Hitters Notebook

The hitters dataset consists of 322 observations of 21 variables with the following information - X (name), At-Bat, Hits, HmRun (home runs), Runs, RBI, Walks, Years, CAtBat, CHits, CHmRun, CRuns, CRBI, CWalks, League, Division, PutOuts, Assists, Errors, Salary, New League. Here League, Division and NewLeagues are factor variabes with 2 categories. We drop rows with missing entries and are left with 263 observations.

```
rm(list=ls())
hitter <- read.csv("hitters.csv")
# str(hitter)
hitter <- na.omit(hitter)
# str(hitter)
```

The leaps package in R does subset selection with the regsubsets function. By default, the maximum number of subsets, this function uses is 8. We extend this to do a complete subset selection by changing the default value of nvmax argument in this function. Note that CRBI is in the model with 1 to 6 variables but not in the model with 7 and 8 variables.

```
#install.packages("leaps")
library(leaps)
?regsubsets
hitters <- hitter[, 2:21]
model1 <- regsubsets(Salary ~ ., hitters)
summary(model1)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., 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
```

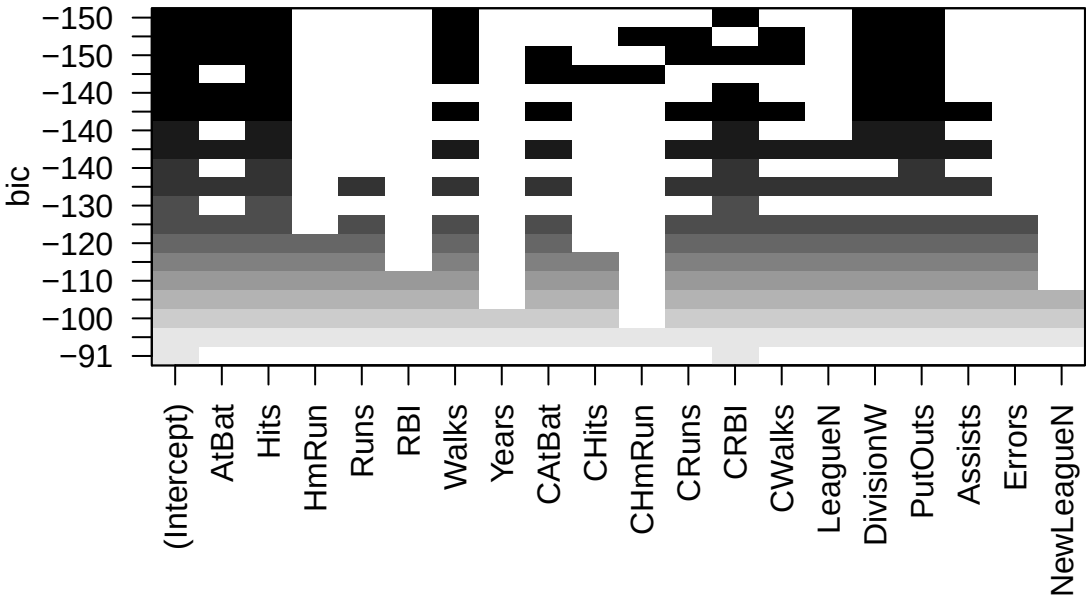
```
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 ) " "   " "  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 2  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 3  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 4  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 5  ( 1 ) "*"   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 6  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   " "   " "    " "   " "   "*"
## 7  ( 1 ) " "   "*"  " "   " "  " " " " "*"   " "   "*"   "*"    "*"   " "   " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   " "   " "    "*"   "*"   " "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
```

```
model2 <- regsubsets(Salary ~ ., hitters, nvmax = 19)
summary(model2)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., hitters, nvmax = 19)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 )  " "   " "  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 2  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 3  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
```

```
## 4  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    " "   "*"
## 5  ( 1 )  "*"   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    " "   "*"
## 6  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   " "    " "   "*"
## 7  ( 1 )  " "   "*"  " "   " "  " " " " "*"   " "   "*"    "*"   "*"    " "   " "
## 8  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   "*"    "*"   " "
## 9  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   "*"    " "   " "    "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   "*"    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " " " "*"   " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " " " "*"   " "   "*"    " "   " "    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " " " "*"   " "   "*"    " "   " "    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " " " "*"   " "   "*"    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
plot(model2)
```



