Ch.6 Exercises: Linear Model Selection and Regularization

```
library(ISLR)
library(MASS)
library(leaps)
library(glmnet)
library(pls)
library(caTools)
```

Conceptual

1. (a)

- Best subset selection as the lowest training RSS as it fits models for every possible combination of predictors. When p is very large, this increases the chance of finding models that fit the training data very well.

(b)

- Best test RSS could be provided by any of the models. Bet subset considers more models than the other two, and the best model found on the training set could also be the best one for a test set. Forward and backward stepwise consider a lot fewer models, but might find a model that fits the test set very well as it tends to avoid over fitting when compared to best subset.

(c)

- True; the k+1 model is derived by adding a predictor that gives greatest improvement to previous k model.
- True; the k model is derived by removing the least useful predictor from the k+1 model.
- False; forward and backward stepwise can select different predictors.
- False; forward and backward stepwise can select different predictors.

2. (a)

(iii) Less flexible and will give improved prediction accuracy when its increase in bias is less than its decrease in variance. As lambda increases, flexibility of fit decreases, and so the estimated coefficients decrease with some being zero. This leads to a substantial decrease in the variance of the predictions for a small increase in bias.

(b)

(iii) same as (a), except every variable has a non-zero coefficient.

(c)

(ii) Non-linear models will generally be more flexible, and so predictions tend to have a higher variance and lower bias. So predictions will improve if the variance rises less than a decrease in the bias (bias-variance trade off).

3. (a)

- (i) Decrease steadily. As s increases the constraint on beta decreases and the RSS reduces until we reach the least squares answer.

(b) - (ii) Decreases initially as RSS in reduced from the maximum value (when B=0) as we move towards the best training value for B1. Eventually starts increasing as the B values start fitting the training set extremely well, and so over fitting the test set.

(c)

- (iii) Steadily increase.

(d)

- (iv) Steadily decrease.

(e)

- (v) Remains constant.

4. (a)

- (iii) Steadily increase. As lambda increases the constraint on beta also increases, this means beta becomes progressively smaller. As such the training RSS will steadily increase to the maximum value (when B $\sim$ 0 at very large lambda values.)

(b)

- (ii) Decreases initially and then starts increasing in a U shape.When lambda=0, we get a least squares fit that has high variance and low bias. As lambda increases, the flexibility of the fit decreases, so reducing variance of predictions for a small increase in bias. This results in more accurate predictions and so the test RSS will decrease initially. As lambda increases beyond the ideal point, we start seeing a much greater increase in bias than the reduction in variance, and so predictions will become more biased. Consequently, we will see a rise in the test RSS.

(c)

- (iv) Steadily decrease.

(d)

- (iii) Steadily increase.

(e)

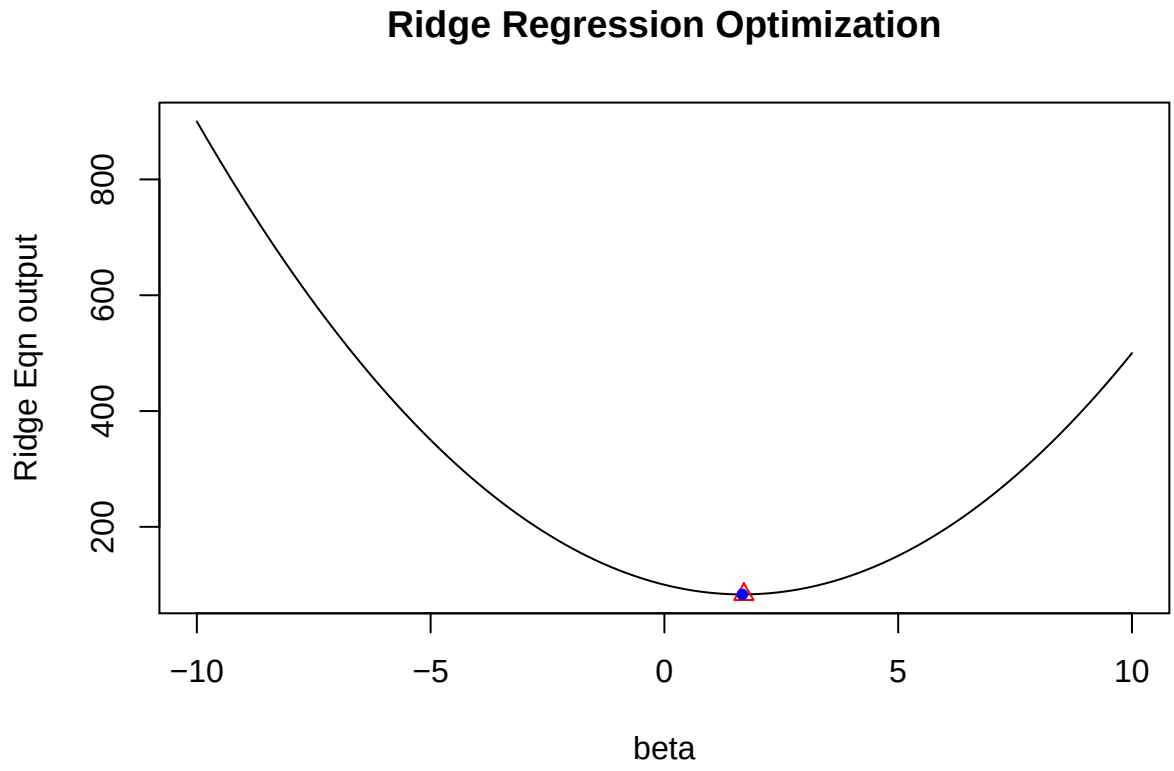
- (v) Remains constant.

6. (a)

```r
# Chart of Ridge Regression Eqn(6.12) against beta.
y = 10
beta = seq(-10,10,0.1)
lambda = 5

# Eqn (6.12) output
eqn1 = (y - beta)^2 + lambda*(beta^2)
which.min(eqn1) #index at minimum, similar to using (6.14).
```

```
## [1] 118
```

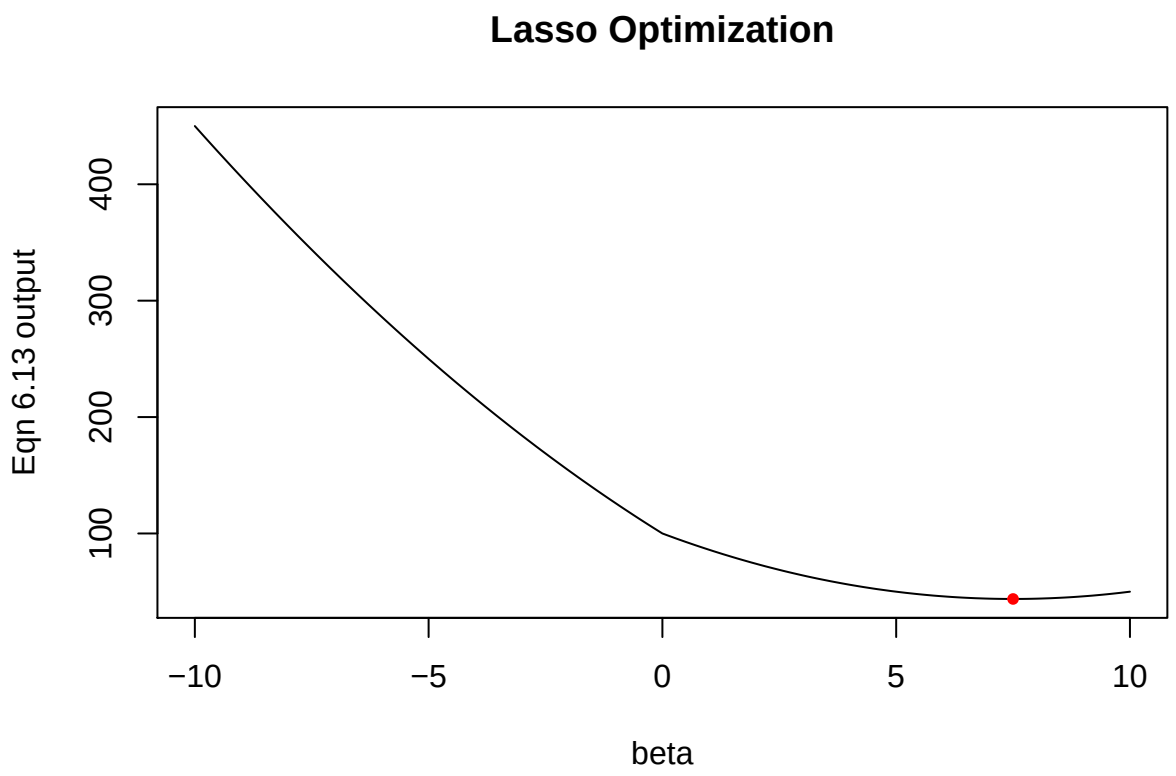
```r
# Minimum beta estimated using (6.14).
est.beta = y/(1 + lambda)
est.value = (y - est.beta)^2 + lambda*(est.beta^2)

# Plot
plot(beta, eqn1, main="Ridge Regression Optimization", xlab="beta", ylab="Ridge Eqn output",type="l")
points(beta[118],eqn1[118],col="red", pch=24,type = "p")
points(est.beta, est.value,col="blue",pch=20,type ="p")
```



