model Selection methods

Vinoth Aryan Nagabosshanam

March 25, 2017

Model Selection

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.3.3
```

```r
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
```

affter see the summary its clear that the na values in the salary variable to remove that we using the foolowing code

```r
Hitters=na.omit(Hitters)
summary(Hitters)
```

```
##      AtBat            Hits           HmRun            Runs
##  Min.   : 19.0   Min.   :  1.0   Min.   : 0.00   Min.   :  0.00
##  1st Qu.:282.5   1st Qu.: 71.5   1st Qu.: 5.00   1st Qu.: 33.50
##  Median :413.0   Median :103.0   Median : 9.00   Median : 52.00
##  Mean   :403.6   Mean   :107.8   Mean   :11.62   Mean   : 54.75
##  3rd Qu.:526.0   3rd Qu.:141.5   3rd Qu.:18.00   3rd Qu.: 73.00
##  Max.   :687.0   Max.   :238.0   Max.   :40.00   Max.   :130.00
##       RBI            Walks            Years           CAtBat
##  Min.   :  0.00   Min.   :  0.00   Min.   : 1.000   Min.   :   19.0
##  1st Qu.: 30.00   1st Qu.: 23.00   1st Qu.: 4.000   1st Qu.:  842.5
##  Median : 47.00   Median : 37.00   Median : 6.000   Median : 1931.0
##  Mean   : 51.49   Mean   : 41.11   Mean   : 7.312   Mean   : 2657.5
##  3rd Qu.: 71.00   3rd Qu.: 57.00   3rd Qu.:10.000   3rd Qu.: 3890.5
##  Max.   :121.00   Max.   :105.00   Max.   :24.000   Max.   :14053.0
##      CHits           CHmRun           CRuns            CRBI
##  Min.   :   4.0   Min.   :  0.00   Min.   :   2.0   Min.   :   3.0
##  1st Qu.: 212.0   1st Qu.: 15.00   1st Qu.: 105.5   1st Qu.:  95.0
##  Median : 516.0   Median : 40.00   Median : 250.0   Median : 230.0
##  Mean   : 722.2   Mean   : 69.24   Mean   : 361.2   Mean   : 330.4
##  3rd Qu.:1054.0   3rd Qu.: 92.50   3rd Qu.: 497.5   3rd Qu.: 424.5
##  Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.0
##      CWalks       League  Division   PutOuts          Assists
##  Min.   :   1.0   A:139   E:129   Min.   :   0.0   Min.   :  0.0
##  1st Qu.:  71.0   N:124   W:134   1st Qu.: 113.5   1st Qu.:  8.0
##  Median : 174.0                   Median : 224.0   Median : 45.0
##  Mean   : 260.3                   Mean   : 290.7   Mean   :118.8
##  3rd Qu.: 328.5                   3rd Qu.: 322.5   3rd Qu.:192.0
##  Max.   :1566.0                   Max.   :1377.0   Max.   :492.0
##      Errors          Salary       NewLeague
##  Min.   : 0.000   Min.   :  67.5   A:141
##  1st Qu.: 3.000   1st Qu.: 190.0   N:122
##  Median : 7.000   Median : 425.0
##  Mean   : 8.593   Mean   : 535.9
##  3rd Qu.:13.000   3rd Qu.: 750.0
##  Max.   :32.000   Max.   :2460.0
```

Best Subset regression

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

```r
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.3.3
```

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

It gives by default best-subsets up to size 8; lets increase that to 19, i.e. all the variables