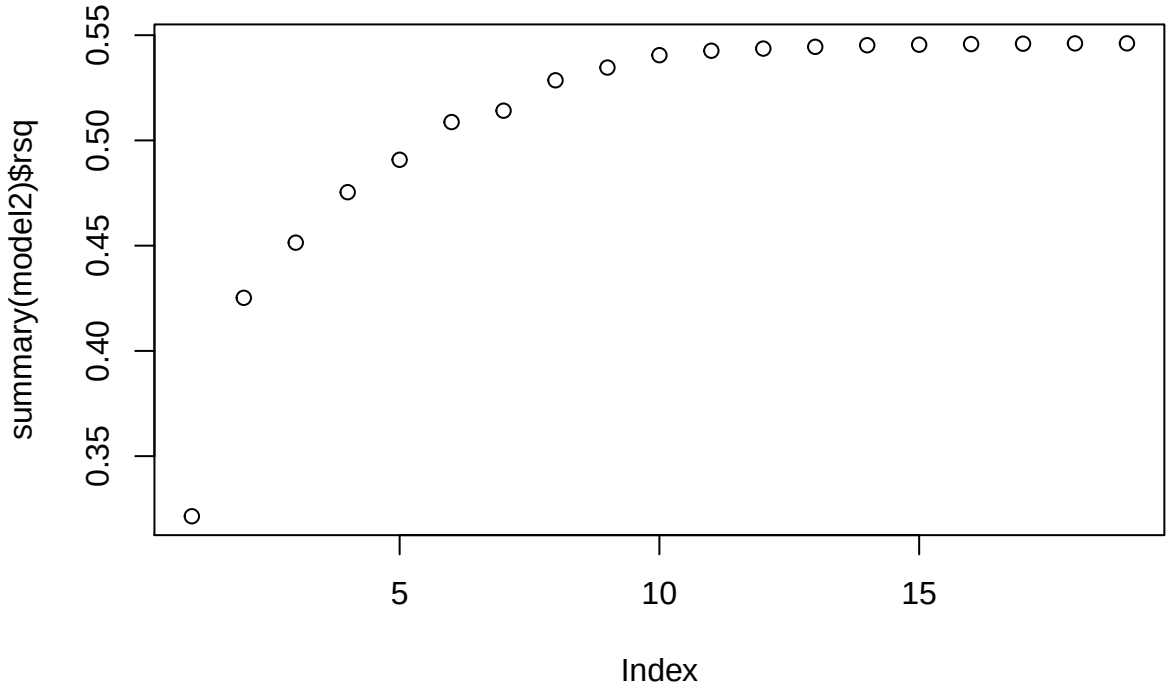

```
names(summary(model2))
```

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

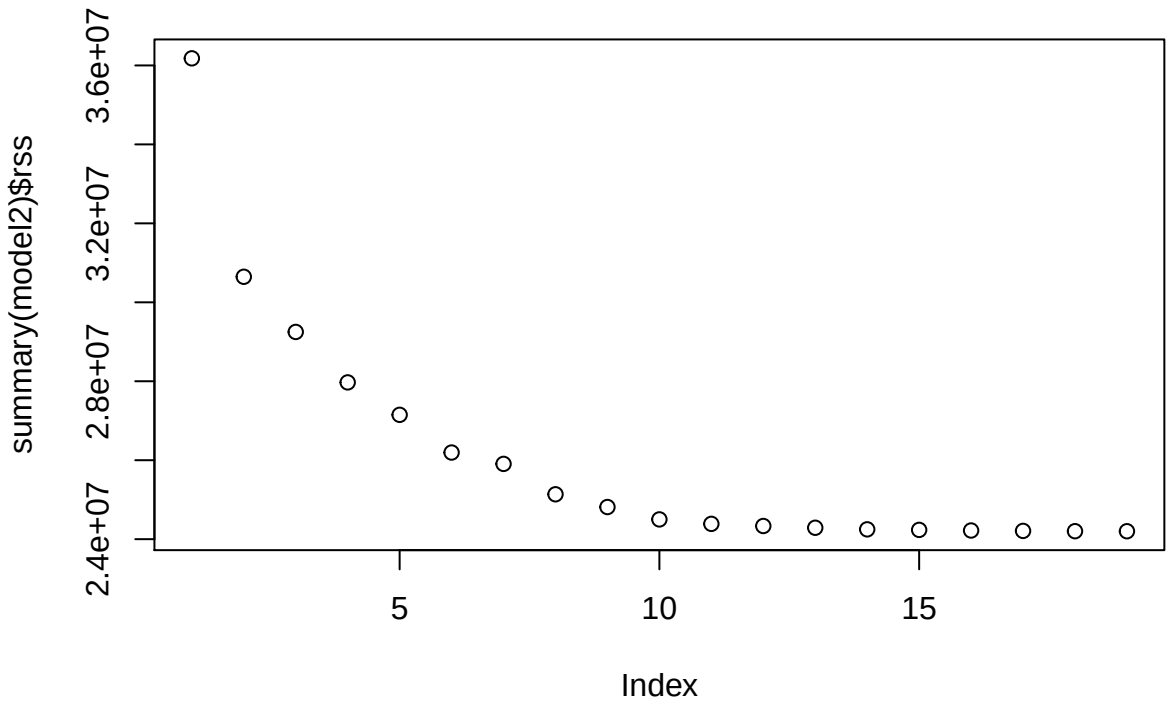
```
summary(model2)$rsq
```

```
##  [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227
##  [8] 0.5285569 0.5346124 0.5404950 0.5426153 0.5436302 0.5444570 0.5452164
## [15] 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```

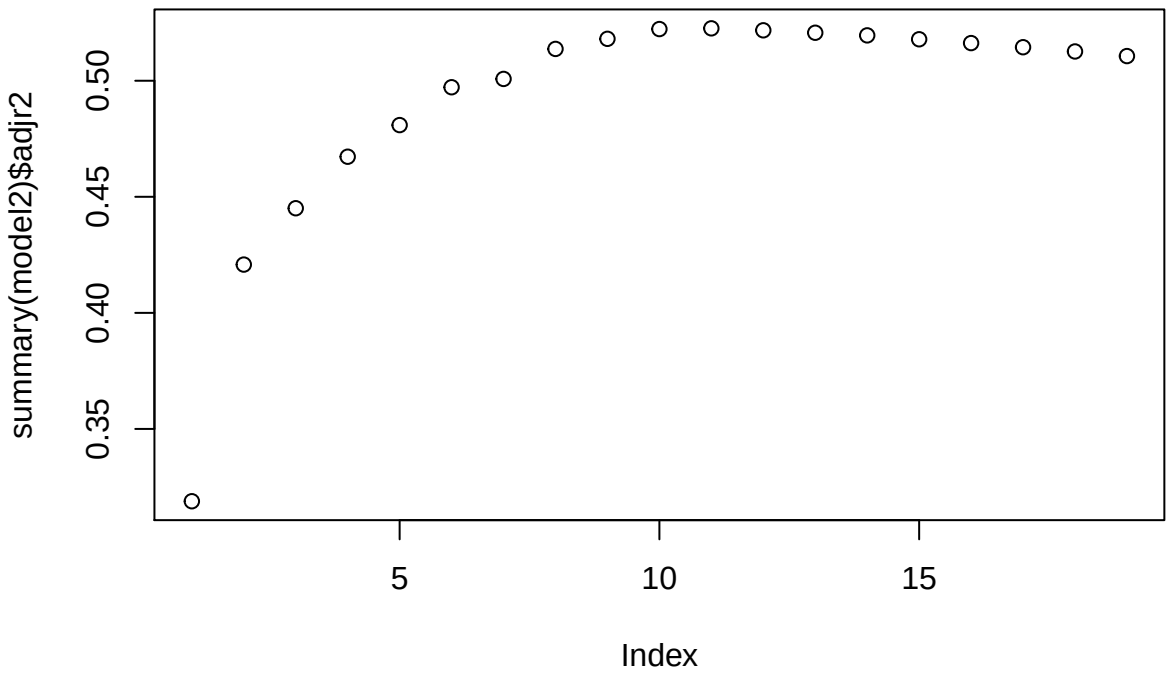
```
plot(summary(model2)$rsq)
```




```
plot(summary(model2)$rss)
```




```
plot(summary(model2)$adjr2)
```




```
which.max(summary(model2)$adjr2)
```

```
## [1] 11
```

```
coef(model2,11)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914    1.4553310
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##    0.7852528   -0.8228559   43.1116152 -111.1460252    0.2894087    0.2688277
```

The figures indicate that R-squared increase as the number of variables in the subset increases and likewise the residual sum of squared (sum of squared errors) decreases as the size of the subsets increases. On the other hand the adjusted R-squared increases first and then decreases.

Forward stepwise selection: In this example, the best model identified by the forward stepwise selection is the same as that obtained by the best subset selection. It is also possible to run this algorithm using a backward method where you drop variables one a time rather add. In general, the solutions from these two methods can be different.

