HW5

Ramtin Boustani - SUID# 05999261

Problem 1

Chapter 6, Exercise 1

(a)

Best subset selection has the samllest training RSS because of doing exhaustive search among all possible combinations

(b)

Mostly Best subset selection because of doing exhaustive search among all possible combinations but there is a chance other methods performing better.

(c)

i.

True

ii.

True

iii.

False, there is no relation between predictors chosen in backward and forward subset selection.

iv.

False, there is no relation between predictors chosen in backward and forward subset selection.

V.

False

Problem 2

Chapter 6, Exercise 3

(a)

(iv) steadily decrease.
By increasing s, coefficients are less restrictive and they have more variance and less bias and model will have less training RSS.

(b)

(ii) decrease intially and eventually U shape. General pattern, model become more and more flexible and up to some point and then overfitting issue causes increase test RSS.

(c)

(iii) steadily increase, the same behaviour mentioned above.

(d)

(iv) steadily decrease, by increasing s we have more and more variance and less and less bias.

(e)

(v) remain constant. irreducible eror is independent of s and betas.

Problem 3

Chapter 6, Exercise 8

(a)

```
set.seed(1)
X = rnorm(100)
eps = rnorm(100)
```

(b)

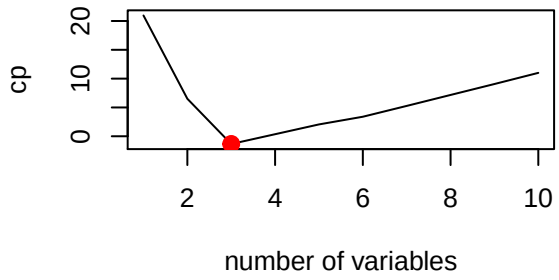
All non-zero coefficients equal to 0.5

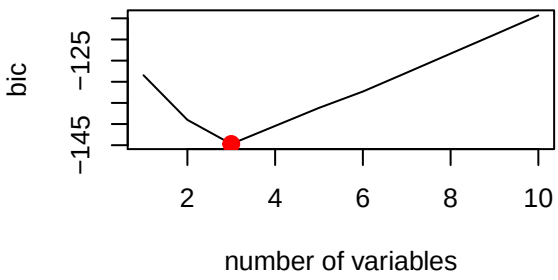
```
beta0 = 0.5
beta1 = 0.5
beta2 = 0.5
beta3 = 0.5
Y = beta0 + beta1 * X + beta2 * X^2 + beta3 * X^3 + eps
```

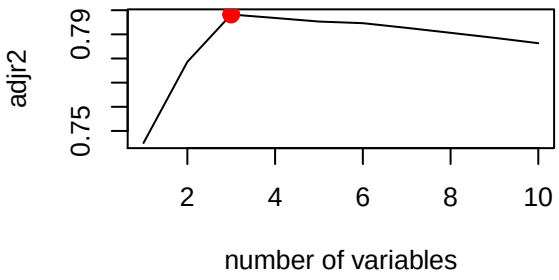
(c)

Best subset selection

```
library(leaps)
data = data.frame( Y=Y, poly(X, 10, raw = T))
regfit.data = data
regfit.full = regsubsets(Y ~ . , data = regfit.data, nvmax = 10, method = "exhaustive")
regfit.summary = summary(regfit.full)
par(mfrow = c(2,2))
plot(regfit.summary$cp, xlab = "number of variables", ylab = "cp", type="l")
points(which.min(regfit.summary$cp), regfit.summary$cp[which.min(regfit.summary$cp)], col="red", cex=2,
plot(regfit.summary$bic, xlab = "number of variables", ylab = "bic", type="l")
points(which.min(regfit.summary$bic), regfit.summary$bic[which.min(regfit.summary$bic)],col="red", cex=
plot(regfit.summary$adjr2, xlab="number of variables", ylab = "adjr2", type="l")
points(which.max(regfit.summary$adjr2), regfit.summary$adjr2[which.max(regfit.summary$adjr2)],col = "re
```