```
regfit.full=regsubsets(Salary~.,data=Hitters, nvmax=19)
reg.summary=summary(regfit.full)
names(reg.summary)
```
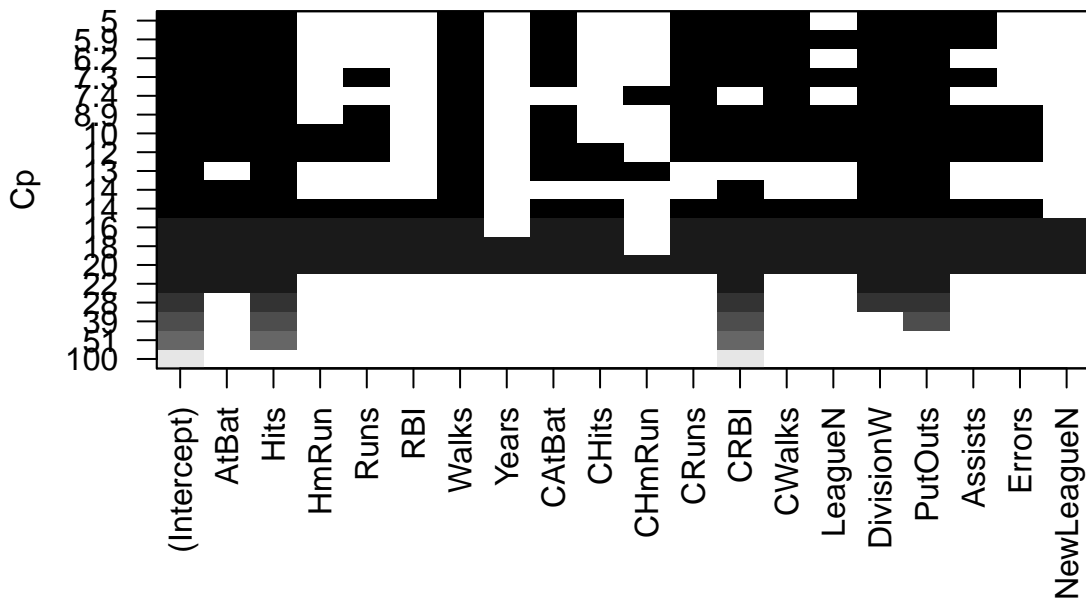
```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
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")
```
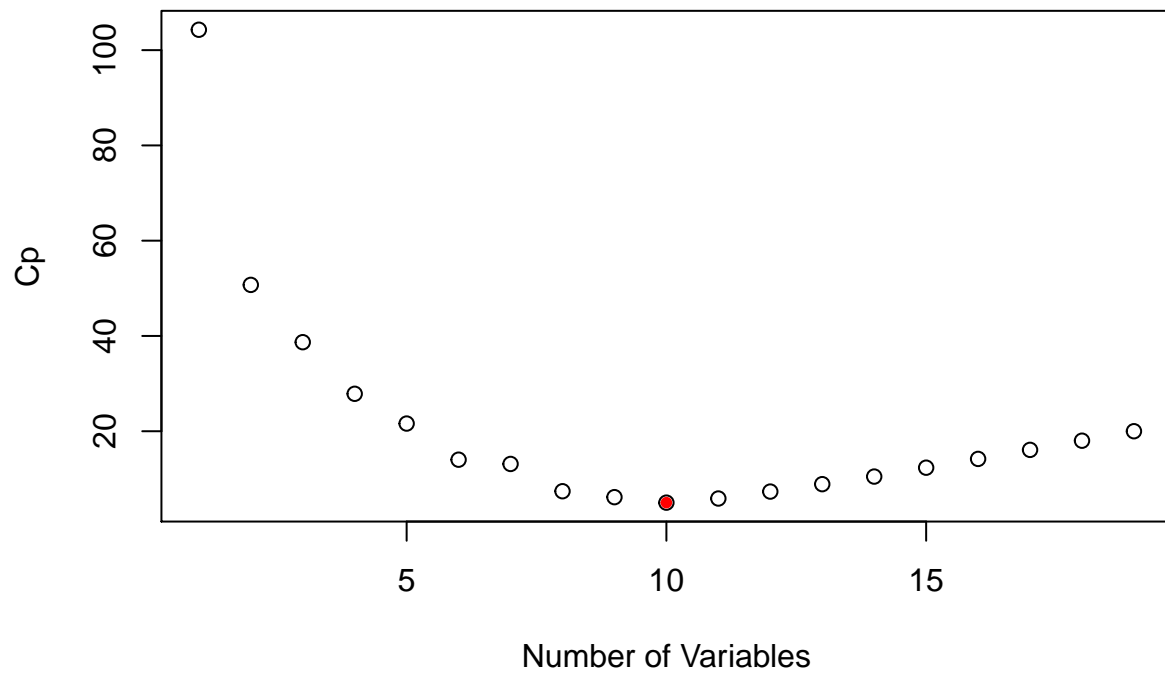

There is a plot method for the `regsubsets` object

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

