Chapter 06: Linear Model Selection and Regularization

Solutions to Exercises

February 03, 2023

CONCEPTUAL

EXERCISE 1:

Part a)

Best subset will have the smallest train RSS because the models will optimize on the training RSS and best subset will try every model that forward and backward selection will try.

Part b)

The best test RSS model could be any of the three. Best subset could easily overfit if the data has large p predictors relative to n observations. Forward and backward selection might not converge on the same model but try the same number of models and hard to say which selection process would be better.

Part c)

- i. TRUE
- ii. TRUE
- iii. FALSE
- iv. FALSE
- v. FALSE

EXERCISE 2:

Part a)

iii. is TRUE - lasso puts a budget constraint on least squares (less flexible)

Part b)

iii. is TRUE - ridge also puts a budget constraint on least squares (less flexible)

Part c)

ii. is TRUE - a non-linear model would be more flexible and have higher variance, less bias

---

EXERCISE 3:

Part a)

iv. is TRUE - as s is increased, there is less and less constraint on the model and it should always have a better training error (if s is increased to s', then the best model using a budget of s would be included when using a budget of s')

Part b)

ii. is TRUE - test error will improve (decrease) to a point and then will worsen (increase) as constraints loosen and model overfits

Part c)

iii. is TRUE - variance always increases with fewer constraints

Part d)

iv. is TRUE - bias always decreases with more model flexibility

Part e)

v. is TRUE - the irreducible error is a constant value, not related to model selection

---

EXERCISE 4:

This problem is similar to Excercise 3, but for ridge instead of lasso and using λ instead of s. For each question part, ridge and lasso should be the same directionally except that increasing λ puts a heavier penalty in the equation, equivalent to reducing the budget s, so the answers to Exercise 4 should be flipped (horizontally) from answers in Exercise 3.

Part a)

iii. is TRUE - training error increases steadily

Part b)

ii. is TRUE - test error will decrease initially and then increase

Part c)

iv. is TRUE - variance always decrease with more constraints

Part d)

iii. is TRUE - bias always increase with less model flexibility

Part e)

v. is TRUE - the irreducible error is a constant value, not related to model selection

---

EXERCISE 5:

Part a)

Ridge: minimize $(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$

Part b)

Step 1: Expanding the equation from Part a:

$$\begin{gathered}(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2) \\ = (y_1^2 + \hat{\beta}_1^2 x_{11}^2 + \hat{\beta}_2^2 x_{12}^2 - 2\hat{\beta}_1 x_{11} y_1 - 2\hat{\beta}_2 x_{12} y_1 + 2\hat{\beta}_1 \hat{\beta}_2 x_{11} x_{12}) \\ +(y_2^2 + \hat{\beta}_1^2 x_{21}^2 + \hat{\beta}_2^2 x_{22}^2 - 2\hat{\beta}_1 x_{21} y_2 - 2\hat{\beta}_2 x_{22} y_2 + 2\hat{\beta}_1 \hat{\beta}_2 x_{21} x_{22}) \\ +\lambda\hat{\beta}_1^2 + \lambda\hat{\beta}_2^2\end{gathered}$$

Step 2: Taking the partial derivative to $\hat{\beta}_1$ and setting equation to 0 to minimize:

$$\frac{\partial}{\partial \hat{\beta}_1} : (2\hat{\beta}_1 x_{11}^2 - 2x_{11} y_1 + 2\hat{\beta}_2 x_{11} x_{12}) + (2\hat{\beta}_1 x_{21}^2 - 2x_{21} y_2 + 2\hat{\beta}_2 x_{21} x_{22}) + 2\lambda\hat{\beta}_1 = 0$$

Step 3: Setting $x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$ and dividing both sides of the equation by 2:

$$(\hat{\beta}_1 x_1^2 - x_1 y_1 + \hat{\beta}_2 x_1^2) + (\hat{\beta}_1 x_2^2 - x_2 y_2 + \hat{\beta}_2 x_2^2) + \lambda\hat{\beta}_1 = 0$$

