Chapter 6

Lab

```
hitters <- as.data.table(ISLR::Hitters)

# ggpairs(hitters)
```

```
sum(is.na(hitters$Salary))
```

```
[1] 59
```

```
full <- nrow(hitters)
```

```
hitters <- hitters[ complete.cases(hitters), ]

1 - (nrow(hitters) / full) # drop about 18% of the data
```

```
[1] 0.1832298
```

```
regfit.full <- regsubsets(Salary ~., hitters)
summary(regfit.full)
```

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

```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
reg.summary$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
```

```r
par(mfrow = c(2,2))
plot(reg.summary$rss, xlab = "Number of Variables", ylab = "RSS")
which.min(reg.summary$rss)
```

```
[1] 19
```

```r
points(19, reg.summary$rss[19], col="red", cex=2, pch=20)
plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adj. R^2")
which.max(reg.summary$adjr2)
```

```
[1] 11
```

```r
points(11, reg.summary$adjr2[11], col="red", cex=2, pch=20)
plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
```

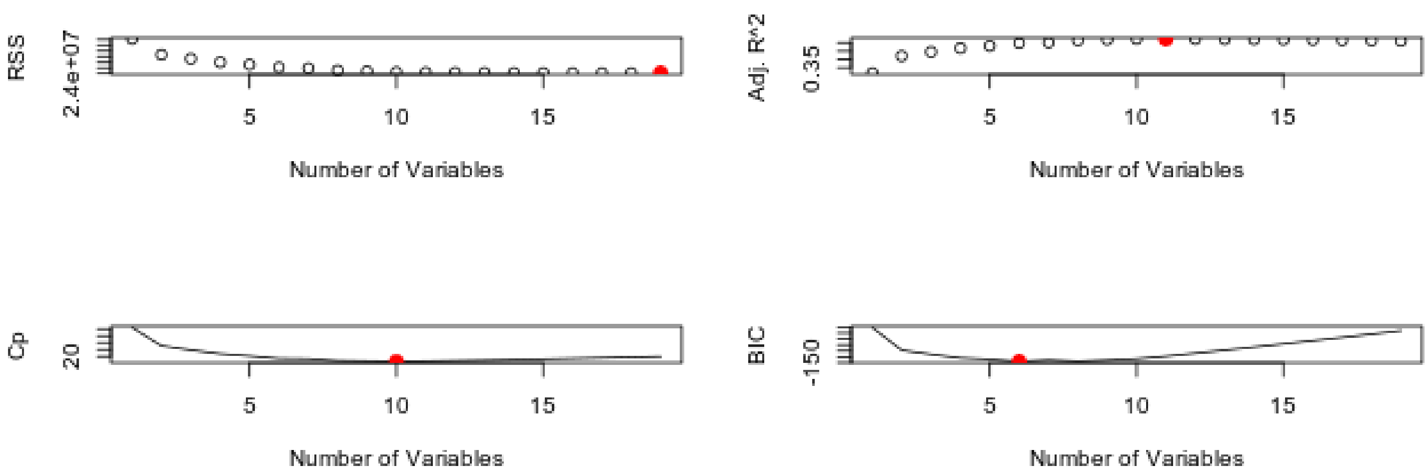
```
which.min(reg.summary$cp)
```

```
[1] 10
```

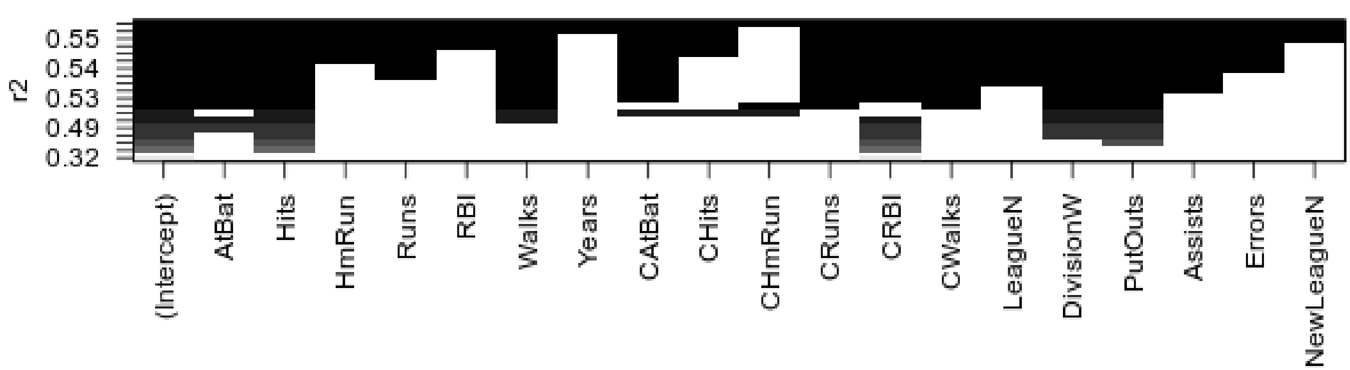
```
points(10, reg.summary$cp[10], col="red", cex=2, pch=20)
which.min(reg.summary$bic)
```

```
[1] 6
```

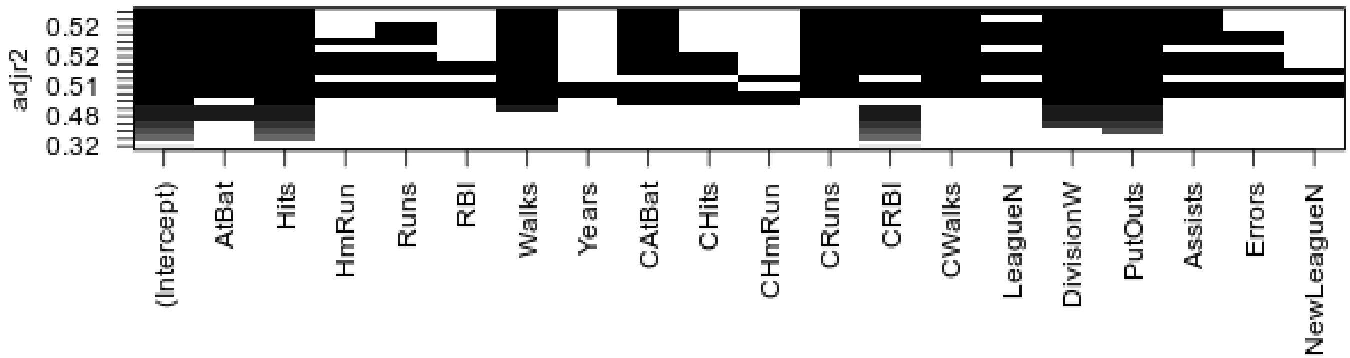
```
plot(reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
points(6, reg.summary$bic[6], col = "red", cex = 2, pch = 20)
```



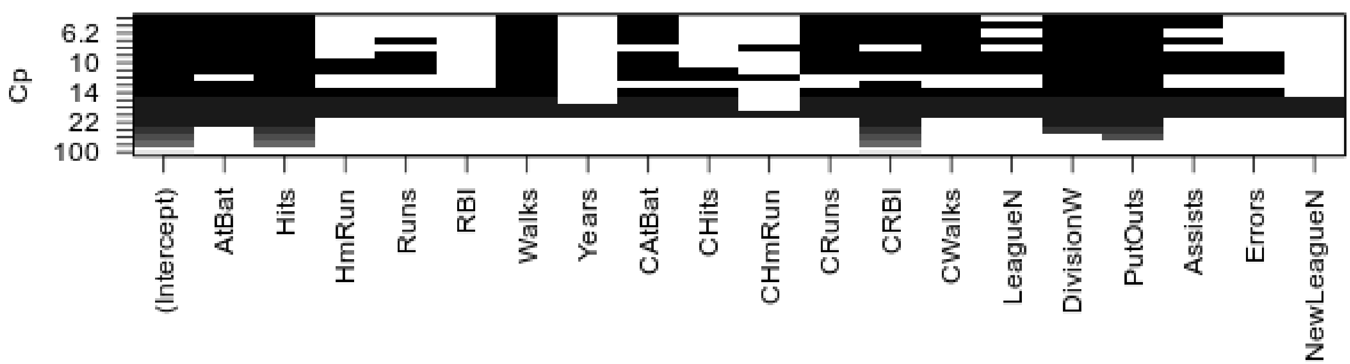

```
plot(regfit.full, scale = "r2")
```



