Linear Models & Regularization

By: Udit (based on ISLR)

Setup

Using **Hitters** dataset from **ISLR2** package. Using **Leaps** package with **regsubsets** for Best Subset Selection. Using **glmnet** package with **glmnet()** for Lasso & Ridge shrinkage. Using **pls** package with **pcr()** for Principal Components regression and **plsr()** for Partial Lease Square regression.

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

```
## Loading required package: Matrix
```

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

```
library(leaps)
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
# Data Review
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
```

```
summary(Hitters) # Salary has missing values
```

```
##      AtBat            Hits         HmRun            Runs
##  Min.   : 16.0   Min.   :  1   Min.   : 0.00   Min.   :  0.00
##  1st Qu.:255.2   1st Qu.: 64   1st Qu.: 4.00   1st Qu.: 30.25
```

```
##  Median :379.5   Median : 96   Median : 8.00   Median : 48.00
##  Mean   :380.9   Mean   :101   Mean   :10.77   Mean   : 50.91
##  3rd Qu.:512.0   3rd Qu.:137   3rd Qu.:16.00   3rd Qu.: 69.00
##  Max.   :687.0   Max.   :238   Max.   :40.00   Max.   :130.00
##
##       RBI             Walks            Years            CAtBat
##  Min.   :  0.00   Min.   :  0.00   Min.   : 1.000   Min.   :   19.0
##  1st Qu.: 28.00   1st Qu.: 22.00   1st Qu.: 4.000   1st Qu.:  816.8
##  Median : 44.00   Median : 35.00   Median : 6.000   Median : 1928.0
##  Mean   : 48.03   Mean   : 38.74   Mean   : 7.444   Mean   : 2648.7
##  3rd Qu.: 64.75   3rd Qu.: 53.00   3rd Qu.:11.000   3rd Qu.: 3924.2
##  Max.   :121.00   Max.   :105.00   Max.   :24.000   Max.   :14053.0
##
##      CHits            CHmRun           CRuns             CRBI
##  Min.   :   4.0   Min.   :  0.00   Min.   :   1.0   Min.   :   0.00
##  1st Qu.: 209.0   1st Qu.: 14.00   1st Qu.: 100.2   1st Qu.:  88.75
##  Median : 508.0   Median : 37.50   Median : 247.0   Median : 220.50
##  Mean   : 717.6   Mean   : 69.49   Mean   : 358.8   Mean   : 330.12
##  3rd Qu.:1059.2   3rd Qu.: 90.00   3rd Qu.: 526.2   3rd Qu.: 426.25
##  Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.00
##
##      CWalks          League  Division    PutOuts          Assists
##  Min.   :   0.00   A:175   E:157    Min.   :   0.0   Min.   :  0.0
##  1st Qu.:  67.25   N:147   W:165    1st Qu.: 109.2   1st Qu.:  7.0
##  Median : 170.50                    Median : 212.0   Median : 39.5
##  Mean   : 260.24                    Mean   : 288.9   Mean   :106.9
##  3rd Qu.: 339.25                    3rd Qu.: 325.0   3rd Qu.:166.0
##  Max.   :1566.00                    Max.   :1378.0   Max.   :492.0
##
##      Errors          Salary       NewLeague
##  Min.   : 0.00   Min.   :  67.5   A:176
##  1st Qu.: 3.00   1st Qu.: 190.0   N:146
##  Median : 6.00   Median : 425.0
##  Mean   : 8.04   Mean   : 535.9
##  3rd Qu.:11.00   3rd Qu.: 750.0
##  Max.   :32.00   Max.   :2460.0
##                  NA's   :59
```

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

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

```r
# Drop missing values
d_Hitters = na.omit(Hitters)
dim(d_Hitters)
```

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

Best Subset Selection

Looks through 2^p models, and identifies best model for each value of p. An asterisk indicates that a given variable is included in the corresponding model.

```r
regfit.full <- regsubsets(Salary ~., d_Hitters, nvmax=dim(d_Hitters)-1)  #nvmax=8 by default
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., d_Hitters, nvmax = dim(d_Hitters) -
##     1)
## 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 ) "*"    "*"    "*"      "*"     "*"    "*"    "*"
```

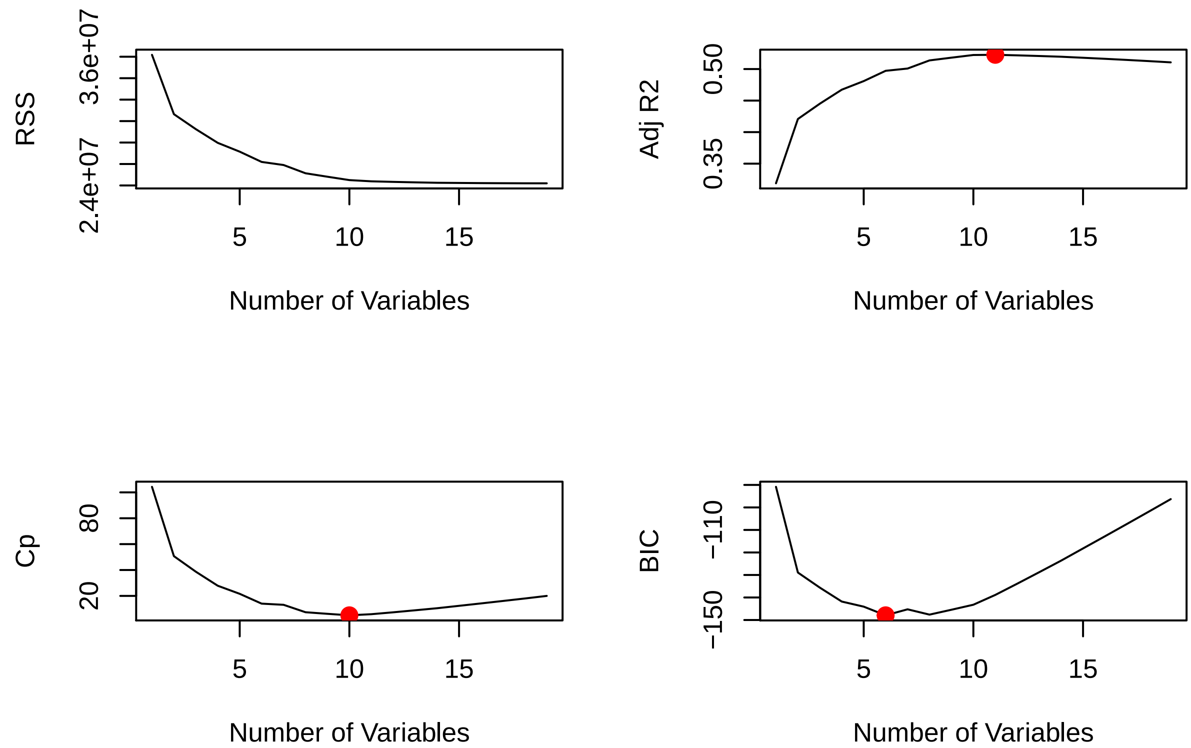
```
names(summary(regfit.full))
```

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

```
#summary(regfit.full)$rsq
#summary(regfit.full)$adjr2
summary(regfit.full)$bic
```

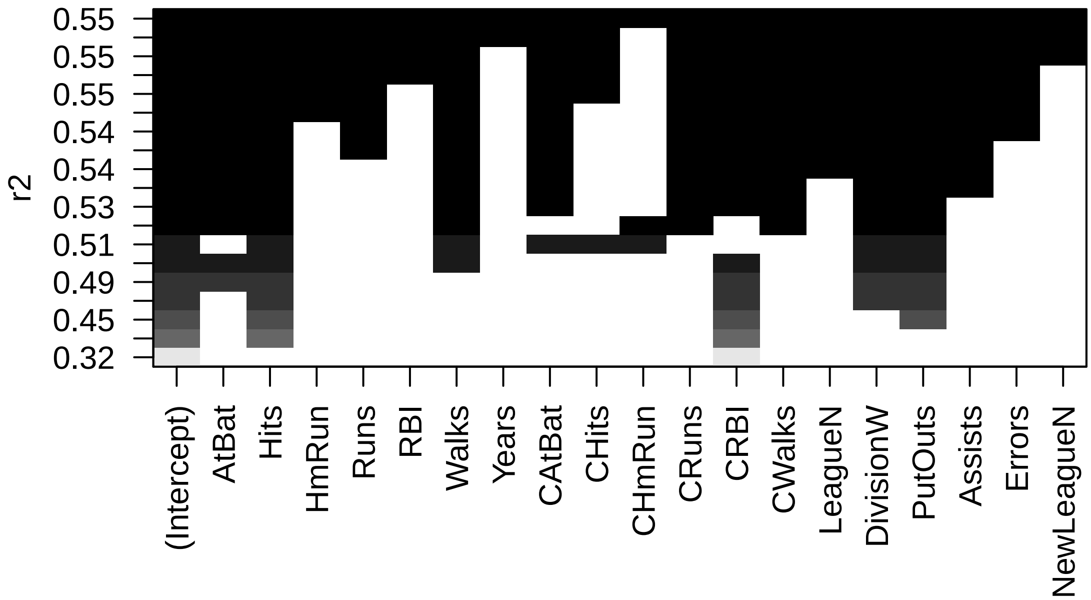
```
##  [1]  -90.84637 -128.92622 -135.62693 -141.80892 -144.07143 -147.91690
##  [7] -145.25594 -147.61525 -145.44316 -143.21651 -138.86077 -133.87283
## [13] -128.77759 -123.64420 -118.21832 -112.81768 -107.35339 -101.86391
## [19]  -96.30412
```

```
reg.summary = summary(regfit.full)

# Plot for easier viewing
par (mfrow = c(2, 2))

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

plot (reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adj R2", type = "l")
  idx <- which.max(reg.summary$adjr2)
  points(idx, reg.summary$adjr2[idx], col="red", cex=2, pch=20)

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

plot (reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
  idx <- which.min(reg.summary$bic)
  points (idx, reg.summary$bic[idx], col="red", cex=2, pch=20)
```