```
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

Here we use the `regsubsets` function but specify the 'method="forward" option:

```
regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward")
summary(regfit.fwd)
```
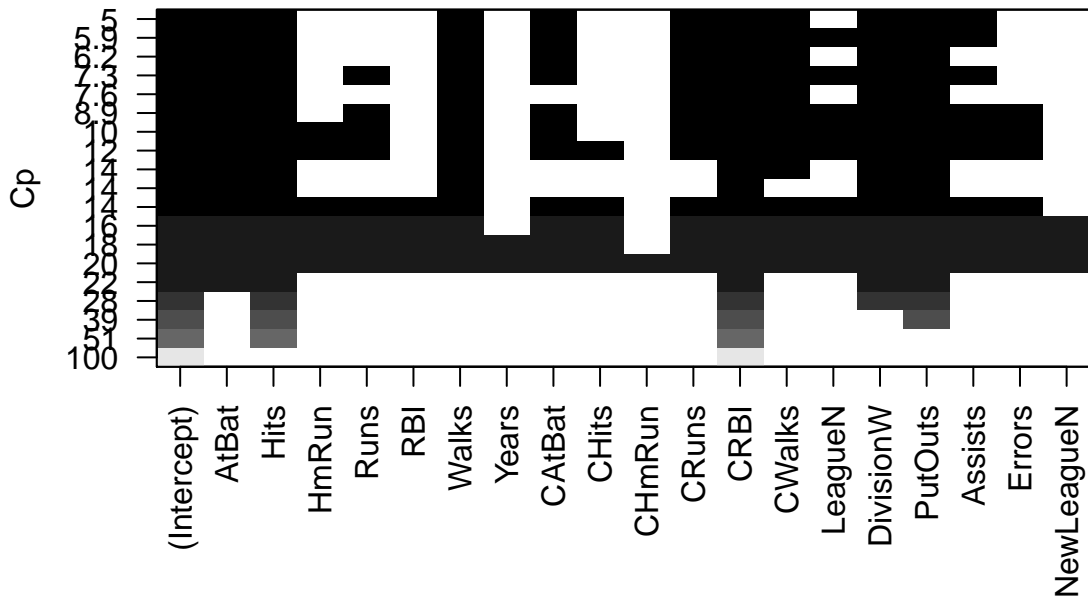
```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           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
## 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

Lets make a training and validation set, so that we can choose a good subset model. We will do it using a slightly different approach from what was done in the the book.

```r
dim(Hitters)
```
```
## [1] 263  20
```
```r
set.seed(1)
train=sample(seq(263),180,replace=FALSE)
train
```
```
[1] 70 98 150 237 53 232 243 170 161 16 259 45 173 97 192 124 178
[18] 245 94 190 228 52 158 31 64 92   4 91 205 80 113 140 115 43
[35] 244 153 181 25 163 93 184 144 174 122 117 251   6 104 241 149 102
[52] 183 224 242 15 21 66 107 136 83 186 60 211 67 130 210 95 151
[69] 17 256 207 162 200 239 236 168 249 73 222 177 234 199 203 59 235
[86] 37 126 22 230 226 42 11 110 214 132 134 77 69 188 100 206 58
[103] 44 159 101 34 208 75 185 201 261 112 54 65 23   2 106 254 257
[120] 154 142 71 166 221 105 63 143 29 240 212 167 172   5 84 120 133
[137] 72 191 248 138 182 74 179 135 87 196 157 119 13 99 263 125 247
[154] 50 55 20 57   8 30 194 139 238 46 78 88 41   7 33 141 32
```

```
## [171] 180 164 213  36 215  79 225 229 198  76
```

```
regfit.fwd=regsubsets(Salary~.,data=Hitters[train,],nvmax=19,method="forward")
```

Now we will make predictions on the observations not used for training. We know there are 19 models, so we set up some vectors to record the errors. We have to do a bit of work here, because there is no predict method for regsubsets.