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

```
## [1] 3
```

```
coef(regfit.full, id= which.max(regfit.summary$adjr2))
```

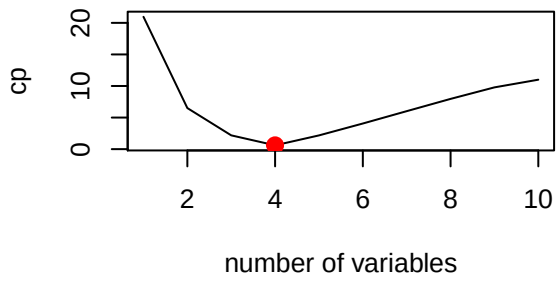
```
## (Intercept)          X1          X2          X5
##  0.57219472  0.94514720  0.34323764  0.09022577
```

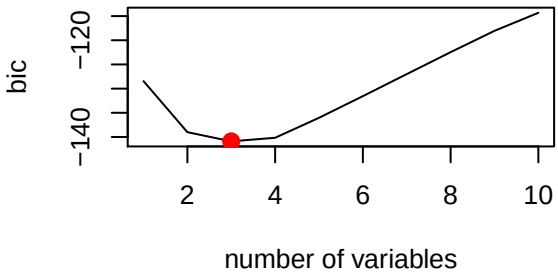
Insted of X3, X5 has chosen.
beta0 is a good estimate but coefficients of X1, X2 and X5 is not close 0.5!

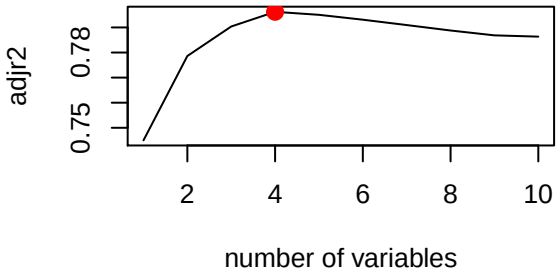
(d)

Forward subset selection

```
regfit.fwd = regsubsets(Y ~ . , data = regfit.data, nvmax = 10, method = "forward")
regfit.summary = summary(regfit.fwd)
par(mfrow = c(2,2))
plot(regfit.summary$cp, xlab = "number of variables", ylab = "cp", type="l")
points(which.min(regfit.summary$cp), regfit.summary$cp[which.min(regfit.summary$cp)], col="red", cex=2,
plot(regfit.summary$bic, xlab = "number of variables", ylab = "bic", type="l")
points(which.min(regfit.summary$bic), regfit.summary$bic[which.min(regfit.summary$bic)],col="red", cex=
plot(regfit.summary$adjr2, xlab="number of variables", ylab = "adjr2", type="l")
points(which.max(regfit.summary$adjr2), regfit.summary$adjr2[which.max(regfit.summary$adjr2)],col = "re
```








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

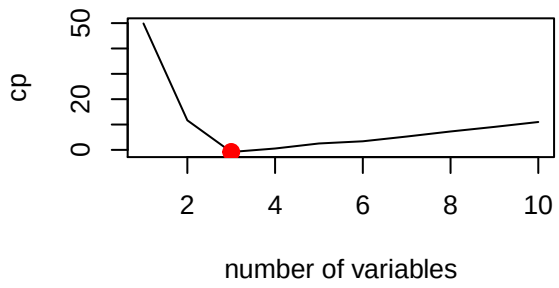
```
## [1] 4
```

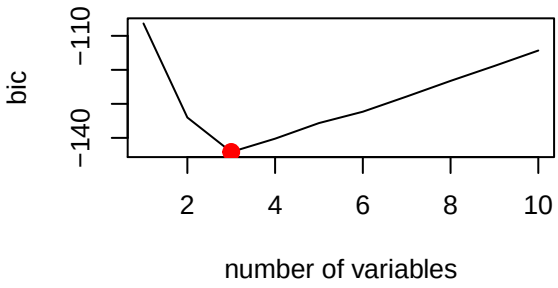
```
coef(regfit.fwd, id= which.max(regfit.summary$adjr2))
```