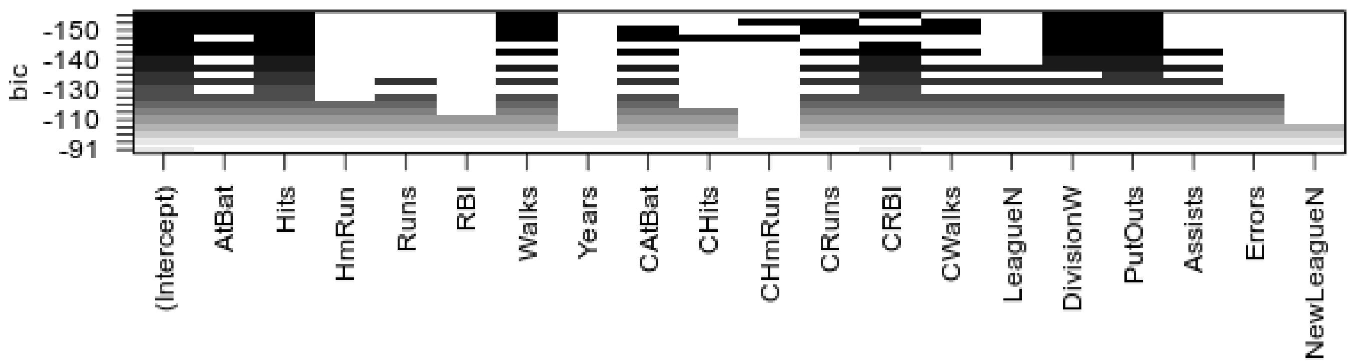
```r
plot(regfit.full, scale = "adjr2")
```




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




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




```r
coef(regfit.full, 6)
```

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

```r
regfit.fwd <- regsubsets(Salary ~., data = hitters, nvmax = 19)
summary(regfit.full)
```

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

```r
regfit.bwd <- regsubsets(Salary ~., data = hitters, nvmax = 19)
summary(regfit.bwd)
```

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

Cross-Validation using best subset selection:

```r
train <- sample(c(T, F), size = nrow(hitters), replace = T)
test <- (!train)
```

```r
regfit.best <- regsubsets(Salary ~., data = hitters[train,], nvmax = 19)
test_mat <- model.matrix(Salary ~., data = hitters)

val_errors <- rep(NA, 19)
for(i in 1:19)
{
   coefi <- coef(regfit.best, id = i)
   pred <- test_mat[, names(coefi)] %*% coefi
   val_errors[i] <- mean((hitters$Salary[test] - pred)^2)
}
```

```
Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length
```

```
Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length
```

```
Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length

Warning in hitters$Salary[test] - pred: longer object length is not a multiple
of shorter object length
```

```
which.min(val_errors)
```

```
[1] 1
```

```
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
}

regfit.best <- regsubsets(Salary ~ ., data = hitters, nvmax = 19)
coef(regfit.best, 10)
```

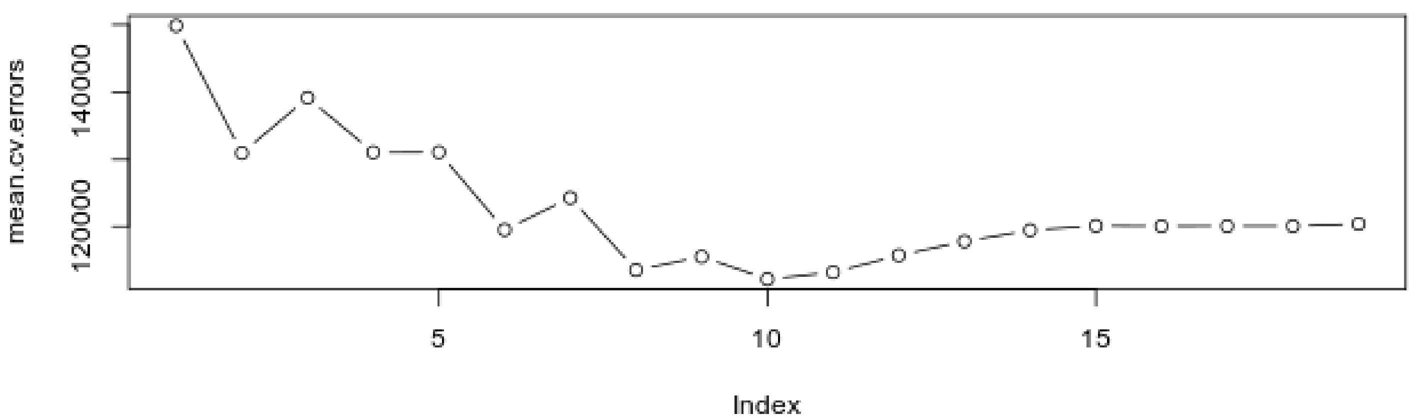
```
 (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
 162.5354420   -2.1686501    6.9180175    5.7732246   -0.1300798    1.4082490
        CRBI       CWalks    DivisionW      PutOuts      Assists
   0.7743122   -0.8308264 -112.3800575    0.2973726    0.2831680
```

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

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

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
```

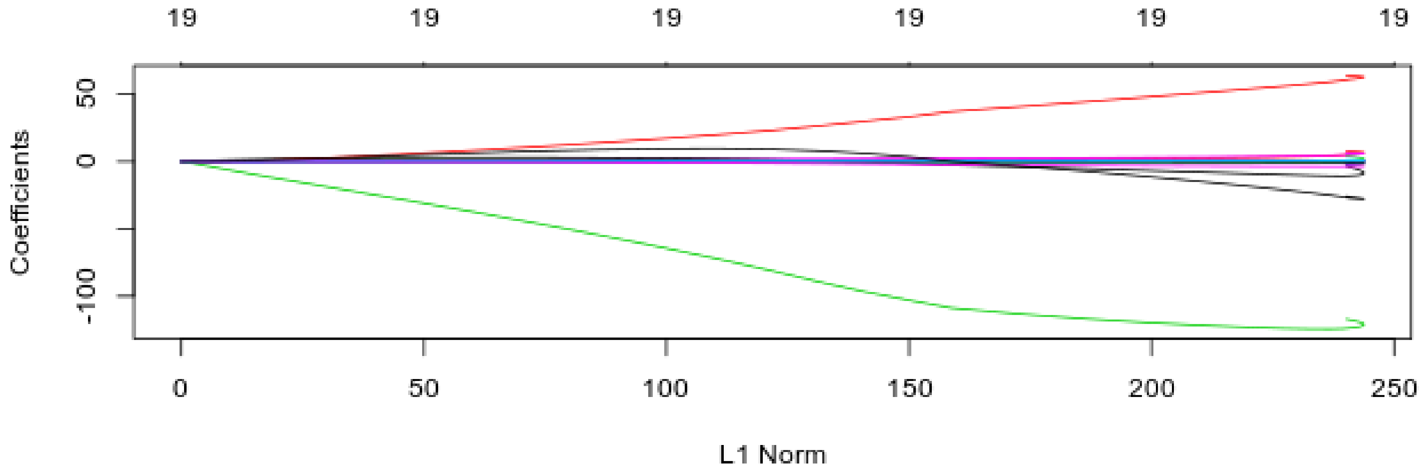
```
       9        10        11        12        13        14        15        16 
115556.5 112216.7 113251.2 115755.9 117820.8 119481.2 120121.6 120074.3 
      17        18        19 
120084.8 120085.8 120403.5 
```

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



Ridge Regression

```
x <- model.matrix(Salary ~ ., hitters)[, -1]
y <- hitters$Salary

grid <- 10^seq(10, -2, length = 100)

ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
plot(ridge.mod)
```




```r
dim(coef(ridge.mod))
```

```
[1]  20 100
```

```r
ridge.mod$lambda[50]
```

```
[1] 11497.57
```

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

```
  (Intercept)         AtBat          Hits         HmRun          Runs 
407.356050200   0.036957182   0.138180344   0.524629976   0.230701523 
          RBI         Walks         Years        CAtBat         CHits 
  0.239841459   0.289618741   1.107702929   0.003131815   0.011653637 
       CHmRun         CRuns          CRBI        CWalks       LeagueN 
  0.087545670   0.023379882   0.024138320   0.025015421   0.085028114 
    DivisionW       PutOuts       Assists        Errors    NewLeagueN 
 -6.215440973   0.016482577   0.002612988  -0.020502690   0.301433531 
```

```r
sqrt(sum(coef(ridge.mod)[-1, 50])^2)
```

```
[1] 3.08789
```

```r
index <- 60
ridge.mod$lambda[index]
```

```
[1] 705.4802
```

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

```
 (Intercept)        AtBat         Hits        HmRun         Runs          RBI 
 54.32519950   0.11211115   0.65622409   1.17980910   0.93769713   0.84718546 
       Walks        Years       CAtBat        CHits       CHmRun        CRuns 
```

```
  1.31987948   2.59640425   0.01083413   0.04674557   0.33777318   0.09355528
        CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
  0.09780402   0.07189612  13.68370191 -54.65877750   0.11852289   0.01606037
      Errors   NewLeagueN
 -0.70358655   8.61181213
```