```
val.errors=rep(NA,19)
x.test=model.matrix(Salary~.,data=Hitters[-train,])# notice the -index!
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")
legend("topright",legend=c("Training","Validation"),col=c("blue","black"),pch=19)
```
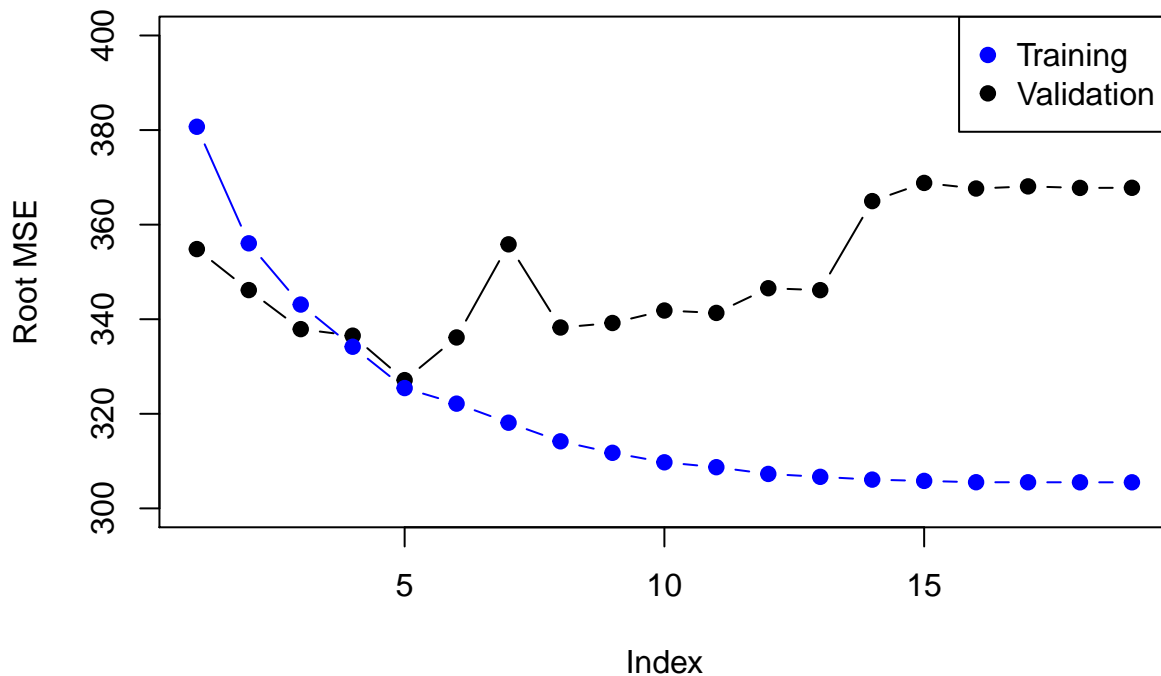

As we expect, the training error goes down monotonically as the model gets bigger, but not so for the validation error.

This was a little tedious - not having a predict method for 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
```

```
}
```

Model Selection by Cross-Validation

We will do 10-fold cross-validation. Its really easy!

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

```
##   [1]  3  1  4  4  7  7  3  5  5  2  5  2  8  3  3  3  9  2  9  8 10  5  8
##  [24]  5  5  5  5 10 10  4  4  7  6  7  7  7  3  4  8  3  6  8 10  4  3  9
##  [47]  9  3  4  9  8  7 10  6 10  3  6  9  4  2  8  2  5  6 10  7  2  8  8
##  [70]  1  3  6  2  5  8  1  1  2  8  1 10  1  2  3  6  6  5  8  8 10  4  2
##  [93]  6  1  7  4  8  3  7  8  7  1 10  1  6  2  9 10  1  7  7  4  7  4 10
## [116]  3  6 10  6  6  9  8 10  6  7  9  6  7  1 10  2  2  5  9  9  6  1  1
## [139]  2  9  4 10  5  3  7  7 10 10  9  3  3  7  3  1  4  6  6 10  4  9  9
## [162]  1  3  6  8 10  8  5  4  5  6  2  9 10  3  7  7  6  6  2  3  2  4  4
## [185]  4  4  8  2  3  5  9  9 10  2  1  3  9  6  7  3  1  9  4 10 10  8  8
## [208]  8  2  5  9  8 10  5  8  2  4  1  4  4  5  5  2  1  9  5  2  9  9  5
## [231]  3  2  1  9  1  7  2  5  8  1  1  7  6  6  4  5 10  5  7  4  8  6  9
## [254]  1  2  5  7  1  3  1  3  1  2
```

```
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){
  best.fit=regsubsets(Salary~.,data=Hitters[folds!=k,],nvmax=19,method="forward")
  for(i in 1:19){
    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")
```
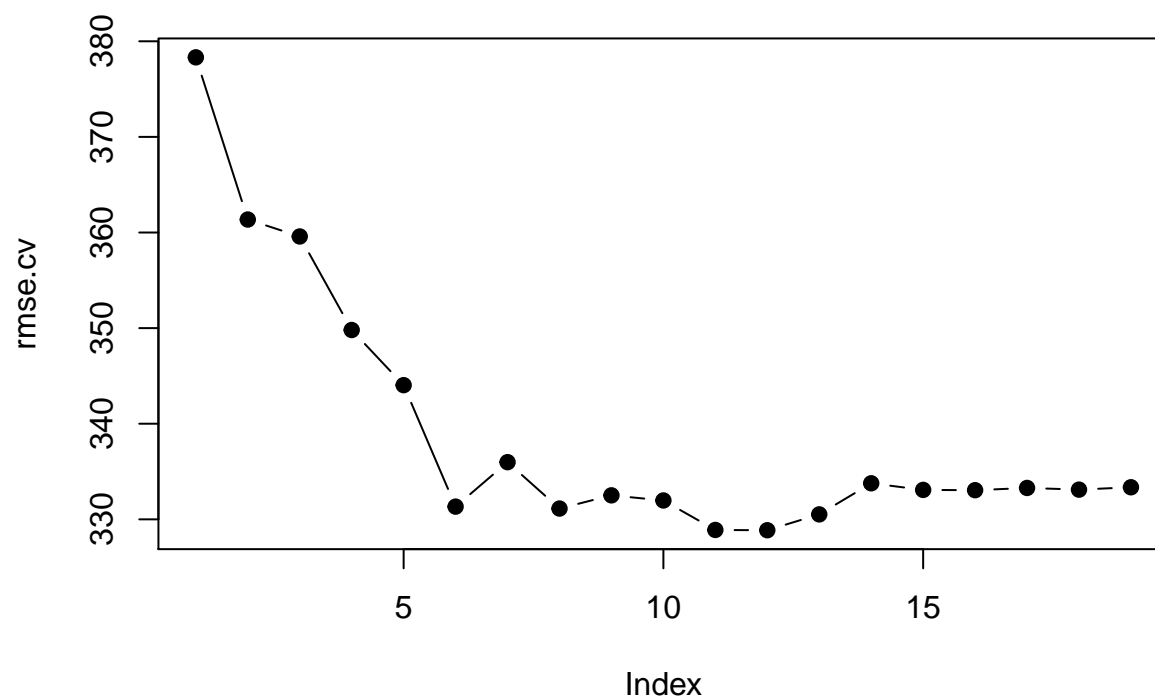
Ridge Regression and the Lasso

