Model Selection

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents.

```
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 some missing values here, remove them.

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

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

Best Subset Regression

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

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

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

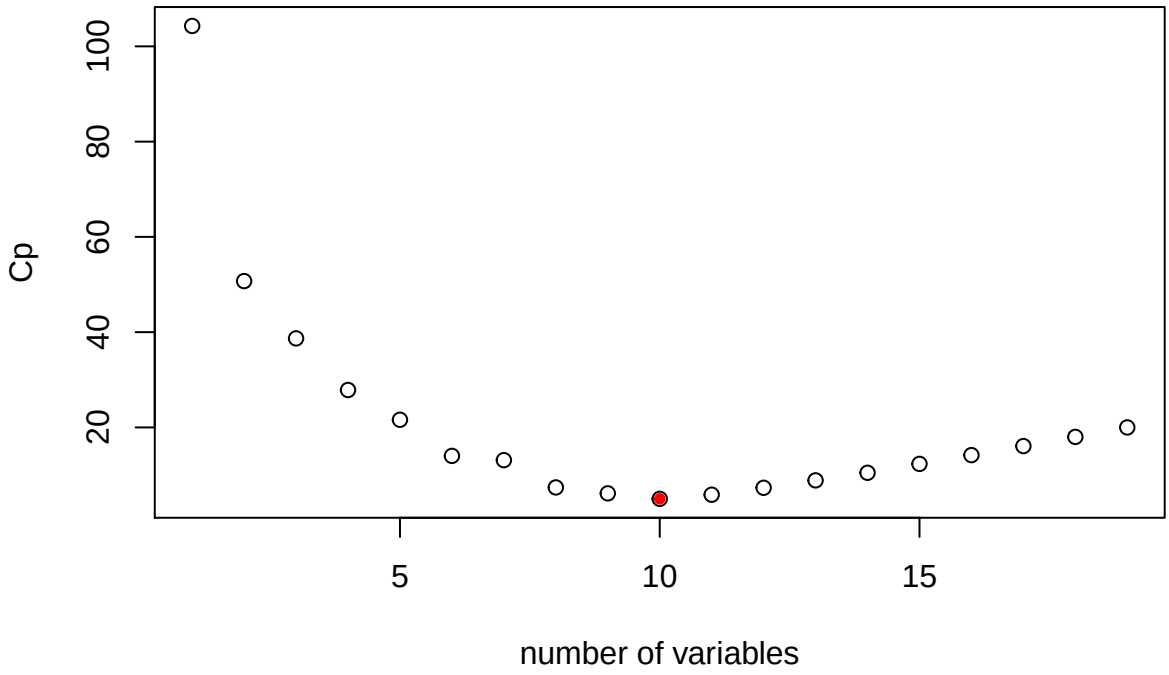
```
regfit.full = regsubsets(Salary~., data=Hitters, nvmax=19)
reg.summary = summary(regfit.full)
names(reg.summary)
```

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

```
plot(reg.summary$cp, xlab='number of variables', ylab='Cp')
which.min(reg.summary$cp)
```

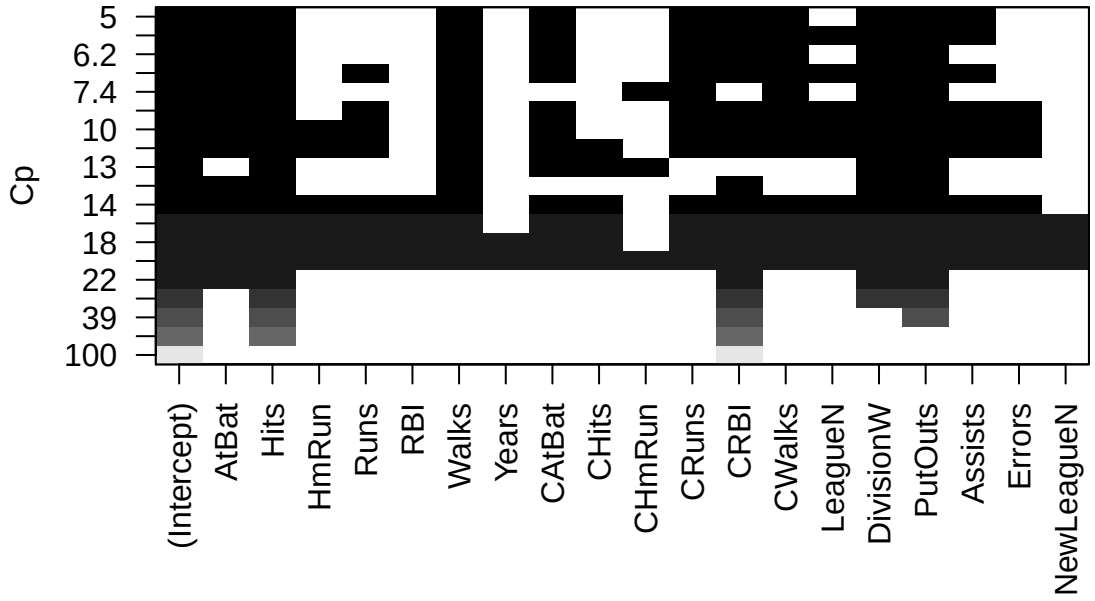
```
## [1] 10
```

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



There is a plot method for the `regsubsets` object

