an argument for it; l
predict_regsubsets <- function(regfit_object, testing_set, id, ...){
  form <- as.formula(regfit_object$call[[2]]) # Return the formula used in the regsubsets() command, th
```

```
  mat <- model.matrix(form, testing_set) # Create the model matrix using the training set and the formu
  coefi <- coef(regfit_object, 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 dataset, and select the best ten-variable model. It is important that we make use of the full dataset in order to obtain more accurate coefficient estimates. Note that we perform best subset selection on the full dataset and select the best ten variable model, rather than simply using the variables that were obtained from the training set, because the best seven-variable model on the full dataset may differ from the corresponding model on the training set.

```
# We saw that the seven-variable model performed best, so let's rerun the regsubsets() using the testin
regfit.bester <- regsubsets(salary ~ ., data=Hitters, nvmax=19)
coef(regfit.bester, 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
```

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

We now try to choose among the models of different sizes using cross-validation. Thsi 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.

```
# Finally, let's do some k-fold cross-validation
k <- 10
set.seed(1)
folds <- sample(1:k, nrow(Hitters), replace=TRUE)
```

Now we write a for loop that performs cross-validation. 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 (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`.

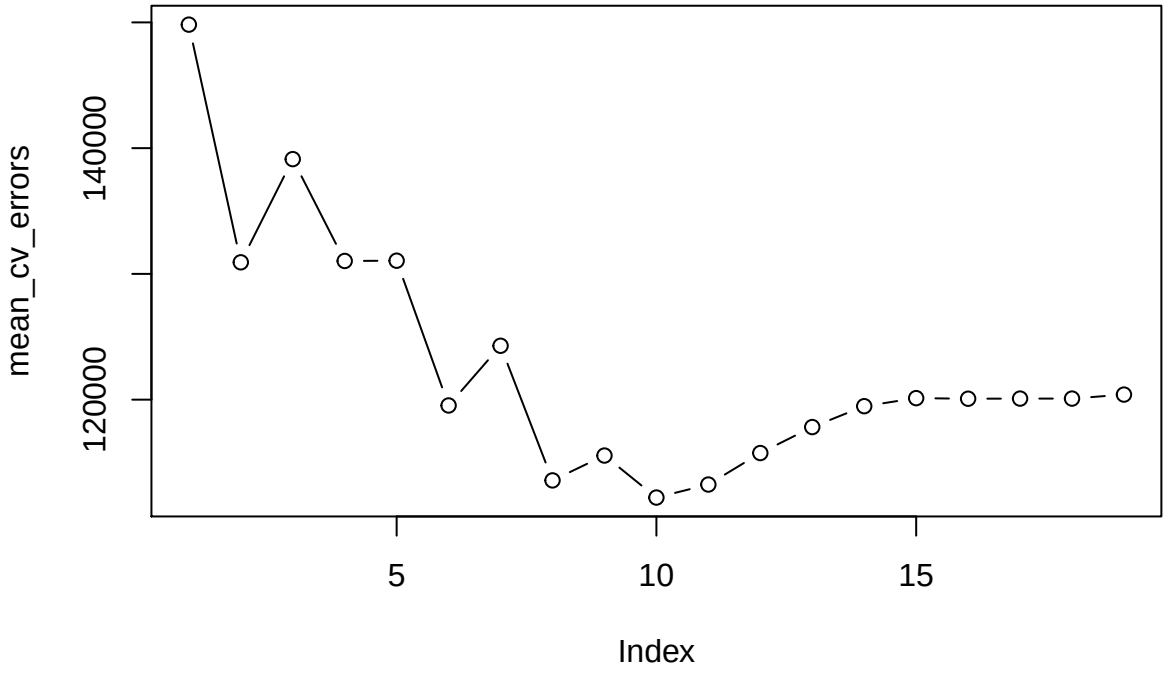
```
# In the jth fold, the elements of Hitters belonging to j are considered the test set, while everything
cv_errors <- matrix(NA, k, 19, dimnames = list(NULL, paste(1:19)))
for (j in 1:k){
  best.fit <- regsubsets(salary ~ ., data=Hitters[folds!=j,], nvmax=19)
  max_vars <- best.fit$call[[4]]
  for (i in 1:19){
    pred <- predict_regsubsets(best.fit, Hitters[folds == j,], id = i) #MLEM
    cv_errors[j,i] <- mean((Hitters$salary[folds == j]-pred)^2)
  }
}
```

This has given us a 10x19 matrix, of which the (i,j)the element corresponds to the test MSE for the ith 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 jth element is the cross-validation erro for the j-variable model.

```
# Use the apply function to average over the columns of this matrix to find the average MSE for each mo
mean_cv_errors <- apply(cv_errors, 2, mean)
mean_cv_errors
```

```
##        1        2        3        4        5        6        7        8
## 149821.1 130922.0 139127.0 131028.8 131050.2 119538.6 124286.1 113580.0
##        9       10       11       12       13       14       15       16
## 115556.5 112216.7 113251.2 115755.9 117820.8 119481.2 120121.6 120074.3
##       17       18       19
## 120084.8 120085.8 120403.5
```

```
# Plot the means to visually find the best performing model
par(mfrow = c(1,1))
plot(mean_cv_errors, type="b")
```



We see that the cross-validation selects a 10-variable model. We now perform best subset selection on the full dataset in order to obtain the 10-variable model.

```
# It should be obvious that the model with the lowest test MSE is the model with the 10 variables, so r
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
```