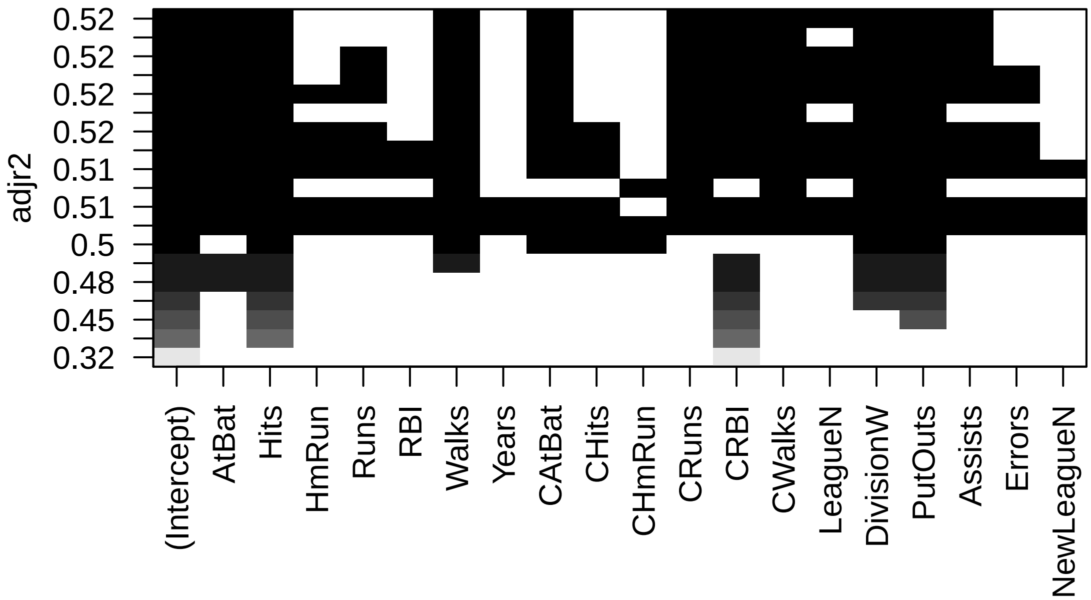


The regsubsets() function has a built-in plot() command which can be used to display the selected variables for the best model with a given number of predictors, ranked according to the BIC, Cp, adjusted R2, or AIC. **Black square** for each variable selected according to the optimal model associated with that statistic.

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




```r
plot(regfit.full, scale = "adjr2")
```




```r
#plot(regfit.full, scale = "Cp")
#plot(regfit.full, scale = "bic")
```

Coefficients for model with best fit

```r
coef(regfit.full, 10)         #based on Adj R2 and Cp
```

```
##   (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
```

```r
coef(regfit.full, 6)         #based on BIC
```

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

```r
# Fitting the Final Regression Model on full data
summary(lm(Salary~AtBat +Hits +Walks +CRBI +Division +PutOuts +CAtBat +CRuns
           +CWalks +Assists, d_Hitters))
```

```
##
## Call:
## lm(formula = Salary ~ AtBat + Hits + Walks + CRBI + Division +
##     PutOuts + CAtBat + CRuns + CWalks + Assists, data = d_Hitters)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -939.11 -176.87  -34.08  130.90 1910.55
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  162.53544   66.90784   2.429 0.015830 *
## AtBat         -2.16865    0.53630  -4.044 7.00e-05 ***
## Hits           6.91802    1.64665   4.201 3.69e-05 ***
## Walks          5.77322    1.58483   3.643 0.000327 ***
## CRBI           0.77431    0.20961   3.694 0.000271 ***
## DivisionW   -112.38006   39.21438  -2.866 0.004511 **
## PutOuts        0.29737    0.07444   3.995 8.50e-05 ***
## CAtBat        -0.13008    0.05550  -2.344 0.019858 *
## CRuns          1.40825    0.39040   3.607 0.000373 ***
## CWalks        -0.83083    0.26359  -3.152 0.001818 **
## Assists        0.28317    0.15766   1.796 0.073673 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 311.8 on 252 degrees of freedom
## Multiple R-squared:  0.5405, Adjusted R-squared:  0.5223
## F-statistic: 29.64 on 10 and 252 DF,  p-value: < 2.2e-16
```

```
summary(lm(Salary~AtBat +Hits +Walks +CRBI +Division +PutOuts, d_Hitters))
```

```
##
## Call:
## lm(formula = Salary ~ AtBat + Hits + Walks + CRBI + Division +
##     PutOuts, data = d_Hitters)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -873.11 -181.72  -25.91  141.77 2040.47
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   91.51180   65.00006   1.408 0.160382
## AtBat         -1.86859    0.52742  -3.543 0.000470 ***
## Hits           7.60440    1.66254   4.574 7.46e-06 ***
## Walks          3.69765    1.21036   3.055 0.002488 **
## CRBI           0.64302    0.06443   9.979  < 2e-16 ***
## DivisionW   -122.95153   39.82029  -3.088 0.002239 **
## PutOuts        0.26431    0.07477   3.535 0.000484 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 319.9 on 256 degrees of freedom
## Multiple R-squared:  0.5087, Adjusted R-squared:  0.4972
```

```
## F-statistic: 44.18 on 6 and 256 DF,  p-value: < 2.2e-16
```

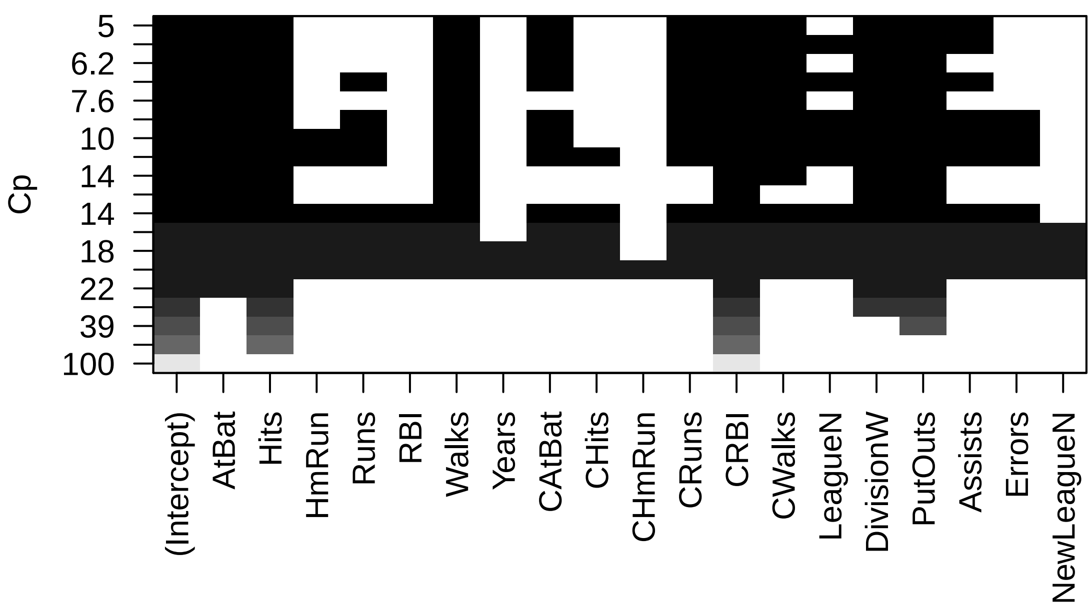
Forward & Backward Selection

```r
#Forward
regfit.fwd <- regsubsets(Salary ~., d_Hitters, nvmax=dim(d_Hitters)-1, method="forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., d_Hitters, nvmax = dim(d_Hitters) - 
##     1, 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 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

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

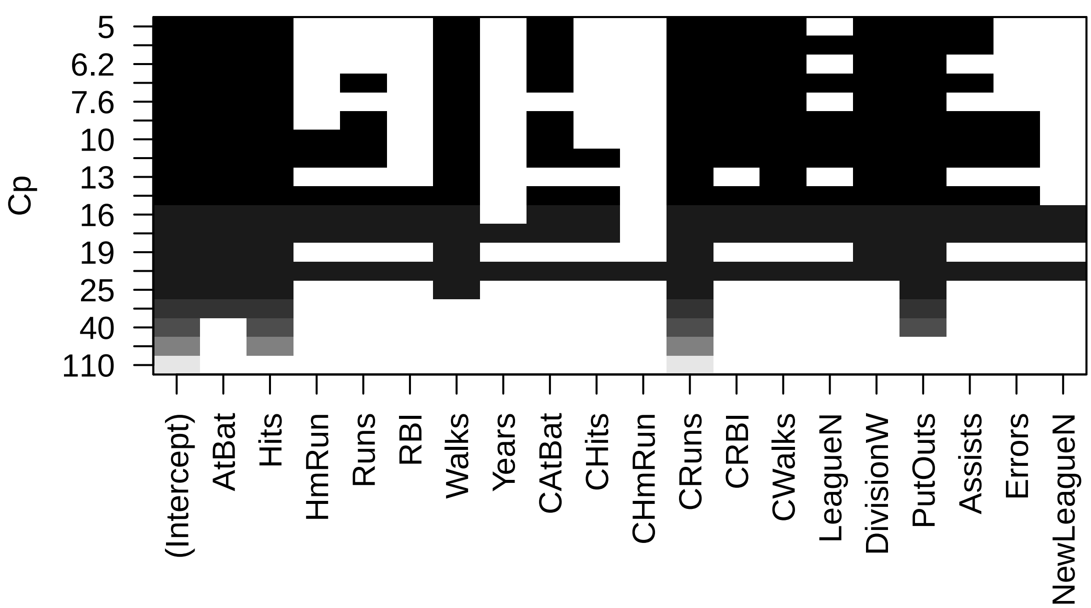


