Ch 5 1 ModelSelect

Analyze the ISLR (Introduction to Statistical Learning with R) data package's baseball 'Hitters' data frame:

```r
library(ISLR)
summary(Hitters)
```

```
##      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                
```

There are missing values, before we proceed we will remove them:

```r
with(Hitters, sum(is.na(Salary)))
```

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

```r
Hitters=na.omit(Hitters)
with(Hitters, sum(is.na(Salary)))
```

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

Best Subset regression

We will now use the package `leaps` to evaluate all the best-subset models.

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

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns
## 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 ) " "  "*"    " "     "*"       "*"     " "     " "    " "
```

By default, it gives the first 8 variables best-subset models. Let's do it again for all the variables:

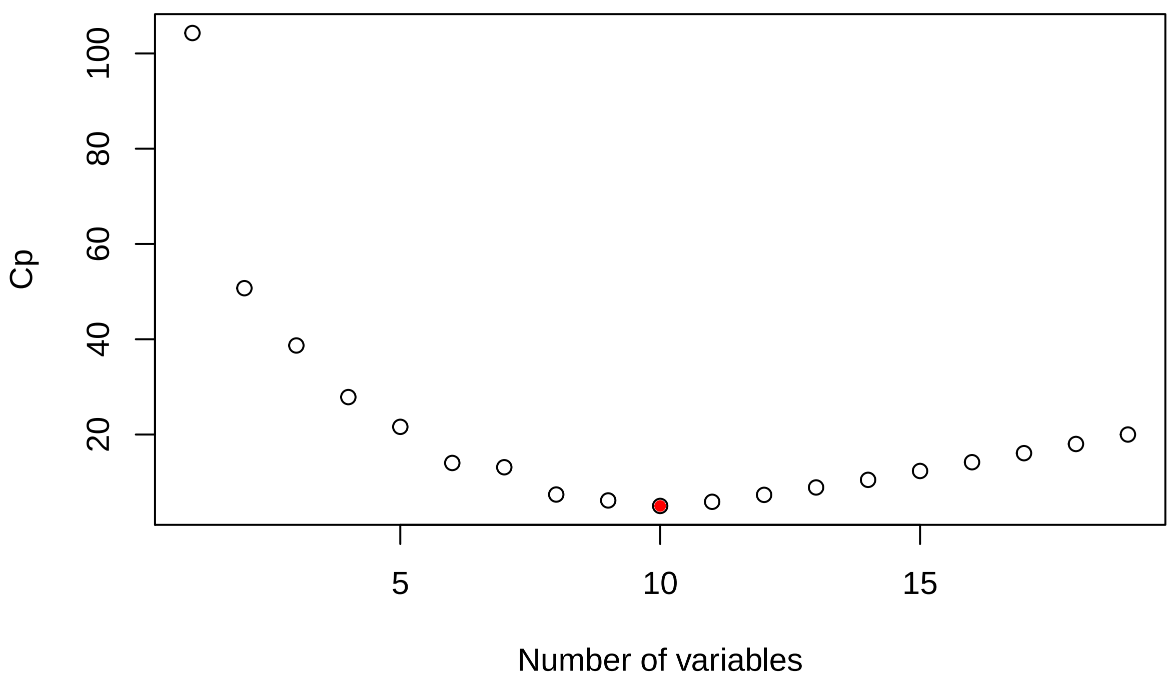
```r
regfit.full = regsubsets(Salary~., data=Hitters, nvmax=19)
reg.summary = summary(regfit.full)
names(reg.summary)
```

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

```r
plot(reg.summary$cp, xlab="Number of variables", ylab="Cp")
which.min(reg.summary$cp)
```

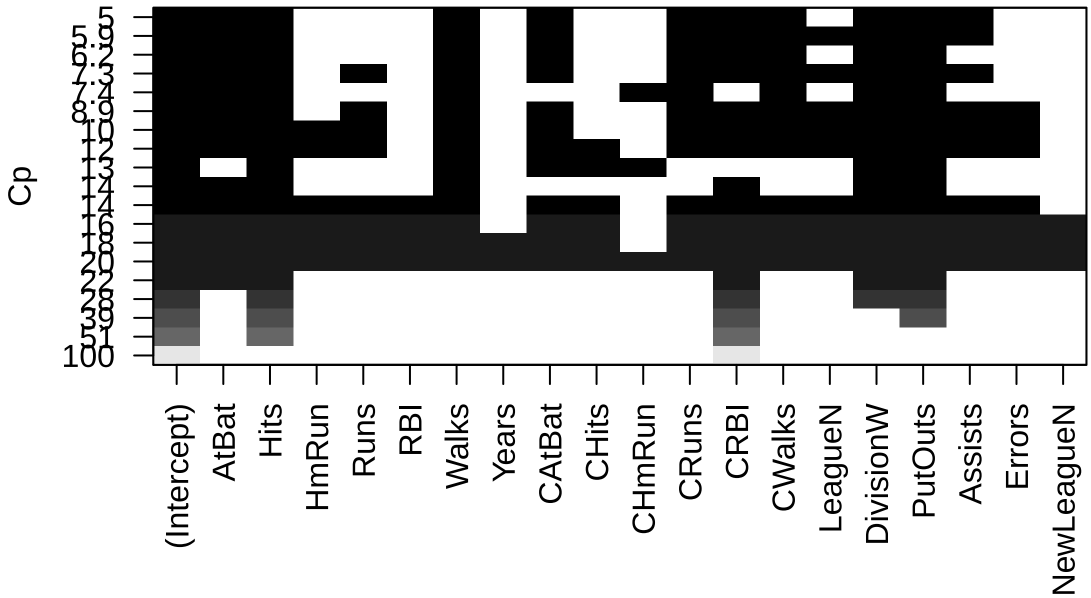
```
## [1] 10
```