```
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

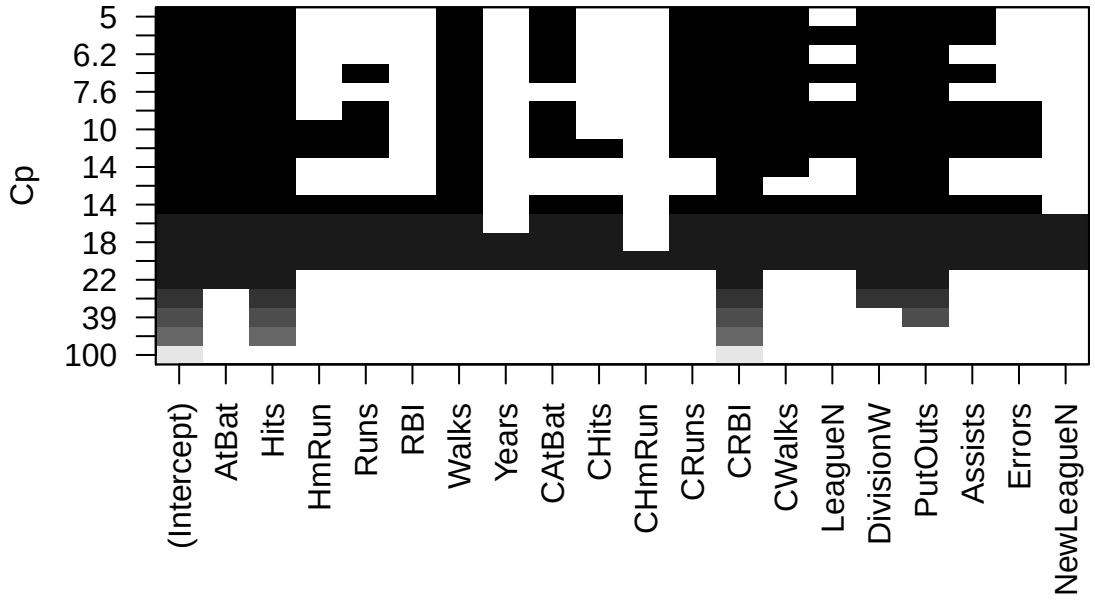
Here we use the `regsubsets` function but specify the method=`forward` option:

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

```
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 textbook.