- For y=10 and lambda=5, beta=10/(1 + 5)= 1.67 minimizes the ridge regression equation.
- As can also be seen on the graph the minimum beta is at 1.67.

(b)

```r
# Chart of Lasso Eqn (6.13) against beta.
# Eqn (6.13) output
eqn2 = (y - beta)^2 + lambda*(abs(beta))

# Minimum beta estimated using (6.15).
est.beta2 = y - lambda/2
est.value2 = (y - est.beta2)^2 + lambda*(abs(est.beta2))

# Plot
plot(beta, eqn2, main="Lasso Optimization", xlab="beta", ylab="Eqn 6.13 output",type="l")
points(est.beta2, est.value2,col="red",pch=20,type ="p")
```



- For y=10 and lambda=5, beta=10-(5/2)= 7.5 minimizes the lasso equation.
- As can also be seen on the graph the minimum beta is at 7.5.

Applied

(8) (a) (b)

```r
set.seed(1)
X = c(rnorm(100))
e = rnorm(100, mean=0, sd=0.25)
b0 = 50 ; b1 = 6 ; b2 = 3 ; b3 = 1.5
```

```
Y = c(b0 + b1*X + b2*X^2 + b3*X^3 + e)

# Dataframe with Y (response and X,X^1..X^10 variables)
df = data.frame(X,X^2,X^3,X^4,X^5,X^6,X^7,X^8,X^9,X^10,Y)
```

(c)

```
# Best subset selection.
regfit.full = regsubsets(Y~.,data=df,nvmax=10)
reg.summary = summary(regfit.full)
reg.summary$cp
```

```
##  [1] 3.129646e+04 8.595767e+03 2.185943e+00 6.067483e-01 2.178201e+00
##  [6] 3.995581e+00 5.786906e+00 7.169409e+00 9.153558e+00 1.100000e+01
```

```
reg.summary$bic
```

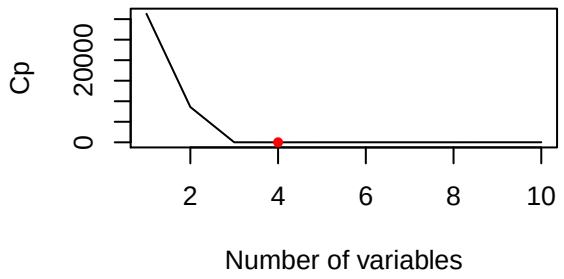
```
##  [1] -160.3392 -284.1762 -732.0341 -731.3031 -727.1721 -722.7696 -718.3966
##  [8] -714.4815 -709.8941 -705.4614
```

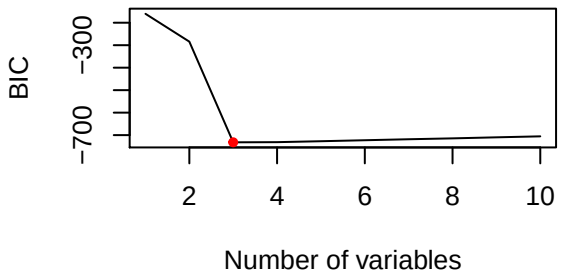
```
reg.summary$adjr2
```

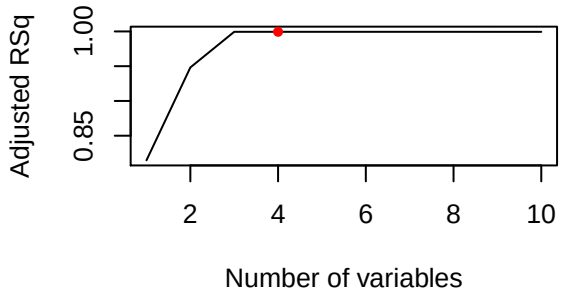
```
##  [1] 0.8146192 0.9481556 0.9994322 0.9994480 0.9994448 0.9994400 0.9994352
##  [8] 0.9994329 0.9994267 0.9994213
```

- Cp reduces substantially from the one and two variable model to the three variable model. It reduces slightly in the four variable model and rises in small increments thereafter.
- BIC value is lowest for three variable model.
- Adjusted R2 increases to 0.999 in the three variable model from the two model value of 0.95. The metric does not change much in the higher variable models.
- These statistical metrics point to the three variable model (with the squared and cubed terms) as being the best choice. We can confirm this visually in the charts below.

```
# Plot of Cp, bic and adjr2 metrics.
par(mfrow=c(2,2))
plot(reg.summary$cp,xlab="Number of variables", ylab="Cp",type="l")
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)],
       col = "red", pch = 20)

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", pch = 20)

plot(reg.summary$adjr2,xlab="Number of variables", ylab="Adjusted RSq",type="l")
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)],
       col = "red", pch = 20)
```







(d)

```
# Forward stepwise selection.
regfit.fwd = regsubsets(Y~.,data=df,nvmax=10,method='forward')
regfwd.summary = summary(regfit.fwd)
regfwd.summary
```

```
## Subset selection object
## Call: regsubsets.formula(Y ~ ., data = df, nvmax = 10, method = "forward")
## 10 Variables  (and intercept)
##      Forced in Forced out
## X        FALSE      FALSE
## X.2      FALSE      FALSE
## X.3      FALSE      FALSE
## X.4      FALSE      FALSE
## X.5      FALSE      FALSE
## X.6      FALSE      FALSE
## X.7      FALSE      FALSE
## X.8      FALSE      FALSE
## X.9      FALSE      FALSE
## X.10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: forward
##           X   X.2 X.3 X.4 X.5 X.6 X.7 X.8 X.9 X.10
## 1  ( 1 )  " " " " "*" " " " " " " " " " " " " " "
## 2  ( 1 )  " " "*" "*" " " " " " " " " " " " " " "
```

```
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " " " " " " " "
## 4  ( 1 )  "*" "*" "*" " " "*" " " " " " " " " " " " " " " "
## 5  ( 1 )  "*" "*" "*" " " "*" "*" " " " " " " " " " " " " "
## 6  ( 1 )  "*" "*" "*" " " "*" "*" " " " " " " "*" " " " "
## 7  ( 1 )  "*" "*" "*" " " "*" "*" "*" " " " " "*" " " " "
## 8  ( 1 )  "*" "*" "*" " " "*" "*" "*" "*" "*" " " " "
## 9  ( 1 )  "*" "*" "*" " " "*" "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
regfwd.summary$cp
```

```
##  [1] 3.129646e+04 1.607895e+04 2.185943e+00 6.067483e-01 2.178201e+00
##  [6] 4.062107e+00 6.016511e+00 7.958257e+00 9.784279e+00 1.100000e+01
```

```
regfwd.summary$bic
```

```
##  [1] -160.3392 -222.0568 -732.0341 -731.3031 -727.1721 -722.6957 -718.1412
##  [8] -713.6008 -709.1892 -705.4614
```

```
regfwd.summary$adjr2
```

```
##  [1] 0.8146192 0.9035099 0.9994322 0.9994480 0.9994448 0.9994396 0.9994338
##  [8] 0.9994279 0.9994227 0.9994213
```