```r
sqrt(sum(coef(ridge.mod)[-1, index])^2)
```

```
[1] 24.62435
```

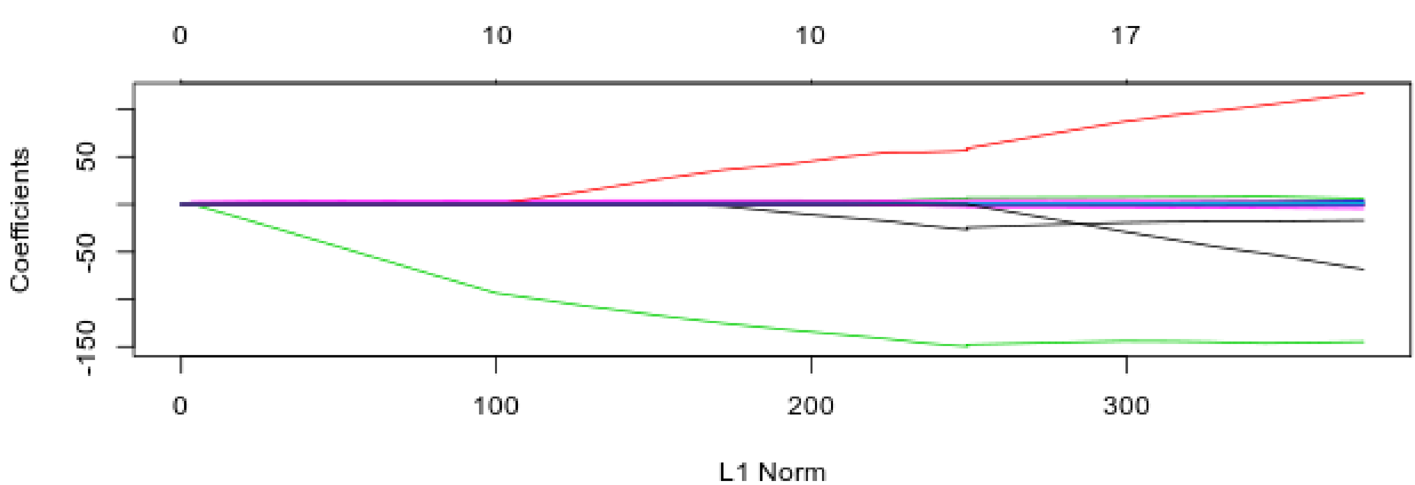
```r
set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)

y.test <- y[test]

ridge.mod <- glmnet(x[train,], y[train], alpha = 0, lambda = grid, threshold = 1e-12)

ridge.pred <- predict(ridge.mod, s = 4, newx = x[test,])
mean((ridge.pred - y.test)^2)
```

```
[1] 142226.5
```

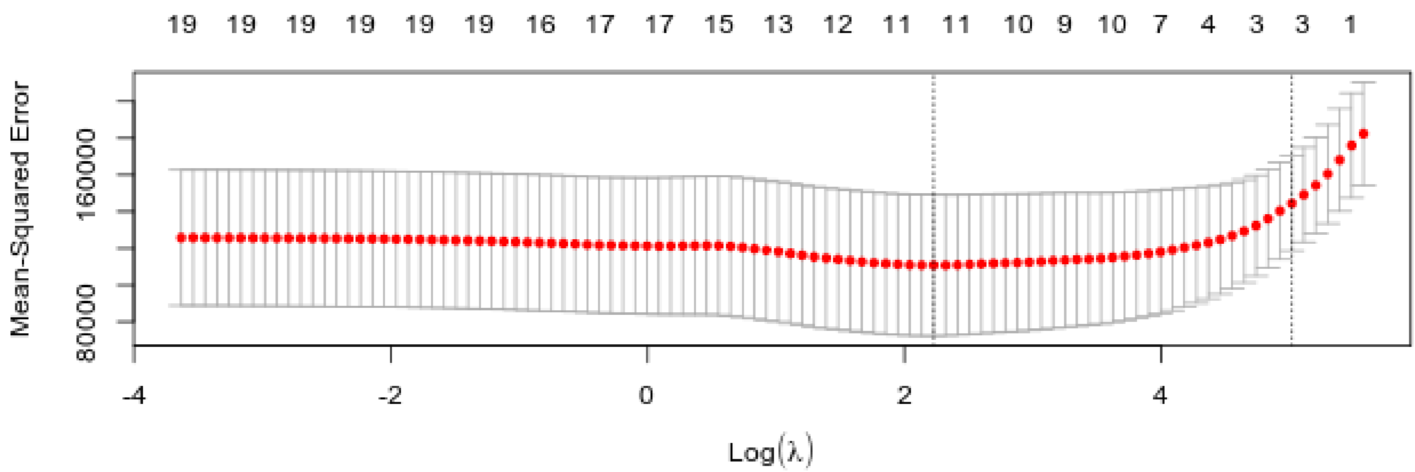
Lasso

```r
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```

```
Warning in regularize.values(x, y, ties, missing(ties)): collapsing to unique
'x' values
```



```r
set.seed(1)

cv.out <- cv.glmnet(x[train, ], y[train], alpha = 1)

plot(cv.out)
```




```r
bestlam <- cv.out$lambda.min
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])

mean((lasso.pred - y.test)^2)
```

```
[1] 143673.6
```

```r
out <- glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:20,]
lasso.coef
```

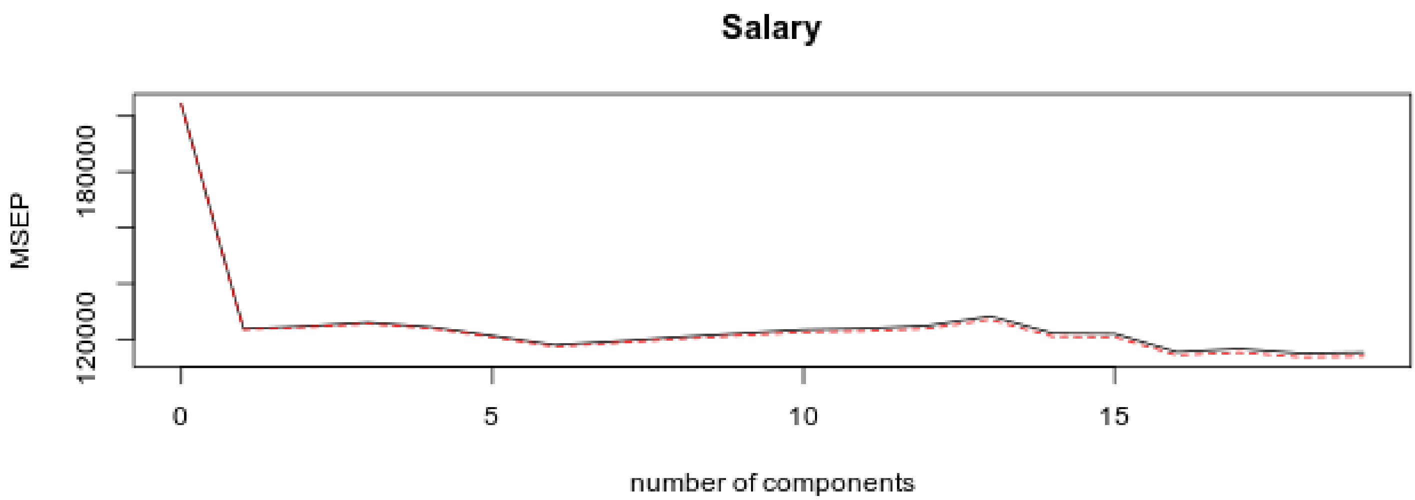
```
  (Intercept)         AtBat          Hits         HmRun          Runs
   1.27479059   -0.05497143    2.18034583    0.00000000    0.00000000
          RBI         Walks         Years        CAtBat         CHits
   0.00000000    2.29192406   -0.33806109    0.00000000    0.00000000
       CHmRun         CRuns          CRBI        CWalks       LeagueN
   0.02825013    0.21628385    0.41712537    0.00000000   20.28615023
    DivisionW       PutOuts       Assists        Errors    NewLeagueN
-116.16755870    0.23752385    0.00000000   -0.85629148    0.00000000
```

Principal Components Regression

```r
set.seed(2)

pcr.fit <- pcr(Salary ~., data = hitters, scale = T, validation = "CV")

summary(pcr.fit)
```

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

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV             452    351.9    353.2    355.0    352.8    348.4    343.6
adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
CV        349.7     349.4     339.9     341.6     339.2     339.6
adjCV     348.0     347.7     338.2     339.7     337.2     337.6

TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
        16 comps  17 comps  18 comps  19 comps
X          99.89     99.97     99.99    100.00
Salary     53.01     53.85     54.61     54.61
```

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




```r
set.seed(1)

pcr.fit <- pcr(Salary ~ ., data = hitters, subset = train, scale = T,
               validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```


```r
pcr.pred <- predict(pcr.fit, x[test,], ncomp = 7)
mean((pcr.pred - y.test)^2)
```

```
[1] 140751.3
```