```r
#Backward
regfit.back <- regsubsets(Salary ~., d_Hitters, nvmax=dim(d_Hitters)-1, method="backward")
summary(regfit.back)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., d_Hitters, nvmax = dim(d_Hitters) -
##     1, 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 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

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




```
#Comparing 7 variable models from two approaches
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
```

```
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
```

```
coef(regfit.back, 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
```

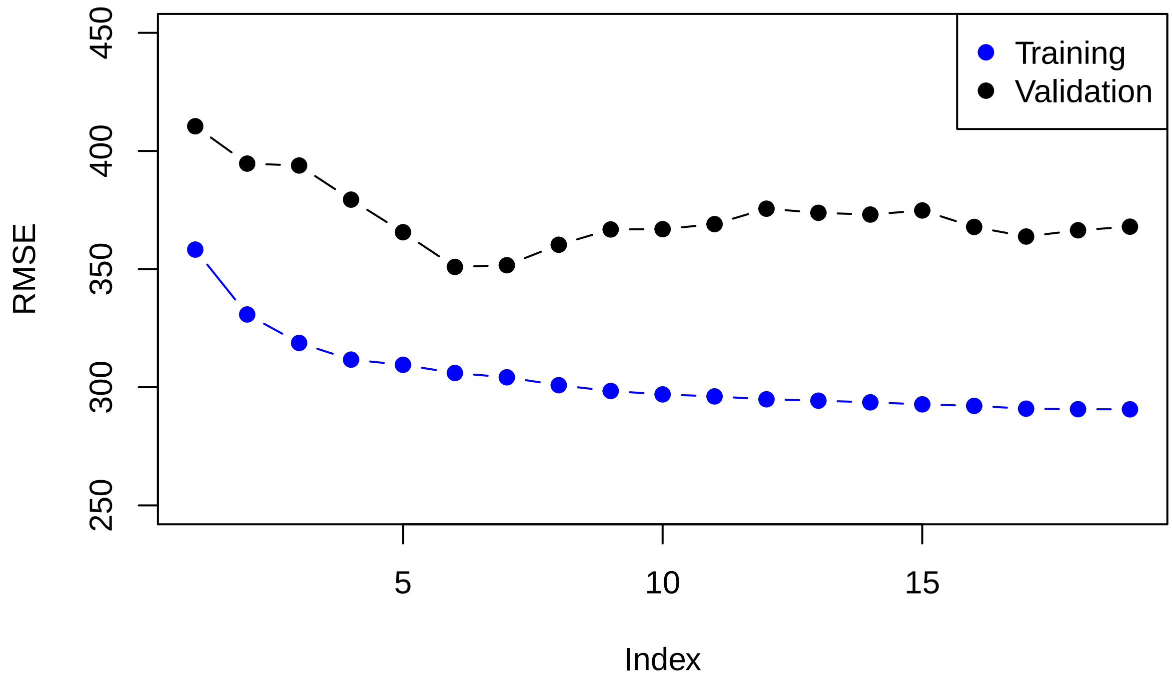
Choosing among models using Train & Validation set method

Model.matrix() function is used in many regression packages to build an "X" matrix from data.

```
set.seed(1)

dim(d_Hitters)
```

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

```
train = sample(seq(263),180, replace = FALSE)    #2/3rd training
regfit.train.fwd <- regsubsets(Salary ~., d_Hitters[train,], nvmax=dim(d_Hitters)-1, method="forward")

RMSE = rep(NA, 19)
x.test = model.matrix(Salary ~., d_Hitters[-train,])

# regsubset does not have a 'Predict' function, so developing our own
for (i in 1:19){
  coefi = coef(regfit.train.fwd, id=i)
  predi = x.test[,names(coefi)]%*%coefi
  RMSE[i] = sqrt(mean( (d_Hitters$Salary[-train]-predi)^2 ))
}

#Model with 6 variables has lowest Test Error
plot(RMSE, ylab="RMSE", ylim=c(250,450),pch=19, type="b")

points(sqrt(regfit.train.fwd$rss[-1]/180), col="blue", pch=19, type="b")
legend("topright", legend=c("Training", "Validation"), col=c("blue", "black"), pch=19)
```



Writing a Predict function for future use, since regsubsets doesn't work with generic predict function:

```
print(regfit.train.fwd$call[[2]])
```

```
## Salary ~ .
```

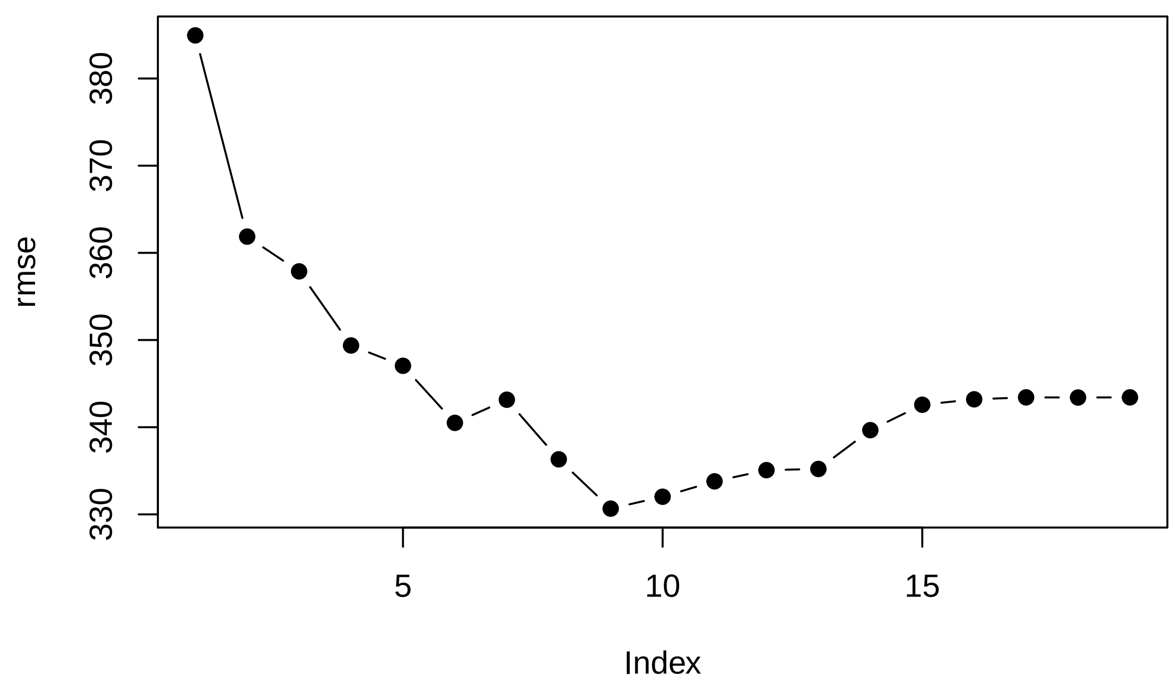
```
predict.regsubsets = function(object, newdata, id,...){
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id)
  mat[,names(coefi)]%*%coefi
}
```

Choosing among models using Cross Validation method

```
set.seed(11)

#Assign each row to a fold ranging from 1 to 10
folds = sample(rep(1:10, length=nrow(d_Hitters)))
table(folds)
```

```
## folds
##  1  2  3  4  5  6  7  8  9 10
## 27 27 27 26 26 26 26 26 26 26
```

```
#Initialize empty error matrix
cv.errors = matrix(NA, 10, 19) #10 folds, 19 variables

for(k in 1:10){
  cv_fit_k = regsubsets(Salary~., d_Hitters[folds!=k,], nvmax=19, method="forward")
  for(i in 1:19){
    pred_ki = predict(cv_fit_k, d_Hitters[folds==k,], id=i)
    cv.errors[k,i] = mean((d_Hitters$Salary[folds==k]-pred_ki)^2)
  }
}

#Average error
rmse = sqrt( apply(cv.errors, 2, mean))
plot(rmse, pch=19, type="b")
```




```
#Cross Validation approach select a 9-variable model.
#Perform best subset selection on full data set.
which.min(rmse)
```

```
## [1] 9
```

```r
coef( regsubsets(Salary~., data=d_Hitters, nvmax=19), 9)
```

```
##   (Intercept)         AtBat          Hits         Walks        CAtBat 
##  146.24960033   -1.93676754    6.65672102    5.55204413   -0.09953904 
##         CRuns          CRBI        CWalks     DivisionW       PutOuts 
##    1.25067124    0.66176849   -0.77798498 -115.34950146    0.27773062
```

Shrinkage: Ridge and Lasso

This function has a different syntax from other model-fitting functions. In particular, we must pass in an x matrix as well as a y vector.