```
model3<-regsubsets(Salary~.,data=hitters,nvmax=19,method="forward")
which.max(summary(model3)$adjr2)
```

```
## [1] 11
```

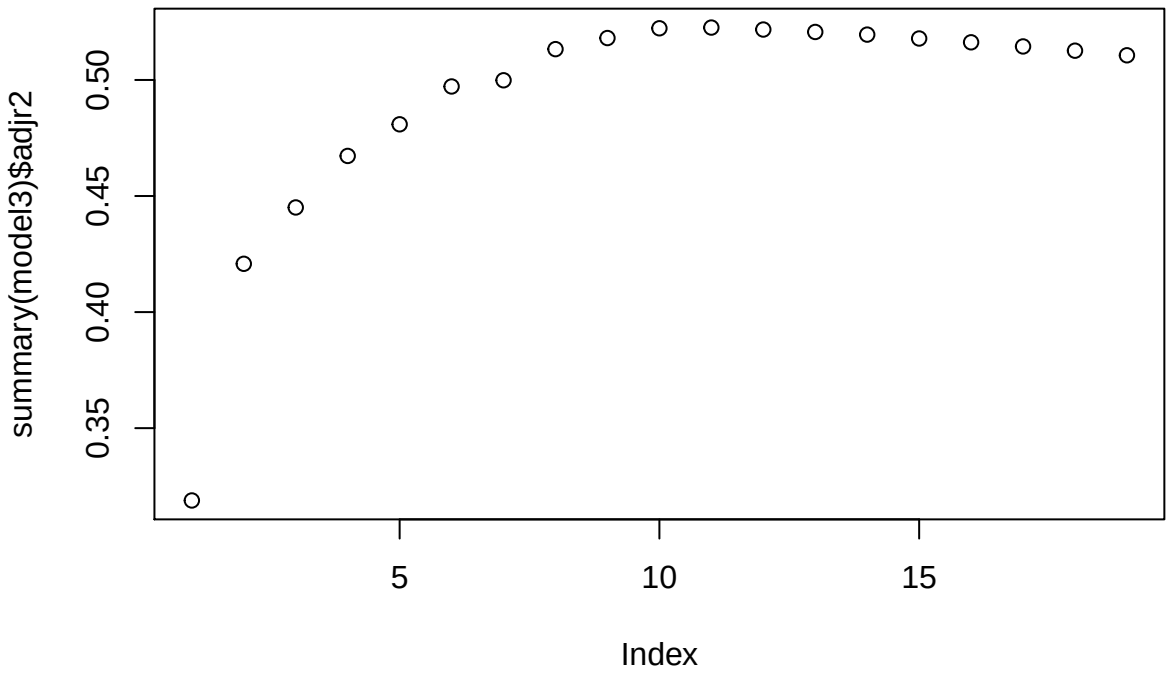
```
coef(model3,11)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914    1.4553310
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##    0.7852528   -0.8228559   43.1116152 -111.1460252    0.2894087    0.2688277
```

```
summary(model2)$adjr2-summary(model3)$adjr2
```

```
##  [1] 3.330669e-16 1.110223e-16 0.000000e+00 0.000000e+00 1.110223e-16
##  [6] 0.000000e+00 9.185854e-04 4.314850e-04 1.110223e-16 1.110223e-16
## [11] 1.110223e-16 0.000000e+00 0.000000e+00 2.220446e-16 1.110223e-16
## [16] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

```
plot(summary(model3)$adjr2)
```




```
model4<-regsubsets(Salary~.,data=hitters,nvmax=19,method="backward")
which.max(summary(model4)$adjr2)
```

```
## [1] 11
```

```
coef(model4,11)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914    1.4553310
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##    0.7852528   -0.8228559   43.1116152 -111.1460252    0.2894087    0.2688277
```

```
summary(model4)
```

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

VALIDATION SET:

Split into training and validation set. `model.matrix` creates the $\mathbf{X}$-matrix. Then we create a loop to find the coefficients for the best subset of size `i` for `i` running from 1 to 19. We use the coefficients to predict the Y-values and compute MSE for the *validation set*

```
# break the dataset into 2 parts: training and validation set
set.seed(1)
train <- sample(c(TRUE, FALSE), nrow(hitters), rep = TRUE)
valset <- (!train) # val set is the complement of train set

# validation set approach
# trains only on the training data
model_valset <- regsubsets(Salary ~ .,data = hitters[train,], nvmax = 19)
summary(model_valset)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = hitters[train, ], nvmax = 19)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
```

```
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 )  " "   " "  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 2  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 3  ( 1 )  " "   " "  " "   " "  " " " "*" " "   "*"    "*"   " "    " "   " "
## 4  ( 1 )  " "   " "  " "   " "  " " " "*" " "   "*"    "*"   " "    " "   " "
## 5  ( 1 )  " "   " "  " "   " "  " " " "*" " "   "*"    "*"   " "    " "   " "
## 6  ( 1 )  " "   " "  " "   " "  " " " "*" " "   "*"    "*"   " "    "*"   " "
## 7  ( 1 )  "*"   "*"  " "   " "  " " " "*" " "   " "    " "   " "    "*"   " "
## 8  ( 1 )  "*"   "*"  "*"   " "  " " " "*" " "   " "    " "   " "    "*"   " "
## 9  ( 1 )  "*"   "*"  "*"   " "  " " " "*" " "   " "    " "   "*"    "*"   " "
## 10  ( 1 ) "*"   "*"  "*"   " "  " " " "*" " "   "*"    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  "*"   " "  " " " "*" " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  "*"   " "  " " " "*" " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  "*"   " "  " " " "*" " "   "*"    " "   "*"    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   " "  " " " "*" "*"   "*"    " "   "*"    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   " "  " " " "*" "*"   "*"    " "   "*"    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  " " " "*" "*"   "*"    "*"   "*"    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  " " " "*" "*"   "*"    "*"   "*"    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  " " " "*" "*"   "*"    "*"   "*"    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 4  ( 1 )  " "    " "     "*"       " "     " "     " "    " "
## 5  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     " "     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 14  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    "*"
## 16  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    "*"
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
# create validation set matrix of X to compute MSE values
valset.mat <- model.matrix(Salary ~., data = hitters[valset,])
```