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

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.3.3

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 3.3.3

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.3.3

## Loaded glmnet 2.0-16
```
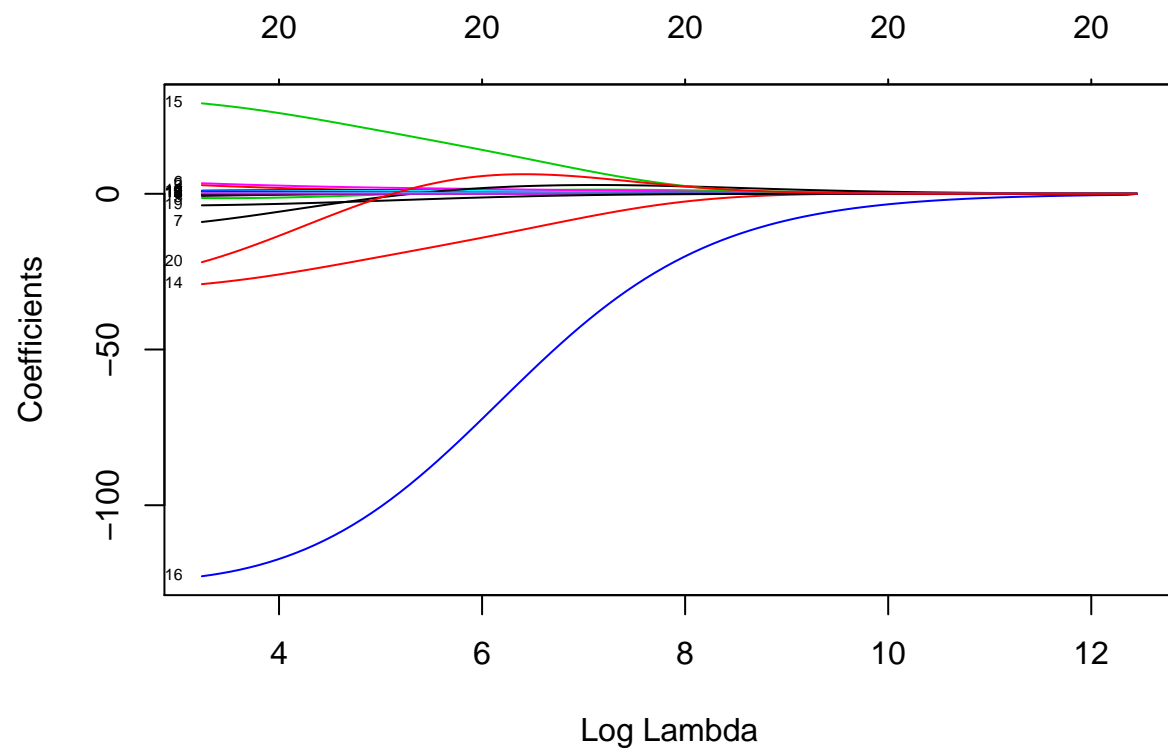
```r
x=model.matrix(Salary~.-1,data=Hitters)
y=Hitters$Salary
```

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

