Lab 6.1 - Subset Selection Methods

An Introduction to Statistical Learning

We will use the `Hitters` dataset.

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

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

There are 59 missing observations for `Salary` so we need to make some cleaning:

```
Hitters = na.omit(Hitters)
attach(Hitters)
```

Subset Selection is done using the `regsubsets()` method, included in the `leaps` library. RSS is used to measure which model is "best".

```
library(leaps)
```

1. Best Subset Selection

By default `regsubsets()` reports result up to the best eight-variable model. We can change this with the parameter `nvmax`:

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

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

An asterisk indicates that a given variable is included in the corresponding model.

If we want to force a varible to appear in the model we can use the `force.in` parameter, with a list of the column indices. In a similar way, we can force a variable to not be included in the model using `force.out`.

`summary()` also includes different statistics used to select the best model: R^2 (`rsq`), Residual Sum of Squares (`rss`), Adjusted R^2 (`adjr2`), Mallow's C_p (`cp`) and Bayesian Information Criterion (`bic`). `which` is a matrix defining which variables are included in each model.

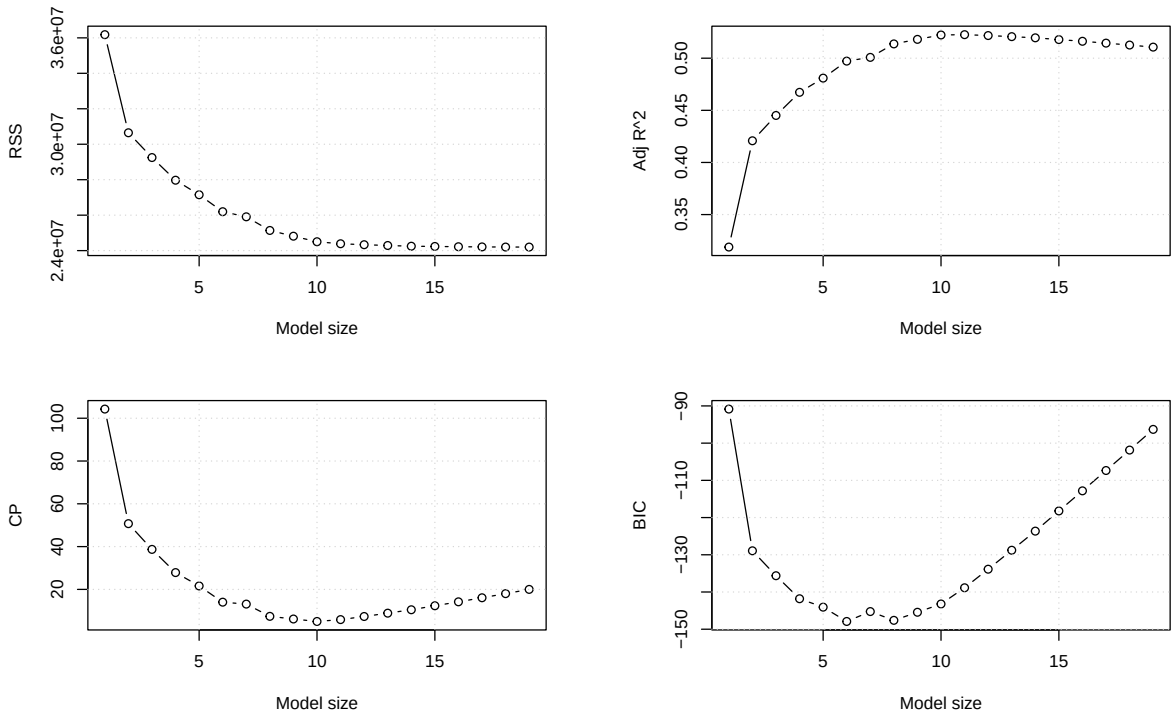
```
names(reg.summary)
```

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

MSE $= \text{RSS}/n$ is generally an underestimate of the test MSE (the model is fitted to get the smallest *training* error, but the same model does not have to be the one with the lowest *test* error, because the training MSE tipycally decreases when we add more variables, but the test MSE can increase). Therefore, training set RSS and training set R^2 are not the best metrics to select the best model. On the other hand, C_p, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) and Adjusted R^2 are computed using techniques for *adjusting* the training error for the model size.