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

Step 4: Add $2\hat{\beta}_1 x_1 x_2$ and $2\hat{\beta}_2 x_1 x_2$ to both sides of the equation:

$$\begin{gathered}\hat{\beta}_1 (x_1^2 + x_2^2 + 2x_1 x_2) + \hat{\beta}_2 (x_1^2 + x_2^2 + 2x_1 x_2) + \lambda\hat{\beta}_1 = x_1 y_1 + x_2 y_2 + 2\hat{\beta}_1 x_1 x_2 + 2\hat{\beta}_2 x_1 x_2 \\ \hat{\beta}_1 (x_1 + x_2)^2 + \hat{\beta}_2 (x_1 + x_2)^2 + \lambda\hat{\beta}_1 = x_1 y_1 + x_2 y_2 + 2\hat{\beta}_1 x_1 x_2 + 2\hat{\beta}_2 x_1 x_2\end{gathered}$$

Step 5: Because $x_1 + x_2 = 0$, we can eliminate the first two terms:

$$\lambda\hat{\beta}_1 = x_1 y_1 + x_2 y_2 + 2\hat{\beta}_1 x_1 x_2 + 2\hat{\beta}_2 x_1 x_2$$

Step 6: Similarly by taking the partial derivative to $\hat{\beta}_2$, we can get the equation:

$$\lambda\hat{\beta}_2 = x_1 y_1 + x_2 y_2 + 2\hat{\beta}_1 x_1 x_2 + 2\hat{\beta}_2 x_1 x_2$$

Step 7: The left side of the equations for both $\lambda\hat{\beta}_1$ and $\lambda\hat{\beta}_2$ are the same so we have:

$$\lambda\hat{\beta}_1 = \lambda\hat{\beta}_2$$

$$\hat{\beta}_1 = \hat{\beta}_2$$

Part c)

Lasso: minimize $(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(|\hat{\beta}_1| + |\hat{\beta}_2|)$

Part d)

Replacing the constraint term from Part b, the derivative term to β is:

$$\frac{\partial}{\partial\hat{\beta}}(\lambda|\beta|) : \lambda\frac{|\beta|}{\beta}$$

Following through the steps in Part b, we get:

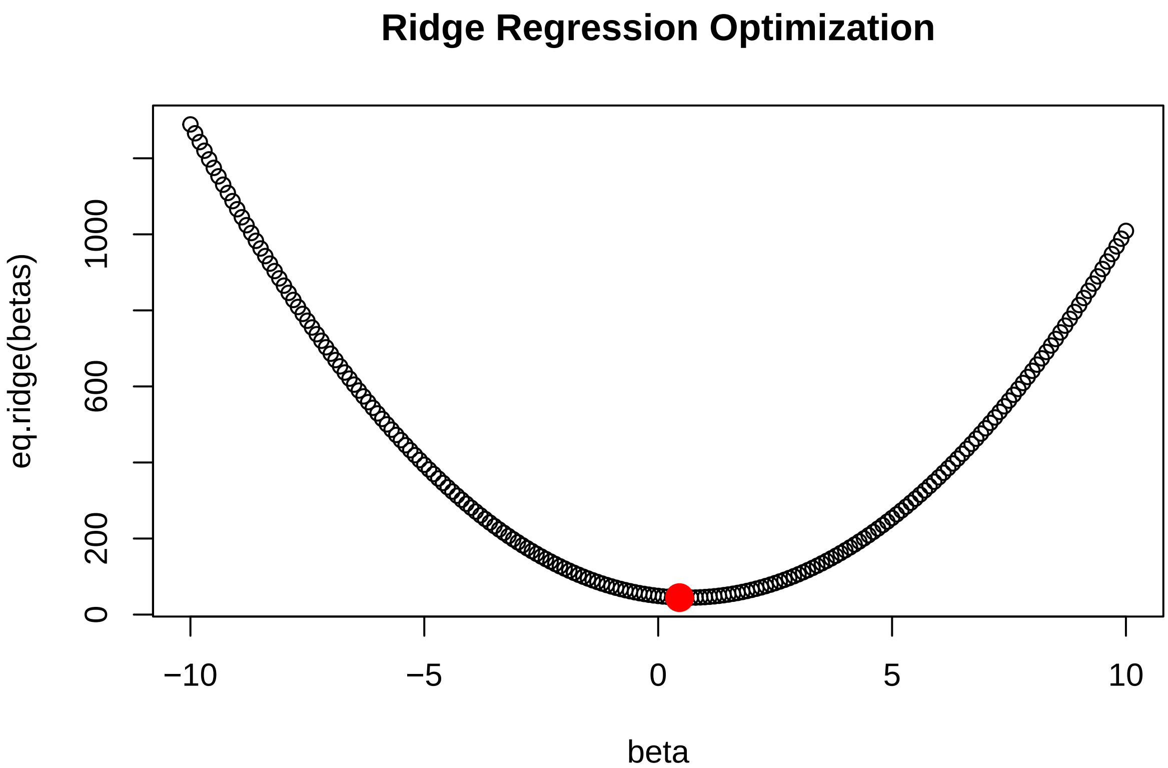
$$\lambda\frac{|\beta_1|}{\beta_1} = \lambda\frac{|\beta_2|}{\beta_2}$$