```r
fit.ridge=glmnet(x,y,alpha=0)
plot(fit.ridge,xvar="lambda",label=TRUE)
```
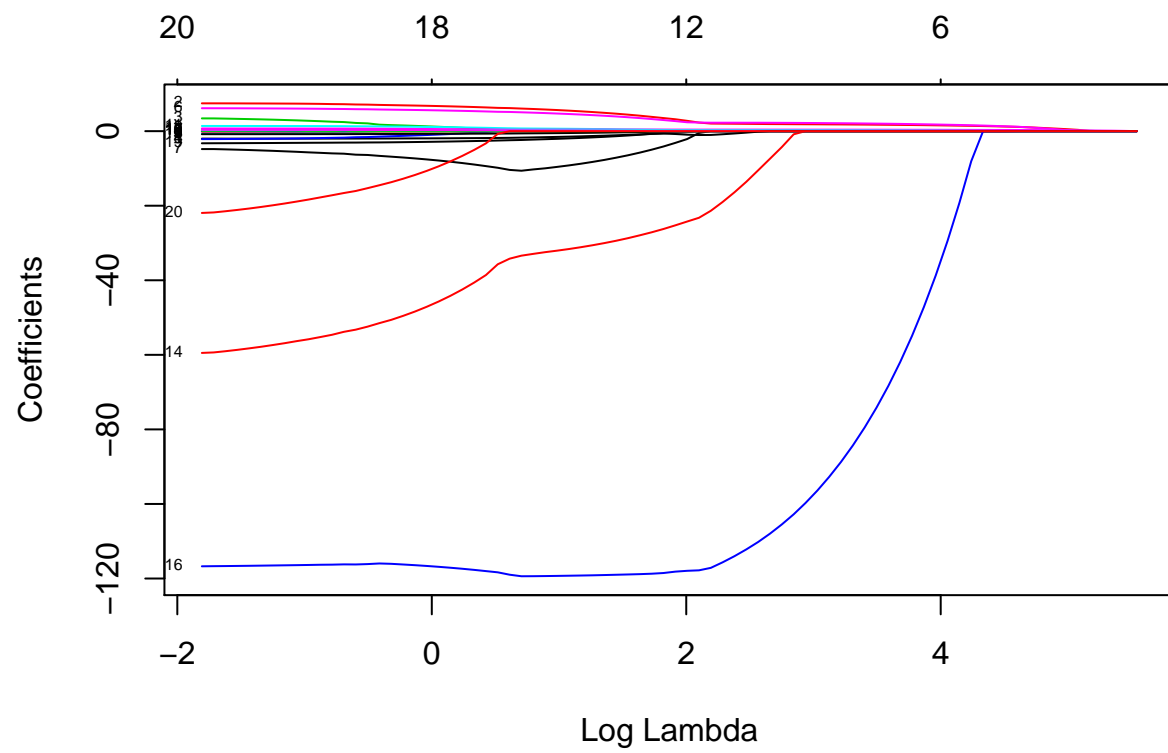
```
cv.ridge=cv.glmnet(x,y,alpha=0)
plot(cv.ridge)
```
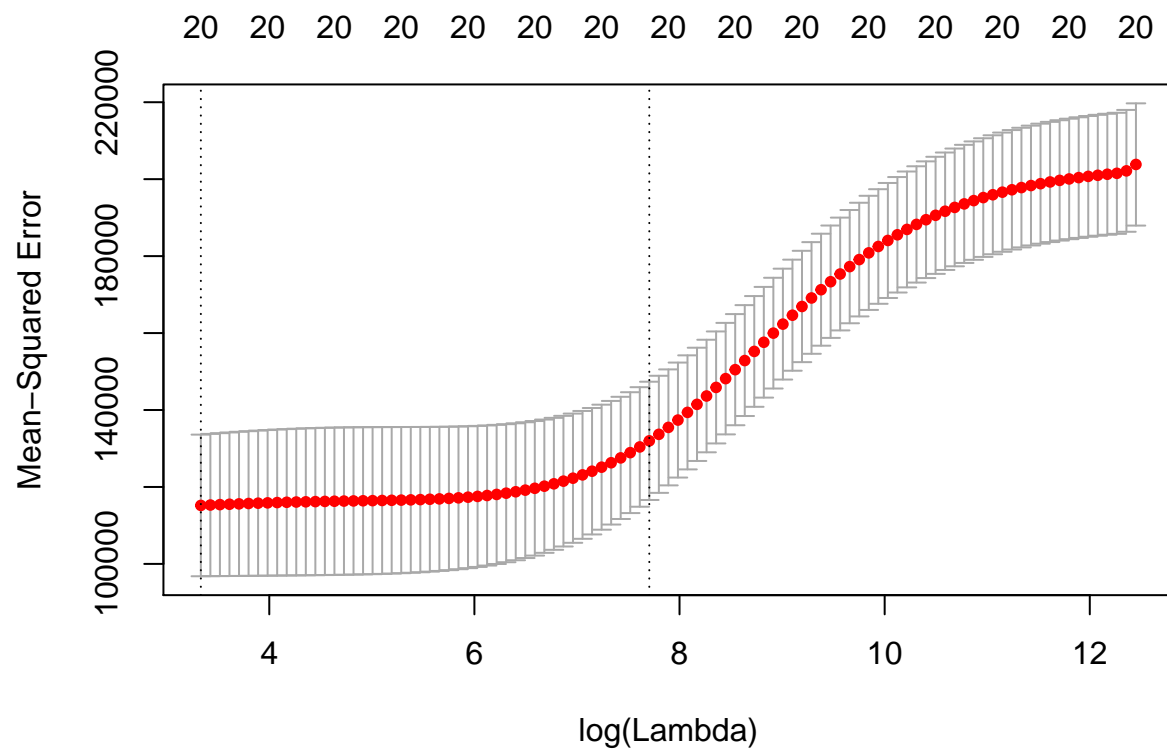