```
dim(Hitters)
```

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

```
set.seed(1)
train = sample(seq(263),180,replace=FALSE)
train
```

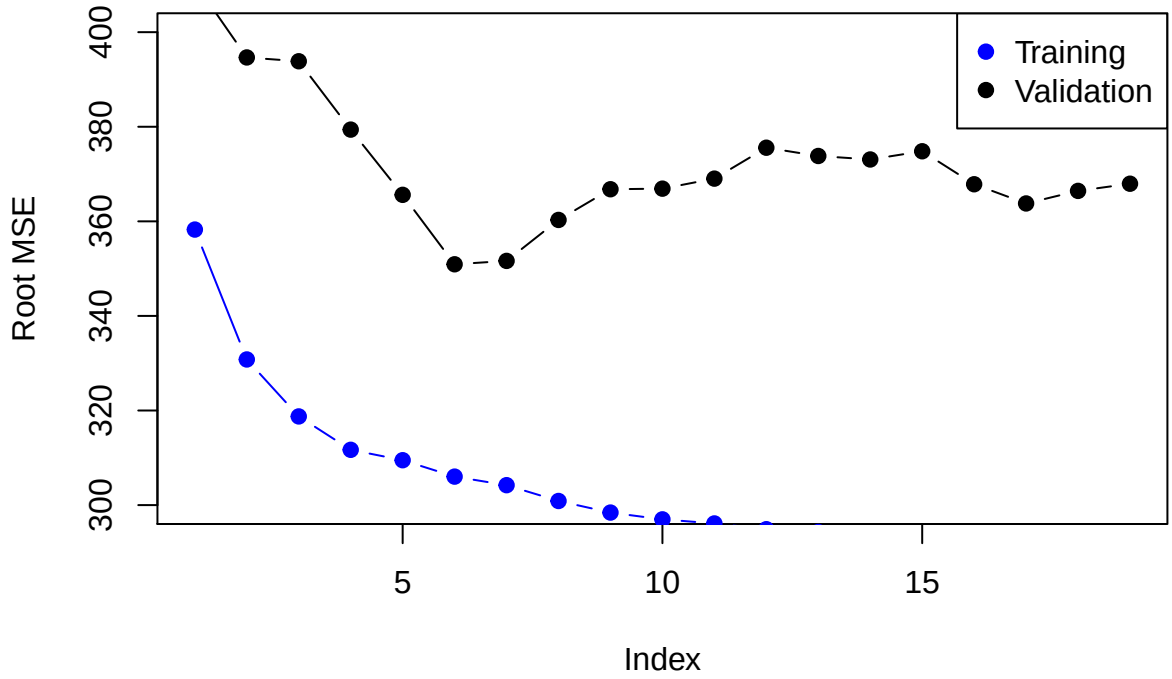
```
##   [1] 167 129 187  85  79 213  37 105 217 110 229 165  34 106 126  89 172
##  [18] 207  33  84 163  70  74  42 166 111 148 156  20  44 121  87 242 233
##  [35]  40 247  25 119 198 122  39 179 240 134  24 160  14 130  45 146  22
##  [52] 206 193 115 104 231 208 209 103  75  13 253 176 248  23 254 244 205
##  [69]  29 141 150 236 108  48 245 215 149  31 102 145  73 232  83 118  90
##  [86] 190 107  64 196  60  51 251 138 262  43  26 143 195 152 178 223 219
## [103] 202 181 222 169   1 239  78 211 246  28 116 257  61 113  86  71 225
## [120]  99 173 234  49 256 174 194  50 135 238 235 230 263  53 100 164  65
## [137] 142 175 140 124 224  77 159  98  66  19  17 228 204 186  35 144  46
## [154] 180 109 210  16 161   9 137  92 162  10 259  32 243  95 154  93  12
## [171] 255 177  15   2 128  67 183 117 197   5
```

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

Now we will make predictions on validation set. There are 19 models in total, 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,])
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.

Write a prediction method for `regsubsets`

```
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.