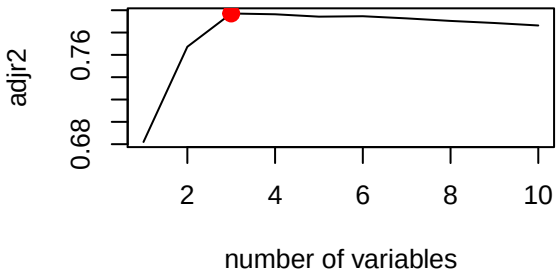
```
## (Intercept)          X1          X2          X3          X5
##  0.57200775  0.88745596  0.34575641  0.05797426  0.08072292
```

Backward subset selection

```
regfit.bwd = regsubsets(Y ~ . , data = regfit.data, nvmax = 10, method = "backward")
regfit.summary = summary(regfit.bwd)
par(mfrow = c(2,2))
plot(regfit.summary$cp, xlab = "number of variables", ylab = "cp", type="l")
points(which.min(regfit.summary$cp), regfit.summary$cp[which.min(regfit.summary$cp)], col="red", cex=2,
plot(regfit.summary$bic, xlab = "number of variables", ylab = "bic", type="l")
points(which.min(regfit.summary$bic), regfit.summary$bic[which.min(regfit.summary$bic)],col="red", cex=
plot(regfit.summary$adjr2, xlab="number of variables", ylab = "adjr2", type="l")
points(which.max(regfit.summary$adjr2), regfit.summary$adjr2[which.max(regfit.summary$adjr2)],col = "re
```








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

```
## [1] 3
```

```
coef(regfit.bwd, id= which.max(regfit.summary$adjr2))
```

```
## (Intercept)          X1          X4          X5
##  0.69884006  0.98672610  0.07758979  0.08472786
```

Best subset selection chosen 3 variables (Intercept=0.57, X1=0.94, X2=0.34, X5=0.09)
Forward subset selection chosen 4 variables (Intercept=0.57, X1= 0.88, X2=0.34, X3=0.05, X5=0.08)
Backward subset selection chosen 3 variables (Intercept=0.69, X1=0.98, X4=0.07, X5=0.08)