```r
points(10, reg.summary$cp[10], pch=20, col="red")
```



There is a method for the `regsubset` object:

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




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

Forward Stepwise Selection

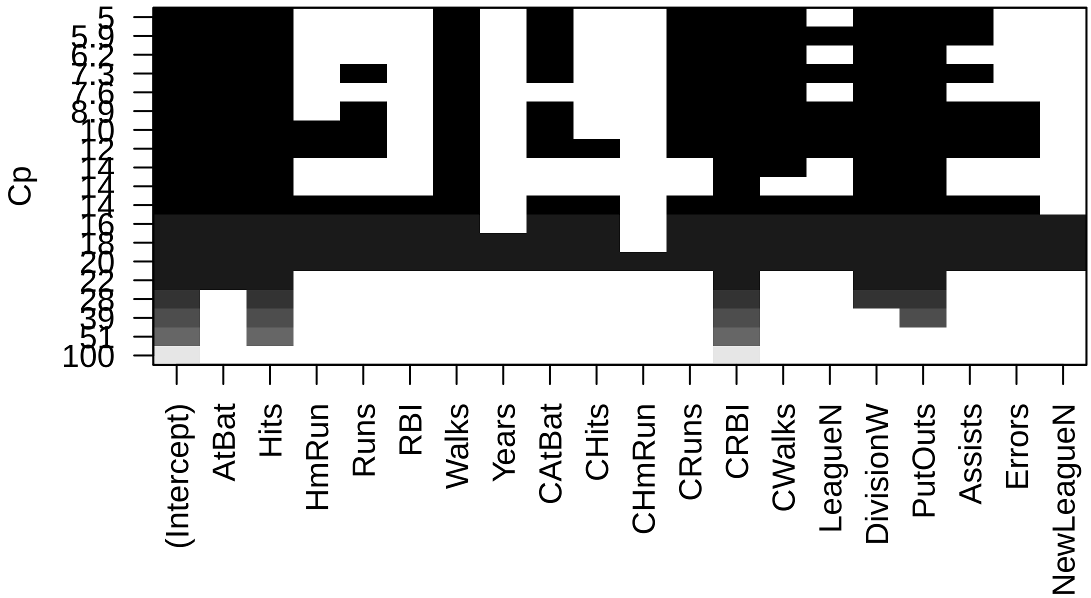
We use `regsubset` again, but specify the `method="forward"` option.

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

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



Model Selection Using a Validation Set

Let's make a training and validation set, so that we can choose a good subset model.

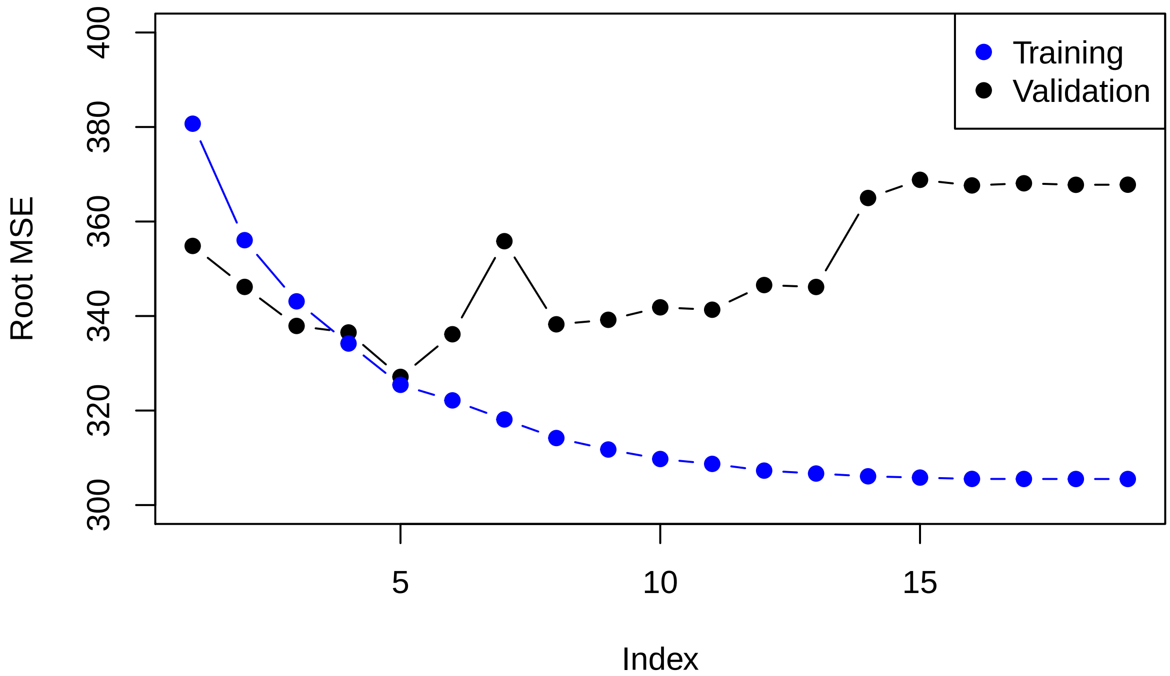
```
dim(Hitters)
```

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