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

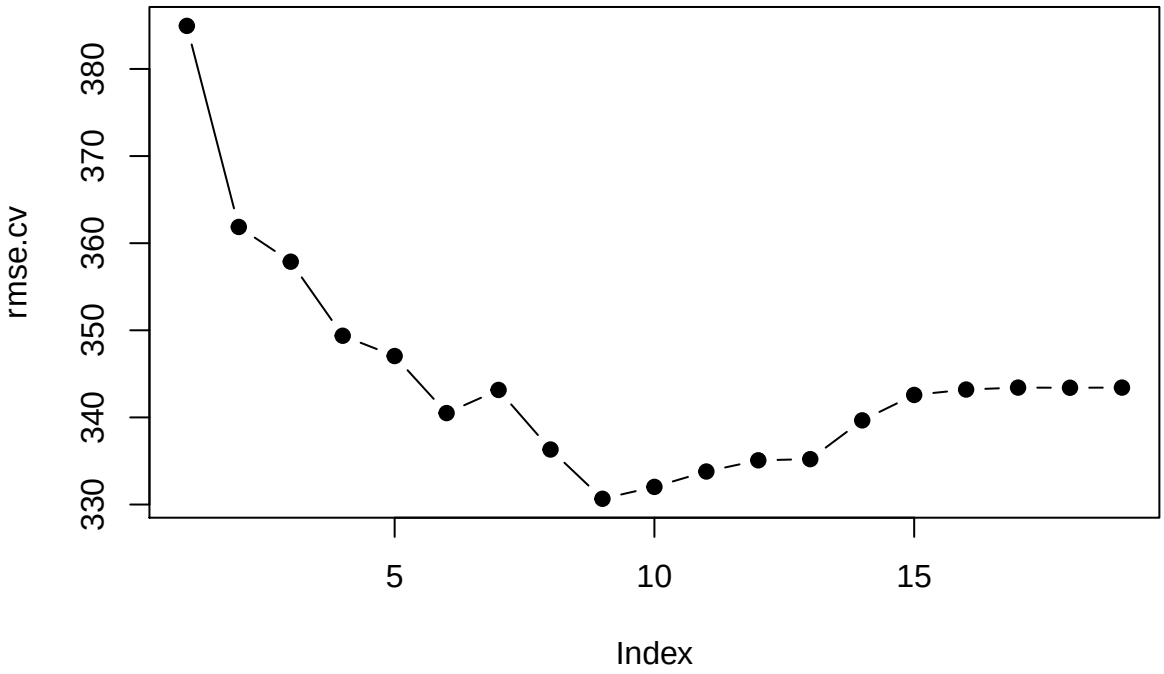
```
##   [1] 10  4  4  4  3  7 10  8 10  3  4  2  2  1  1  5  3  3  5  9  5  2  7
##  [24]  3  8  7  4  6  7  7  7 10  2  6  6  1  8  8 10  1  8  3  7 10  1  6
##  [47] 10  3  8  7  9  3  6  2  5  6  5  5  2  6  4  7 10  7  1  8  1  2  8
##  [70]  2  1  8  2  6  4  3  1  1  5  1  2 10  5  8  8  3  9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

```
## [208]  3  4  6  8  2  9  5  7  2 10  4  1  6 10  3  5  5  3  5  7  3  6  5
## [231]  4  1  3  7  1  5  2  9  4  6  8  3  2  4  5 10  8  5  2  2  2  6  3
## [254]  4  7  2  6  4  8  7  1  2  6
```

```
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

package glmnet

```
library(glmnet)
```

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

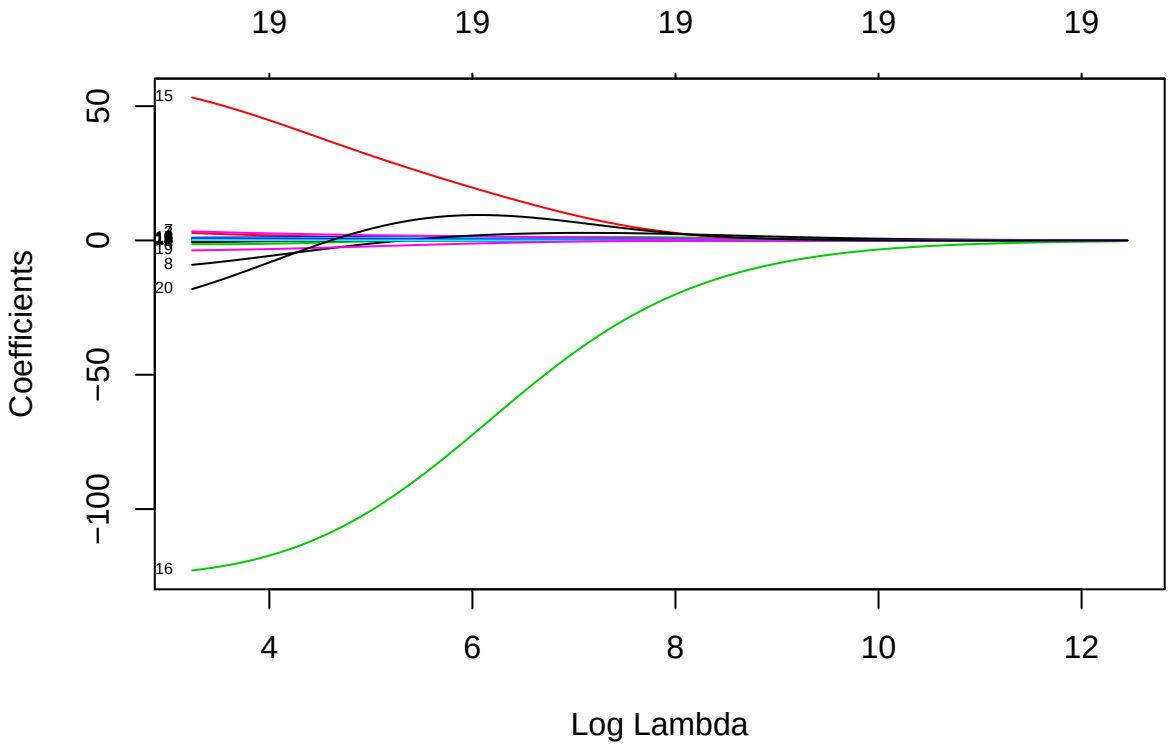
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