(e)

```
library(glmnet)
```

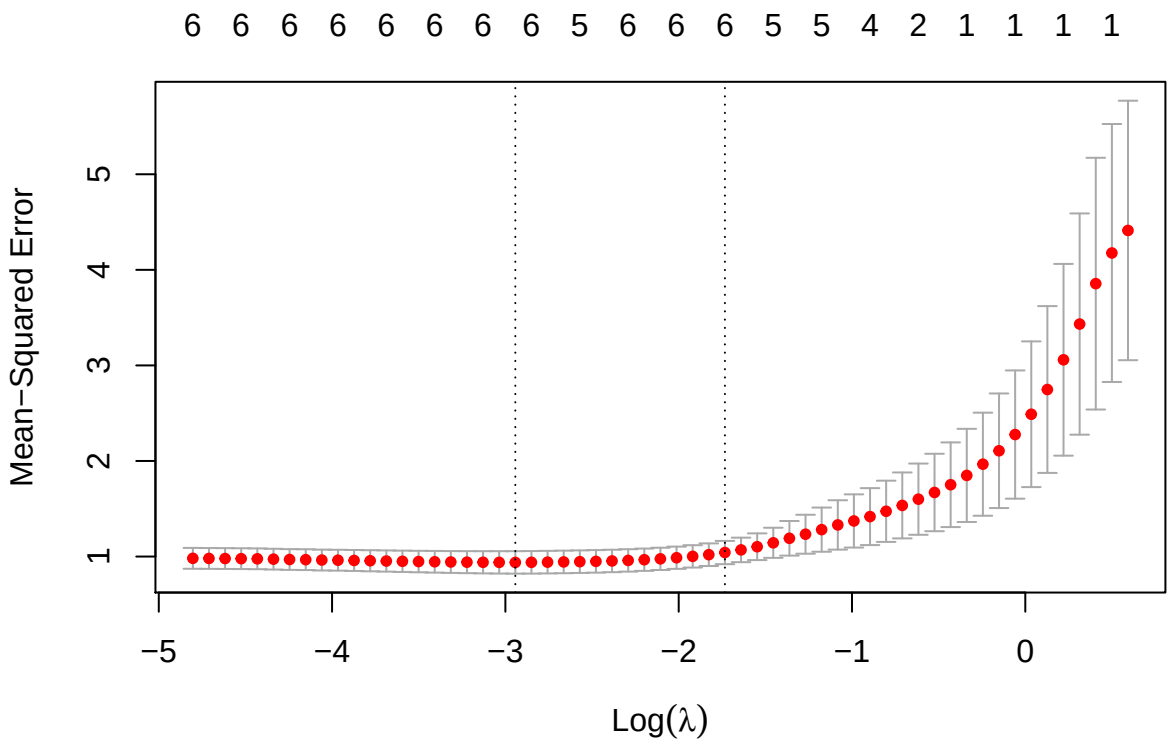
```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0
```

```
xmat = model.matrix(Y ~ poly(X, 10, raw = T), data = data)[,-1]
mod.lasso = cv.glmnet(xmat, Y, alpha=1)
best.lamda = mod.lasso$lambda.min
best.lamda
```

```
## [1] 0.05272586
```

```
plot(mod.lasso)
```



Based on the best lamda predicating using lasso

```
fit.lasso = glmnet(xmat, Y, alpha=1)
predict(fit.lasso, s=best.lamda, type = "coefficients" )
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                      1
## (Intercept)             0.6620354260
## poly(X, 10, raw = T)1   0.7553408606
## poly(X, 10, raw = T)2   0.1503593044
## poly(X, 10, raw = T)3   0.1743999059
## poly(X, 10, raw = T)4   0.0391080323
## poly(X, 10, raw = T)5   0.0567537978
## poly(X, 10, raw = T)6   0.0002078909
## poly(X, 10, raw = T)7   .
## poly(X, 10, raw = T)8   .
## poly(X, 10, raw = T)9   .
## poly(X, 10, raw = T)10  .
```

lasso picks 6 variables which X1 has the most weight and X4, X5 and X6 have small wight

(f)

All non-zero coefficients equal to 0.5

Lasso

```
beta7= 0.5
Y = beta0 + beta7 * X^7 + eps
data = data.frame( Y=Y, poly(X, 10, raw = T))
xmat = model.matrix(Y ~ poly(X, 10, raw = T), data = data)[,-1]
```

```
mod.lasso = cv.glmnet(xmat, Y ,alpha=1)
best.lamda = mod.lasso$lambda.min
fit.lasso = glmnet(xmat, Y, alpha = 1)
predict(fit.lasso, s=best.lamda, type = "coefficients" )
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)           0.4714183710
## poly(X, 10, raw = T)1  .
## poly(X, 10, raw = T)2  .
## poly(X, 10, raw = T)3  .
## poly(X, 10, raw = T)4  .
## poly(X, 10, raw = T)5  .
## poly(X, 10, raw = T)6  .
## poly(X, 10, raw = T)7  0.4936609079
## poly(X, 10, raw = T)8  .
## poly(X, 10, raw = T)9  0.0007375787
## poly(X, 10, raw = T)10 .
```

best subset selection

```
regfit.full = regsubsets( Y~. , data=data, method = "exhaustive", nvmax = 10)
regfit.summary = summary(regfit.full)
coef(regfit.full, id = which.min(regfit.summary$cp) )
```

```
## (Intercept)          X2          X7
##   0.5704904  -0.1417084   0.5015552
```