```
set.seed(1)
train = sample(seq(263),180,replace=FALSE)
regfit.fwd = regsubsets(Salary~., data=Hitters[train,], nvmax=19, method="forward")
```

Now, we separate the data to two parts, one for training and another for test/validation to make prediction. There is no `prediction` method for `regsubsets`, so we need to write that part. We also create a vector to store the results of the 19 different models.

```
val.errors = rep(NA, 19)
x.test = model.matrix(Salary~., data=Hitters[-train,])
for(i in 1:19){
  coefi = coef(regfit.fwd, id=i)
  pred = x.test[,names(coefi)]%*%coefi
  val.errors[i] = mean((Hitters$Salary[-train]-pred)^2)
}
plot(sqrt(val.errors), ylab="Root MSE", ylim=c(300,400), pch=19, type="b")
points(sqrt(regfit.fwd$rss[-1]/180),col="blue",pch=19,type="b") # -1 excludes null model
legend("topright", legend=c("Training","Validation"),col=c("blue","black"),pch=19)
```



As expected, the model error goes down monotonically as the model gets bigges, but not so for the validation error.

This was a little tedious - not having a `predict` method `regsubsets`. So we will write one!

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

```
## Salary ~ .
```

Model Seletion by Cross-Validation

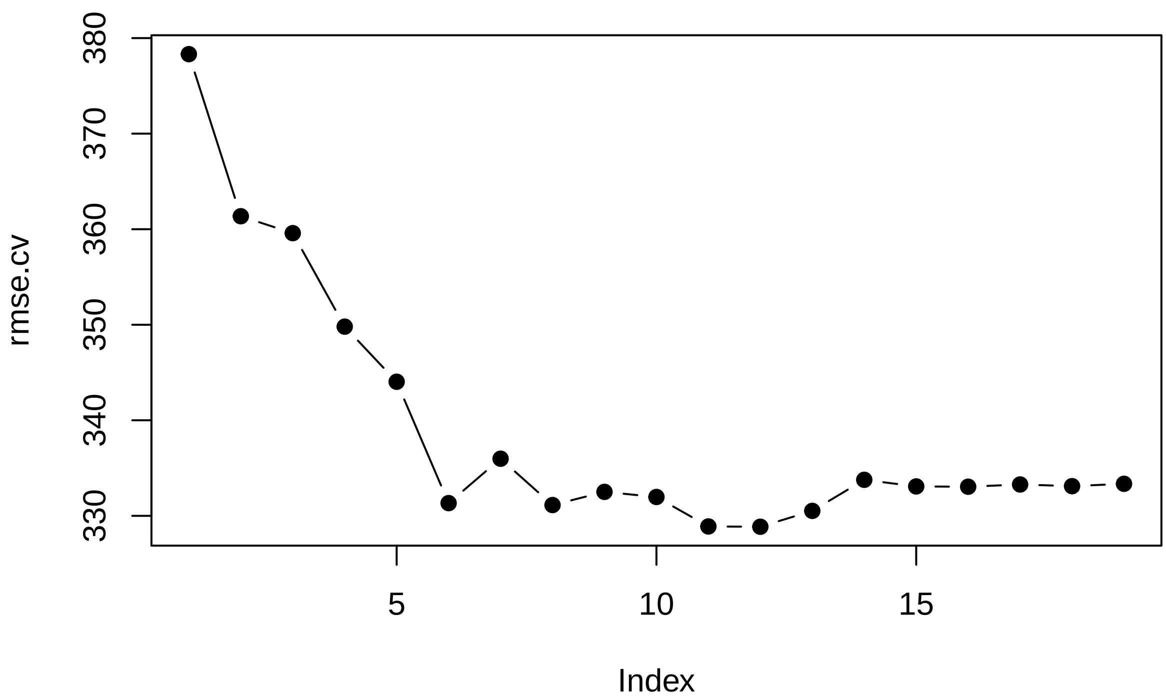
We will do a 10-fold cross-validation.