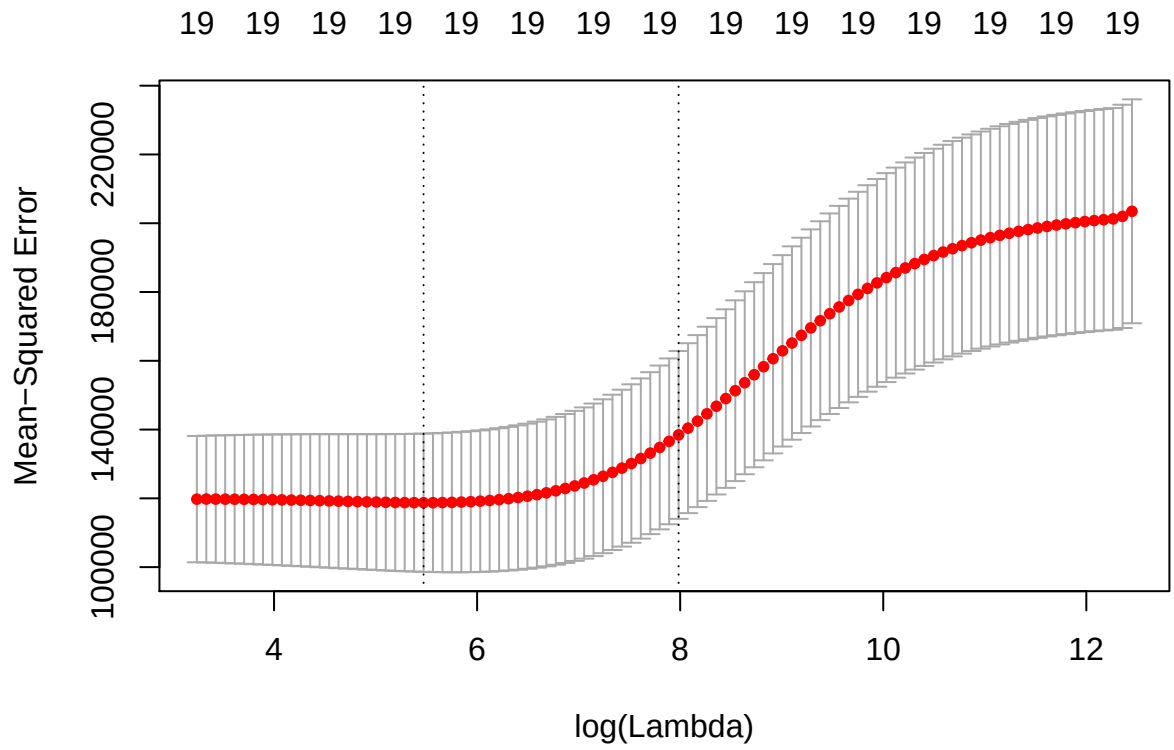
```
x = model.matrix(Salary~.,data=Hitters)
y = Hitters$Salary
```

First we do ridge regression. This is achieved by calling `glmnet` with `alpha=0`. Also `cv.glmnet` will do cross-validation.