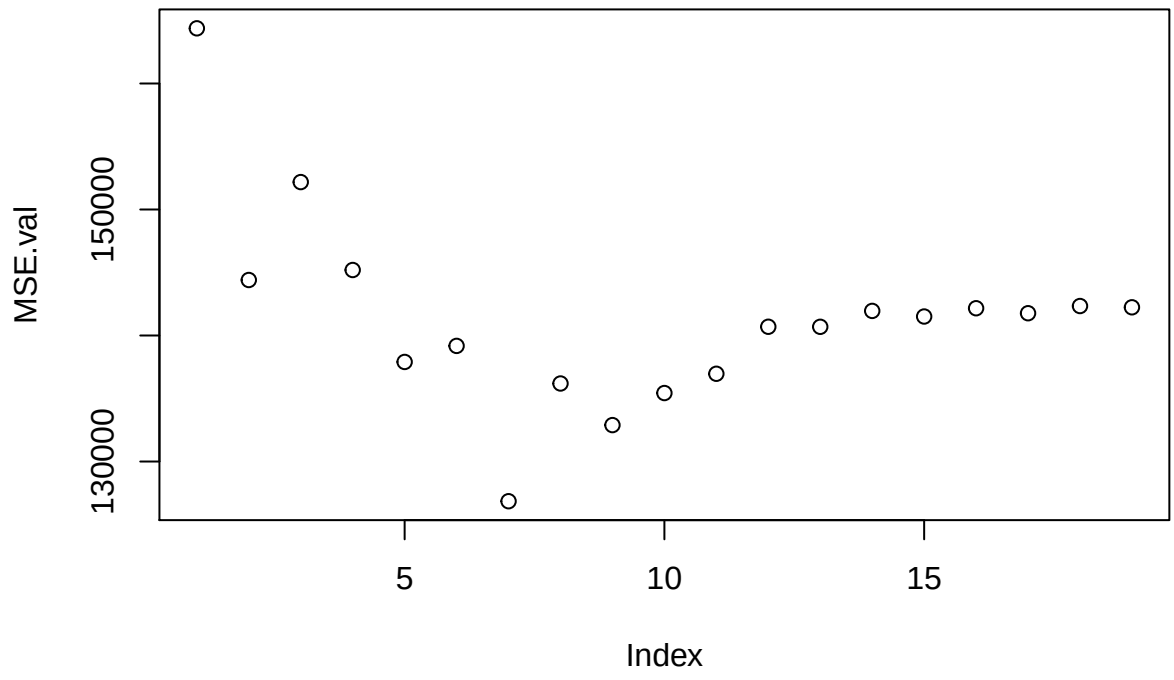
```
# valset.mat

# MSE values code
MSE.val = rep(NA ,19)
for (i in 1:19) {
  coefi = coef(model_valset, id = i)
  pred = valset.mat[,names(coefi)] %*% coefi # matrix multiplication
  MSE.val[i] = mean((hitters$Salary[valset] - pred)^2)
}
```

The best model turns out to be the one with 7 variables. Notice that if we now look at the best subset of size 7 on the entire data set, it is a bit different.

```
MSE.val
```

```
##  [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0 136191.4
##  [9] 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2 141508.2 142164.4
## [17] 141767.4 142339.6 142238.2
```

```
plot(MSE.val)
```




```
which.min(MSE.val)
```

```
## [1] 7
```

```
coef(model_valset, 7) #### Coefficients from the entire training set of hitters (without NA values)
```

```
##  (Intercept)        AtBat         Hits        Walks        CRuns       CWalks
##   67.1085369   -2.1462987    7.0149547    8.0716640    1.2425113   -0.8337844
##    DivisionW      PutOuts
## -118.4364998    0.2526925
```

```
coef(model2, 7)          #### Coefficients from the entire data set of hitters (without NA values)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits       CHmRun
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073    1.4420538
##    DivisionW      PutOuts
## -129.9866432    0.2366813
```

Unfortunately `regsubsets` do not have a `prediction` method in-built. Hence write a function for prediction:

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

pred <- predict.regsubsets(model_valset, hitters[valset,], id = 7)
MSE.valset7 <- mean((hitters$Salary[valset] - pred)^2)

MSE.valset7
```

```
## [1] 126849
```

```
MSE.val[7]
```

```
## [1] 126849
```

k-FOLD CROSS VALIDATION: We now try to choose among the models of different sizes using cross validation. This approach is somewhat involved, as we must perform best subset selection within each of the `k` training sets. Despite this, we see that with its clever subsetting syntax, R makes this job quite easy. First, we create a vector that allocates each observation to one of k = 10 folds, and we create a matrix in which we will store the results.

```
k <- 10
set.seed (1)
folds <- sample(1:k, nrow(hitters), replace = TRUE)
cv.errors <- matrix(NA, k, 19, dimnames = list(NULL, paste(1:19)))
folds
```

```
##   [1]  9  4  7  1  2  7  2  3  1  5  5 10  6 10  7  9  5  5  9  9  5  5  2 10  9
##  [26]  1  4  3  6 10 10  6  4  4 10  9  7  6  9  8  9  7  8  6 10  7  3 10  6  8
##  [51]  2  2  6  6  1  3  3  8  6  7  6  8  7  1  4  8  9  9  7  4  7  6  1  5  6
```

```
##  [76]  1  9  7  7  3  6  2 10 10  7  3  2 10  1 10 10  8 10  5  7  8  5  6  8  1
## [101]  3 10  3  1  6  6  4  9  5  1  3  6  3  7  3  3  1  9  2  8  6  1  2  7  7
## [126]  4  9  8  3  5  3  4  2  1  7  9 10 10  2  2  3  1  2  3  3  3  8  9  2 10
## [151]  8 10  4  5  9  5  7  5  6  4  2  1  3  8  9  6  1  4  5  9  5  8  4  1  9
## [176]  5  1  5  4 10 10  9  8  5  5  6  6  2  2  8  4 10  8  5  5  8  8  7  4  4
## [201]  1 10  4  9  9  9  9  6  6  4  3  3  9  9  7  9  5  7  4  4 10  8  1 10  2
## [226] 10  1  1  4  5  5  6  9  8  5  1  2  1  8  5  8 10  7  7  2  9  4  2  5  2
## [251]  4  3  6  9  7  5  5  1  1 10  1  3 10
```