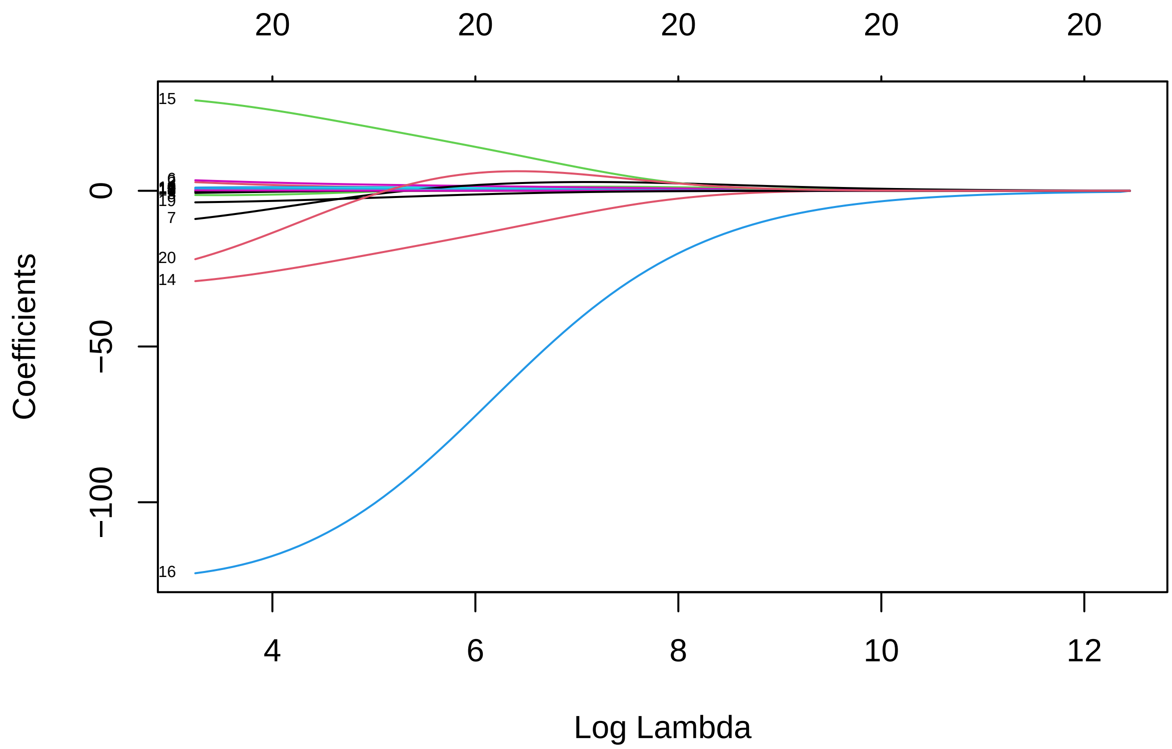
Argument **alpha** determines the model type. The penalty is defined as $(1-\alpha)/2||\beta||_2^2 + \alpha||\beta||_1$ where $0 \leq \alpha \leq 1$.

Therefore, **alpha=1 is the lasso penalty, and alpha=0 the ridge penalty**. For alpha between 0 and 1, we get elastic-net.

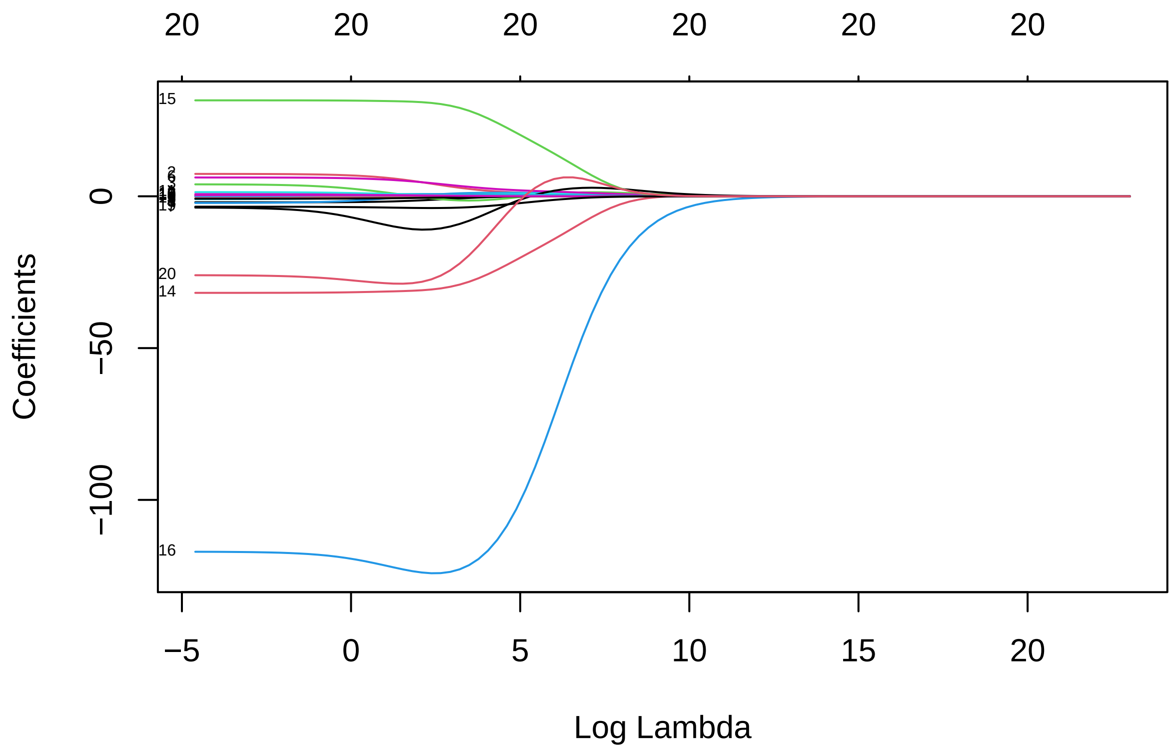
Note that by default, the glmnet() function standardizes the variables so that they are on the same scale. To turn off this default setting, use the argument standardize = FALSE.

Ridge Penalty - no feature selection

```r
# Ridge penalty
x = model.matrix(Salary~.-1, data=d_Hitters)  #dropping the intercept
y = d_Hitters$Salary

# Fit with default lambda grid
fit.ridge = glmnet(x,y,alpha=0)
plot(fit.ridge, xvar="lambda", label=TRUE)
```




```r
# User-defined lambda grid
grid = 10^seq(10,-2,length=100)
ridge.mod <- glmnet (x, y, alpha = 0, lambda = grid)
plot(ridge.mod, xvar="lambda", label=TRUE)
```




```r
# Checking Coefficients for different lamdbas
ridge.mod$lambda[50]
```

```
## [1] 11497.57
```

```r
coef(ridge.mod)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 407.399396904   0.036958511   0.138184676   0.524663826   0.230712132
##           RBI         Walks         Years        CAtBat         CHits
##   0.239850635   0.289621892   1.107710345   0.003131822   0.011653657
##        CHmRun         CRuns          CRBI        CWalks       LeagueA
##   0.087547281   0.023380056   0.024138479   0.025015505  -0.082038139
##       LeagueN     DivisionW       PutOuts       Assists        Errors
##   0.082040144  -6.215425597   0.016482325   0.002612463  -0.020523535
##    NewLeagueN
##   0.298876792
```