Both Lasso and Best subset pick one wrong extra predictor (lasso X9 & best subset X2)
Weight of estimation for beta7 and intercept in best subset is better than lasso

Problem 4

Chapter 6, Exercise 9

(a)

```
library(ISLR)
set.seed(1)
n = dim(College)[1]
train.size = n/2
train = sample( 1:n, size = train.size)
College.train = College[train, ]
College.test = College[-train, ]
```

(b)

```
lm.fit = lm(Apps ~ ., data=College.train)
lm.predict = predict(lm.fit, College.test)
lm.test = mean((College.test[,"Apps"]-lm.predict)^2)
lm.test
```

```
## [1] 1135758
```

(c)

```
train.xmat = model.matrix(Apps~. , data=College.train )
test.xmat = model.matrix(Apps~. , data=College.test )
train.y = College.train[,"Apps"]
test.y = College.test[,"Apps"]
mod.ridge = cv.glmnet(x = train.xmat, y = train.y, alpha=0)
best.lamda = mod.ridge$lambda.min
pred.ridge = predict(mod.ridge, s=best.lamda, newx = test.xmat)
ridge.test = mean((test.y - pred.ridge)^2)
ridge.test
```

```
## [1] 976261.5
```

Test error for ridge regression is higher than linear regression

(d)

```
mod.lasso = cv.glmnet(x = train.xmat, y = train.y, alpha=1)
best.lamda = mod.lasso$lambda.min
pred.lasso = predict(mod.lasso, s=best.lamda, newx = test.xmat)
lasso.test = mean((test.y - pred.lasso)^2)
lasso.test
```

```
## [1] 1115901
```

Test error for lasso is higher than ridge regression

```
mod.lasso = glmnet( x= model.matrix( Apps~. , data =College), y=College[,"Apps"], alpha = 1)
predict(mod.lasso, s=best.lamda , type="coefficients")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -471.39372069
## (Intercept)    .
## PrivateYes  -491.04485135
## Accept         1.57033288
## Enroll        -0.75961467
## Top10perc     48.14698891
## Top25perc    -12.84690694
## F.Undergrad    0.04149116
## P.Undergrad    0.04438973
## Outstate      -0.08328388
## Room.Board     0.14943472
## Books          0.01532293
## Personal       0.02909954
## PhD           -8.39597537
## Terminal      -3.26800340
## S.F.Ratio     14.59298267
## perc.alumni   -0.04404771
## Expend         0.07712632
## Grad.Rate      8.28950241
```

PrivateYes, Accept, Top10perc, Top25perc, PhD, Terminal, S.F.Ratio, Grad.Rate have high weight
F.Undergrad, P.Undergrad, Books, Personal, perc.alumni have very low weight