- The statistical metrics are very similar to that for best subset selection.

```
# Backward stepwise selection.
regfit.bwd = regsubsets(Y~.,data=df,nvmax=10,method='backward')
regbwd.summary = summary(regfit.bwd)
regbwd.summary
```

```
## Subset selection object
## Call: regsubsets.formula(Y ~ ., data = df, nvmax = 10, method = "backward")
## 10 Variables  (and intercept)
##      Forced in Forced out
## X        FALSE      FALSE
## X.2      FALSE      FALSE
## X.3      FALSE      FALSE
## X.4      FALSE      FALSE
## X.5      FALSE      FALSE
## X.6      FALSE      FALSE
## X.7      FALSE      FALSE
## X.8      FALSE      FALSE
## X.9      FALSE      FALSE
## X.10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: backward
##           X   X.2 X.3 X.4 X.5 X.6 X.7 X.8 X.9 X.10
## 1  ( 1 )  "*" " " " " " " " " " " " " " " " " " " "
## 2  ( 1 )  "*" "*" " " " " " " " " " " " " " " " " "
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " " "
```

```
## 4  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "*" " "
## 5  ( 1 )  "*" "*" "*" " " " " " " " " " " " "*" "*" " "
## 6  ( 1 )  "*" "*" "*" " " " " " " " " " " " "*" "*" "*"
## 7  ( 1 )  "*" "*" "*" " " " " " " "*" " " " "*" "*" "*"
## 8  ( 1 )  "*" "*" "*" "*" " " " "*" " " " "*" "*" "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" " " " "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
regbwd.summary$cp
```

```
##  [1] 3.337929e+04 8.595767e+03 2.185943e+00 9.808795e-01 2.731015e+00
##  [6] 4.418100e+00 5.925033e+00 7.169409e+00 9.153558e+00 1.100000e+01
```

```
regbwd.summary$bic
```

```
##  [1] -153.9152 -284.1762 -732.0341 -730.8911 -726.5609 -722.3012 -718.2429
##  [8] -714.4815 -709.8941 -705.4614
```

```
regbwd.summary$adjr2
```

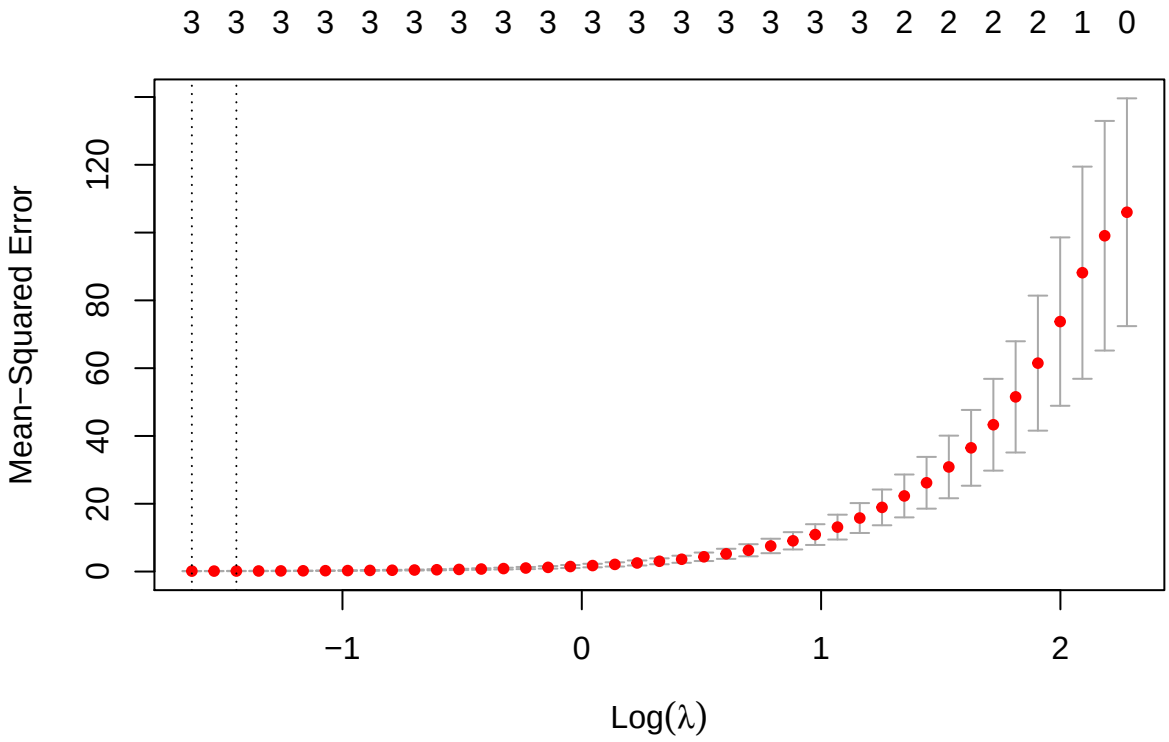
```
##  [1] 0.8023195 0.9481556 0.9994322 0.9994458 0.9994414 0.9994374 0.9994343
##  [8] 0.9994329 0.9994267 0.9994213
```

- Results are very similar to best subset and forward selection. As expected, all these metrics show that the three variable model with the squared and cubed term is the best.

(e)

```
# Matrix of predictors for a response vector y.
x = model.matrix(Y~.,df)[,-1]
y = df$Y

# Training and test sets.
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
y.test = y[test]

# Lasso model.
lasso = glmnet(x[train,], y[train], alpha=1)
```

```
cv.out = cv.glmnet(x[train,],y[train], alpha=1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
```

- Higher values of lambda result in an increase in the MSE.
- Best value of lambda is 0.2.

```
# Coefficients from lasso model with best lambda.
out = glmnet(x,y,alpha=1)
lasso.coef = predict(out, type="coefficients",s=0.20)[1:11,]
lasso.coef
```

```
##  (Intercept)            X          X.2          X.3          X.4          X.5
## 50.164059615  5.841735821  2.801284260  1.484136848  0.004163542  0.000000000
##          X.6          X.7          X.8          X.9         X.10
##  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
```

- The lasso model creates a sparse model with four variables. The intercept and coefficients for X, X^2 and X^3 closely match the ones chosen in 8(b) whilst the value for x^4 is very small. Therefore, this model provides an accurate estimation of the response Y.

(f)

```
b7 = 5
Y2 = c(b0 + b7*X^7 + e)
df2 = data.frame(X,X^2,X^3,X^4,X^5,X^6,X^7,X^8,X^9,X^10,Y2)
```

```r
# Best subset selection.
regfit.full2 = regsubsets(Y2~.,data=df2,nvmax=10)
reg.summary = summary(regfit.full2)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(Y2 ~ ., data = df2, nvmax = 10)
## 10 Variables  (and intercept)
##      Forced in Forced out
## X        FALSE      FALSE
## X.2      FALSE      FALSE
## X.3      FALSE      FALSE
## X.4      FALSE      FALSE
## X.5      FALSE      FALSE
## X.6      FALSE      FALSE
## X.7      FALSE      FALSE
## X.8      FALSE      FALSE
## X.9      FALSE      FALSE
## X.10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           X   X.2 X.3 X.4 X.5 X.6 X.7 X.8 X.9 X.10
## 1  ( 1 )  " " " " " " " " " " " " "*" " " " " " "
## 2  ( 1 )  " " "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  " " "*" " " " " " "*" " " "*" " " " " " "
## 4  ( 1 )  "*" "*" "*" " " " " " " " " "*" " " " " " "
## 5  ( 1 )  "*" "*" "*" "*" " " " " " "*" " " " " " "
## 6  ( 1 )  "*" " " "*" " " " " " "*" "*" "*" " " "*"
## 7  ( 1 )  "*" " " "*" " " "*" "*" "*" "*" " " "*"
## 8  ( 1 )  "*" "*" "*" "*" " " "*" "*" "*" " " "*"
## 9  ( 1 )  "*" "*" "*" "*" " " "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```r
reg.summary$cp
```

```
##  [1] -0.05835158 -0.57457577 -0.26264076  0.73888307  2.40063344  3.84493124
##  [7]  5.81405698  7.18226563  9.16504068 11.00000000
```

```r
reg.summary$bic
```

```
##  [1] -1420.684 -1418.737 -1415.955 -1412.444 -1408.213 -1404.224 -1399.653
##  [8] -1395.754 -1391.168 -1386.748
```

```r
reg.summary$adjr2
```

```
##  [1] 0.9999994 0.9999994 0.9999994 0.9999994 0.9999994 0.9999994 0.9999994
##  [8] 0.9999994 0.9999994 0.9999994
```

- Cp is lowest for one variable model with the x^7 term.
- BIC value is lowest for the one variable model.