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

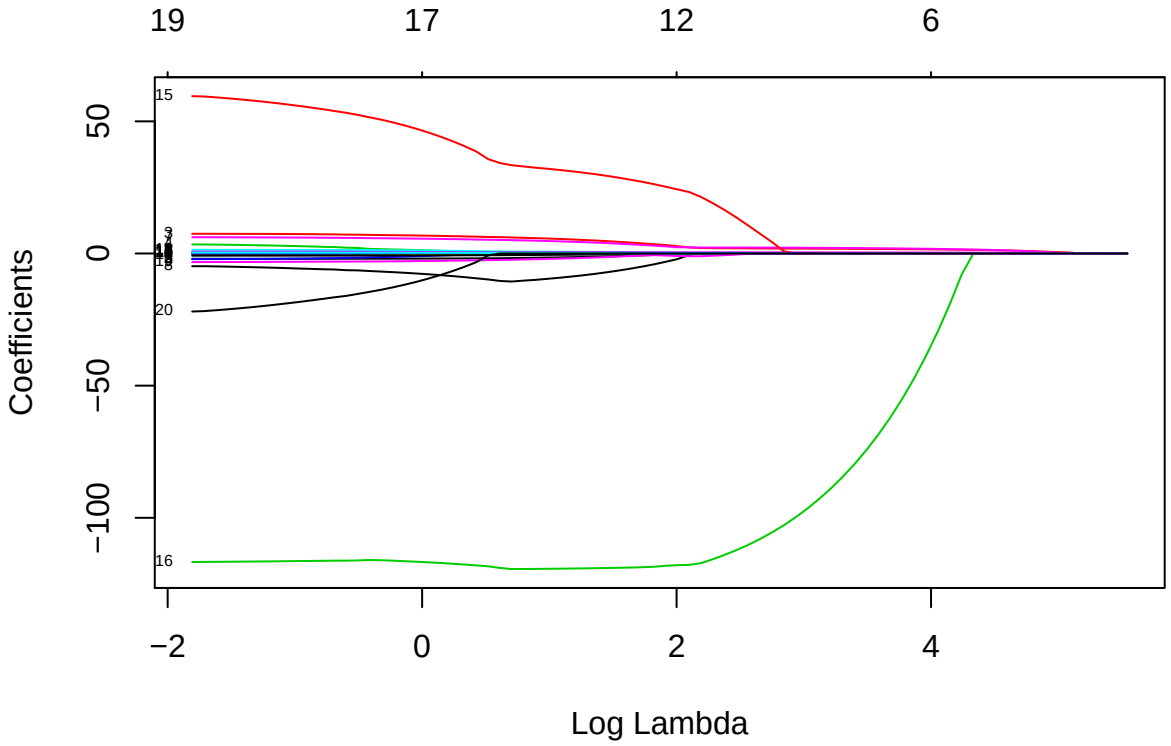



```
cv.ridge = cv.glmnet(x,y,alpha=0)
plot(cv.ridge)
```