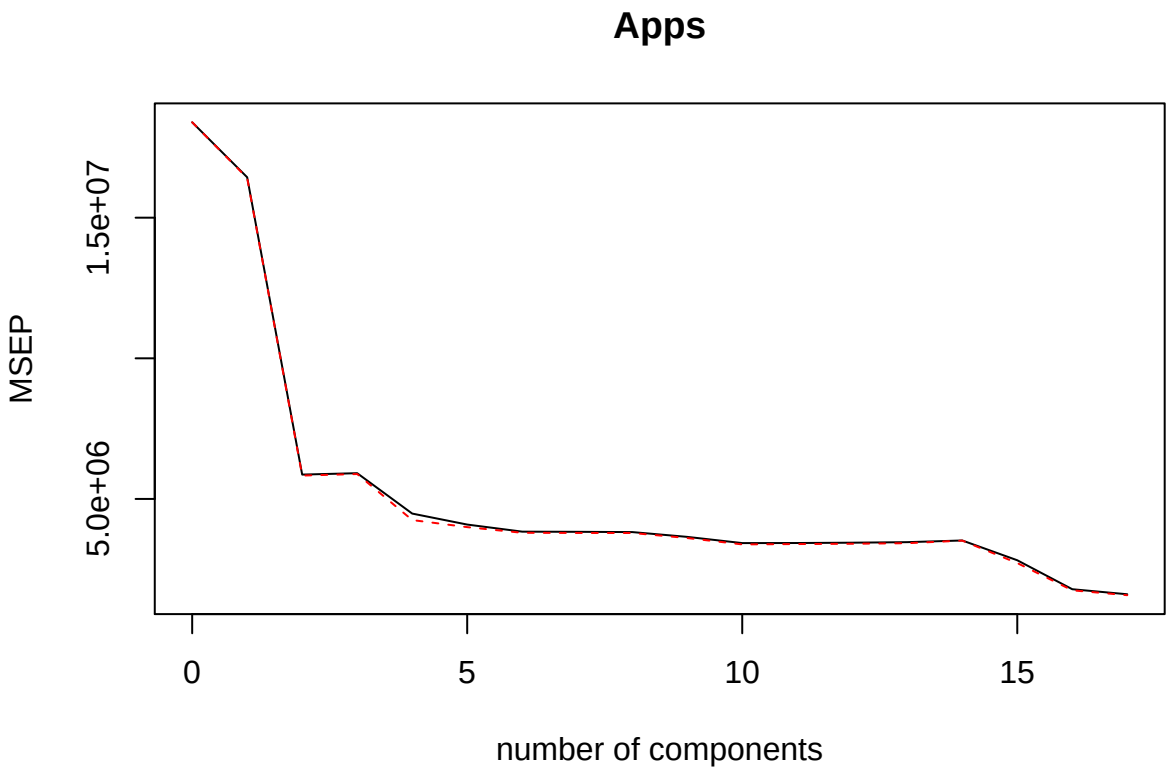
(e)

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
fit.pcr = pcr( Apps~., data= College.train, scale=TRUE, validation="CV" )
validationplot(fit.pcr, val.type="MSEP")
```




```
pcr.pred = predict(fit.pcr, College.test, ncomp=10)
pcr.test = mean((test.y - pcr.pred)^2)
pcr.test
```

```
## [1] 1723100
```

Test error for PCR is higger than lasso

(f)

```
pls.fit <- plsr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
pls.pred <- predict(pls.fit, College.test, ncomp = 10)
pls.test = mean((pls.pred - test.y)^2)
pls.test
```

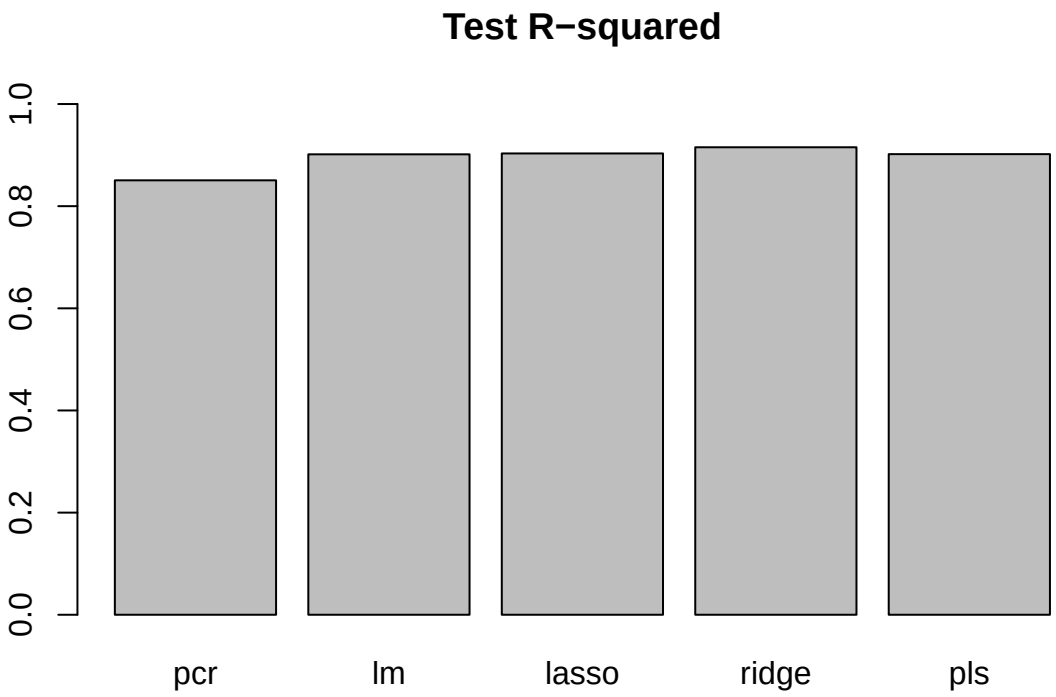
```
## [1] 1131661
```

Test error is lower than PCR

(g)

R-square