So it seems that the lasso just requires that β_1 and β_2 are both positive or both negative (ignoring possibility of 0…)

EXERCISE 6:

Part a)

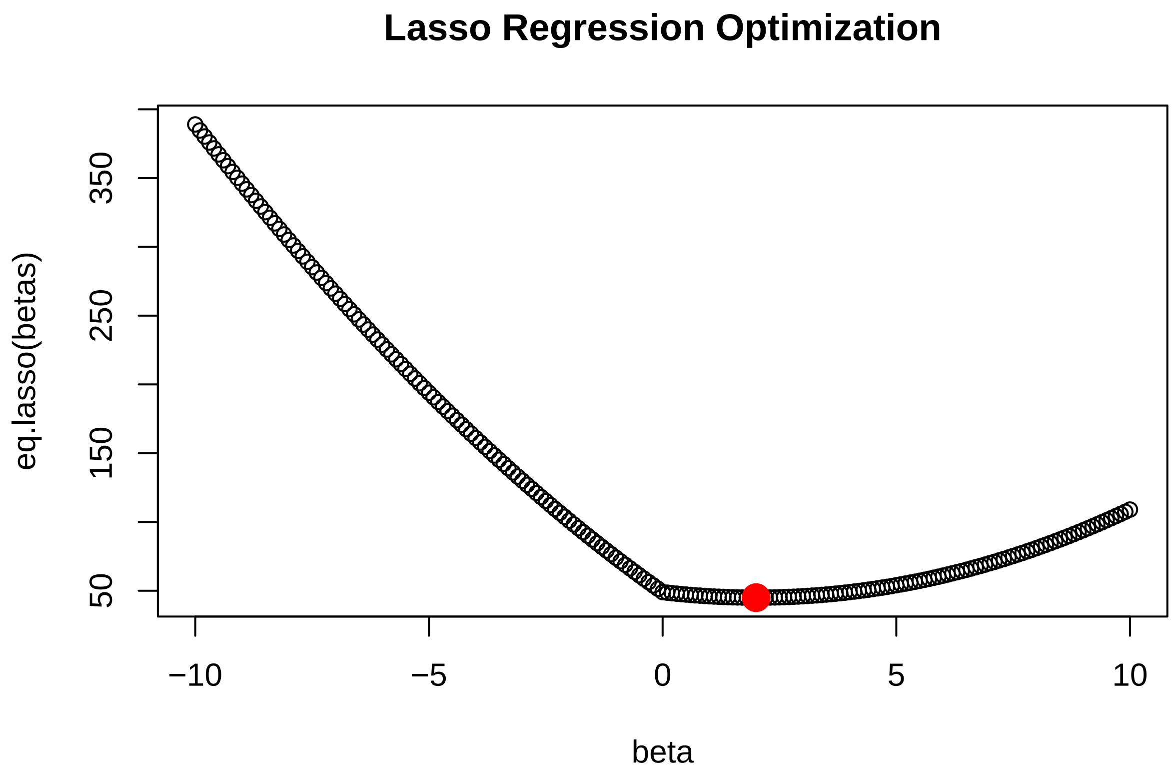
```
betas <- seq(-10,10,0.1)
eq.ridge <- function(beta, y=7, lambda=10) (y-beta)^2 + lambda*beta^2
plot(betas, eq.ridge(betas), xlab="beta", main="Ridge Regression Optimization", pch=1)
points(5/(1+10), eq.ridge(7/(1+10)), pch=16, col="red", cex=2)
```



For $y = 7$ and $\lambda = 10$, $\hat{\beta} = \frac{7}{1+10}$ minimizes the ridge regression equation

Part b)

```
betas <- seq(-10,10,0.1)
eq.lasso <- function(beta, y=7, lambda=10) (y-beta)^2 + lambda*abs(beta)
plot(betas, eq.lasso(betas), xlab="beta", main="Lasso Regression Optimization", pch=1)
points(7-10/2, eq.lasso(7-10/2), pch=16, col="red", cex=2)
```