- Adjusted R2 is 0.999+ for the one variable model.
- These statistical metrics point to the one variable model with the x^7 term as being the best choice.

```
# Lasso model.
# Matrix of predictors for a response vector y.
x2 = model.matrix(Y2~.,df2)[,-1]
y2 = df2$Y2

# Training and test sets.
train2 = sample(1:nrow(x2), nrow(x2)/2)
test2 = (-train2)
y2.test = y2[test2]
```

```
# Cross validation to find best value for lambda.
cv.out = cv.glmnet(x[train,],y[train], alpha=1)
cv.out$lambda.min
```

```
## [1] 0.1960455
```

```
# Lasso model with best lambda value.
lasso2 = glmnet(x2, y2, alpha=1)
lasso.coef2 = predict(lasso2,type="coefficients",s=0.2)[1:11,]
lasso.coef2
```

```
## (Intercept)           X         X.2         X.3         X.4         X.5
##   50.604887    0.000000    0.000000    0.000000    0.000000    0.000000
##         X.6         X.7         X.8         X.9        X.10
##    0.000000    4.854434    0.000000    0.000000    0.000000
```

- Lasso model using best value of lambda results in a sparse model with one variable. It assigns a non-zero coefficient to the variable (X^7) that best explains the response Y, and assigns a zero to the rest.

(9) (a) (b)

```
set.seed(2)
x = model.matrix(Apps~.,College)[,-1]
y = College$Apps
grid = 10^seq(10,-2,length=100)

# Training and test sets
train = sample(1:nrow(x), nrow(x)/1.3)
test = (-train)
y.test = y[test]
```

```
# Linear model using least squares (ridge regression with lambda=0) and test MSE.
linear.model = glmnet(x[train,], y[train], alpha=0, lambda=grid, thresh=1e-12)
linear.pred = predict(linear.model, s=0, newx=x[test,],exact=T,x=x[train,],y=y[train])
mean((linear.pred-y.test)^2)
```

```
## [1] 1367387
```

```
# Dataframes for use with lm() function.
train.df = data.frame(College[train,])
test.df = data.frame(College[test,])

# Test MSE using lm() function.
lm.fit = lm(Apps~., data=train.df)
lm.pred = predict(lm.fit, test.df, type=c("response"))
err.lm = mean((lm.pred-test.df$Apps)^2)
err.lm
```

```
## [1] 1367393
```

- The MSE from ridge regression with lambda=0 and lm() function are reasonably similar, and the slight discrepancy is likely due to approximation used by glmnet().

(c)

```
# Ridge regression using lambda chosen by CV.
cv.out = cv.glmnet(x[train,],y[train],alpha=0)
bestlam = cv.out$lambda.min

ridge.mod = glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
ridge.pred = predict(ridge.mod, s=bestlam, newx=x[test,])
err.ridge = mean((ridge.pred-y.test)^2)
err.ridge
```

```
## [1] 1355876
```

- Best value of lambda is 387.9 and the test MSE is slightly lower than for least squares.

(d)

```
# Lasso model using lambda chosen by CV.
cv.out = cv.glmnet(x[train,],y[train],alpha=1)
bestlam = cv.out$lambda.min

lasso.mod = glmnet(x[train,],y[train],alpha=1,lambda=grid)
lasso.pred = predict(lasso.mod, s=bestlam, newx=x[test,])
err.lasso = mean((lasso.pred-y.test)^2)
err.lasso
```

```
## [1] 1357174
```

```
lasso.coef = predict(lasso.mod, type="coefficients", s=bestlam)[1:18,]
length(lasso.coef[lasso.coef!= 0])
```

```
## [1] 18
```

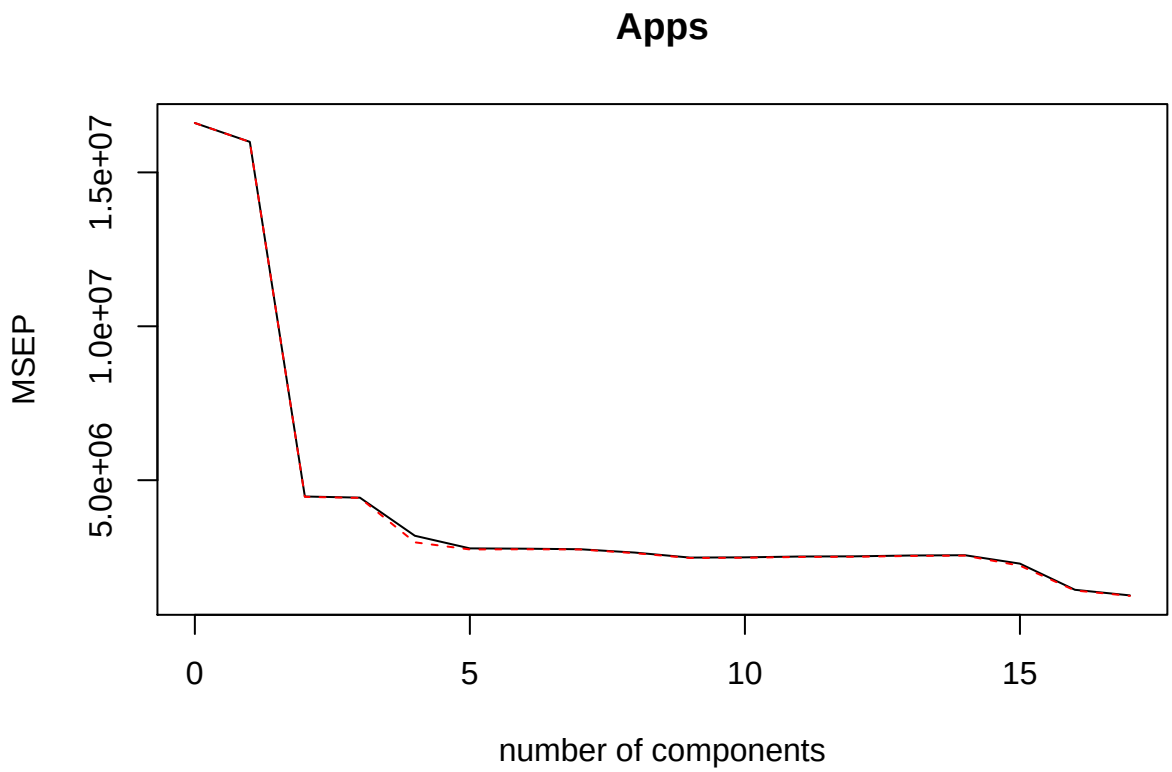
```
lasso.coef
```

```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -821.56743989 -376.48364328    1.61159267   -1.10449603   42.86250271
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
##  -11.29156663    0.08949160    0.05954328   -0.07693368    0.18154296
##         Books      Personal           PhD      Terminal     S.F.Ratio
##    0.21045953    0.05559108   -6.07750546   -4.90698208   21.51866518
##   perc.alumni        Expend     Grad.Rate
##    3.28302767    0.07234029    5.11006362
```

- Test MSE is slightly lower than least squares regression.
- There are no non-zero variables, but many variables are heavily shrunk.