```
cv.errors
```

```
##       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
##  [1,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [2,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [3,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [4,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [5,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [6,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [7,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [8,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [9,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [10,] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
```

Now we write a for loop that performs cross-validation. In the j-th fold, the elements of folds that equal j are in the test set, and the remainder are in the training set. We make our predictions for each model size (using our new `predict.regsubsets` function), compute the test errors on the appropriate subset, and store them in the appropriate slot in the matrix cv.errors.

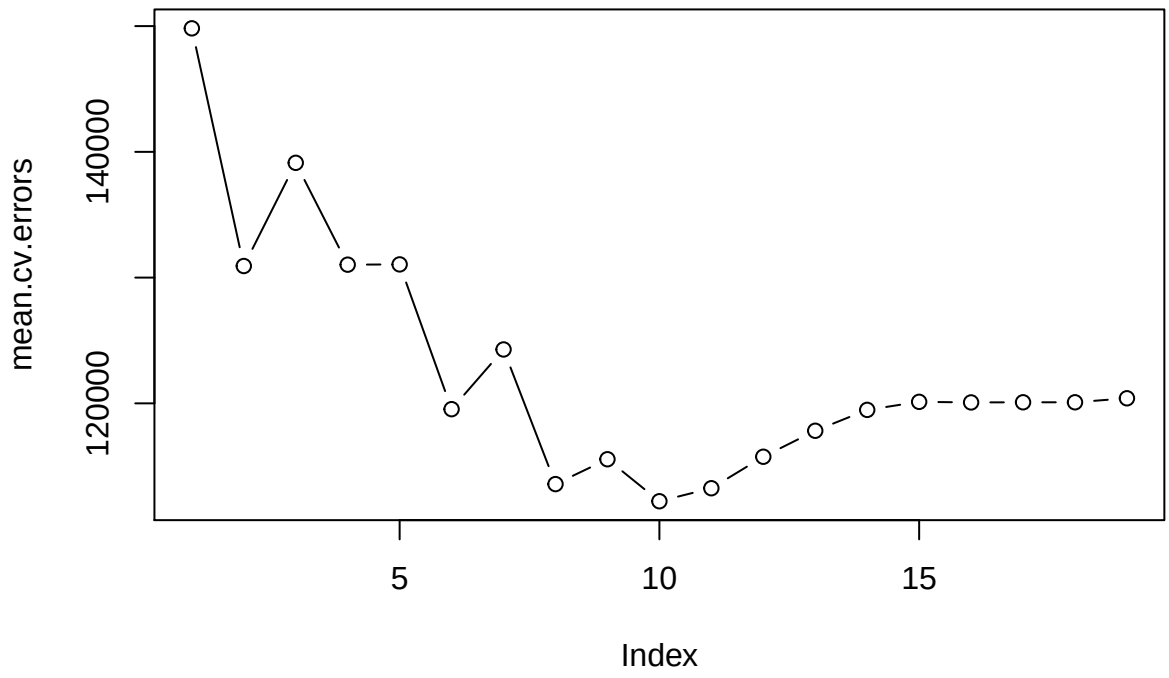
```
for (j in 1:k) {
  # best.cv means the best subset
 best.cv <- regsubsets(Salary ~ ., data = hitters[folds !=j,], nvmax = 19)
 for (i in 1:19) {
 pred <- predict.regsubsets(best.cv, hitters[folds == j,], id = i)
 cv.errors[j,i] <- mean((hitters$Salary[folds == j] - pred)^2)
 }
}
```

We see that cross-validation selects an 10-variable model. We now perform best subset selection on the full data set in order to obtain the 10-variable model.

```
mean.cv.errors <- apply(cv.errors , 2, mean)
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 149821.1 130922.0 139127.0 131028.8 131050.2 119538.6 124286.1 113580.0
##        9       10       11       12       13       14       15       16
## 115556.5 112216.7 113251.2 115755.9 117820.8 119481.2 120121.6 120074.3
##       17       18       19
## 120084.8 120085.8 120403.5
```

```
par(mfrow = c(1,1))
plot(mean.cv.errors, type = 'b')
```




```
which.min(mean.cv.errors)
```

```
## 10
## 10
```

```
coef(model2,10)
```

```
##  (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
```

```
coef(best.cv,10)
```

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

LASSO: The generalized linear model with penalized maximum likelihood package glmnet in R implements the LASSO method. To run the `glmnet()` function, we need to pass in the arguments as X (input matrix), y (output vector), rather than the y~X format that we used thus far. The model.matrix() function produces a matrix corresponding to the 19 predictors and the intercept and helps transform qualitative variables into dummy quantitative variables. This is important since `glmnet()` works only with quantitative variabes.

```r
#install.packages("glmnet")
library(glmnet)
```

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

```
## Loaded glmnet 4.0-2
```

```r
# lasso
# same as above, define the matrix for multiplication, but this time round its for the entire dataset
X <- model.matrix(Salary ~., hitters)
y <- hitters$Salary
# str(X)
```

We now choose a range for the lambda parameters and create a training and test set. We then build the LASSO model on this data. The output from the model provides the Df (number of nonzeros), %Dev and Lambda values. The deviance measure is given as 2(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation). Null deviance is defined to be 2(loglike_sat - loglike(NULL)) where the NULL model refers to the intercept model only. The deviance ratio is dev.ratio=1-deviance/nulldev. As lambda decreases, the dev.ratio increases (more importance given to model fit than model complexity).

```r
# lasso on train set, do cross-validation on test set
grid <- 10^seq(10, -2, length = 100)
set.seed(1)

# make the training set half the data
train <- sample(1:nrow(X), nrow(X) / 2)
test <- -train
modellasso <- glmnet(X[train,], y[train], lambda = grid)
summary(modellasso)
```

```
##           Length Class     Mode
## a0         100   -none-    numeric
## beta      2000   dgCMatrix S4
## df         100   -none-    numeric
## dim          2   -none-    numeric
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         4   -none-    call
## nobs         1   -none-    numeric
```

```r
modellasso
```

```
##
## Call:  glmnet(x = X[train, ], y = y[train], lambda = grid)
##
##     Df  %Dev    Lambda
```