```
set.seed(11)
folds = sample(rep(1:10,length=nrow(Hitters)))
table(folds)
```

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

```
cv.errors = matrix(NA,10,19)
for(k in 1:10){ # Loop for folds
  best.fit = regsubsets(Salary~., data=Hitters[folds!=k,],nvmax=19,method="forward")
  for(i in 1:19){ # Loop for sizes of picked feature best subsets
```

```
    pred = predict(best.fit,Hitters[folds==k,], id=i)
    cv.errors[k,i] = mean((Hitters$Salary[folds==k]-pred)^2)
  }
}
rmse.cv = sqrt(apply(cv.errors,2,mean))
plot(rmse.cv, pch=19, type="b")
```



Ridge regression and the Lasso

We will use the `glmnet` package, which does not use the formula language, so we will set up an `x` and `y`.

```
library(glmnet)
```

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

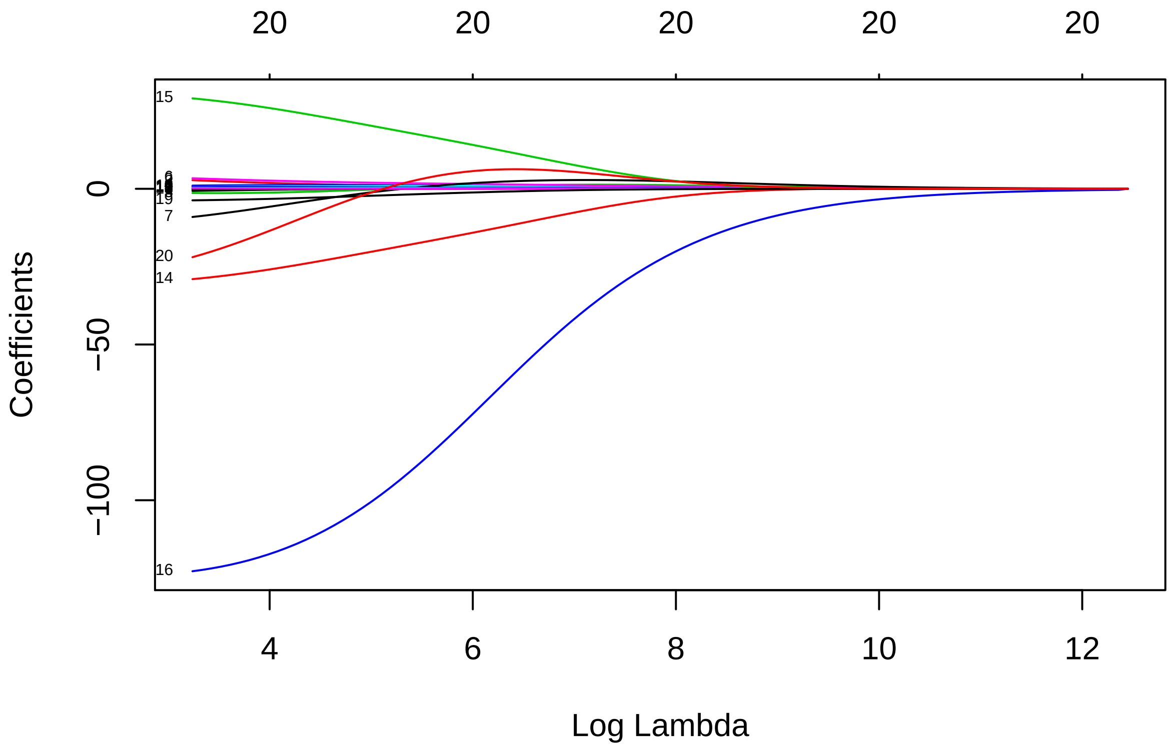
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-2
```