(e)

```
# PCR model with M chosen by cross validation.
pcr.fit = pcr(Apps~., data=College, subset=train, scale=T, validation="CV")
validationplot(pcr.fit, val.type="MSEP")
```




```
#summary(pcr.fit)
```

- From the graph we can observe that the MSE reduces rapidly as M increases and is lowest at M=16. However, the reduction from M=5 to M=16 is small when compared the reduction from M=1 to M=5.

- The test MSE for both M values are shown below.

```
pcr.pred = predict(pcr.fit, x[test,], ncomp=5)
mean((pcr.pred-y.test)^2)
```

```
## [1] 1945054
```

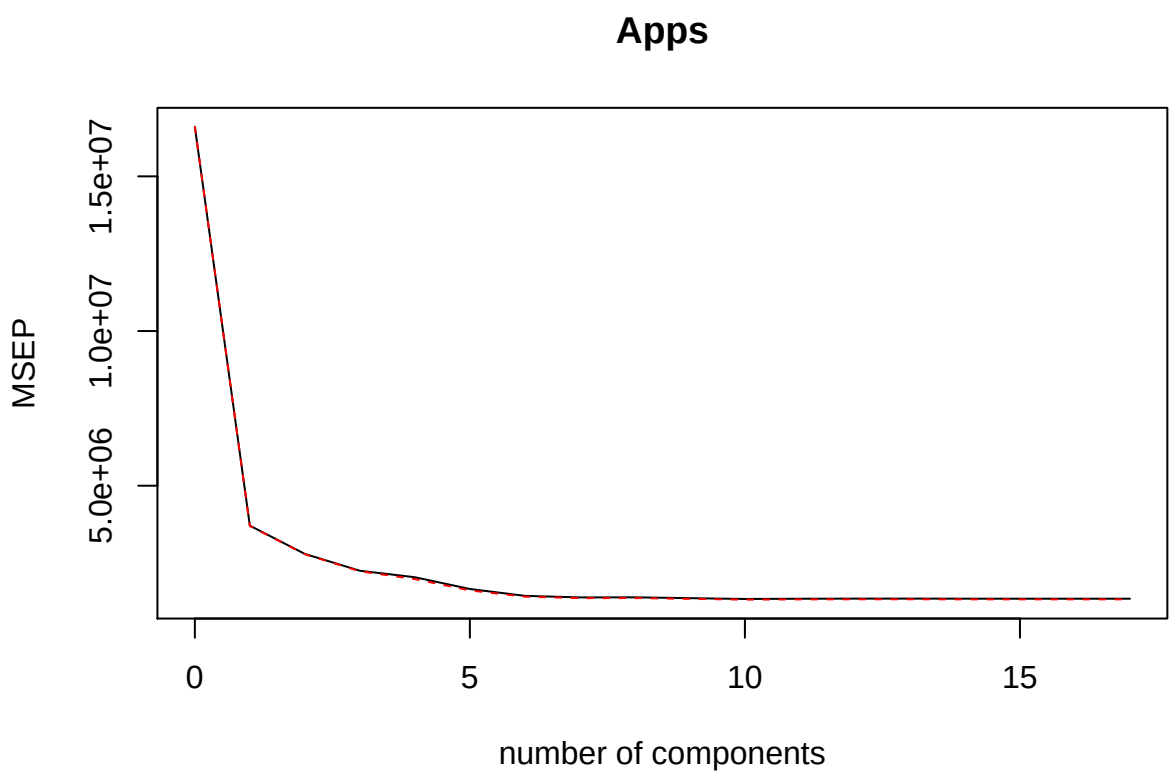
```
pcr.pred = predict(pcr.fit, x[test,], ncomp=16)
err.pcr = mean((pcr.pred-y.test)^2)
err.pcr
```

```
## [1] 1230277
```

- Test MSE for M=5 is 1945054, which is much larger than least squares.
- Best MSE is achieved when M=16, which gives a test MSE reasonably lower than least squared.
- Using M=17 gives the same result as least squares.

(f)

```
# PLS model with M chosen by cross validation.
pls.fit = plsr(Apps~., data=College, subset=train, scale=T, validation="CV")
validationplot(pls.fit, val.type="MSEP")
```



- The lowest MSE occurs when m~8.

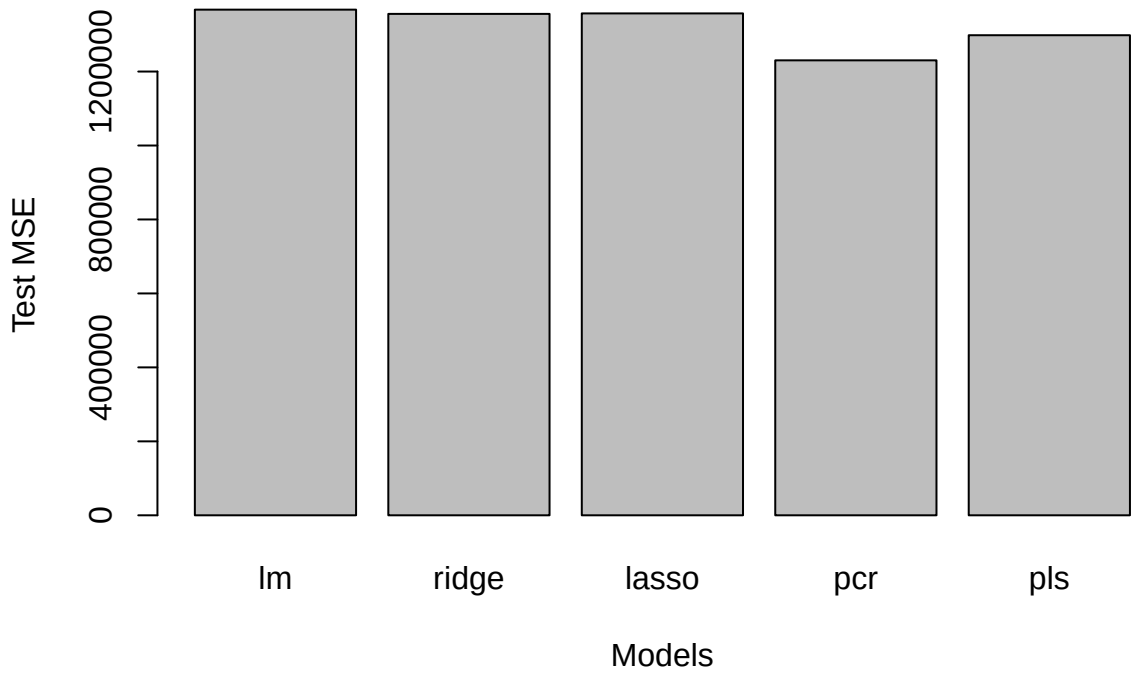
```
pls.pred = predict(pls.fit, x[test,], ncomp=8)
err.pls = mean((pls.pred-y.test)^2)
err.pls
```

```
## [1] 1298214
```

- The test MSE is similar to PCR and slightly lower than least squares.

(g)