Partial Least Squares

```r
set.seed(1)

pls.fit <- plsr(Salary ~ ., data = hitters, subset = train, scale = T,
                validation = "CV")

summary(pls.fit)
```

```
Data:   X dimension: 131 19
    Y dimension: 131 1
Fit method: kernelpls
Number of components considered: 19

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           428.3    325.5    329.9    328.8    339.0    338.9    340.1
adjCV        428.3    325.0    328.2    327.2    336.6    336.1    336.6
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       339.0    347.1    346.4     343.4     341.5     345.4     356.4
adjCV    336.2    343.4    342.8     340.2     338.3     341.8     351.1
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
CV        348.4     349.1     350.0     344.2     344.5     345.0
adjCV     344.2     345.0     345.9     340.4     340.6     341.1

TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X         39.13    48.80    60.09    75.07    78.58    81.12    88.21    90.71
Salary    46.36    50.72    52.23    53.03    54.07    54.77    55.05    55.66
        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X         93.17     96.05     97.08     97.61     97.97     98.70     99.12
Salary    55.95     56.12     56.47     56.68     57.37     57.76     58.08
        16 comps  17 comps  18 comps  19 comps
X          99.61     99.70     99.95    100.00
Salary     58.17     58.49     58.56     58.62
```

```r
pls.pred <- predict(pls.fit, x[test, ], ncomp = 2)
mean((pls.pred - y.test)^2)
```

```
[1] 145367.7
```

```r
pls.fit <- plsr(Salary ~ ., data = hitters, scale = T, ncomp = 2)
summary(pls.fit)
```

```
Data:   X dimension: 263 19
```

```
    Y dimension: 263 1
Fit method: kernelpls
Number of components considered: 2
TRAINING: % variance explained
        1 comps  2 comps
X         38.08    51.03
Salary    43.05    46.40
```

Conceptual

1.)

We perform best subset, forward stepwise, and backward stepwise selection on a single data set. For each approach, we obtain $p + 1$ models, containing $0, 1, 2, \ldots, p$ predictors.

a.) Which of the three models with k predictors has the smallest *training* RSS?

The model with the smallest training RSS will be the $C\binom{k}{p}$ *model with p = k.*

b.) Which of the three models with k predictors has the smallest *test* RSS?

Difficult to say. The best subset selection technique looks at more models, however, forward or backward selection could pick a better model by chance.

c.) T/F

i.) The predictors in the k-variable model identified by foward stepwise selection are a subset of the predictors in the (k+1)-variable model identified by forward stepwise selection.

T

ii.) The predictors isn the k-variable model identified by backward stepwise are a subset of the predictors in the (k+1)-variable model identified by backward stepwise selection.

T

iii.) The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the (k+1)-variable model identified by forward stepwise selection.

F. There is no link between these models predictors.

iv.) The predictors in the k-variable model identifed by forward stepwise are a subset of the predictors in the (k+1)-variable model identifed by backward stepwise selection.

F. There is no link between these models predictors.

v.) The predictors in the k-variable model identified by best subset are a subset of the predictors in the (k+1)-variable model identifed by best subset selection.

F. There is no link between these models predictors.

2.)

For parts (a) through (c), indicate which of i through iv. is correct.

a.) The lasso, relative to least squares, is:

i.) More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

T

ii.) More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

F

iii.) Less flexible and hence will give improved prediction accuracyt when its increase in bias is less than its decrease in variance.

F

iv.) Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

F

3.)

Suppose we estimate the regression coefficients in a linear regression model by minimizing:

$\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{i,j})$

subject to,

$\sum_{j=1}^{p} (|\beta_j|) \leq s$

a.) As we increase s from 0, the training RSS will:

Steadily decrease. As we increase s from 0, we are restricting the βj coefficients less and less (the coefficients will increase to their least squares estimates), and so the model is becoming more and more flexible which provokes a steady decrease in the training RSS.

b.) Test RSS will:

Decrease initially, and then eventually start increasing in a U shape. As we increase s from 0, we are restricting the βj coefficients less and less (the coefficients will increase to their least squares estimates), and so the model is becoming more and more flexible which provokes at first a decrease in the test RSS before increasing again after that in a typical U shape.

c.) Variance will:

Steadily increase. As we increase s from 0, we are restricting the βj coefficients less and less (the coefficients will increase to their least squares estimates), and so the model is becoming more and more flexible which provokes a steady increase in variance.

d.) (squared) bias will:

Steadily decrease. As we increase s from 0, we are restricting the βj coefficients less and less (the coefficients will increase to their least squares estimates), and so the model is becoming more and more flexible which provokes a steady decrease in bias.

e.) irreducible error will:

remain unchanged.

4.)

Suppose we estimate the regression coefficients in a linear regression model by minimizing:

$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}) + \lambda\sum_{j=1}^{p}\beta_j^2$

for a particular value of λ.

a.) As we increase λ from 0, the training RSS will:

Steadily increase.

b.) Test RSS will:

Decrease initially, then turn to a U shape.

c.) Variance will:

Steadily decrease.

d.) (squared) bias will:

Steadily increase

e.) Irreducible error will:

Remain constant.

5.)

It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting.

Suppose that $n = 2, p = 2, x_{11} = x_{12} = x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ so the estimate for the intercept in a least squares, ridge regresion or lasso model is zero: $\hat{\beta}_0 = 0$

Write out the ridge regression optimization problem in this setting:

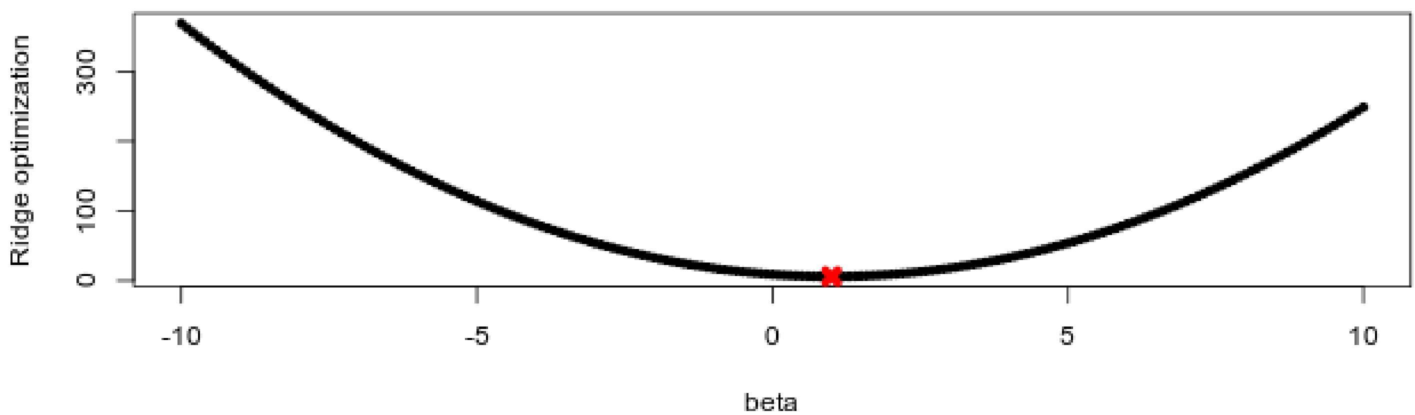
$(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2)^2 + (y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$

6.)

We will now explore (6.12) and (6.13) further.

a.) Consider (6.12) with p=1. For some choice of y1 and λ>0, plot (6.12) as a function of β1. Your plot should confirm that (6.12) is solved by (6.14).

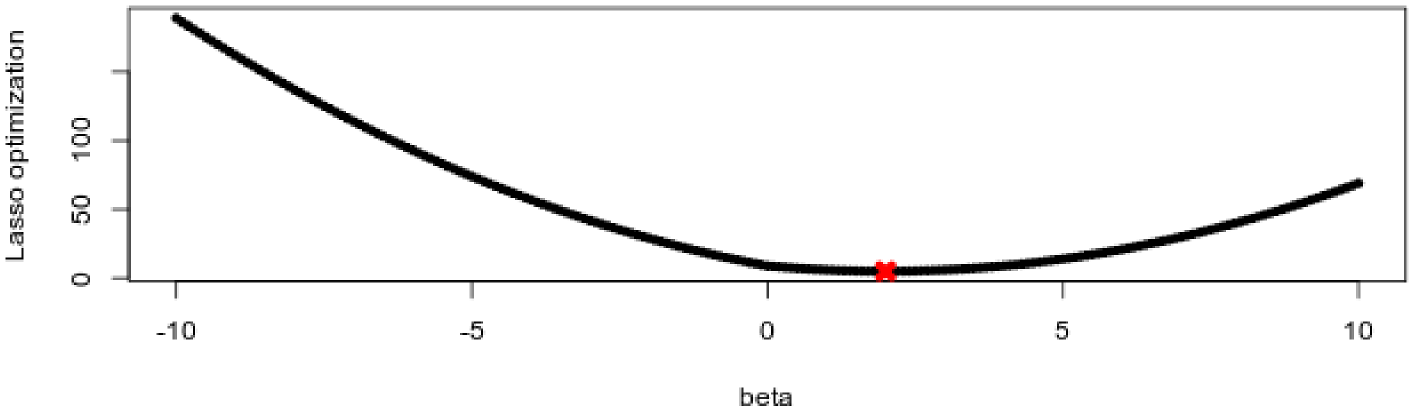
```
y <- 3
lambda <- 2
beta <- seq(-10, 10, 0.1)
plot(beta, (y - beta)^2 + lambda * beta^2, pch = 20, xlab = "beta", ylab = "Ridge optimization"
beta.est <- y / (1 + lambda)
points(beta.est, (y - beta.est)^2 + lambda * beta.est^2, col = "red", pch = 4, lwd = 5)
```