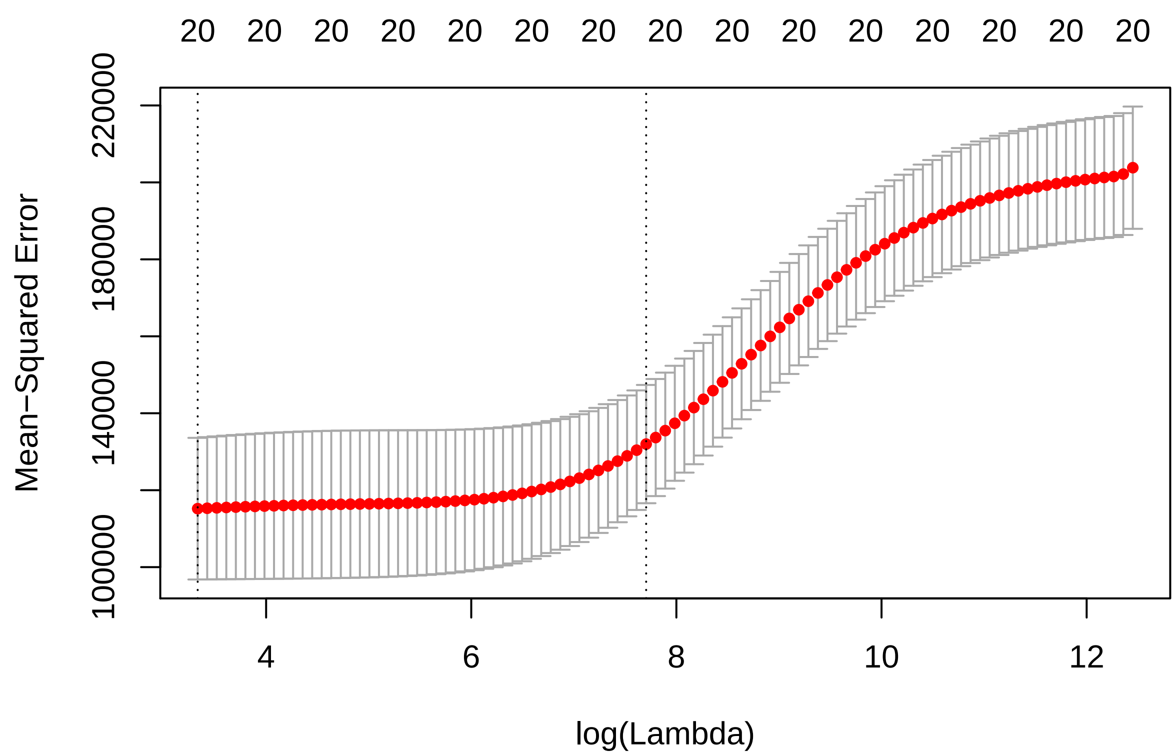
```
x=model.matrix(Salary~.-1, data=Hitters) # -1 means no intercept
y=Hitters$Salary
```

First, we will fit a ridge regression model. This is achived by calling `glmnet` with `alpha=0` (see the help file). There is also a `cv.glmnet` function, which will do the cross validation for us.

```
fit.ridge = glmnet(x,y,alpha=0)
plot(fit.ridge, xvar="lambda", label=TRUE)
```

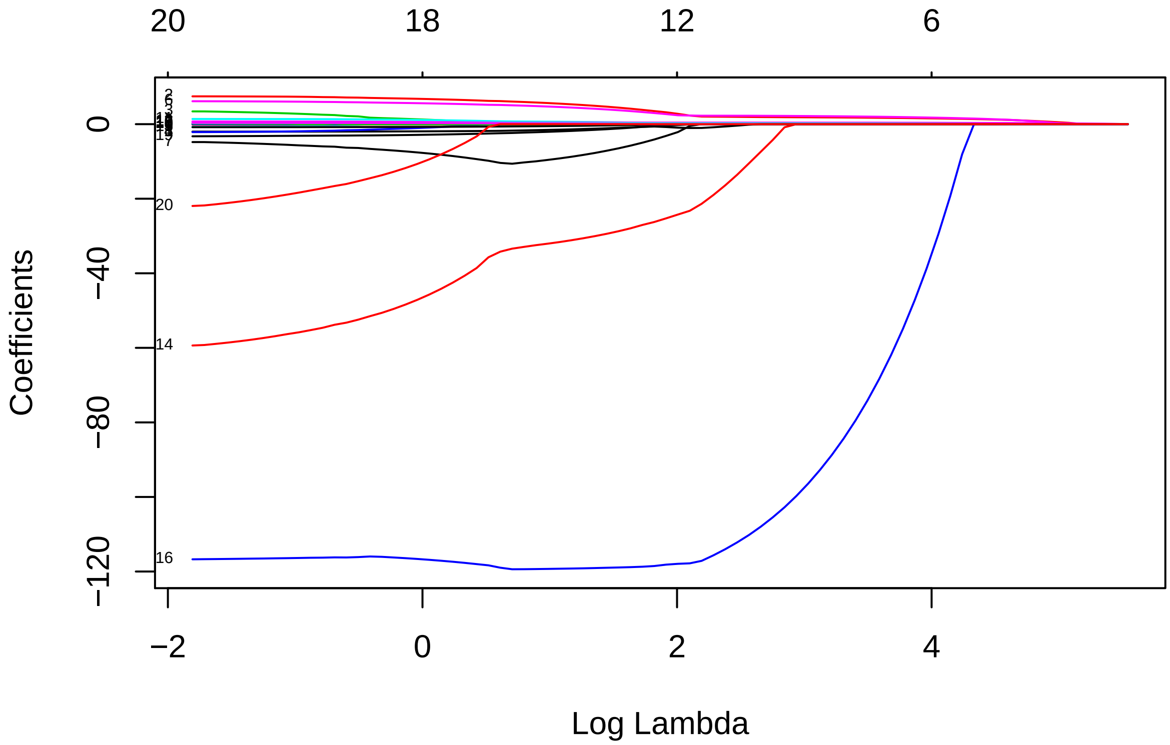



```
cv.ridge = cv.glmnet(x,y,alpha=0)
plot(cv.ridge)
```