```
test.avg <- mean(College.test$Apps)
tss =  mean((test.y-test.avg)^2)
pcr.r2 = 1 - pcr.test/tss
lm.r2 = 1 - lm.test/tss
lasso.r2 = 1 - lasso.test/tss
ridge.r2 = 1 - ridge.test/tss
pls.r2 = 1 - pls.test/tss
barplot(c(pcr.r2, lm.r2, lasso.r2, ridge.r2, pls.r2), names.arg = c("pcr", "lm", "lasso", "ridge", "pls
```



pcr < lm < lasso < ridge < pls

Problem 5

Chapter 6, Exercise 11

(a)

```
library(MASS)
set.seed(1)
```

best subset selection

```
#implement predict for best subset
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
}
```

```r
#using 10 fold corss validation
k=10
p = ncol(Boston) - 1
folds = sample(rep(1:k, length = nrow(Boston)))
cv.errors = matrix(NA, k, p)
for (i in 1:k) {
    best.fit = regsubsets(crim ~ ., data = Boston[folds != i, ], nvmax = p)
    for (j in 1:p) {
        pred = predict(best.fit, Boston[folds == i, ], id = j)
        cv.errors[i, j] = mean((Boston$crim[folds == i] - pred)^2)
    }
}
mse.cv = apply(cv.errors, 2, mean)
which.min(mse.cv)
```

```
## [1] 9
```

```r
mse.cv[which.min(mse.cv)]
```

```
## [1] 42.81453
```

Lasso

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

```
##
## Call:  cv.glmnet(x = x, y = y, type.measure = "mse", alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Measure    SE Nonzero
## min  0.207   44.84 18.13       7
## 1se  4.066   62.75 23.76       1
```

Ridge

```r
x = model.matrix( crim ~ ., data = Boston)[,-1]
y = Boston[,1]
cv.ridge = cv.glmnet(x, y, alpha=0, type.measure = "mse")
cv.ridge
```