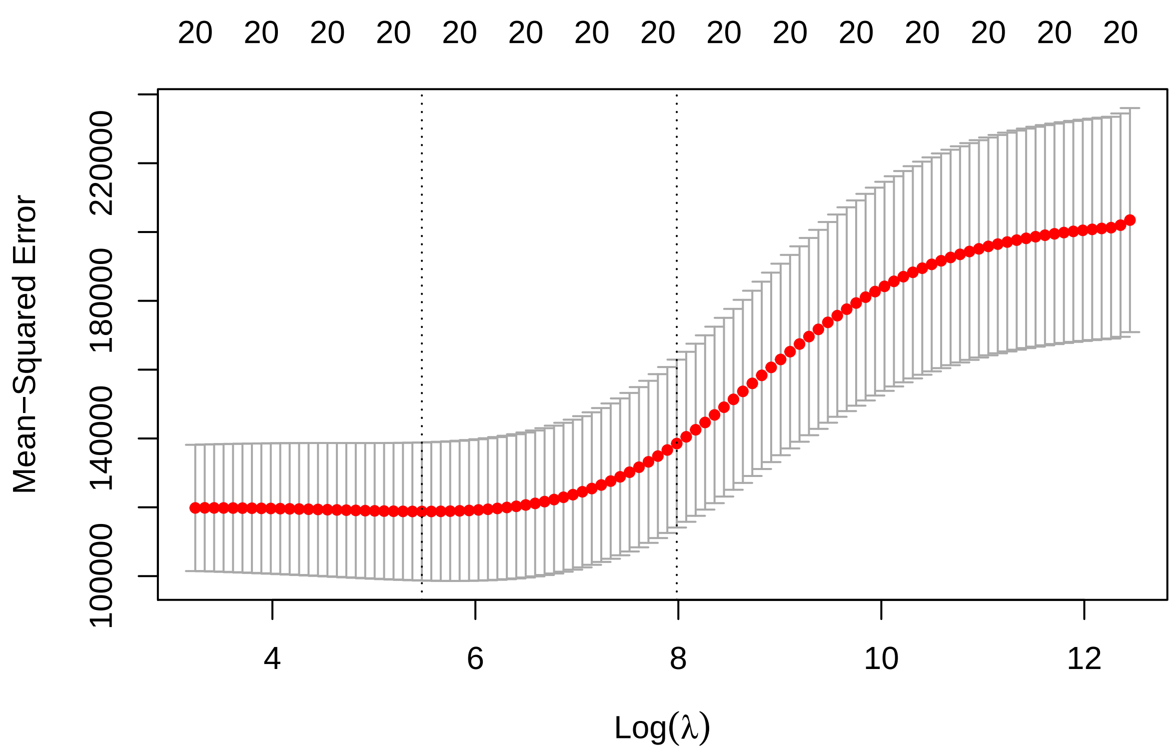
```r
ridge.mod$lambda[60]
```

```
## [1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
##   (Intercept)        AtBat         Hits        HmRun         Runs          RBI 
##   61.72599501   0.11266871   0.65790290   1.19513175   0.94289598   0.85041323 
##         Walks        Years       CAtBat        CHits       CHmRun        CRuns 
##    1.31891024   2.60105804   0.01082533   0.04671252   0.33835636   0.09359447 
##          CRBI       CWalks      LeagueA      LeagueN    DivisionW      PutOuts 
##    0.09777821   0.07182962 -10.49032970  10.48716517 -54.62727686   0.11823107 
##       Assists       Errors   NewLeagueN 
##    0.01577698  -0.72133826   6.22986245
```

```
# In-build Cross Validation (k=10 by default)
cv.ridge = cv.glmnet(x,y,alpha=0)
plot(cv.ridge)      #two dashed lines
```



Review best fit values of **lambda** lambda.min = value of lambda that gives minimum cvm (mean cross-validated error). lambda.1se = largest value of lambda st. error is within 1 std error of the min.

```
# Extract best value for lambda based on CV
cv.ridge$lambda.min
```

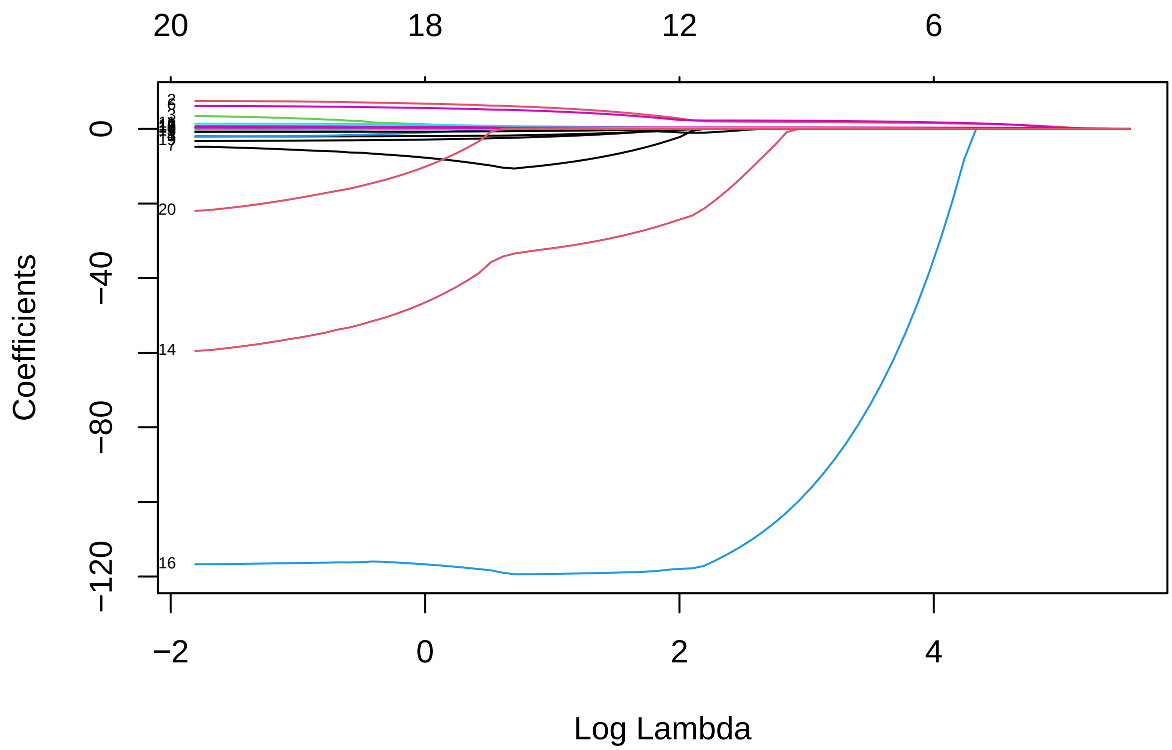
```
## [1] 238.0769
```

```
cv.ridge$lambda.1se
```

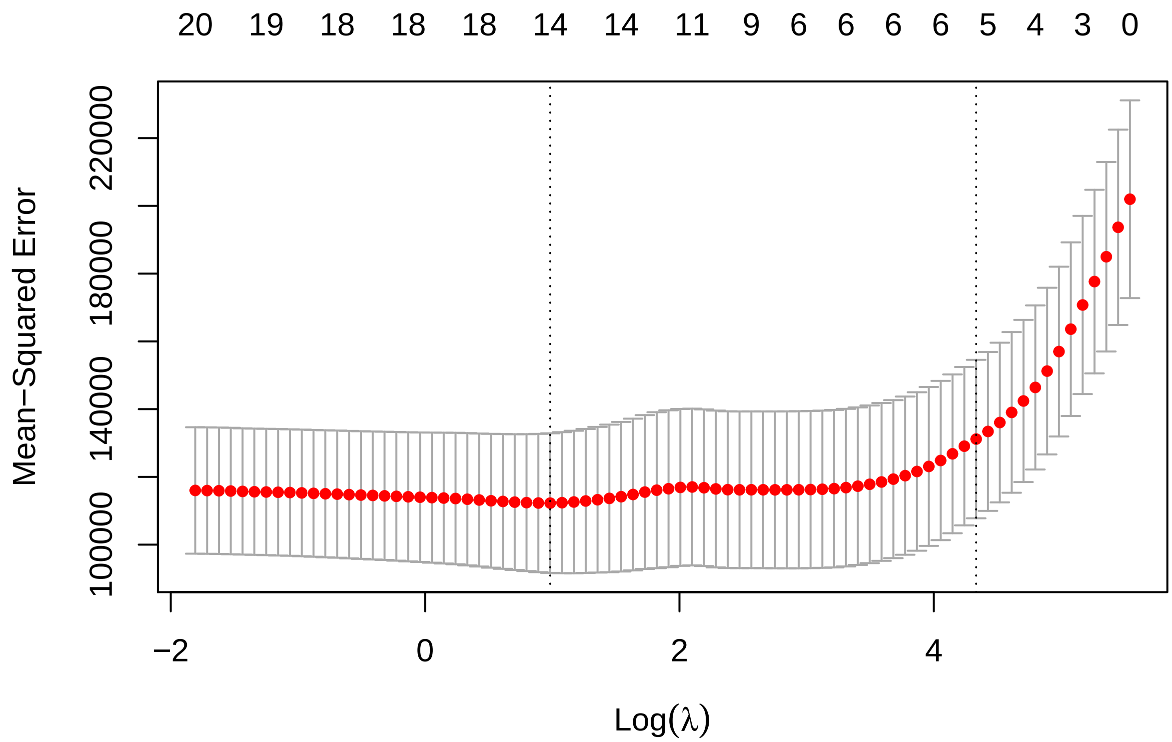
```
## [1] 2935.124
```

Lasso Penalty - shrinkage + feature selection

```
# Lasso penalty
fit.lasso = glmnet(x,y,alpha=1)
plot(fit.lasso, xvar="lambda", label=TRUE)
```




```
# In-build Cross Validation
cv.lasso = cv.glmnet(x,y,alpha=1)
plot(cv.lasso)
```




```
# Extract best value for lambda based on CV
cv.lasso$lambda.min
```

```
## [1] 2.674375
```

```
cv.lasso$lambda.1se
```

```
## [1] 76.16717
```