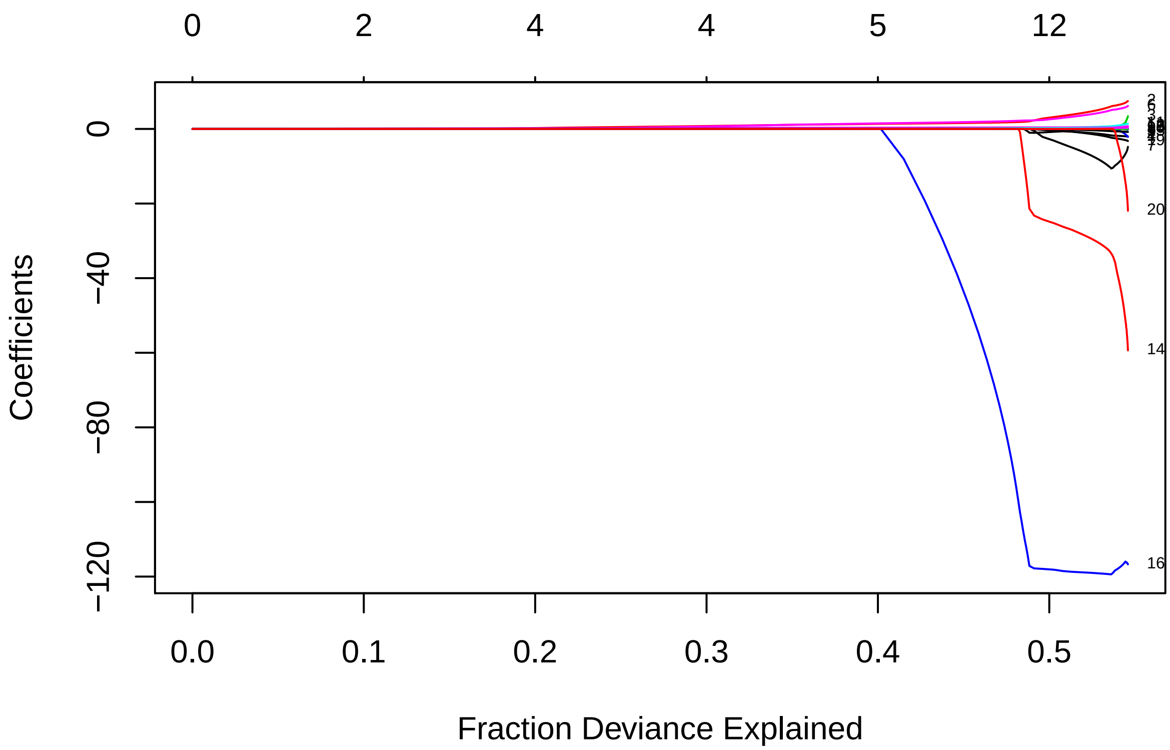


Now, we fit the lasso; in `glmnet` it means `alpha=1`

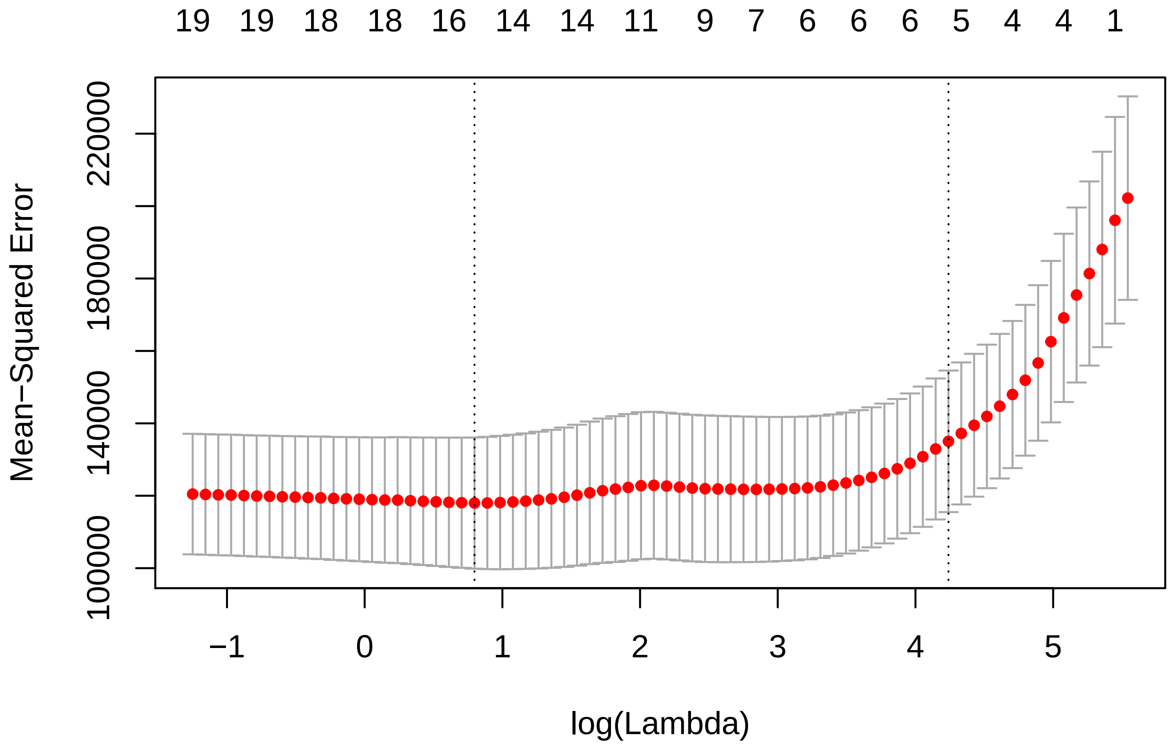
```r
fit.lasso = glmnet(x,y)
plot(fit.lasso, xvar="lambda", label=TRUE)
```