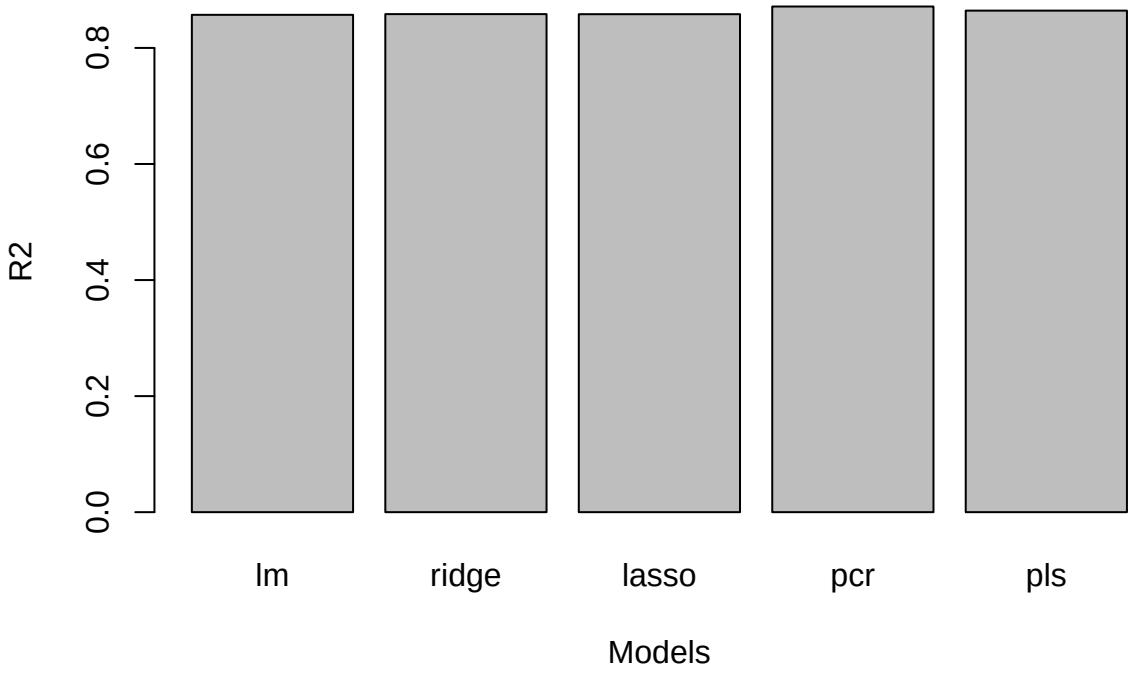
```
err.all = c(err.lm, err.ridge, err.lasso, err.pcr, err.pls)
barplot(err.all, xlab="Models", ylab="Test MSE", names=c("lm", "ridge", "lasso", "pcr", "pls"))
```



- All the models give reasonably similar results, with PCR and PLS giving slightly lower test MSE's.
- We will now compute the R2 metric for each model.

```
test.avg = mean(y.test)
lm.r2 = 1 - mean((lm.pred - y.test)^2) / mean((test.avg - y.test)^2)
ridge.r2 = 1 - mean((ridge.pred - y.test)^2) / mean((test.avg - y.test)^2)
lasso.r2 = 1 - mean((lasso.pred - y.test)^2) / mean((test.avg - y.test)^2)
pcr.r2 = 1 - mean((pcr.pred - y.test)^2) / mean((test.avg - y.test)^2)
pls.r2 = 1 - mean((pls.pred - y.test)^2) / mean((test.avg - y.test)^2)
```

```r
barplot(c(lm.r2, ridge.r2, lasso.r2, pcr.r2, pls.r2), xlab="Models", ylab="R2",
        names=c("lm", "ridge", "lasso", "pcr", "pls"))
```



- Every model has a a R2 metric of around 0.85 or above. SO we can be reasonably confident about the accuracy of the predictions.

10. (a)

```r
# Simulated data set.
set.seed(111)

n = 1000
p = 20
X = matrix(rnorm(n*p), n, p)
#B = rnorm(p)
B = sample(-10:10, 20)
B[1] = 0
B[4] = 0
B[7] = 0
B[11] = 0
B[15] = 0
B[19] = 0
e = rnorm(1000, mean=0, sd=0.1)
```

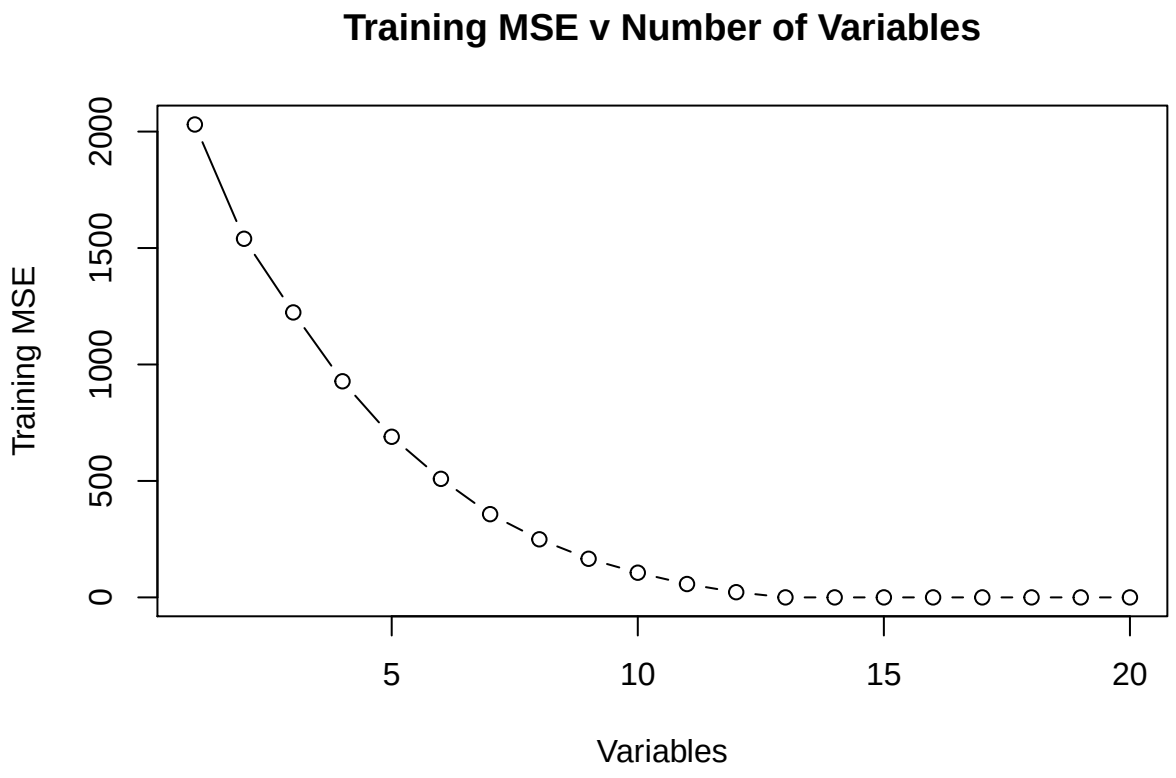
```r
Y = X%*%B + e  # %*% does matrix multiplication

df = data.frame(X, Y)
```

(b)

```r
# Train and test sets.
library(caTools)
sample = sample.split(df$Y, 0.10)
train = subset(df, sample==T)
test = subset(df, sample==F)
```

(c)

```r
# Best subset selection on training set.
regfit.full = regsubsets(Y~., data=train, nvmax=20)
reg.summary = summary(regfit.full)

# Training MSE for best model of each size.
train.mse = (reg.summary$rss)/length(train)
plot(1:20,train.mse,xlab = "Variables",ylab = "Training MSE",
     main = "Training MSE v Number of Variables", pch = 1, type = "b")
```