For $y = 7$ and $\lambda = 10$, $\hat{\beta} = 7 - \frac{10}{2}$ minimizes the ridge regression equation

---

EXERCISE 7:

Part a)

[... *will come back to this. maybe.*]

Part b)

[... *will come back to this. maybe.*]

Part c)

[... *will come back to this. maybe.*]

Part d)

[... *will come back to this. maybe.*]

Part e)

[... *will come back to this. maybe.*]

APPLIED

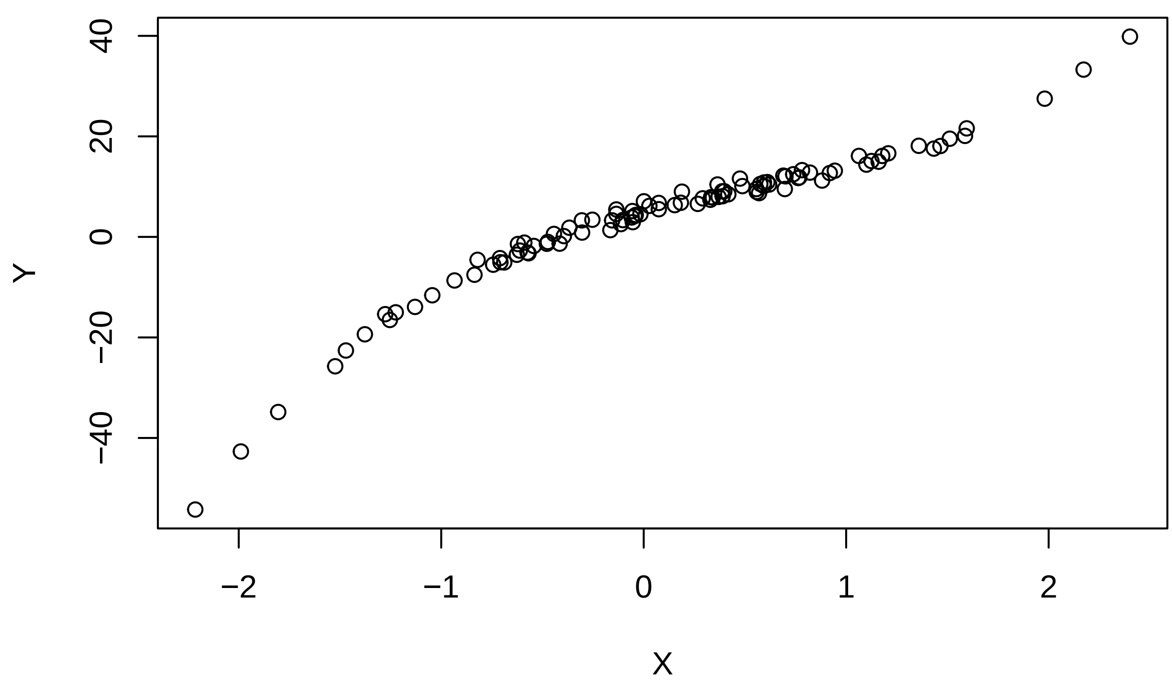
EXERCISE 8:

Part a)

```
set.seed(1)
X <- rnorm(100)
eps <- rnorm(100)
```

Part b)

```
Y <- 5 + 10*X - 3*X^2 + 2*X^3 + eps
plot(X,Y)
```



Part c)