We may see that the function is minimized at β=y/(1+λ).

b.) Consider (6.13) with p=1. For some choice of y1 and λ>0, plot (6.13) as a function of β1. Your plot should confirm that (6.13) is solved by (6.15).

```
y <- 3
lambda <- 2
beta <- seq(-10, 10, 0.1)
plot(beta, (y - beta)^2 + lambda * abs(beta), pch = 20, xlab = "beta", ylab = "Lasso optimizati
beta.est <- y - lambda / 2
points(beta.est, (y - beta.est)^2 + lambda * abs(beta.est), col = "red", pch = 4, lwd = 5)
```



We may see that the function is minimized at $\beta = y - \lambda/2$ as $y > \lambda/2$.

Applied

8.)

In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

a.) Use the rnorm() function to generate a predictor X of length n=100, as well as a noise vector ε of length n=100.

```
set.seed(1)

n <- 100

x = rnorm(n)
eps = rnorm(n)
```

b.) Generate a response vector Y of length n=100 according to the model

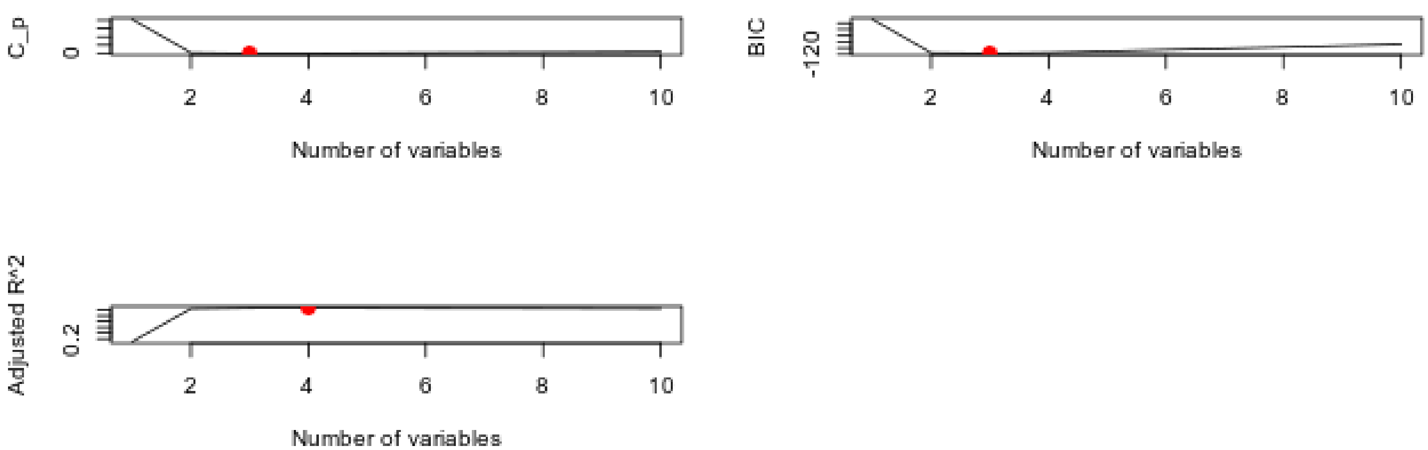
$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$

where β_0, β_1, β_2 and β_3 are constants of your choice.

```
b0 <- 2
b1 <- 3
b2 <- 0.3
b3 <- -1

y <- b0 + b1 * x + b2 * x^2 + b3 * x^3 + eps
```

c.) Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors X,X2,□,X10. What is the best model obtained according to Cp, BIC, and adjusted R2 ? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the data.frame() function to create a single data set containing both X and Y.

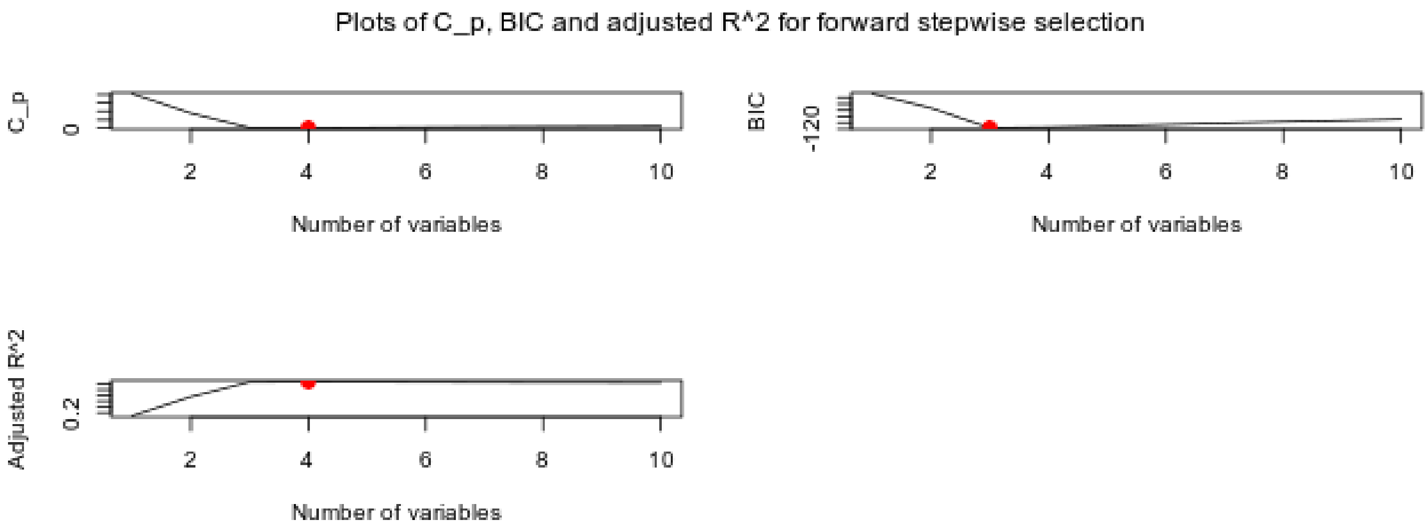
```
data.full <- data.frame(y = y, x = x)
regfit.full <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8)
reg.summary <- summary(regfit.full)

par(mfrow = c(2, 2))
plot(reg.summary$cp, xlab = "Number of variables", ylab = "C_p", type = "l")
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)], col = "red", cex =
plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)], col = "red", ce
plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)], col = "re
```



d.) Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c) ?

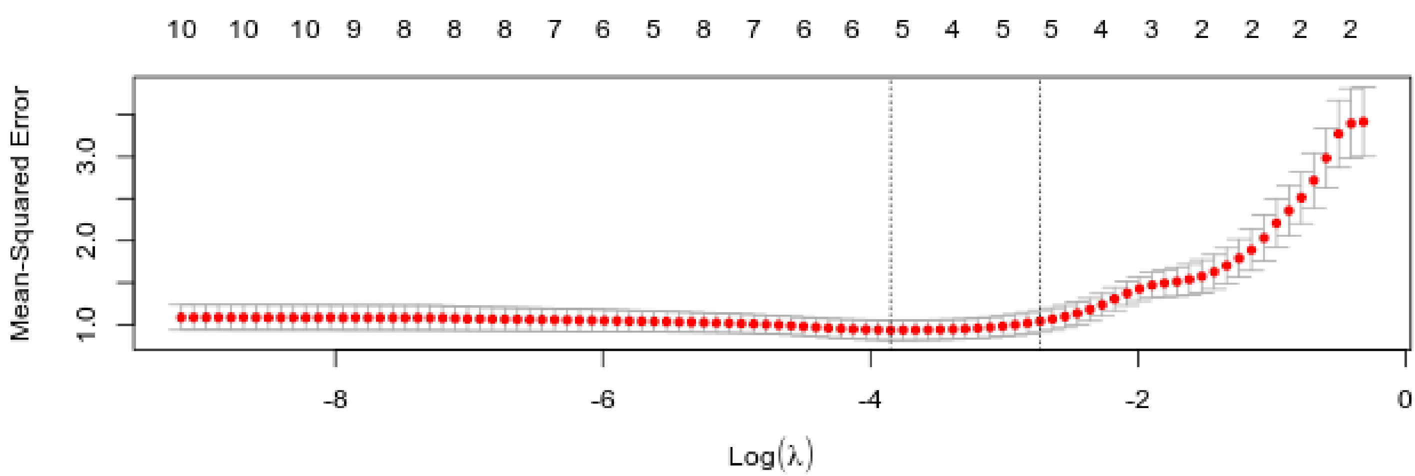
```
regfit.fwd <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) +
reg.summary.fwd <- summary(regfit.fwd)

par(mfrow = c(2, 2))
plot(reg.summary.fwd$cp, xlab = "Number of variables", ylab = "C_p", type = "l")
points(which.min(reg.summary.fwd$cp), reg.summary.fwd$cp[which.min(reg.summary.fwd$cp)], col =
plot(reg.summary.fwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(reg.summary.fwd$bic), reg.summary.fwd$bic[which.min(reg.summary.fwd$bic)], col
plot(reg.summary.fwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(reg.summary.fwd$adjr2), reg.summary.fwd$adjr2[which.max(reg.summary.fwd$adjr2)
```

```
mtext("Plots of C_p, BIC and adjusted R^2 for forward stepwise selection", side = 3, line = -2,
```