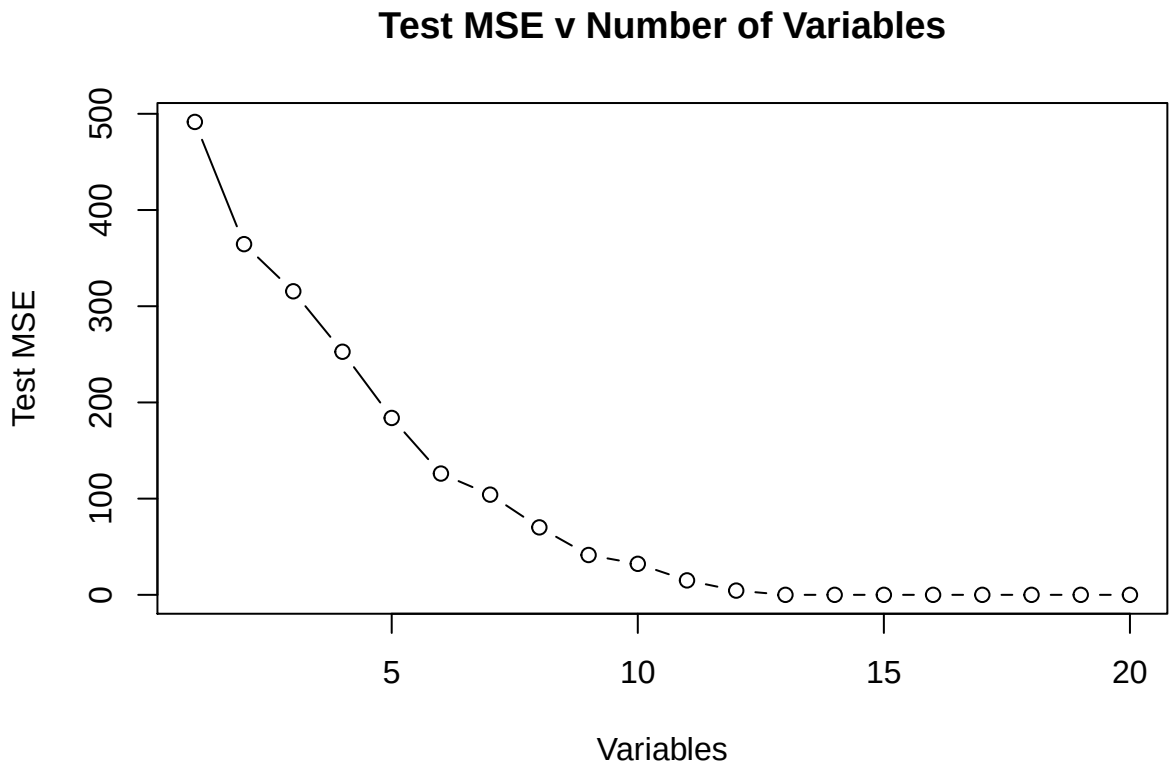


- As expected, training MSE decreases monotonically as the number of variables increase.

- Minimum training MSE is at maximum number of variables: 20.

(d)

```
# For loop to compute test MSE for each best model.
library(HH)
test.mse = rep(NA,20)

for(i in 1:20){
    model=lm.regsubsets(regfit.full, i)
    model.pred = predict(model, newdata=test, type=c("response"))
    test.mse[i] =  mean((test$Y-model.pred)^2)
  }

# Plot
plot(1:20,test.mse,xlab = "Variables",ylab = "Test MSE",
     main = "Test MSE v Number of Variables", pch = 1, type = "b")
```



- Test MSE decreases rapidly as the number of variables increase, but the minimum is not at the max number of variables.
- Minimum test MSE is when number of variables: 13.

(e)

- Minimum Test MSE occurs at a model with 13 variables. The test MSE deceases rapidly until it reaches the minimum and then starts to rise thereafter.

- As the model flexibility increases, it is better able to fit the data set. This results in the Test MSE decreasing rapidly until it reaches a minimum. Thereafter, further increases in model flexibility causes over fitting and hence results in an increase in the Test MSE.

(f)

```
# Coefficients of best model found through subset selection.
coef(regfit.full, 13)
```

```
##   (Intercept)            X2            X3            X5            X6
##  -0.001334982  -6.003548929  -7.008083537 -10.009342703   1.999414193
##            X8            X9           X10           X13           X14
##   6.978017828  -2.989193466   4.997145917   6.005053131   3.973003175
##           X16           X17           X18           X20
##   8.007345963  -7.999568356   9.993296372   3.005914845
```

```
B
```

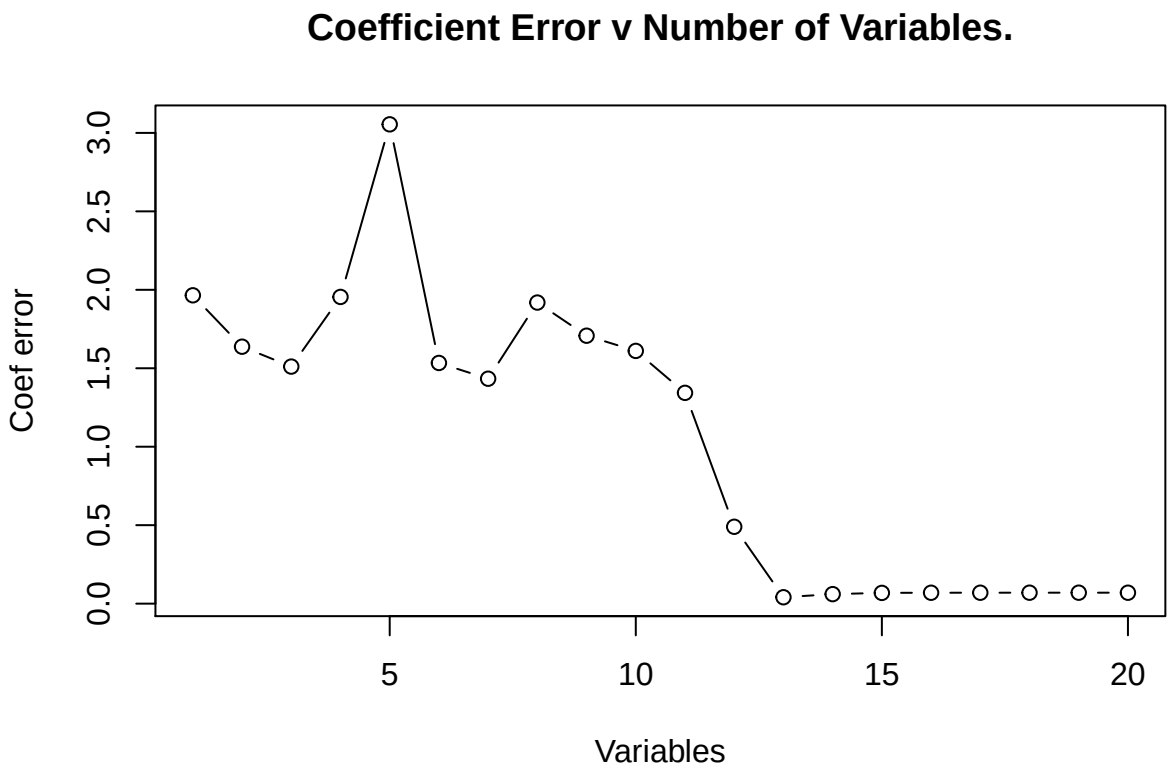
```
##  [1]   0  -6  -7   0 -10   2   0   7  -3   5   0   0   6   4   0   8  -8  10   0
## [20]   3
```

- Best model variables exactly match the 13 non-zero variables from the original model, and their respective coefficients are highly similar.

(g)

```
# Original coefficient values transposed and columns renamed.
B = as.data.frame(t(B))
names(B) = paste0('X', 1:(ncol(B)))
```

```
# Loop to calculate root squared errors of the true and estimated coefficients.
coef.err = rep(NA,20)
for (i in 1:20){
  a = coef(regfit.full, i)
  coef.err[i] = sqrt(sum(((a[-1] - B[names(a)[-1]])^2)))
}

plot(1:20,coef.err,xlab = "Variables",ylab = "Coef error",
     main="Coefficient Error v Number of Variables.", pch = 1, type = "b")
```




```
which.min(coef.err)
```