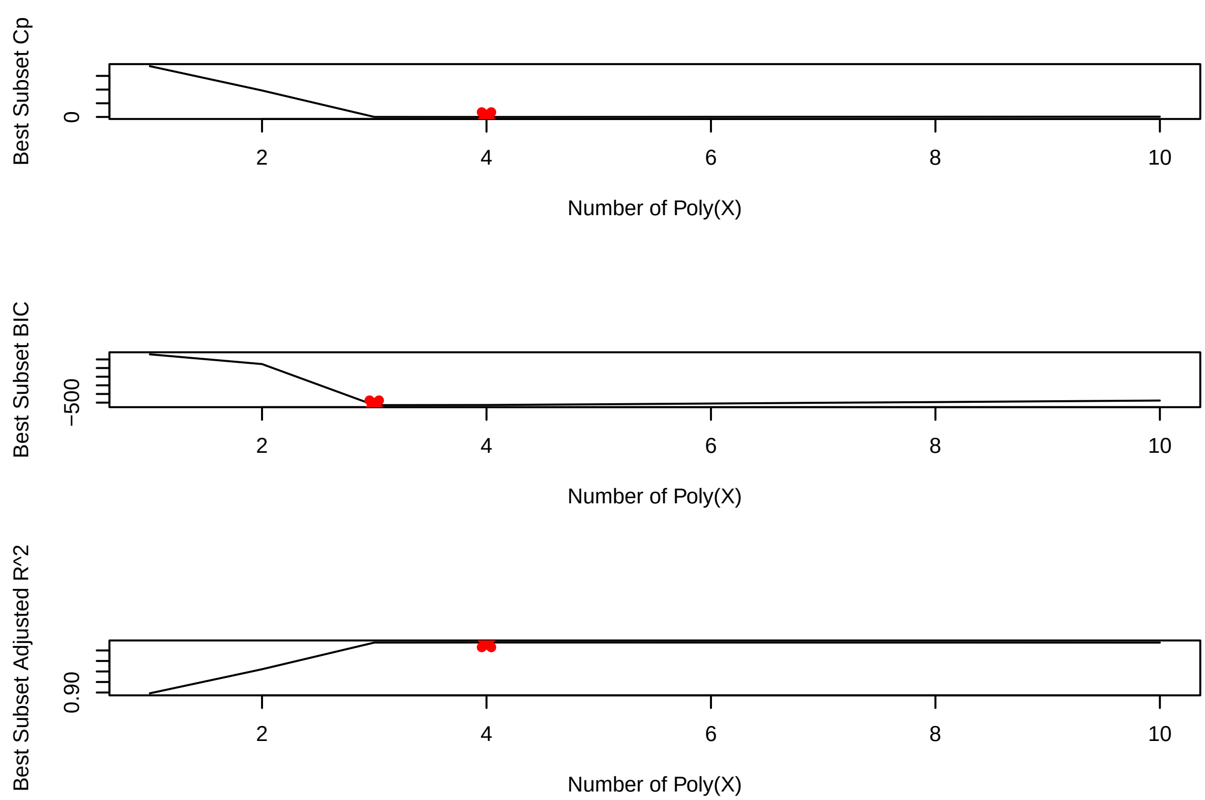
```
require(leaps)
regfit.full <- regsubsets(Y~poly(X,10,raw=T), data=data.frame(Y,X), nvmax=10)
(reg.summary <- summary(regfit.full))
```

```
## Subset selection object
## Call: regsubsets.formula(Y ~ poly(X, 10, raw = T), data = data.frame(Y,
```

```
##     X), nvmax = 10)
## 10 Variables  (and intercept)
##                        Forced in Forced out
## poly(X, 10, raw = T)1      FALSE      FALSE
## poly(X, 10, raw = T)2      FALSE      FALSE
## poly(X, 10, raw = T)3      FALSE      FALSE
## poly(X, 10, raw = T)4      FALSE      FALSE
## poly(X, 10, raw = T)5      FALSE      FALSE
## poly(X, 10, raw = T)6      FALSE      FALSE
## poly(X, 10, raw = T)7      FALSE      FALSE
## poly(X, 10, raw = T)8      FALSE      FALSE
## poly(X, 10, raw = T)9      FALSE      FALSE
## poly(X, 10, raw = T)10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           poly(X, 10, raw = T)1 poly(X, 10, raw = T)2 poly(X, 10, raw = T)3
## 1  ( 1 )  "*"                   " "                   " "
## 2  ( 1 )  "*"                   "*"                   " "
## 3  ( 1 )  "*"                   "*"                   "*"
## 4  ( 1 )  "*"                   "*"                   "*"
## 5  ( 1 )  "*"                   "*"                   "*"
## 6  ( 1 )  "*"                   "*"                   "*"
## 7  ( 1 )  "*"                   "*"                   "*"
## 8  ( 1 )  "*"                   "*"                   "*"
## 9  ( 1 )  "*"                   "*"                   "*"
## 10  ( 1 ) "*"                   "*"                   "*"
##           poly(X, 10, raw = T)4 poly(X, 10, raw = T)5 poly(X, 10, raw = T)6
## 1  ( 1 )  " "                   " "                   " "
## 2  ( 1 )  " "                   " "                   " "
## 3  ( 1 )  " "                   " "                   " "
## 4  ( 1 )  " "                   "*"                   " "
## 5  ( 1 )  " "                   "*"                   "*"
## 6  ( 1 )  " "                   " "                   " "
## 7  ( 1 )  " "                   "*"                   "*"
## 8  ( 1 )  "*"                   " "                   "*"
## 9  ( 1 )  "*"                   "*"                   "*"
## 10  ( 1 ) "*"                   "*"                   "*"
##           poly(X, 10, raw = T)7 poly(X, 10, raw = T)8 poly(X, 10, raw = T)9
## 1  ( 1 )  " "                   " "                   " "
## 2  ( 1 )  " "                   " "                   " "
## 3  ( 1 )  " "                   " "                   " "
## 4  ( 1 )  " "                   " "                   " "
## 5  ( 1 )  " "                   " "                   " "
## 6  ( 1 )  "*"                   "*"                   "*"
## 7  ( 1 )  " "                   "*"                   " "
## 8  ( 1 )  " "                   "*"                   "*"
## 9  ( 1 )  " "                   "*"                   "*"
## 10  ( 1 ) "*"                   "*"                   "*"
##           poly(X, 10, raw = T)10
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
## 5  ( 1 )  " "
```