e.) Now fit a lasso model to the simulated data, again using $X, X^2, \ldots, x^{10}$

```
xmat <- model.matrix(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(

cv.lasso <- cv.glmnet(xmat, y, alpha = 1)

par(mfrow = c(1, 1))
plot(cv.lasso)
```




```
bestlam <- cv.lasso$lambda.min
bestlam
```

```
[1] 0.02129764
```

```
fit.lasso <- glmnet(xmat, y, alpha = 1)
predict(fit.lasso, s = bestlam, type = "coefficients")[1:11, ]
```

```
  (Intercept)   (Intercept)             x        I(x^2)        I(x^3) 
 2.1473403999  0.0000000000  2.8922592433  0.0162368913 -0.9623042225 
       I(x^4)        I(x^5)        I(x^6)        I(x^7)        I(x^8) 
 0.0000000000  0.0000000000  0.0036472725  0.0000000000  0.0007535099 
       I(x^9) 
 0.0000000000 
```

The lasso method picks X, X2, X3 and X5 as variables for the model.

9.)

In this exercise, we will predict the number of applications recieved using the other variables in the **College** data set.

a.) Split the data into a training set and a test set.

```
college <- as.data.table(ISLR::College)

unif <- runif(nrow(college))

train <- unif < .7
test <- !(train)

train <- college[train]
test <- college[test]
```

b.) Fit a linear model using least squares on the training set, and repor the test error obtained.

```
lm.fit <- lm(Apps ~ ., data = train)

mean((test$Apps - predict(lm.fit, newdata = test))^2)
```

```
[1] 1171752
```

c.) Fit a ridge regression on the training set, with λ chosen by cross-validation. Report the error.

```
train.mat <- model.matrix(Apps ~ ., data = train)

cv.ridge <- cv.glmnet(train.mat, train$Apps, data = train, alpha = 0)

bestlam <- cv.ridge$lambda.min
bestlam
```

```
[1] 381.8302
```

```r
test.mat <- model.matrix(Apps ~ ., data = test)

fit.ridge <- glmnet(test.mat, test$Apps, alpha = 1, lambda = bestlam)
pred <- predict(fit.ridge, s = bestlam, newx = test.mat, type = "response")

mean( (test$Apps - pred )^2 )
```

```
[1] 1455488
```

d.) Fit a lasso regresion on the training set, with λ chosen by cross-validation. Report the error.

```r
cv.lasso <- cv.glmnet(train.mat, train$Apps, data = train, alpha = 1)

bestlam <- cv.lasso$lambda.min
bestlam
```

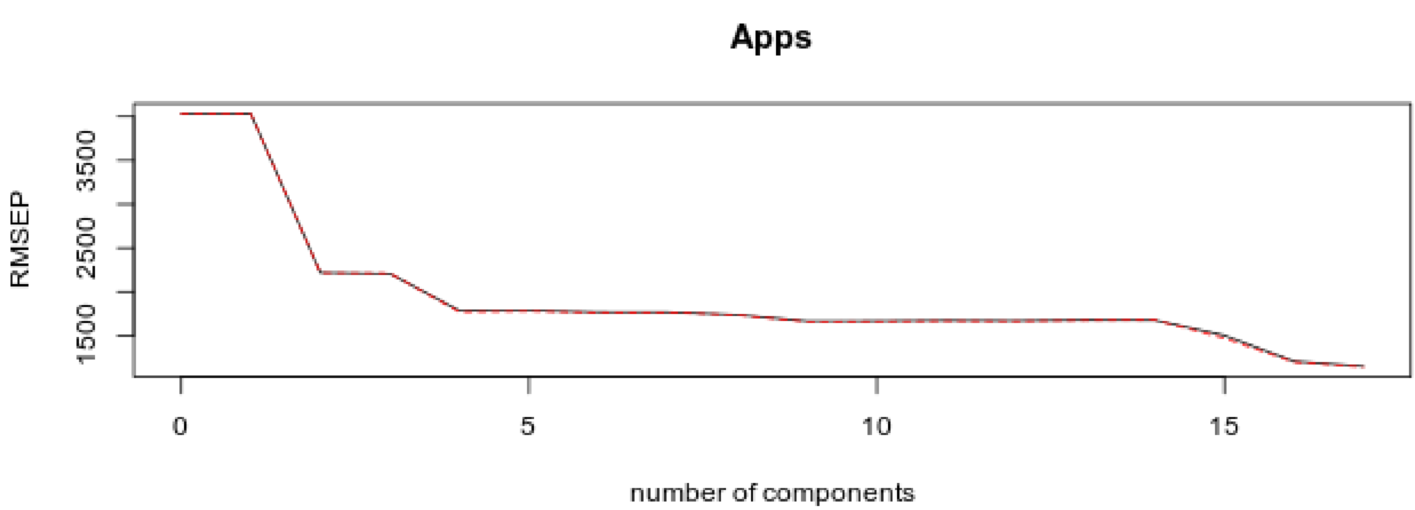
```
[1] 1.856688
```

```r
fit.lasso <- glmnet(test.mat, test$Apps, alpha = 1, lambda = bestlam)

pred <- predict(fit.lasso, test.mat, s = bestlam)

mean( (test$Apps - pred)^2)
```

```
[1] 945906
```

e.) Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error, along with the value of M.

```r
fit.pcr <- pcr(Apps ~ ., data = train, scale = T, validation = "CV")
validationplot(fit.pcr)
```



```r
fit.pcr$ncomp
```

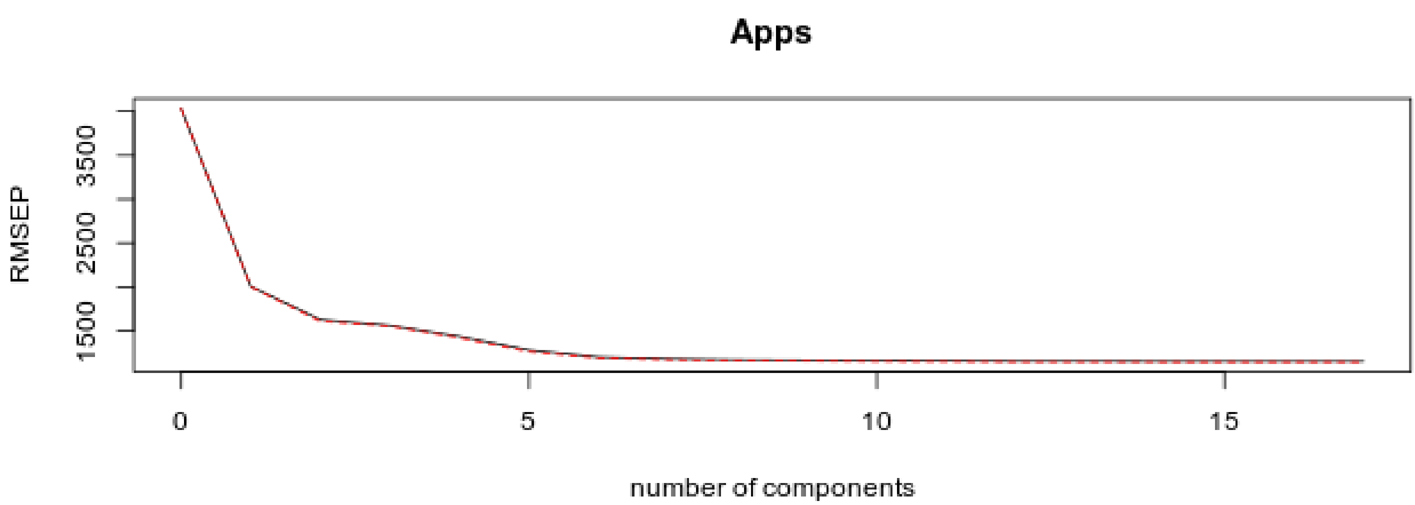
```
[1] 17
```

```r
pred.pcr <- predict(fit.pcr, test, ncomp = 10)
mean((pred.pcr - test$Apps)^2)
```

```
[1] 1347194
```

f.) Fit a PLS model on the training set, with M chosen by cross-validation. Report the test error and M.

```r
pls.fit <- plsr(Apps ~ ., data = train, scale = T,
                validation = "CV")

validationplot(pls.fit)
```




```r
pred.pls <- predict(pls.fit, test, ncomp = 10)
mean((pred.pls - test$Apps)^2)
```

```
[1] 1157333
```

10.)