```
## 1    0  0.00 1.000e+10
## 2    0  0.00 7.565e+09
## 3    0  0.00 5.722e+09
## 4    0  0.00 4.329e+09
## 5    0  0.00 3.275e+09
## 6    0  0.00 2.477e+09
## 7    0  0.00 1.874e+09
## 8    0  0.00 1.417e+09
## 9    0  0.00 1.072e+09
## 10   0  0.00 8.111e+08
## 11   0  0.00 6.136e+08
## 12   0  0.00 4.642e+08
## 13   0  0.00 3.511e+08
## 14   0  0.00 2.656e+08
## 15   0  0.00 2.009e+08
## 16   0  0.00 1.520e+08
## 17   0  0.00 1.150e+08
## 18   0  0.00 8.697e+07
## 19   0  0.00 6.579e+07
## 20   0  0.00 4.977e+07
## 21   0  0.00 3.765e+07
## 22   0  0.00 2.848e+07
## 23   0  0.00 2.154e+07
## 24   0  0.00 1.630e+07
## 25   0  0.00 1.233e+07
## 26   0  0.00 9.326e+06
## 27   0  0.00 7.055e+06
## 28   0  0.00 5.337e+06
## 29   0  0.00 4.037e+06
## 30   0  0.00 3.054e+06
## 31   0  0.00 2.310e+06
## 32   0  0.00 1.748e+06
## 33   0  0.00 1.322e+06
## 34   0  0.00 1.000e+06
## 35   0  0.00 7.565e+05
## 36   0  0.00 5.722e+05
## 37   0  0.00 4.329e+05
## 38   0  0.00 3.275e+05
## 39   0  0.00 2.477e+05
## 40   0  0.00 1.874e+05
## 41   0  0.00 1.417e+05
## 42   0  0.00 1.072e+05
## 43   0  0.00 8.111e+04
## 44   0  0.00 6.136e+04
## 45   0  0.00 4.642e+04
## 46   0  0.00 3.511e+04
## 47   0  0.00 2.656e+04
## 48   0  0.00 2.009e+04
## 49   0  0.00 1.520e+04
## 50   0  0.00 1.150e+04
## 51   0  0.00 8.697e+03
## 52   0  0.00 6.579e+03
## 53   0  0.00 4.977e+03
## 54   0  0.00 3.765e+03
```

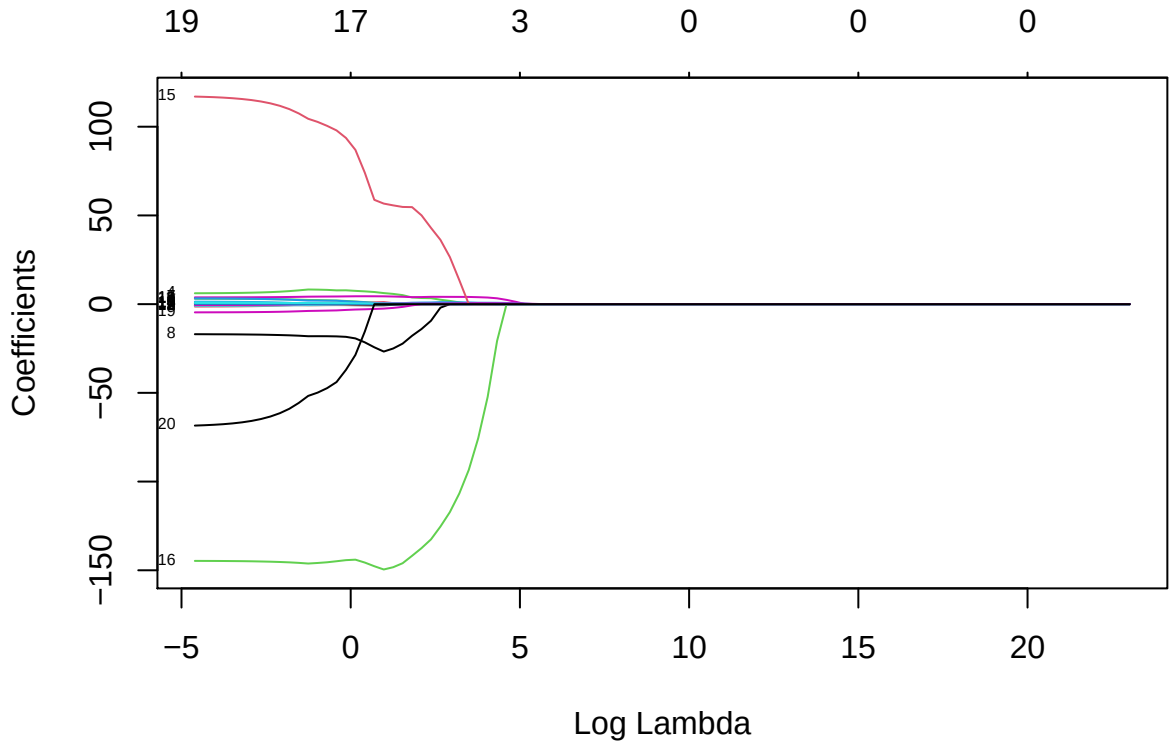
```
## 55   0  0.00 2.848e+03
## 56   0  0.00 2.154e+03
## 57   0  0.00 1.630e+03
## 58   0  0.00 1.233e+03
## 59   0  0.00 9.330e+02
## 60   0  0.00 7.060e+02
## 61   0  0.00 5.340e+02
## 62   0  0.00 4.040e+02
## 63   0  0.00 3.050e+02
## 64   1  9.18 2.310e+02
## 65   2 22.05 1.750e+02
## 66   3 32.38 1.320e+02
## 67   3 38.58 1.000e+02
## 68   4 42.90 7.600e+01
## 69   7 46.07 5.700e+01
## 70   8 48.26 4.300e+01
## 71   8 49.60 3.300e+01
## 72  10 50.62 2.500e+01
## 73  10 51.45 1.900e+01
## 74  11 52.14 1.400e+01
## 75  10 53.11 1.100e+01
## 76  10 53.58 8.000e+00
## 77  12 53.97 6.000e+00
## 78  12 54.50 5.000e+00
## 79  13 54.99 4.000e+00
## 80  14 55.36 3.000e+00
## 81  14 56.40 2.000e+00
## 82  15 57.19 2.000e+00
## 83  16 57.67 1.000e+00
## 84  17 58.00 1.000e+00
## 85  16 58.24 1.000e+00
## 86  16 58.34 0.000e+00
## 87  17 58.40 0.000e+00
## 88  19 58.43 0.000e+00
## 89  19 58.50 0.000e+00
## 90  19 58.54 0.000e+00
## 91  19 58.57 0.000e+00
## 92  19 58.59 0.000e+00
## 93  19 58.60 0.000e+00
## 94  19 58.60 0.000e+00
## 95  19 58.61 0.000e+00
## 96  19 58.61 0.000e+00
## 97  19 58.61 0.000e+00
## 98  19 58.61 0.000e+00
## 99  19 58.62 0.000e+00
## 100 19 58.62 0.000e+00
```

```
deviance(modellasso)
```

```
##   [1] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 23667021
##   [9] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 23667021
##  [17] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 23667021
##  [25] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 23667021
##  [33] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 23667021
```

```
##  [41] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 23667021
##  [49] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 23667021
##  [57] 23667021 23667021 23667021 23667021 23667021 23667021 23667021 21493596
##  [65] 18448591 16004435 14536141 13513584 12762702 12245648 11927561 11686083
##  [73] 11491020 11326290 11097815 10986274 10893993 10768183 10653596 10564248
##  [81] 10318467 10131203 10018387  9939979  9884276  9860097  9846001  9837861
##  [89]  9822121  9811413  9804973  9801069  9798650  9797130  9796148  9795507
##  [97]  9795081  9794787  9794588  9794444
```

```r
plot(modellasso, xvar = "lambda", label = TRUE)
```




```r
# df = number of variables chosen (ie. number of non zero beta coefficients)
# %Dev = how much of the null model deviance was explained by the model
# ie. summation (y_i - y_bar) ^2
# no parameter in the model (only the intercept)
# how much of this is explained by the beta values
# for huge lambdas, betas are all zero
```

We see from the plot that as lambda increases, many of the coefficients get close to zero. We can retrieve these coefficients as follows. Note that the number of non-zero coefficients does not change in a fully monotonic way, as lambda increases or decreases.