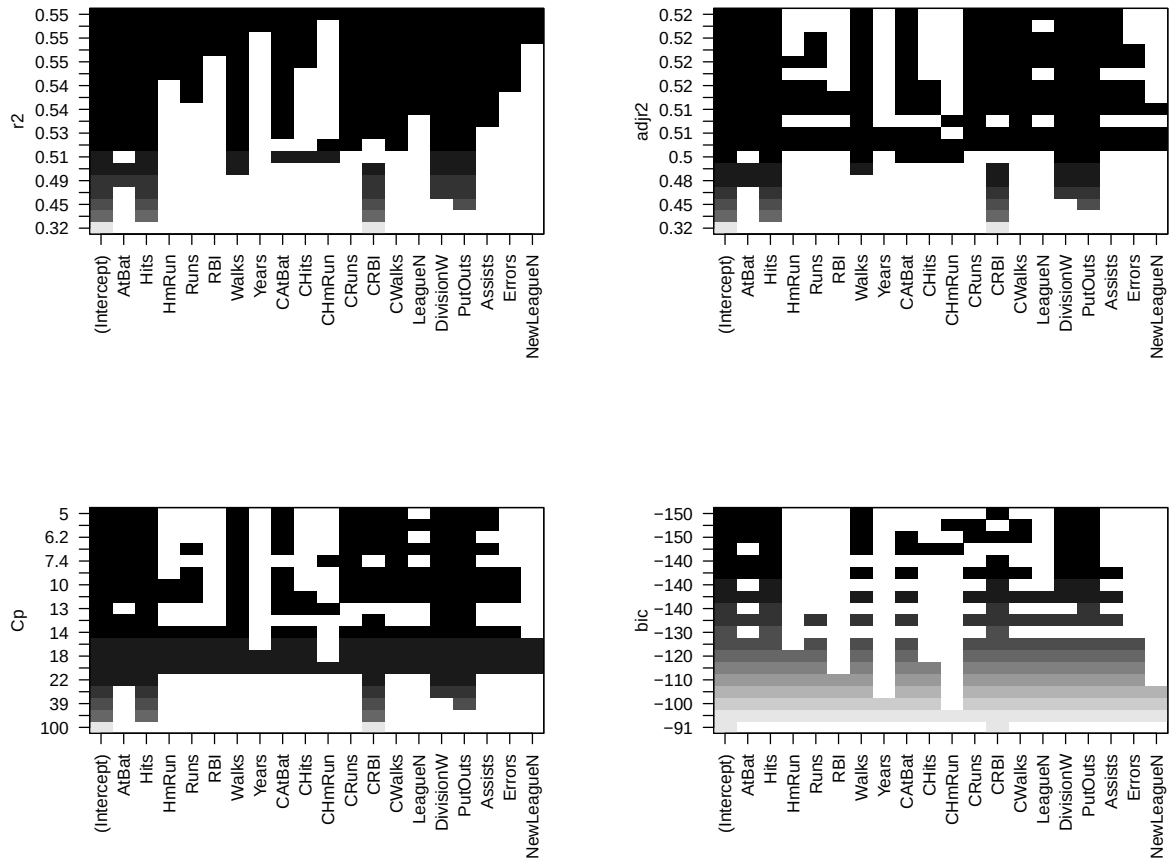
To decide which model to use we can plot some statistics:

```
par(mfrow=c(2, 2), mar=c(4, 4, 3, 4))
plot(reg.summary$rss, type='b', xlab = 'Model size', ylab = 'RSS'); grid()
plot(reg.summary$adjr2, type='b', xlab = 'Model size', ylab = 'Adj R^2'); grid()
plot(reg.summary$cp, type='b', xlab = 'Model size', ylab = 'CP'); grid()
plot(reg.summary$bic, type='b', xlab = 'Model size', ylab = 'BIC'); grid()
```



`regsubsets()` also has a built-in `plot()` function to plot the selected variables for the best model with a given number of predictors, according to the specified loss metric:

```
par(mfrow=c(2, 2), mar=c(4, 4, 3, 4))
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. In this case when we use `adjr2` we obtain a 12-variable model, a 11-variable model for `Cp` and a 7-variable model for `bic`.

To get the coefficients for the 7-variable model:

```
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
```

2. Forward and Backward Stepwise Selection

Forward and Backward stepwise selection are done with `regsubsets()` using the parameter `method`.

```
regfit.fwd = regsubsets(Salary~., data = Hitters, nvmax = 19, method = 'forward')
regfit.bwd = regsubsets(Salary~., data = Hitters, nvmax = 19, method = 'backward')
```

The method returns a similar output than before, with asterisks indicating when a variable has been included in a model.

Like before, `summary(fitted_model)` has information for the different statistics.

3. Choosing among models using the Validation Set approach and Cross-Validation

The Validation Set approach

We create training and test subsets:

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

Now we perform model selection using the training set:

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

Computing the validation set error is more complicated than with other methods, as there is no `predict()` function for `regsubsets()`. We first make a model matrix from the test data. `model.matrix()` creates a model matrix by expanding factors (`League`, `Division`, `NewLeague`) to a set of dummy variables and expanding interactions similarly:

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