```
##
## Call:  cv.glmnet(x = x, y = y, type.measure = "mse", alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Measure    SE Nonzero
## min   0.54   43.33 15.55      13
## 1se  81.70   58.82 19.92      13
```

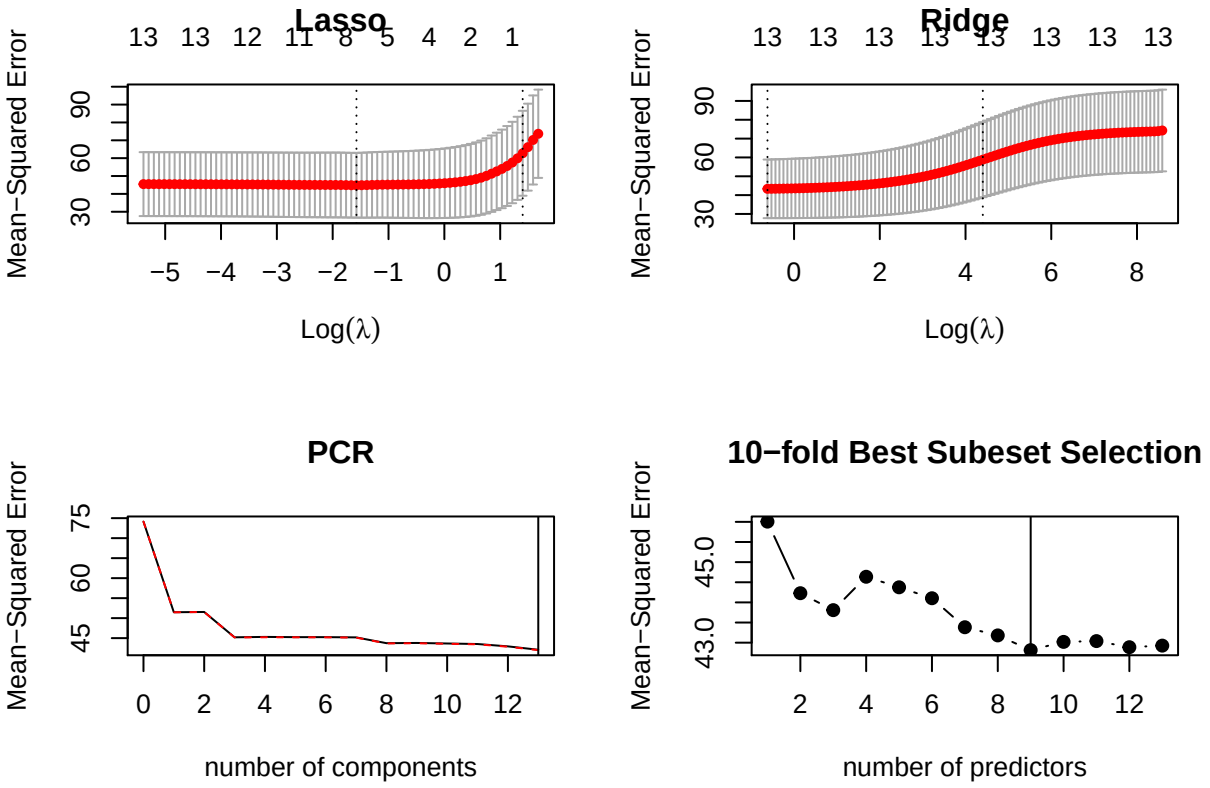
PCR

```r
fit.pcr = pcr( crim ~., data=Boston, scale=TRUE, validation="CV")
#summary(fit.pcr)
MSEP(fit.pcr)
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           74.13    51.48    51.55    45.21    45.31    45.25    45.25
## adjCV        74.13    51.46    51.54    45.17    45.22    45.22    45.21
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       45.19    43.74    43.79     43.66     43.53     42.94      42.1
## adjCV    45.13    43.68    43.73     43.58     43.46     42.85      42.0
```

Lasso (42.52) vs Ridge (45.42) vs PCR(43.09) vs Best subest selection (42.92)

```r
par(mfrow = c(2,2))
plot(cv.lasso , main="Lasso" )
plot(cv.ridge, main = "Ridge")
validationplot(fit.pcr, val.type="MSEP" ,ylab = "Mean-Squared Error", main="PCR")
abline(v=13)
plot(mse.cv, pch = 19, type = "b",  ylab = "Mean-Squared Error", xlab = "number of predictors", main="10
abline(v=which.min(mse.cv))
```



(b)

All 4 models have close test MSE but best subset selection model with CV 10 folds has the best performance

(c)

No, from 13 features
Best subset uses 12
Lasso uses 11

Ridge uses 13
PCR uses 13 components