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



```r
cv.lasso = cv.glmnet(x,y)
plot(cv.lasso)
```




```r
coef(cv.lasso) # Picks model one std from minimum to make more parsimonius
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 127.95694754
## AtBat          .
## Hits           1.42342566
## HmRun          .
## Runs           .
## RBI            .
## Walks          1.58214111
## Years          .
## CAtBat         .
## CHits          .
## CHmRun         .
## CRuns          0.16027975
## CRBI           0.33667715
## CWalks         .
## LeagueA        .
## LeagueN        .
## DivisionW     -8.06171262
## PutOuts        0.08393604
## Assists        .
## Errors         .
## NewLeagueN     .
```

Suppose we want to our earlier train/validation division to select the `lambda` for the lasso.

```r
lasso.tr = glmnet(x[train,], y[train])
lasso.tr
```

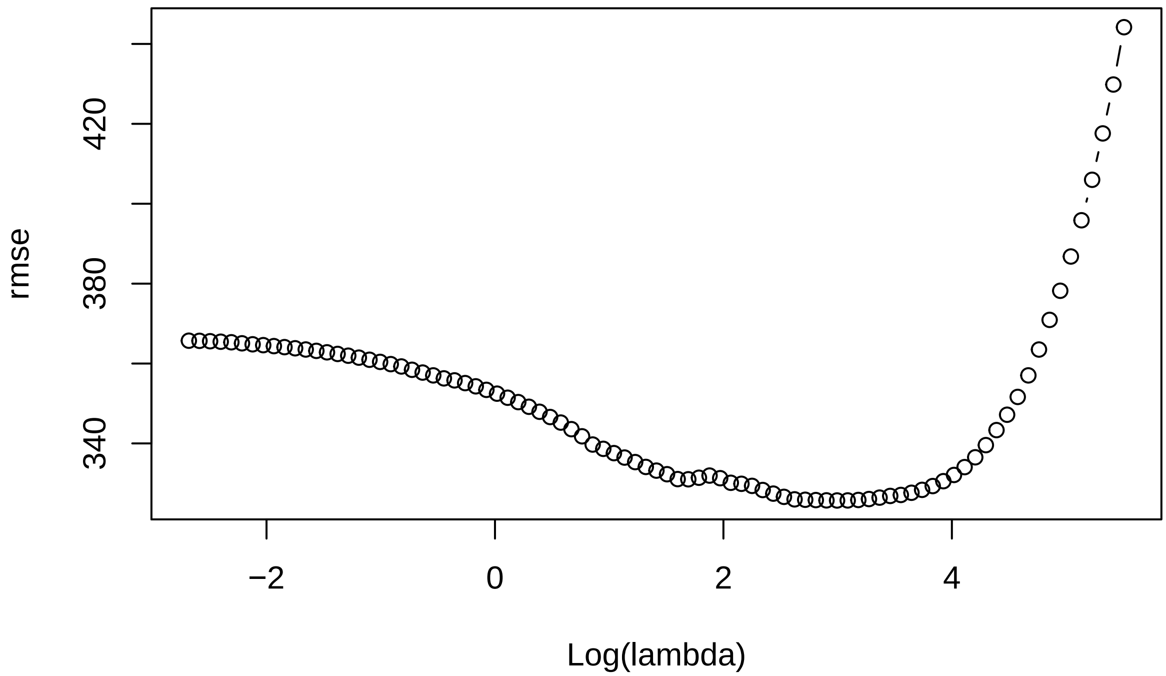
```
##
## Call:  glmnet(x = x[train, ], y = y[train])
##
##       Df    %Dev    Lambda
##  [1,]  0 0.00000 246.40000
##  [2,]  1 0.05013 224.50000
##  [3,]  1 0.09175 204.60000
##  [4,]  2 0.13840 186.40000
##  [5,]  2 0.18000 169.80000
##  [6,]  3 0.21570 154.80000
##  [7,]  3 0.24710 141.00000
##  [8,]  3 0.27320 128.50000
##  [9,]  4 0.30010 117.10000
## [10,]  4 0.32360 106.70000
## [11,]  4 0.34310  97.19000
## [12,]  4 0.35920  88.56000
## [13,]  5 0.37360  80.69000
## [14,]  5 0.38900  73.52000
## [15,]  5 0.40190  66.99000
## [16,]  5 0.41260  61.04000
## [17,]  5 0.42140  55.62000
## [18,]  5 0.42880  50.67000
## [19,]  5 0.43490  46.17000
## [20,]  5 0.43990  42.07000
## [21,]  5 0.44410  38.33000
## [22,]  5 0.44760  34.93000
## [23,]  6 0.45140  31.83000
## [24,]  7 0.45480  29.00000
## [25,]  7 0.45770  26.42000
## [26,]  7 0.46010  24.07000
## [27,]  8 0.46220  21.94000
## [28,]  8 0.46380  19.99000
## [29,]  8 0.46520  18.21000
## [30,]  8 0.46630  16.59000
## [31,]  8 0.46730  15.12000
## [32,]  8 0.46810  13.78000
## [33,]  9 0.47110  12.55000
## [34,]  9 0.47380  11.44000
## [35,]  9 0.47620  10.42000
## [36,] 10 0.48050   9.49500
## [37,]  9 0.48450   8.65200
## [38,] 10 0.48770   7.88300
## [39,] 10 0.49360   7.18300
## [40,] 11 0.49890   6.54500
## [41,] 12 0.50450   5.96300
## [42,] 12 0.51010   5.43400
## [43,] 13 0.51470   4.95100
## [44,] 13 0.51850   4.51100
## [45,] 13 0.52170   4.11000
```