```
##                    (Intercept) AtBat Hits HmRun Runs RBI Walks Years CAtBat
## -Alvin Davis                 1   479  130    18   66  72    76     3   1624
## -Alfredo Griffin             1   594  169     4   74  51    35    11   4408
## -Andre Thornton              1   401   92    17   49  66    65    13   5206
## -Alan Trammell               1   574  159    21  107  75    59    10   4631
## -Buddy Biancalana            1   190   46     2   24   8    15     5    479
## -Bruce Bochy                 1   127   32     8   16  22    14     8    727
##                    CHits CHmRun CRuns CRBI CWalks LeagueN DivisionW PutOuts
## -Alvin Davis         457     63   224  266    263       0         1     880
## -Alfredo Griffin    1133     19   501  336    194       0         1     282
## -Andre Thornton     1332    253   784  890    866       0         0       0
## -Alan Trammell      1300     90   702  504    488       0         0     238
## -Buddy Biancalana    102      5    65   23     39       0         1     102
## -Bruce Bochy         180     24    67   82     56       1         1     202
##                    Assists Errors NewLeagueN
## -Alvin Davis            82     14          0
## -Alfredo Griffin       421     25          0
## -Andre Thornton          0      0          0
## -Alan Trammell         445     22          0
## -Buddy Biancalana      177     16          0
## -Bruce Bochy            22      2          1
```

Now we get the coefficients for the best model obtained by `regsubsets()` for each model size `i`, and multiply them into the appropiate 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) {
  # Get the coefficients for the i-th model
  coef.i = coef(regfit.best, id = i)
  # Make a new model matrix containing only the variables for the i-th model
  test.mat.i = test.mat[, names(coef.i)]
  # Multiply the new matrix by the coefficients for the i-th model
  pred = test.mat.i %*% coef.i
  # Compute validation set errors
```

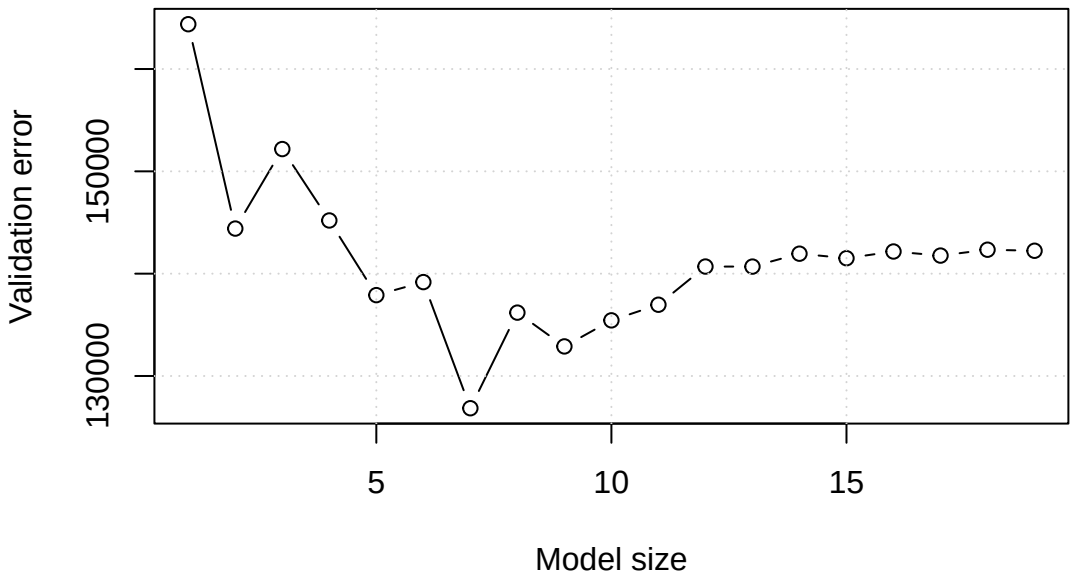
```
  val.errors[i] = mean((Hitters$Salary[test] - pred)^2)
}
```

We can make a function to do this prediction process:

```
pred.regsubsets = function(obj, newdata, id, ...) {
  form = as.formula(obj$call[[2]])
  mat = model.matrix(form, newdata)
  coef.i = coef(obj, id=id)
  xvars = names(coef.i)
  mat[, xvars] %*% coef.i
}
```

Plotting the validation errors we see that the model with the smallest error is the one with 7 variables:

```
plot(val.errors, type='b', xlab = 'Model size', ylab = 'Validation error'); grid()
```



The minimum validation error occurs for the 7-variable model:

```
ix = which.min(val.errors)
cat(sprintf("Model Size: %d [MSE = %.2f]", ix, val.errors[ix]))
```

```
## Model Size: 7 [MSE = 126848.96]
```

Now we can use the complete dataset to create the final 7-variable model:

```
regfit.final = regsubsets(Salary~., data = Hitters, nvmax = 19)
coef(regfit.final, 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
```

Different selection methods can select different variables for the same model size.