```r
# number of non-zero beta coefficients
modellasso$df
```

```
##   [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##  [26]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##  [51]  0  0  0  0  0  0  0  0  0  0  0  0  0  1  2  3  3  4  7  8  8 10 10 11 10
##  [76] 10 12 12 13 14 14 15 16 17 16 16 17 19 19 19 19 19 19 19 19 19 19 19 19 19
```

```
#modellasso$beta
#coef(modellasso)
```

Predictions: We start with a prediction for the model fitted with lambda = 100. The test mean squared error for this model is 151898.3. Suppose, we change lambda to 50, we get 144900 and if we change lambda to 200, we get 183422.8. Note that by default if prediction is done at lambda values that are not tried in the fitting algorithm, it uses linear interpolation to make predictions. We can use `exact = T` in the argument to get the exact value by refitting. In addition, you need to then pass also the original training set data to the function. You get a test error of 115096 with the full model while 193253 with a very large value of lambda. Thus choosing lambda appropriately will be important in the quality of the fit. This can be done with cross-validation.

```
# fit on test set to get test MSE error value
modellasso$lambda
```

```
##   [1] 1.000000e+10 7.564633e+09 5.722368e+09 4.328761e+09 3.274549e+09
##   [6] 2.477076e+09 1.873817e+09 1.417474e+09 1.072267e+09 8.111308e+08
##  [11] 6.135907e+08 4.641589e+08 3.511192e+08 2.656088e+08 2.009233e+08
##  [16] 1.519911e+08 1.149757e+08 8.697490e+07 6.579332e+07 4.977024e+07
##  [21] 3.764936e+07 2.848036e+07 2.154435e+07 1.629751e+07 1.232847e+07
##  [26] 9.326033e+06 7.054802e+06 5.336699e+06 4.037017e+06 3.053856e+06
##  [31] 2.310130e+06 1.747528e+06 1.321941e+06 1.000000e+06 7.564633e+05
##  [36] 5.722368e+05 4.328761e+05 3.274549e+05 2.477076e+05 1.873817e+05
##  [41] 1.417474e+05 1.072267e+05 8.111308e+04 6.135907e+04 4.641589e+04
##  [46] 3.511192e+04 2.656088e+04 2.009233e+04 1.519911e+04 1.149757e+04
##  [51] 8.697490e+03 6.579332e+03 4.977024e+03 3.764936e+03 2.848036e+03
##  [56] 2.154435e+03 1.629751e+03 1.232847e+03 9.326033e+02 7.054802e+02
##  [61] 5.336699e+02 4.037017e+02 3.053856e+02 2.310130e+02 1.747528e+02
##  [66] 1.321941e+02 1.000000e+02 7.564633e+01 5.722368e+01 4.328761e+01
##  [71] 3.274549e+01 2.477076e+01 1.873817e+01 1.417474e+01 1.072267e+01
##  [76] 8.111308e+00 6.135907e+00 4.641589e+00 3.511192e+00 2.656088e+00
##  [81] 2.009233e+00 1.519911e+00 1.149757e+00 8.697490e-01 6.579332e-01
##  [86] 4.977024e-01 3.764936e-01 2.848036e-01 2.154435e-01 1.629751e-01
##  [91] 1.232847e-01 9.326033e-02 7.054802e-02 5.336699e-02 4.037017e-02
##  [96] 3.053856e-02 2.310130e-02 1.747528e-02 1.321941e-02 1.000000e-02
```

```
# s = lambda value
predictlasso1 <- predict(modellasso, newx = X[test,], s = 100)
mean((predictlasso1 - y[test])^2)
```

```
## [1] 151898.3
```

```
predictlasso2 <- predict(modellasso,newx = X[test,], s = 50)
mean((predictlasso2 - y[test])^2)
```

```
## [1] 144900
```

```
predictlasso3 <- predict(modellasso,newx = X[test,], s = 200)
mean((predictlasso3 - y[test])^2)
```

```
## [1] 183422.8
```

```
?predict.glmnet
predictlasso1a <- predict(modellasso, newx = X[test,], s = 100, exact = T, x = X[train,], y = y[train])
mean((predictlasso1a - y[test])^2)
```

```
## [1] 151898.3
```

```
predictlasso2a <- predict(modellasso, newx = X[test,], s = 50, exact = T, x = X[train,], y = y[train])
mean((predictlasso2a - y[test])^2)
```

```
## [1] 145013.4
```

```
predictlasso3a <- predict(modellasso, newx = X[test,], s = 200, exact = T, x = X[train,], y = y[train])
mean((predictlasso3a - y[test])^2)
```

```
## [1] 183132.6
```

```
# s = 0 gives MSE value
predictlasso4 <- predict(modellasso, newx = X[test,], s = 0, exact = T, x = X[train,], y = y[train])
mean((predictlasso4 - y[test])^2)
```

```
## [1] 166929.8
```

```
# s = maximum is the null model
predictlasso5 <- predict(modellasso, newx = X[test,], s = 10^10, exact = T, x = X[train,], y = y[train])
mean((predictlasso5 - y[test])^2)
```

```
## [1] 224669.9
```

Cross-validation: By default, you perform 10 fold cross validation. Note that glmnet uses randomization in choosing the folds which we should be able to control better by setting the seed to be the same. Default k=10. The optimal value of lambda found from cross validation is 8.461927. You can plot the lambda parameter with the cross-validated mean error (cvm). We see that the best fit from model with the optimal lambda gives a much smaller error on the test set than the model which is based on complete linear regression or the model with only an intercept. We can print out the coefficients to identify that 7 variables are chosen (excluding the intercept).