```
## [46,] 14 0.52440   3.74500
## [47,] 14 0.52670   3.41200
## [48,] 15 0.52870   3.10900
## [49,] 15 0.53030   2.83300
## [50,] 15 0.53160   2.58100
## [51,] 16 0.53280   2.35200
## [52,] 17 0.53420   2.14300
## [53,] 18 0.53580   1.95300
## [54,] 18 0.53760   1.77900
## [55,] 18 0.53890   1.62100
## [56,] 18 0.54000   1.47700
## [57,] 18 0.54090   1.34600
## [58,] 18 0.54160   1.22600
## [59,] 18 0.54220   1.11700
## [60,] 18 0.54280   1.01800
## [61,] 18 0.54320   0.92770
## [62,] 18 0.54360   0.84530
## [63,] 18 0.54380   0.77020
## [64,] 19 0.54410   0.70180
## [65,] 19 0.54430   0.63940
## [66,] 19 0.54450   0.58260
## [67,] 19 0.54470   0.53090
## [68,] 19 0.54490   0.48370
## [69,] 20 0.54510   0.44070
## [70,] 20 0.54520   0.40160
## [71,] 20 0.54530   0.36590
## [72,] 20 0.54540   0.33340
## [73,] 20 0.54550   0.30380
## [74,] 20 0.54560   0.27680
## [75,] 20 0.54570   0.25220
## [76,] 20 0.54570   0.22980
## [77,] 20 0.54580   0.20940
## [78,] 20 0.54580   0.19080
## [79,] 20 0.54590   0.17380
## [80,] 20 0.54590   0.15840
## [81,] 20 0.54590   0.14430
## [82,] 20 0.54590   0.13150
## [83,] 20 0.54600   0.11980
## [84,] 19 0.54600   0.10920
## [85,] 19 0.54600   0.09948
## [86,] 19 0.54600   0.09064
## [87,] 19 0.54600   0.08259
## [88,] 20 0.54600   0.07525
## [89,] 20 0.54600   0.06856
```

```r
pred = predict(lasso.tr, x[-train,])
dim(pred)
```

```
## [1] 83 89
```

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




```r
lam.best = lasso.tr$lambda[order(rmse)[1]] # Pick best lambda
lam.best
```

```
## [1] 19.98706
```

```r
coef(lasso.tr, s=lam.best) # sparse matrix format
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  107.9416686
## AtBat          .
## Hits           0.1591252
## HmRun          .
## Runs           .
## RBI            1.7340039
## Walks          3.4657091
## Years          .
## CAtBat         .
## CHits          .
## CHmRun         .
## CRuns          0.5386855
## CRBI           .
## CWalks         .
## LeagueA      -30.0493021
## LeagueN        .
## DivisionW   -113.8317016
## PutOuts        0.2915409
## Assists        .
## Errors         .
## NewLeagueN     2.0367518
```