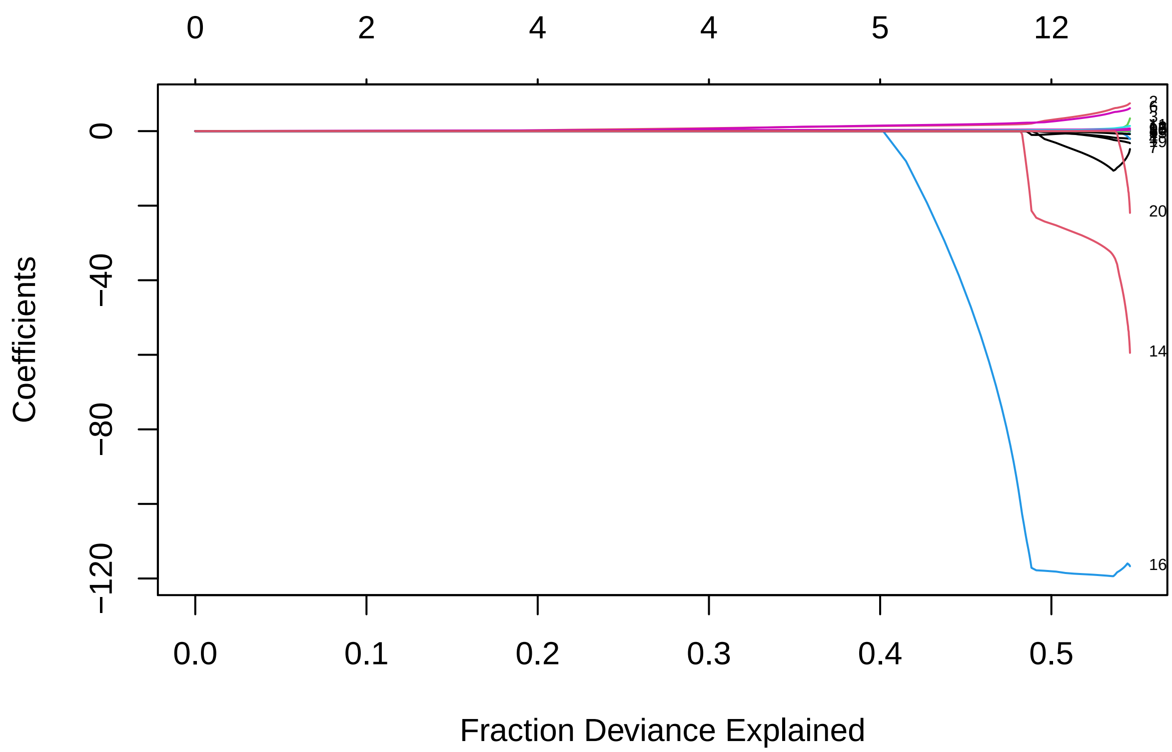
```
# Review Coeff for best fit model
coef(cv.lasso)
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 144.37970485
## AtBat         .
## Hits          1.36380384
## HmRun         .
## Runs          .
## RBI           .
## Walks         1.49731098
## Years         .
## CAtBat        .
## CHits         .
```

```
## CHmRun        .
## CRuns         0.15275165
## CRBI          0.32833941
## CWalks        .
## LeagueA       .
## LeagueN       .
## DivisionW     .
## PutOuts       0.06625755
## Assists       .
## Errors        .
## NewLeagueN    .
```

The plot shows that a lot of R2 is explained by variables with heavily shrunk coefficients And at the end, only a small improvement is caused in R2 by some big increase in coefficients, *possibly implying over-fitting*.

```
plot(fit.lasso, xvar="dev", label=TRUE)
```



Using Train/ Validation split instead fo find best model.

```
# Train/ Test approach
train.lasso = glmnet(x[train,],y[train],alpha=1)
train.lasso
```

```
##
## Call:  glmnet(x = x[train, ], y = y[train], alpha = 1)
```

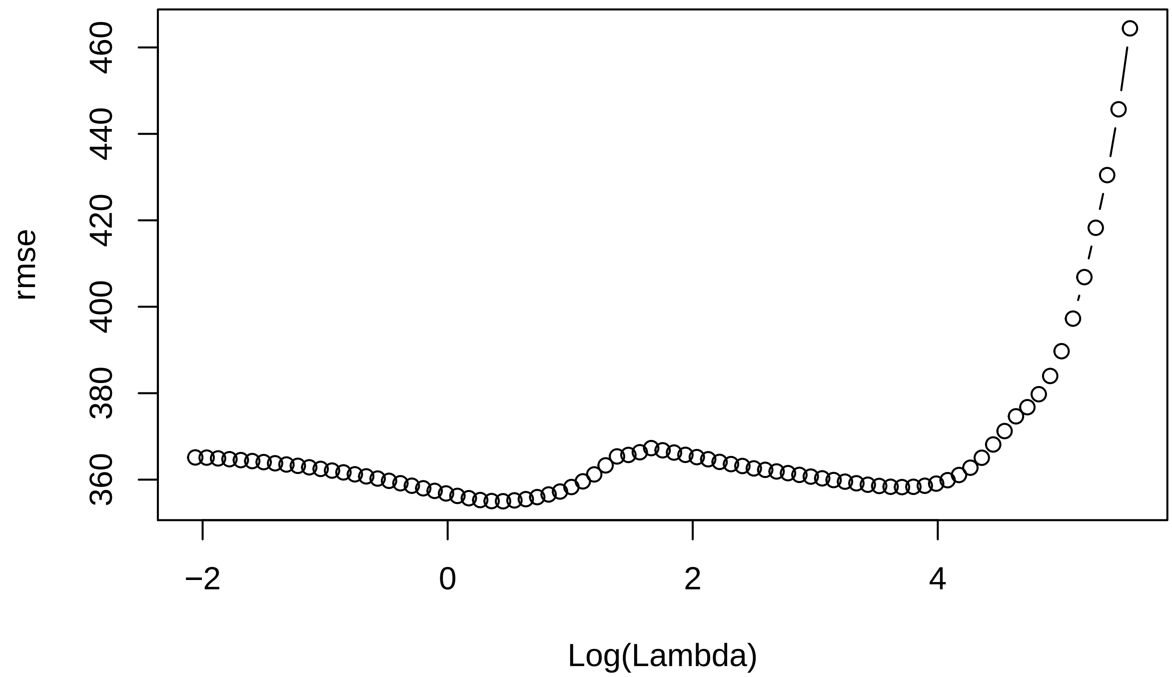
```
##
##     Df  %Dev  Lambda
## 1    0  0.00 262.100
## 2    1  5.92 238.800
## 3    1 10.83 217.600
## 4    1 14.91 198.300
## 5    2 19.72 180.600
## 6    3 23.94 164.600
## 7    3 27.45 150.000
## 8    3 30.37 136.700
## 9    3 32.79 124.500
## 10   3 34.80 113.500
## 11   4 36.50 103.400
## 12   5 38.77  94.190
## 13   6 40.90  85.820
## 14   6 42.73  78.200
## 15   6 44.25  71.250
## 16   6 45.51  64.920
## 17   6 46.55  59.150
## 18   6 47.42  53.900
## 19   6 48.14  49.110
## 20   6 48.74  44.750
## 21   6 49.24  40.770
## 22   6 49.65  37.150
## 23   6 49.99  33.850
## 24   7 50.28  30.840
## 25   7 50.51  28.100
## 26   8 50.71  25.610
## 27   8 50.94  23.330
## 28   8 51.12  21.260
## 29   8 51.28  19.370
## 30   8 51.41  17.650
## 31   8 51.52  16.080
## 32   8 51.60  14.650
## 33   8 51.68  13.350
## 34   9 51.75  12.170
## 35   9 51.99  11.080
## 36  10 52.23  10.100
## 37  10 52.44   9.202
## 38  11 52.64   8.385
## 39  11 52.82   7.640
## 40  11 52.97   6.961
## 41  11 53.09   6.343
## 42  11 53.19   5.779
## 43  12 53.28   5.266
## 44  14 53.53   4.798
## 45  14 53.83   4.372
## 46  15 54.06   3.984
## 47  16 54.45   3.630
## 48  16 54.79   3.307
## 49  16 55.06   3.013
## 50  16 55.29   2.746
## 51  17 55.48   2.502
## 52  17 55.65   2.280
```

```
## 53 17 55.78   2.077
## 54 17 55.89   1.892
## 55 18 56.00   1.724
## 56 18 56.16   1.571
## 57 18 56.30   1.432
## 58 19 56.42   1.304
## 59 19 56.53   1.189
## 60 19 56.63   1.083
## 61 19 56.71   0.987
## 62 19 56.77   0.899
## 63 19 56.83   0.819
## 64 19 56.88   0.746
## 65 19 56.92   0.680
## 66 19 56.95   0.620
## 67 19 56.98   0.565
## 68 19 57.00   0.514
## 69 19 57.02   0.469
## 70 19 57.04   0.427
## 71 19 57.05   0.389
## 72 19 57.06   0.355
## 73 19 57.07   0.323
## 74 19 57.08   0.294
## 75 19 57.08   0.268
## 76 19 57.09   0.244
## 77 19 57.09   0.223
## 78 19 57.10   0.203
## 79 19 57.10   0.185
## 80 19 57.11   0.168
## 81 19 57.11   0.154
## 82 19 57.11   0.140
## 83 19 57.11   0.127
```

```
pred = predict(train.lasso, x[-train,])
dim(pred)  #83 values of lambda and 83 rows in test data
```

```
## [1] 83 83
```

```
# RMSE
rmse = sqrt(apply((y[-train] -pred)^2, 2, mean))
plot(log(train.lasso$lambda), rmse, type="b", xlab="Log(Lambda)")
```




```
# Best Lambda
idx = which.min(rmse)
train.lasso$lambda[idx]
```

```
## [1] 1.571184
```