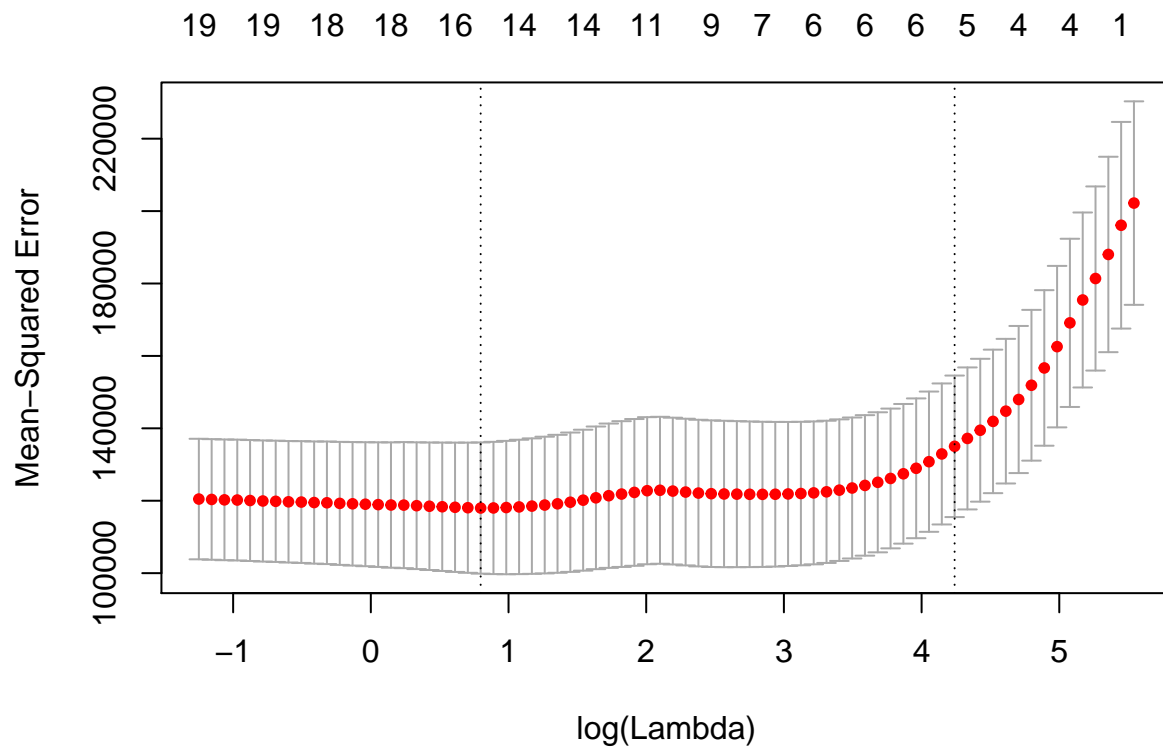
Now we fit a lasso model; for this we use the default `alpha=1`

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

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