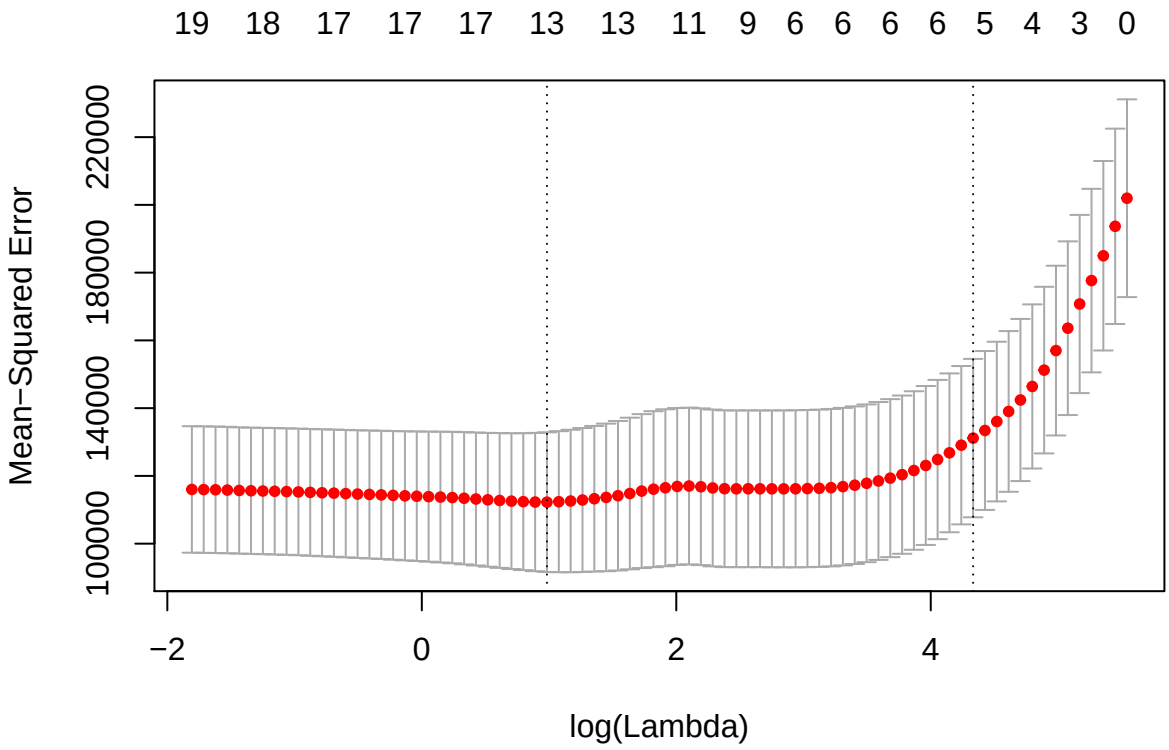


Now we fit a lasso model; for this we use the default `alpha=1`

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




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




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

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) 144.37970485
## (Intercept)   .
## AtBat         .
## Hits          1.36380384
## HmRun         .
## Runs          .
## RBI           .
## Walks         1.49731098
## Years         .
## CAtBat        .
## CHits         .
## CHmRun        .
## CRuns         0.15275165
## CRBI          0.32833941
## CWalks        .
## LeagueN       .
## DivisionW     .
## PutOuts       0.06625755
## Assists       .
## Errors        .
## NewLeagueN    .
```

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

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

```
##
```

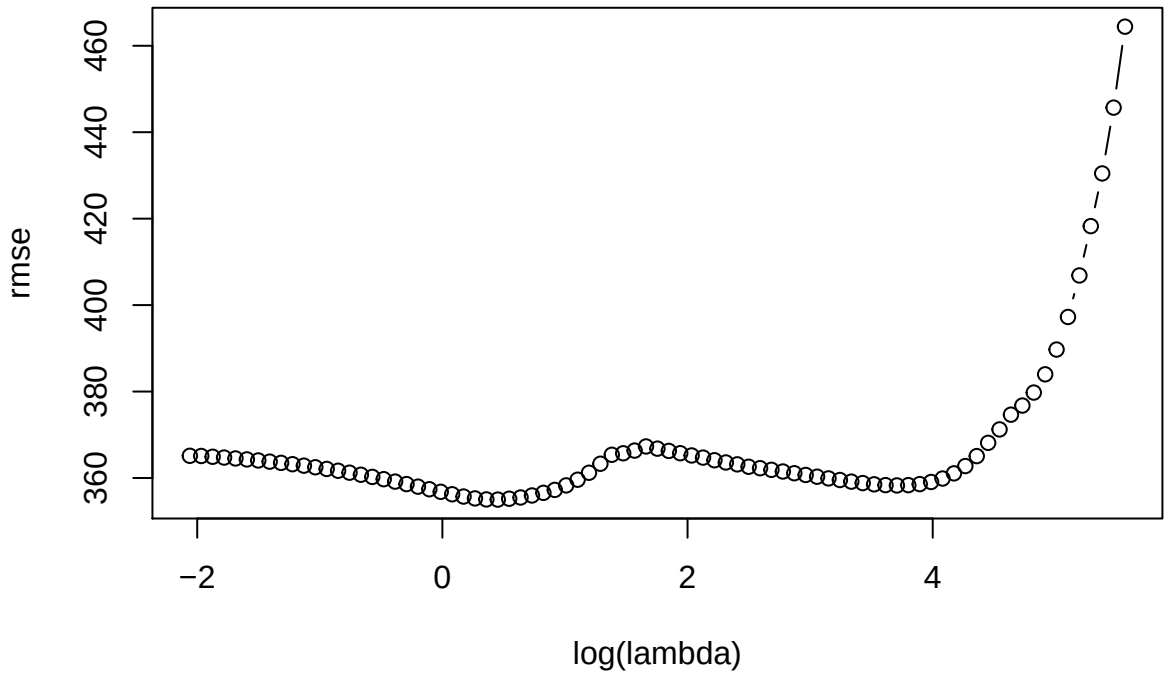
```
## Call:  glmnet(x = x[train, ], y = y[train])
##
##       Df    %Dev   Lambda
##  [1,]  0 0.00000 262.1000
##  [2,]  1 0.05919 238.8000
##  [3,]  1 0.10830 217.6000
##  [4,]  1 0.14910 198.3000
##  [5,]  2 0.19720 180.6000
##  [6,]  3 0.23940 164.6000
##  [7,]  3 0.27450 150.0000
##  [8,]  3 0.30370 136.7000
##  [9,]  3 0.32790 124.5000
## [10,]  3 0.34800 113.5000
## [11,]  4 0.36500 103.4000
## [12,]  5 0.38770  94.1900
## [13,]  6 0.40900  85.8200
## [14,]  6 0.42730  78.2000
## [15,]  6 0.44250  71.2500
## [16,]  6 0.45510  64.9200
## [17,]  6 0.46550  59.1500
## [18,]  6 0.47420  53.9000
## [19,]  6 0.48140  49.1100
## [20,]  6 0.48740  44.7500
## [21,]  6 0.49240  40.7700
## [22,]  6 0.49650  37.1500
## [23,]  6 0.49990  33.8500
## [24,]  7 0.50280  30.8400
## [25,]  7 0.50510  28.1000
## [26,]  8 0.50710  25.6100
## [27,]  8 0.50940  23.3300
## [28,]  8 0.51120  21.2600
## [29,]  8 0.51280  19.3700
## [30,]  8 0.51410  17.6500
## [31,]  8 0.51520  16.0800
## [32,]  8 0.51600  14.6500
## [33,]  8 0.51680  13.3500
## [34,]  9 0.51750  12.1700
## [35,]  9 0.51990  11.0800
## [36,] 10 0.52230  10.1000
## [37,] 10 0.52440   9.2020
## [38,] 11 0.52640   8.3850
## [39,] 11 0.52820   7.6400
## [40,] 11 0.52970   6.9610
## [41,] 11 0.53090   6.3430
## [42,] 11 0.53190   5.7790
## [43,] 12 0.53280   5.2660
## [44,] 14 0.53530   4.7980
## [45,] 14 0.53830   4.3720
## [46,] 15 0.54060   3.9840
## [47,] 16 0.54450   3.6300
## [48,] 16 0.54790   3.3070
## [49,] 16 0.55060   3.0130
## [50,] 16 0.55290   2.7460
## [51,] 17 0.55480   2.5020
```