```
set.seed(2)
?cv.glmnet
cvlasso <- cv.glmnet(X[train,], y[train])

# lambda is chosen automatically here
cvlasso$glmnet.fit
```

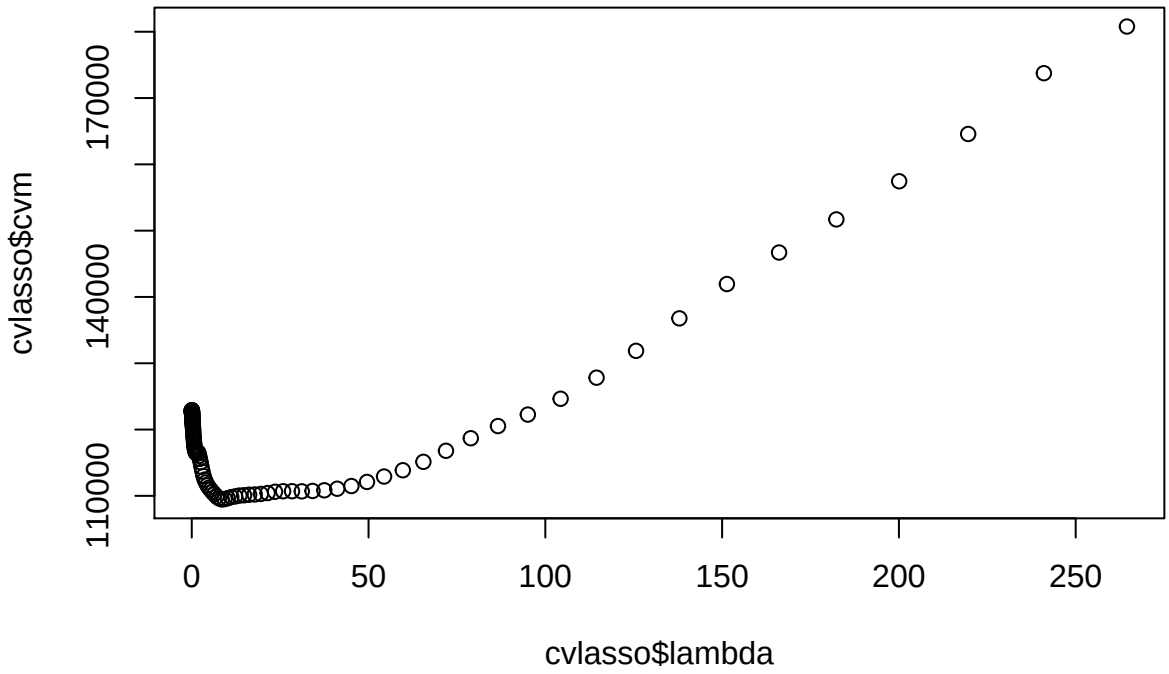
```
##
## Call:  glmnet(x = X[train, ], y = y[train])
##
##     Df  %Dev  Lambda
## 1    0  0.00 264.500
## 2    1  6.57 241.000
## 3    1 12.03 219.600
## 4    2 16.64 200.100
## 5    2 20.51 182.300
## 6    3 24.00 166.100
## 7    3 27.88 151.400
## 8    3 31.11 137.900
## 9    3 33.78 125.700
## 10   3 36.00 114.500
## 11   3 37.85 104.300
## 12   3 39.38  95.050
## 13   4 40.72  86.610
## 14   4 42.28  78.920
## 15   6 43.60  71.910
## 16   6 44.73  65.520
## 17   6 45.67  59.700
## 18   7 46.55  54.390
## 19   8 47.33  49.560
## 20   8 48.00  45.160
## 21   8 48.55  41.150
## 22  10 49.02  37.490
## 23   8 49.43  34.160
## 24   8 49.78  31.130
## 25   9 50.14  28.360
## 26   9 50.47  25.840
## 27  10 50.81  23.550
## 28  10 51.10  21.450
## 29  10 51.35  19.550
## 30  10 51.55  17.810
## 31  10 51.72  16.230
## 32  11 51.95  14.790
## 33  11 52.36  13.470
## 34  11 52.70  12.280
## 35  11 52.99  11.190
## 36  10 53.21  10.190
## 37  10 53.38   9.287
## 38  10 53.52   8.462
## 39  10 53.63   7.710
## 40  11 53.74   7.025
## 41  11 53.87   6.401
## 42  12 54.09   5.832
## 43  12 54.28   5.314
## 44  12 54.44   4.842
## 45  12 54.57   4.412
## 46  13 54.73   4.020
## 47  13 54.91   3.663
## 48  13 55.06   3.338
## 49  13 55.19   3.041
## 50  13 55.29   2.771
```

```
## 51  14 55.44   2.525
## 52  14 55.87   2.300
## 53  14 56.25   2.096
## 54  15 56.57   1.910
## 55  15 56.86   1.740
## 56  15 57.10   1.586
## 57  15 57.30   1.445
## 58  15 57.47   1.316
## 59  16 57.61   1.199
## 60  17 57.73   1.093
## 61  17 57.85   0.996
## 62  17 57.96   0.907
## 63  17 58.05   0.827
## 64  18 58.14   0.753
## 65  17 58.20   0.686
## 66  17 58.25   0.625
## 67  16 58.29   0.570
## 68  16 58.32   0.519
## 69  16 58.34   0.473
## 70  16 58.36   0.431
## 71  16 58.38   0.393
## 72  17 58.40   0.358
## 73  17 58.41   0.326
## 74  17 58.42   0.297
## 75  19 58.44   0.271
## 76  19 58.45   0.247
## 77  19 58.49   0.225
## 78  19 58.49   0.205
## 79  19 58.52   0.187
## 80  19 58.53   0.170
## 81  19 58.54   0.155
## 82  19 58.55   0.141
## 83  19 58.56   0.129
## 84  19 58.57   0.117
## 85  19 58.58   0.107
## 86  19 58.58   0.097
## 87  19 58.59   0.089
## 88  19 58.59   0.081
## 89  19 58.59   0.074
## 90  19 58.60   0.067
## 91  19 58.60   0.061
## 92  19 58.60   0.056
## 93  19 58.60   0.051
## 94  19 58.60   0.046
## 95  19 58.61   0.042
## 96  19 58.61   0.038
## 97  19 58.61   0.035
## 98  19 58.61   0.032
## 99  19 58.61   0.029
## 100 19 58.61   0.026
```

```r
# gives minimum lambda value
cvlasso$lambda.min
```

```
## [1] 8.461927
```

```
# after finding the minimum lambda value, use this lambda to validate on the the test
plot(cvlasso$lambda, cvlasso$cvm)
```




```
predictlassocv <- predict(modellasso, s = 8.461927, newx = X[test,])
mean((predictlassocv - y[test])^2)
```

```
## [1] 143816.9
```

```
coef(modellasso, s = 8.461927)
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  122.2856247
## (Intercept)    .
## AtBat          .
## Hits           .
## HmRun          3.4405109
## Runs           .
## RBI            .
## Walks          4.0062805
## Years        -13.3907376
## CAtBat         .
```

```
## CHits          0.2185291
## CHmRun         0.8996567
## CRuns          .
## CRBI           0.3882897
## CWalks         .
## LeagueN       49.0843552
## DivisionW   -136.8445001
## PutOuts        0.1183245
## Assists        0.1423117
## Errors         .
## NewLeagueN     .
```