We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

a.) Generate a data set with p = 20 features, n = 1,000 observations, and an associated quantitative response vector generated according to the model:

$Y = X\beta + \epsilon$

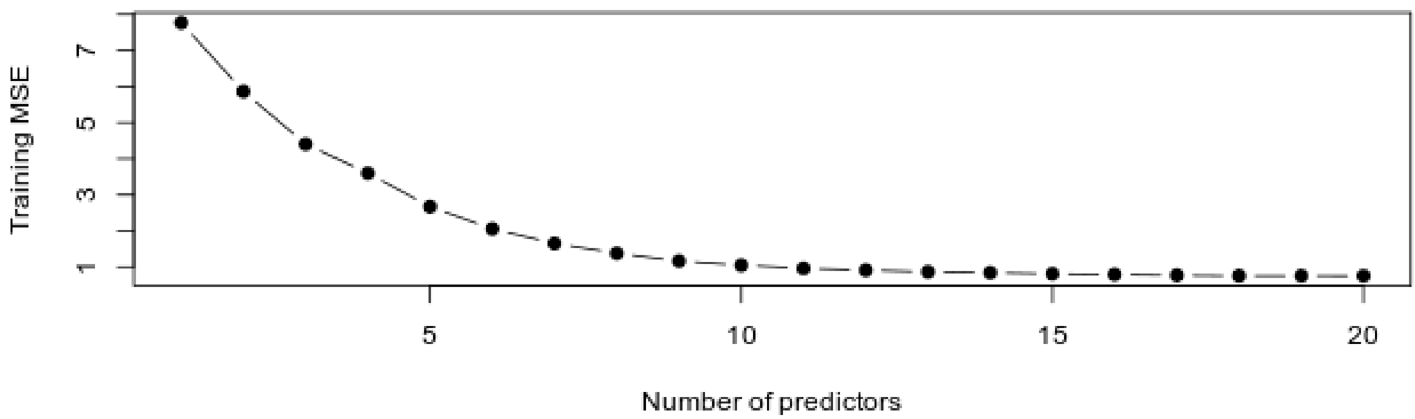
```
set.seed(1)
x <- matrix(rnorm(1000 * 20), 1000, 20)
b <- rnorm(20)
b[3] <- 0
b[4] <- 0
b[9] <- 0
b[19] <- 0
b[10] <- 0
eps <- rnorm(1000)
y <- x %*% b + eps
```

b.) Split your data set into a training set containing 100 observations and a test set containing 900 observations.

```
train <- sample(seq(1000), 100, replace = FALSE)
test <- -train
x.train <- x[train, ]
x.test <- x[test, ]
y.train <- y[train]
y.test <- y[test]
```

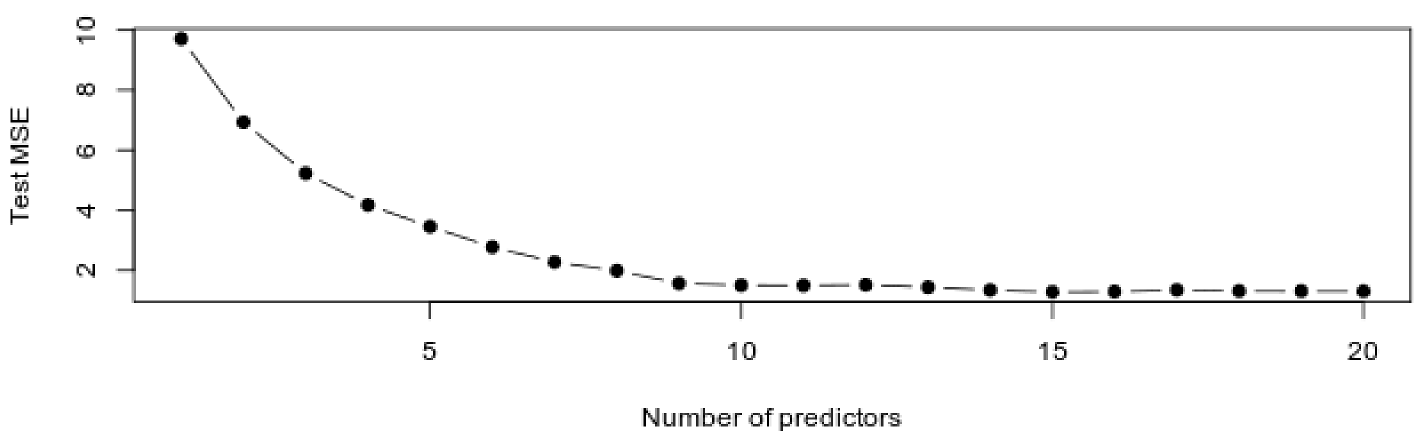
c.) Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

```
data.train <- data.frame(y = y.train, x = x.train)
regfit.full <- regsubsets(y ~ ., data = data.train, nvmax = 20)
train.mat <- model.matrix(y ~ ., data = data.train, nvmax = 20)
val.errors <- rep(NA, 20)
for (i in 1:20) {
    coefi <- coef(regfit.full, id = i)
    pred <- train.mat[, names(coefi)] %*% coefi
    val.errors[i] <- mean((pred - y.train)^2)
}
plot(val.errors, xlab = "Number of predictors", ylab = "Training MSE", pch = 19, type = "b")
```



d.) Plot the test MSE associated with the best model of each size.

```
data.test <- data.frame(y = y.test, x = x.test)
test.mat <- model.matrix(y ~ ., data = data.test, nvmax = 20)
val.errors <- rep(NA, 20)
for (i in 1:20) {
    coefi <- coef(regfit.full, id = i)
    pred <- test.mat[, names(coefi)] %*% coefi
    val.errors[i] <- mean((pred - y.test)^2)
}
plot(val.errors, xlab = "Number of predictors", ylab = "Test MSE", pch = 19, type = "b")
```



e.) For which model size does the test set MSE take on its minimum value ? Comment on your results. It it takes on its minimum value for a model containing only an intercept or a model containing all the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test MSE is minimized for an intermediate model size.

```
which.min(val.errors)
```

```
[1] 15
```

f.) How does the model at which the test set MSE is minimized compare to the true model used to generate the data ? Comment on the coefficient values.

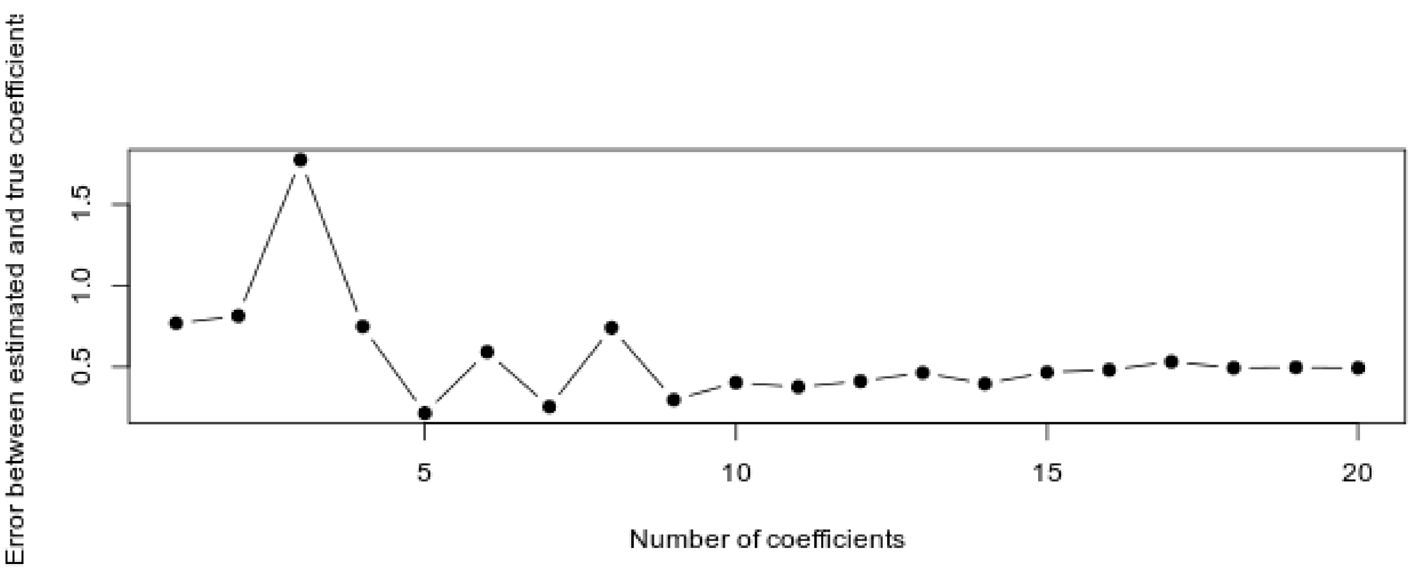
```
coef(regfit.full, which.min(val.errors))
```

```
 (Intercept)          x.2          x.4          x.5          x.6          x.7
-0.003933937  0.359127426  0.202707344  1.036265913 -0.253843053 -1.282753293
         x.8         x.11         x.12         x.13         x.14         x.15
 0.691581077  0.895769881  0.526887865 -0.207638251 -0.507929833 -0.892604795
        x.16         x.17         x.18         x.20
-0.343062241  0.184479252  1.646950451 -1.060191640
```

g.) Create a plot displaying:

$\sqrt{(\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2)}$

```
val.errors <- rep(NA, 20)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
for (i in 1:20) {
    coefi <- coef(regfit.full, id = i)
    val.errors[i] <- sqrt(sum((b[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_cols])^2
}
plot(val.errors, xlab = "Number of coefficients", ylab = "Error between estimated and true coef
```

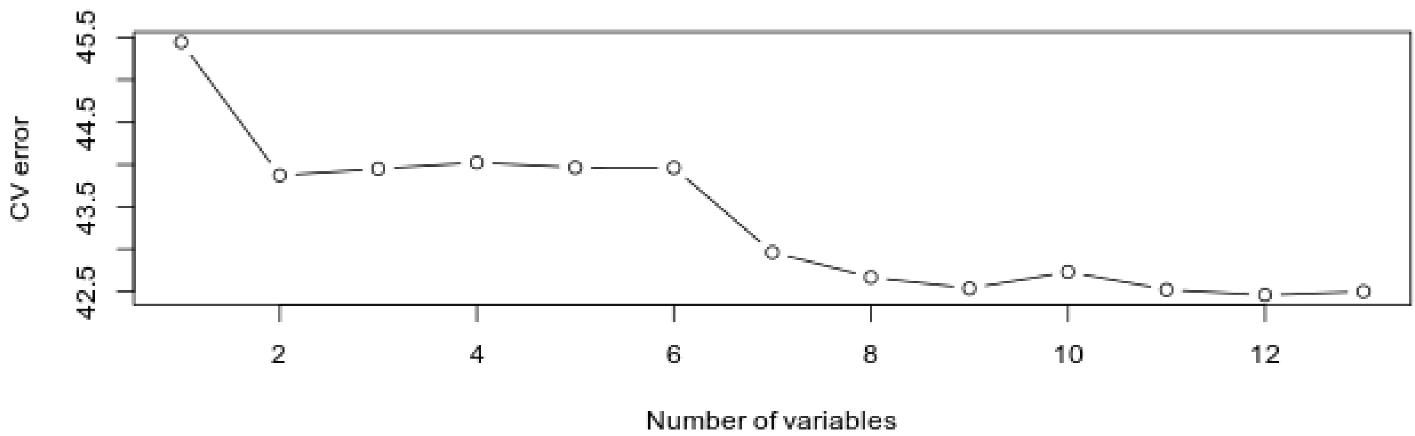


11.)

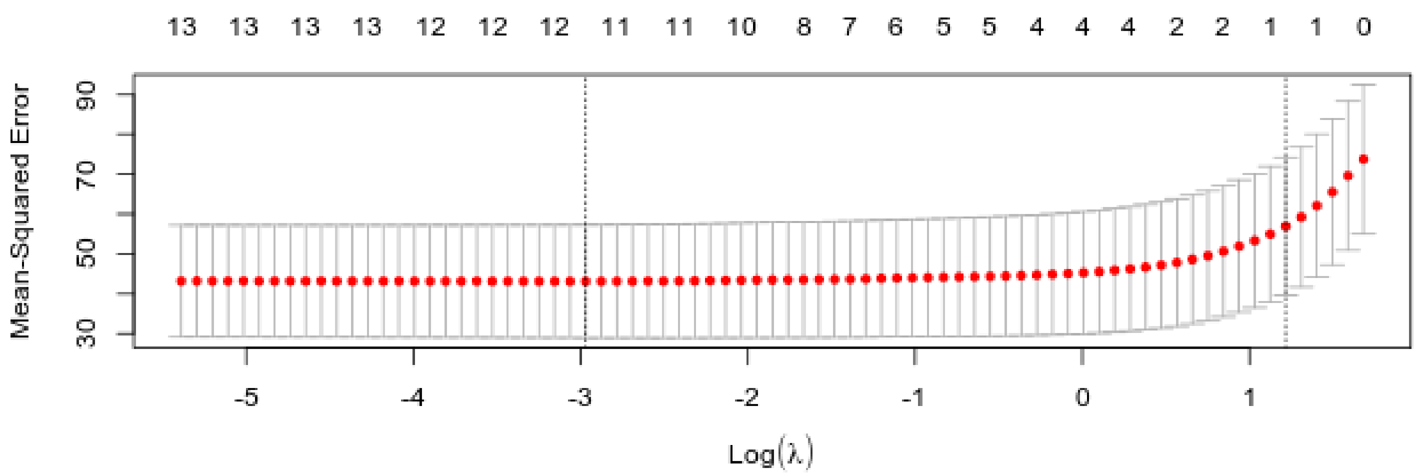
We will now try to predict per capita crime rate in the "Boston" data set.

a.) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression and PCR. Present and discuss results for the approaches that you consider.

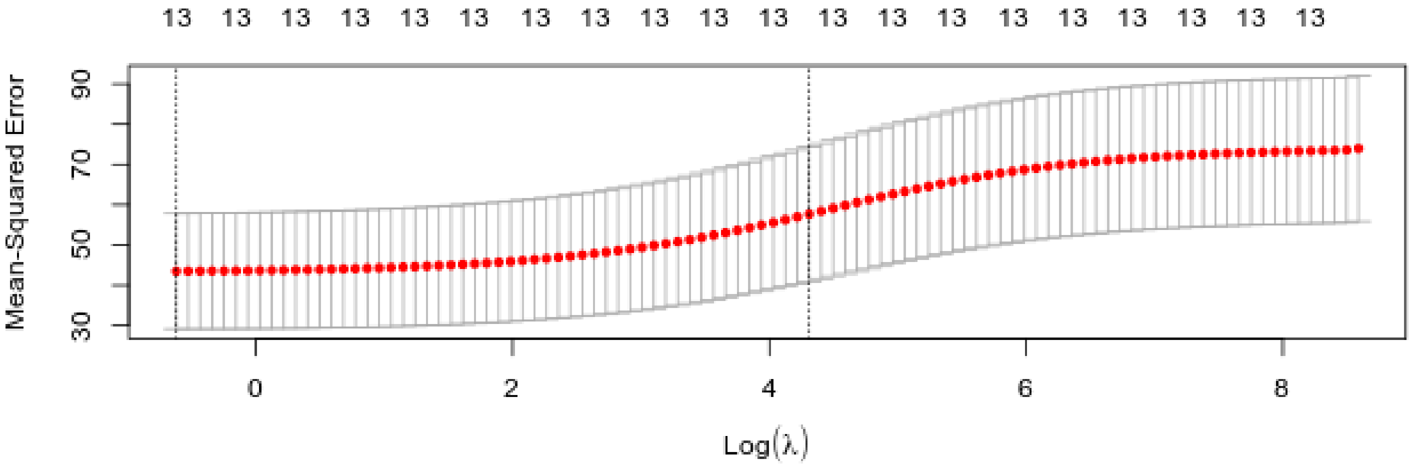
```
data(Boston)
set.seed(1)

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
}

k = 10
folds <- sample(1:k, nrow(Boston), replace = TRUE)
cv.errors <- matrix(NA, k, 13, dimnames = list(NULL, paste(1:13)))
for (j in 1:k) {
    best.fit <- regsubsets(crim ~ ., data = Boston[folds != j, ], nvmax = 13)
    for (i in 1:13) {
        pred <- predict(best.fit, Boston[folds == j, ], id = i)
        cv.errors[j, i] <- mean((Boston$crim[folds == j] - pred)^2)
    }
}

mean.cv.errors <- apply(cv.errors, 2, mean)
plot(mean.cv.errors, type = "b", xlab = "Number of variables", ylab = "CV error")
```




```r
x <- model.matrix(crim ~ ., Boston)[, -1]
y <- Boston$crim
cv.out <- cv.glmnet(x, y, alpha = 1, type.measure = "mse")
plot(cv.out)
```




```r
cv.out <- cv.glmnet(x, y, alpha = 0, type.measure = "mse")
plot(cv.out)
```


13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
Mean-Squared Error
30
50
70
90
0
2
4
6
8
Log(λ)