Principal Components Regression (PCR)

Setting **scale = TRUE** has the effect of standardizing each predictor. Setting **validation = "CV"** causes pcr() to compute the ten-fold cross-validation error for each possible value of M, the number of principal components used.

Note that pcr() reports the root mean squared error.

```
set.seed(2)
pcr.fit <- pcr(Salary~., data=d_Hitters, scale=TRUE, validation="CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
```

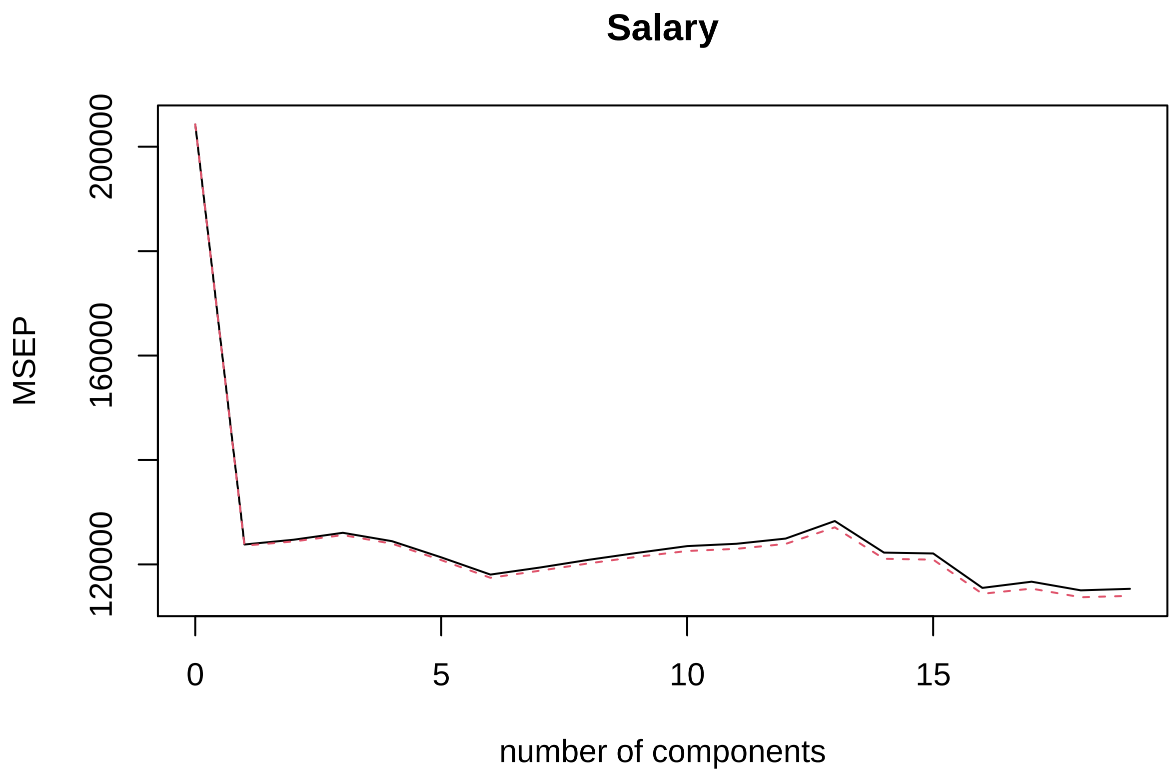
```
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    351.9    353.2    355.0    352.8    348.4    343.6
## adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
## adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        349.7     349.4     339.9     341.6     339.2     339.6
## adjCV     348.0     347.7     338.2     339.7     337.2     337.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

```
pcr.fit$loadings
```

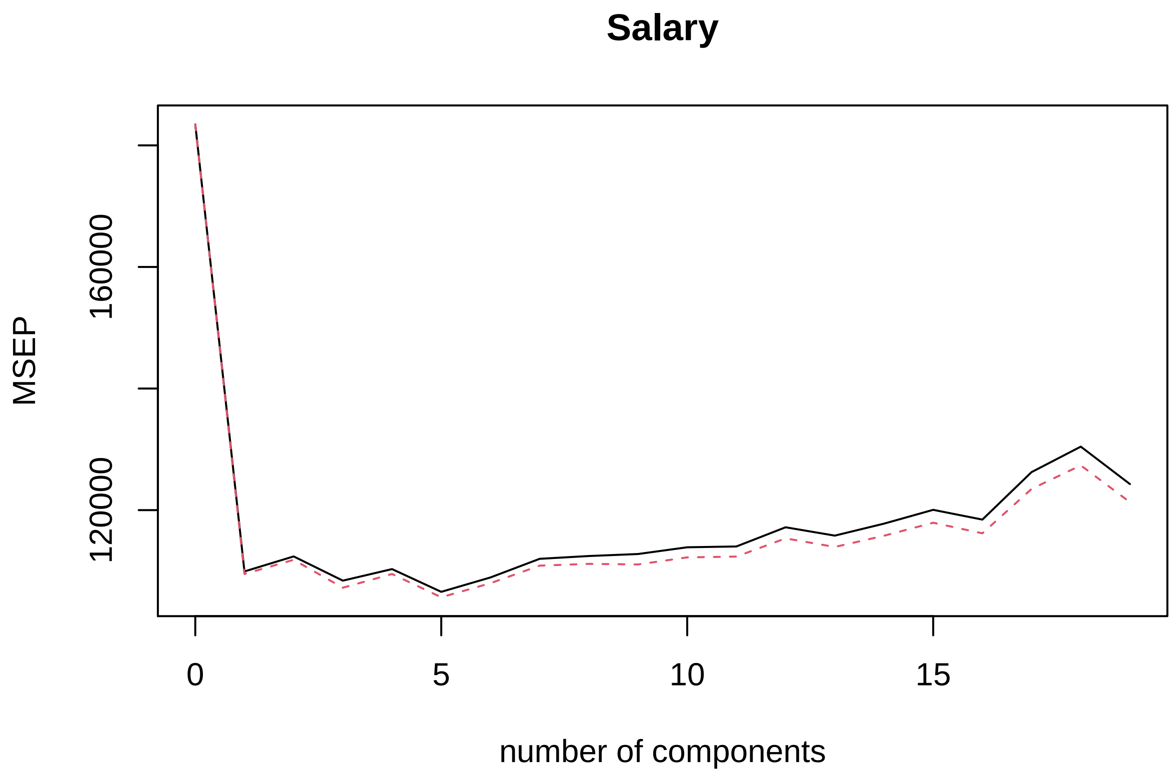
```
##
## Loadings:
##            Comp 1 Comp 2 Comp 3 Comp 4 Comp 5 Comp 6 Comp 7 Comp 8 Comp 9
## AtBat       0.198  0.384                             -0.107  0.270
## Hits        0.196  0.377                             -0.130  0.389
## HmRun       0.204  0.237  0.216 -0.236         0.150  0.506 -0.226  0.127
## Runs        0.198  0.378                      0.137 -0.202  0.115 -0.171
## RBI         0.235  0.315        -0.139         0.112  0.319         0.131
## Walks       0.209  0.230        -0.131               -0.558 -0.623
## Years       0.283 -0.262                                     0.138
## CAtBat      0.330 -0.193                                     0.147
## CHits       0.331 -0.183                                     0.195
## CHmRun      0.319 -0.126                              0.229 -0.249  0.168
## CRuns       0.338 -0.172
## CRBI        0.340 -0.168                              0.119
## CWalks      0.317 -0.192                             -0.178 -0.263
## LeagueN                   -0.548 -0.396         0.137
## DivisionW                               -0.986        -0.113
## PutOuts            0.156        -0.288 -0.106 -0.924
## Assists            0.169 -0.398  0.524                              0.707
## Errors             0.201 -0.383  0.422        -0.148  0.373 -0.301 -0.609
## NewLeagueN                -0.545 -0.418         0.157
##            Comp 10 Comp 11 Comp 12 Comp 13 Comp 14 Comp 15 Comp 16 Comp 17
## AtBat       0.146          -0.103           0.306  -0.532   0.510  -0.139
## Hits        0.130          -0.121           0.211          -0.720   0.167
## HmRun      -0.351  -0.202   0.315   0.109          -0.355  -0.200
## Runs               -0.312   0.322   0.381  -0.267   0.468   0.221  -0.141
## RBI        -0.172   0.243  -0.348  -0.440           0.461   0.237   0.107
## Walks      -0.121   0.176  -0.185          -0.238  -0.177  -0.103
## Years      -0.513   0.192  -0.355   0.605
```