```r
coef(cv.lasso)
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) 127.95694771
## AtBat           .
## Hits          1.42342566
## HmRun           .
## Runs            .
## RBI             .
## Walks         1.58214110
## Years           .
## CAtBat          .
## CHits           .
## CHmRun          .
## CRuns         0.16027975
## CRBI          0.33667715
## CWalks          .
## LeagueA         .
## LeagueN         .
## DivisionW    -8.06171247
## PutOuts       0.08393604
## Assists         .
## Errors          .
## NewLeagueN      .
```

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

to do.

```
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,]   8 0.45480  29.00000
## [25,]   8 0.45770  26.42000
## [26,]   8 0.46010  24.07000
## [27,]   9 0.46220  21.94000
## [28,]   9 0.46380  19.99000
## [29,]   9 0.46520  18.21000
## [30,]   9 0.46630  16.59000
## [31,]   9 0.46730  15.12000
## [32,]   9 0.46810  13.78000
## [33,]  10 0.47110  12.55000
## [34,]  10 0.47380  11.44000
## [35,]  10 0.47620  10.42000
## [36,]  11 0.48050   9.49500
## [37,]  10 0.48450   8.65200
## [38,]  11 0.48770   7.88300
## [39,]  11 0.49360   7.18300
## [40,]  12 0.49890   6.54500
## [41,]  13 0.50450   5.96300
## [42,]  13 0.51010   5.43400
## [43,]  14 0.51470   4.95100
## [44,]  14 0.51850   4.51100
## [45,]  14 0.52170   4.11000
## [46,]  15 0.52440   3.74500
```

```
## [47,] 15 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,] 17 0.54220   1.11700
## [60,] 17 0.54280   1.01800
## [61,] 17 0.54320   0.92770
## [62,] 17 0.54360   0.84530
## [63,] 17 0.54380   0.77020
## [64,] 18 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$lambda),rmse,type="b",xlab="Log(lambda)")
```
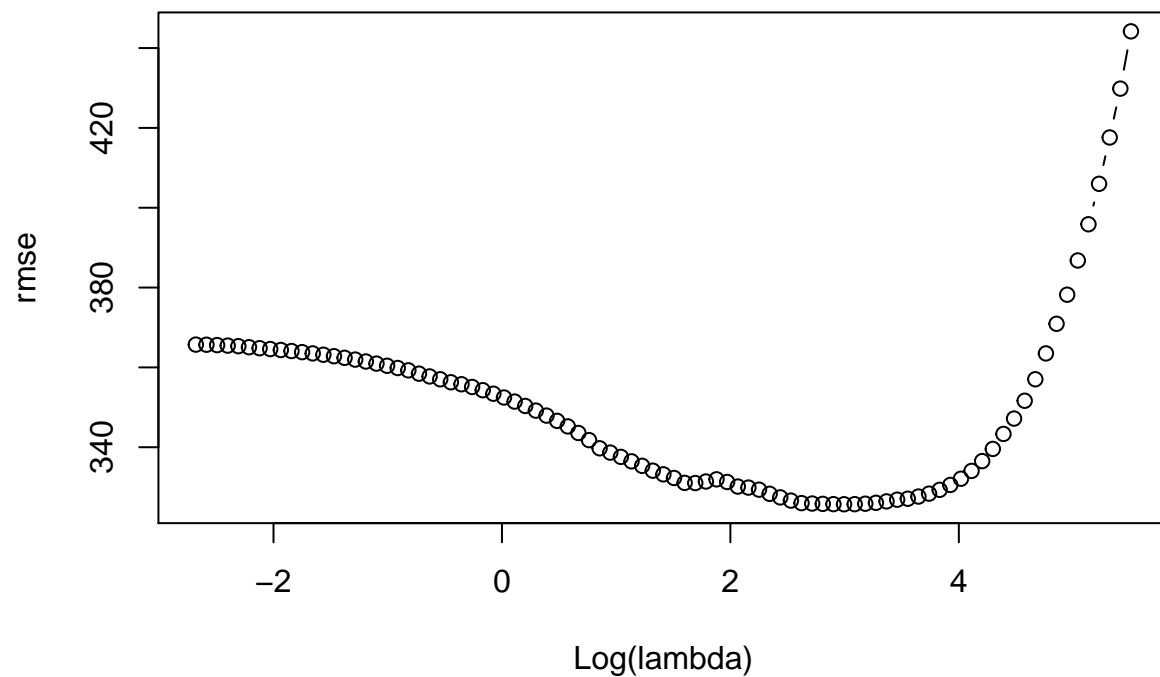
```r
lam.best=lasso.tr$lambda[order(rmse)[1]]
lam.best
```

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

```r
coef(lasso.tr,s=lam.best)
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  106.2272607
## AtBat          .
## Hits           0.1591945
## HmRun          .
## Runs           .
## RBI            1.7340977
## Walks          3.4655873
## Years          .
## CAtBat         .
## CHits          .
## CHmRun         .
## CRuns          0.5386938
## CRBI           .
## CWalks         .
## LeagueA      -28.3582245
## LeagueN        1.8006762
## DivisionW   -113.8285692
## PutOuts        0.2915437
```

```
## Assists        .
## Errors         .
## NewLeagueN     1.9431535
```