To see how well it predicted:

```
preds.val = pred.regsubsets(regfit.final, Hitters[, -19], id = 7)
cat(sprintf("Model Size: %d [MSE = %.2f]", ix, mean((preds.val - Hitters$Salary)^2)))
```

```
## Model Size: 7 [MSE = 98503.98]
```

K-Fold Cross-Validation

We perform best subset selection *within each of the k training sets*. First we assign each row in *Hitters* to one of the k folds:

```
set.seed(1)
k = 10
folds = sample(1:k, nrow(Hitters), replace = T)
head(folds, 20)
```

```
##  [1]  9  4  7  1  2  7  2  3  1  5  5 10  6 10  7  9  5  5  9  9
```

Create an error matrix:

```
cv.errors = matrix(data = 0, nrow = k, ncol = 19,
                   dimnames = list(paste(1:k), paste(1:19)))
```

We now perform cross-validation in a loop:

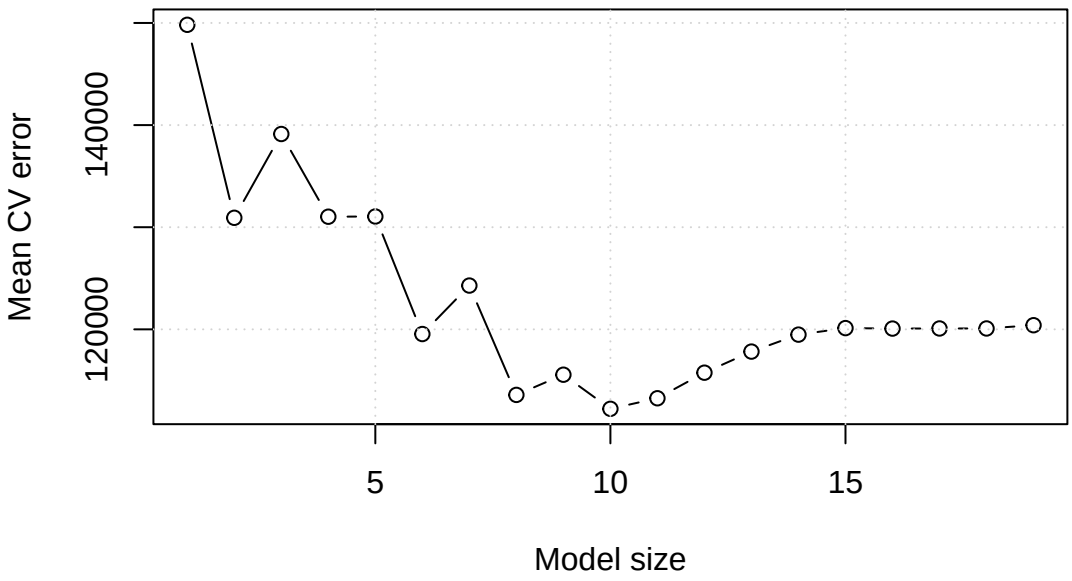
```
for (fold in 1:k) {
  best.fit = regsubsets(Salary~.,
                        data = Hitters[folds!=fold,],
                        nvmax = 19)
  for (nvar in 1:19) {
    pred = pred.regsubsets(obj = best.fit,
                           newdata = Hitters[folds==fold,],
                           id = nvar)
    err = Hitters$Salary[folds==fold]
    cv.errors[fold, nvar] = as.double(mean((err - pred)^2))
  }
}
```

We compute the mean error for each model size, by computing the mean for every fold given a model size (mean of the columns):

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

And we plot the results:

```
plot(mean.cv.errors, type='b', xlab = 'Model size', ylab = 'Mean CV error')
grid()
```



In this case the best model is one with 10 variables:

```
coef(best.fit, id = 10)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  190.7517251   -2.3530620    7.4637992    5.5739960   -0.1196880    1.3104568
##         CRBI       CWalks    DivisionW      PutOuts      Assists
##    0.7578680   -0.7730634 -106.4628564    0.2551175    0.2886913
```

And the error is:

```
ix = which.min(mean.cv.errors)
preds.val = pred.regsubsets(best.fit, Hitters[, -19], id = ix)
cat(sprintf("Model Size: %d [MSE = %.2f]", ix, mean((preds.val - Hitters$Salary)^2)))
```

```
## Model Size: 10 [MSE = 93505.12]
```