```
## CAtBat     -0.101                 -0.149  -0.168  -0.158          -0.182
## CHits                              -0.267  -0.290  -0.137  -0.110
## CHmRun      0.651                   0.330                           0.292
## CRuns              -0.152   0.229  -0.202  -0.129           0.162   0.623
## CRBI        0.281          -0.121          -0.209          -0.143  -0.610
## CWalks             -0.191   0.216  -0.167   0.737   0.245          -0.170
## LeagueN            -0.581  -0.407
## DivisionW
## PutOuts
## Assists
## Errors
## NewLeagueN          0.544   0.429
##            Comp 18 Comp 19
## AtBat       0.107
## Hits
## HmRun
## Runs
## RBI
## Walks
## Years
## CAtBat     -0.720  -0.409
## CHits               0.770
## CHmRun     -0.254   0.166
## CRuns       0.400  -0.344
## CRBI        0.475  -0.260
## CWalks
## LeagueN
## DivisionW
## PutOuts
## Assists
## Errors
## NewLeagueN
##
##                Comp 1 Comp 2 Comp 3 Comp 4 Comp 5 Comp 6 Comp 7 Comp 8 Comp 9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.053  0.053  0.053  0.053  0.053  0.053  0.053  0.053  0.053
## Cumulative Var  0.053  0.105  0.158  0.211  0.263  0.316  0.368  0.421  0.474
##                Comp 10 Comp 11 Comp 12 Comp 13 Comp 14 Comp 15 Comp 16 Comp 17
## SS loadings      1.000   1.000   1.000   1.000   1.000   1.000   1.000   1.000
## Proportion Var   0.053   0.053   0.053   0.053   0.053   0.053   0.053   0.053
## Cumulative Var   0.526   0.579   0.632   0.684   0.737   0.789   0.842   0.895
##                Comp 18 Comp 19
## SS loadings      1.000   1.000
## Proportion Var   0.053   0.053
## Cumulative Var   0.947   1.000
```

```
  # sum(pcr.fit$loadings[,1]^2)   # sum of square of coeffs for any PC adds up to one
  # sum(pcr.fit$loadings[,6]^2)   # sum of square of coeffs for any PC adds up to one
validationplot(pcr.fit, val.type="MSEP")
```




```r
#Using Train/ Test split - 5 component model has best fit
set.seed (1)
train <- sample(1:nrow(d_Hitters), nrow(d_Hitters)/2)
x <- model.matrix(Salary ., d_Hitters)[,-1]
y <- d_Hitters[,"Salary"]

pcr.fit.train <- pcr(Salary~., data=d_Hitters, scale=TRUE, validation="CV",
                     subset=train)   #using both training and CV!
validationplot(pcr.fit.train, val.type="MSEP")
```




```
pcr.pred <- predict(pcr.fit.train, x[-train,], ncomp=5)
mean((pcr.pred - d_Hitters[-train,"Salary"])^2)
```

```
## [1] 142811.8
```

```
#Fitting 5 component model on full-dataset
pcr.fit.5 <- pcr(Salary~., data=d_Hitters, scale=TRUE, ncomp=5)
summary(pcr.fit.5)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps
## X         38.31    60.16    70.84    79.03    84.29
## Salary    40.63    41.58    42.17    43.22    44.90
```

Partial Least Square (PLS) Regression

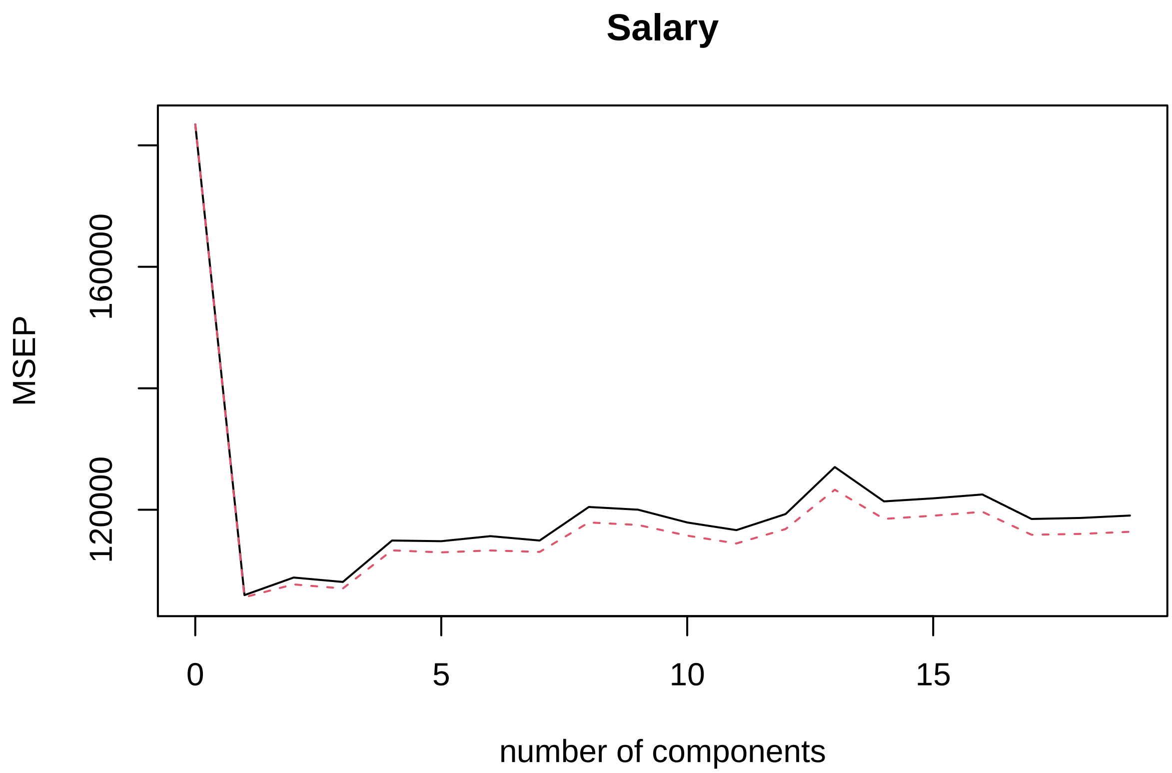
Setting **scale = TRUE** has the effect of standardizing each predictor. Setting **validation = "CV"** causes pcr() to compute the ten-fold cross-validation error for each possible value of M, the number of principal components used.

Note that pcr() reports the root mean squared error.

```r
set.seed(1)
pls.fit <- plsr(Salary~., data=d_Hitters, subset=train, scale=TRUE, validation="CV")
summary(pls.fit)
```

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    325.5    329.9    328.8    339.0    338.9    340.1
## adjCV        428.3    325.0    328.2    327.2    336.6    336.1    336.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       339.0    347.1    346.4     343.4     341.5     345.4     356.4
## adjCV    336.2    343.4    342.8     340.2     338.3     341.8     351.1
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        348.4     349.1     350.0     344.2     344.5     345.0
## adjCV     344.2     345.0     345.9     340.4     340.6     341.1
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         39.13    48.80    60.09    75.07    78.58    81.12    88.21    90.71
## Salary    46.36    50.72    52.23    53.03    54.07    54.77    55.05    55.66
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         93.17     96.05     97.08     97.61     97.97     98.70     99.12
## Salary    55.95     56.12     56.47     56.68     57.37     57.76     58.08
##         16 comps  17 comps  18 comps  19 comps
## X          99.61     99.70     99.95    100.00
## Salary     58.17     58.49     58.56     58.62
```

```r
validationplot(pls.fit, val.type="MSEP")
```




```r
# Performance on Test Set
pls.pred <- predict(pls.fit, x[-train,], ncomp=1)
mean((pls.pred - d_Hitters[-train,"Salary"])^2)
```

```
## [1] 151995.3
```

```r
# Performance on Full dataset
pls.fit.full <- plsr(Salary~., data=d_Hitters, scale=TRUE, ncomp=1)
summary(pls.fit.full)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 1
## TRAINING: % variance explained
##         1 comps
## X         38.08
## Salary    43.05
```

Notice that the percentage of variance in Salary that the one-component PLS fit explains, 43.05 %, is almost as much as that explained using the final five-component model PCR fit, 44.90 %. This is because PCR only attempts to maximize the amount of variance explained in the predictors, while PLS searches for directions that explain variance in both the predictors and the response.