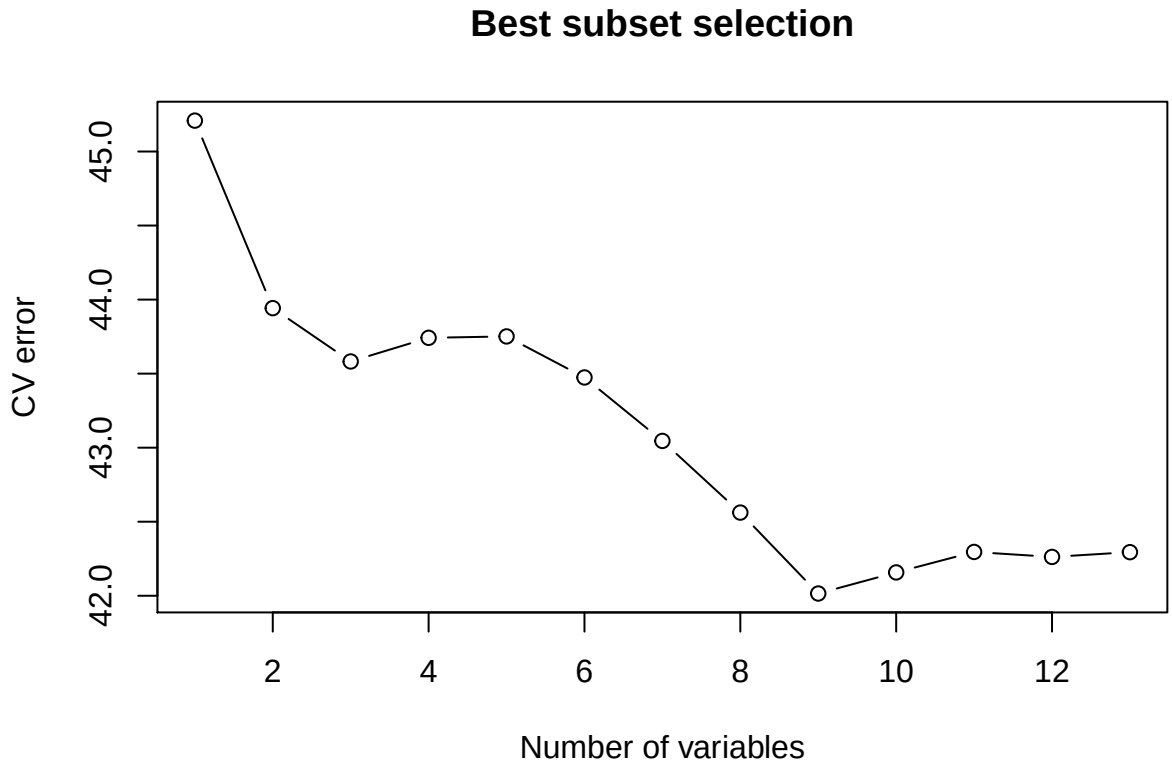
```
## [1] 13
```

- The chart starts in a disjointed manner before the coefficient errors start reducing rapidly. Eventually, it does show a minimum at the same variable size as for the test mse. However, when using a different random seed the coefficient error chart does not always find a minimum at the same variable size as the test mse chart. Therefore, a model that gives a minimum for coefficient error does not necessarily lead to a lower test mse.

11. (a)

```
# Best subset selection using cross validation with 10 folds.

#Predict function from p249.
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
}

set.seed(121)
k = 10
```

```r
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(1:13, mean.cv.errors, xlab = "Number of variables", ylab = "CV error",
     main= "Best subset selection", pch = 1, type = "b")
```



- CV error is lowest for model with 9 variables. CV Error = 42.0151126.

```r
# Using Lasso to create a sparse model.
set.seed(121)

x = model.matrix(crim~.,Boston)[,-1]
y = Boston$crim
grid = 10^seq(10,-2,length=100)

train = sample(1:nrow(x), nrow(x)/1.3)
```

```
test = (-train)
y.test = y[test]
```

```
set.seed(121)
cv.out = cv.glmnet(x[train,], y[train], alpha=1)
bestlam = cv.out$lambda.min

lasso.mod = glmnet(x[train,],y[train],alpha=1,lambda=grid)
lasso.pred = predict(lasso.mod, s=bestlam, newx=x[test,])
mean((lasso.pred-y.test)^2)
```

```
## [1] 31.61342
```

```
lasso.coef = predict(lasso.mod, type="coefficients", s=bestlam)[1:13,]
lasso.coef
```

```
##  (Intercept)           zn        indus         chas          nox           rm
## 11.863977269  0.035551345 -0.069356583 -0.341147083 -7.159328191  0.258906989
##          age          dis          rad          tax      ptratio        black
##  0.000000000 -0.756621224  0.510388304  0.000000000 -0.159640151 -0.009241809
##        lstat
##  0.145509242
```

- Test MSE of 31.6, with only age and tax being exactly zero, we have a best model with 10 variables.

```
# Using Ridge Regression.
cv.out = cv.glmnet(x[train,], y[train], alpha=0)
bestlam = cv.out$lambda.min

glm.mod = glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
glm.pred = predict(glm.mod, s=bestlam, newx=x[test,])
mean((glm.pred-y.test)^2)
```

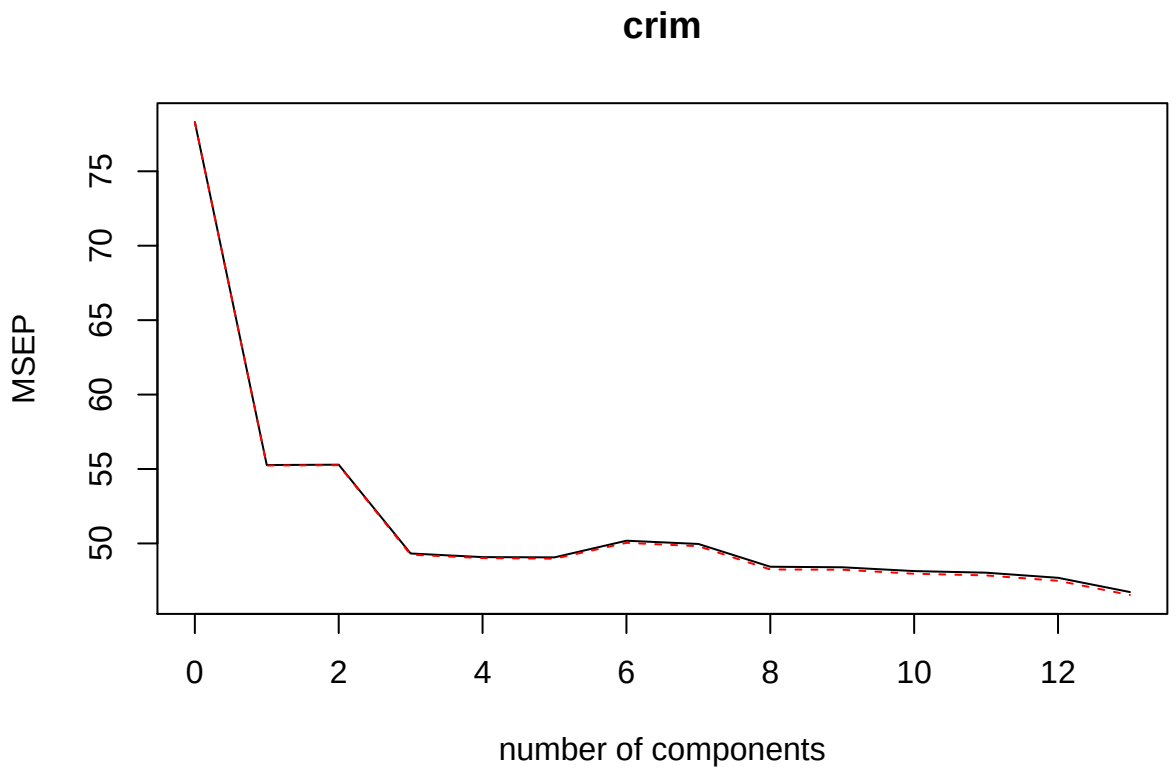
```
## [1] 31.87334
```

```
glm.coef = predict(glm.mod, type="coefficients", s=bestlam)[1:13,]
glm.coef
```

```
##  (Intercept)           zn        indus         chas          nox           rm
##  9.139463700  0.033824862 -0.087751836 -0.530186066 -6.237996789  0.383596934
##          age          dis          rad          tax      ptratio        black
##  0.003905644 -0.708321504  0.425606232  0.003344803 -0.125766385 -0.010097779
##        lstat
##  0.155705890
```

- Lambda chosen by cross validation is close to zero, so both ridge regression and lasso test mse are similar to that provided by least squares.

```r
# Using PCR
pcr.fit = pcr(crim~., data=Boston, subset=train, scale=T, validation="CV")
validationplot(pcr.fit, val.type="MSEP")
```




```r
pcr.pred = predict(pcr.fit, x[test,], ncomp=8)
mean((pcr.pred-y.test)^2)
```

```
## [1] 33.74218
```

- Test MSE of 33.7.

(b) (c)

- I would choose the Lasso model, as it gives the lowest test mse.
- Lasso models are generally more interpretable.
- It results in a sparse model with 10 variables. Two variables whose effect on the response were below the required threshold were removed.