```
## [52,] 17 0.55650   2.2800
## [53,] 17 0.55780   2.0770
## [54,] 17 0.55890   1.8920
## [55,] 18 0.56000   1.7240
## [56,] 18 0.56160   1.5710
## [57,] 18 0.56300   1.4320
## [58,] 19 0.56420   1.3040
## [59,] 19 0.56530   1.1890
## [60,] 19 0.56630   1.0830
## [61,] 19 0.56710   0.9867
## [62,] 19 0.56770   0.8991
## [63,] 19 0.56830   0.8192
## [64,] 19 0.56880   0.7464
## [65,] 19 0.56920   0.6801
## [66,] 19 0.56950   0.6197
## [67,] 19 0.56980   0.5647
## [68,] 19 0.57000   0.5145
## [69,] 19 0.57020   0.4688
## [70,] 19 0.57040   0.4271
## [71,] 19 0.57050   0.3892
## [72,] 19 0.57060   0.3546
## [73,] 19 0.57070   0.3231
## [74,] 19 0.57080   0.2944
## [75,] 19 0.57080   0.2683
## [76,] 19 0.57090   0.2444
## [77,] 19 0.57090   0.2227
## [78,] 19 0.57100   0.2029
## [79,] 19 0.57100   0.1849
## [80,] 19 0.57110   0.1685
## [81,] 19 0.57110   0.1535
## [82,] 19 0.57110   0.1399
## [83,] 19 0.57110   0.1274
```

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

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

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




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

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

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

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                           1
## (Intercept)  193.01429291
## (Intercept)    .
## AtBat         -1.14830040
## Hits           4.92901730
## HmRun          .
## Runs          -3.51936866
## RBI            0.38309009
## Walks          6.01828596
## Years        -20.74174043
## CAtBat        -0.01903175
## CHits          0.08077424
## CHmRun         0.53493799
## CRuns          0.77272746
## CRBI           0.49203970
## CWalks        -0.47458673
## LeagueN       91.21313136
## DivisionW   -161.10222986
## PutOuts        0.28639231
## Assists        0.29245560
## Errors        -4.69237401
## NewLeagueN   -56.95409378
```