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

```
par(mfrow=c(3,1))
min.cp <- which.min(reg.summary$cp)
plot(reg.summary$cp, xlab="Number of Poly(X)", ylab="Best Subset Cp", type="l")
points(min.cp, reg.summary$cp[min.cp], col="red", pch=4, lwd=5)
min.bic <- which.min(reg.summary$bic)
plot(reg.summary$bic, xlab="Number of Poly(X)", ylab="Best Subset BIC", type="l")
points(min.bic, reg.summary$bic[min.bic], col="red", pch=4, lwd=5)
min.adjr2 <- which.max(reg.summary$adjr2)
plot(reg.summary$adjr2, xlab="Number of Poly(X)", ylab="Best Subset Adjusted R^2", type="l")
points(min.adjr2, reg.summary$adjr2[min.adjr2], col="red", pch=4, lwd=5)
```




```
coef(regfit.full, min.cp)
```

```
##           (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##            5.07200775           10.38745596           -3.15424359
## poly(X, 10, raw = T)3 poly(X, 10, raw = T)5
##            1.55797426            0.08072292
```

```r
coef(regfit.full, min.bic)
```

```
##           (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##              5.061507              9.975280             -3.123791
## poly(X, 10, raw = T)3
##              2.017639
```

```r
coef(regfit.full, min.adjr2)
```

```
##           (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##            5.07200775           10.38745596           -3.15424359
## poly(X, 10, raw = T)3 poly(X, 10, raw = T)5
##            1.55797426            0.08072292
```

Part d)

```r
# forward selection
regfit.fwd <- regsubsets(Y~poly(X,10,raw=T), data=data.frame(Y,X), nvmax=10)
(fwd.summary <- summary(regfit.fwd))
```

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

```
## 3  ( 1 )  " "                    " "                    " "
## 4  ( 1 )  " "                    "*"                    " "
## 5  ( 1 )  " "                    "*"                    "*"
## 6  ( 1 )  " "                    " "                    " "
## 7  ( 1 )  " "                    "*"                    "*"
## 8  ( 1 )  "*"                    " "                    "*"
## 9  ( 1 )  "*"                    "*"                    "*"
## 10  ( 1 ) "*"                    "*"                    "*"
##           poly(X, 10, raw = T)7 poly(X, 10, raw = T)8 poly(X, 10, raw = T)9
## 1  ( 1 )  " "                    " "                    " "
## 2  ( 1 )  " "                    " "                    " "
## 3  ( 1 )  " "                    " "                    " "
## 4  ( 1 )  " "                    " "                    " "
## 5  ( 1 )  " "                    " "                    " "
## 6  ( 1 )  "*"                    "*"                    "*"
## 7  ( 1 )  " "                    "*"                    " "
## 8  ( 1 )  " "                    "*"                    "*"
## 9  ( 1 )  " "                    "*"                    "*"
## 10  ( 1 ) "*"                    "*"                    "*"
##           poly(X, 10, raw = T)10
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
## 5  ( 1 )  " "
## 6  ( 1 )  " "
## 7  ( 1 )  "*"
## 8  ( 1 )  "*"
## 9  ( 1 )  "*"
## 10  ( 1 ) "*"
```

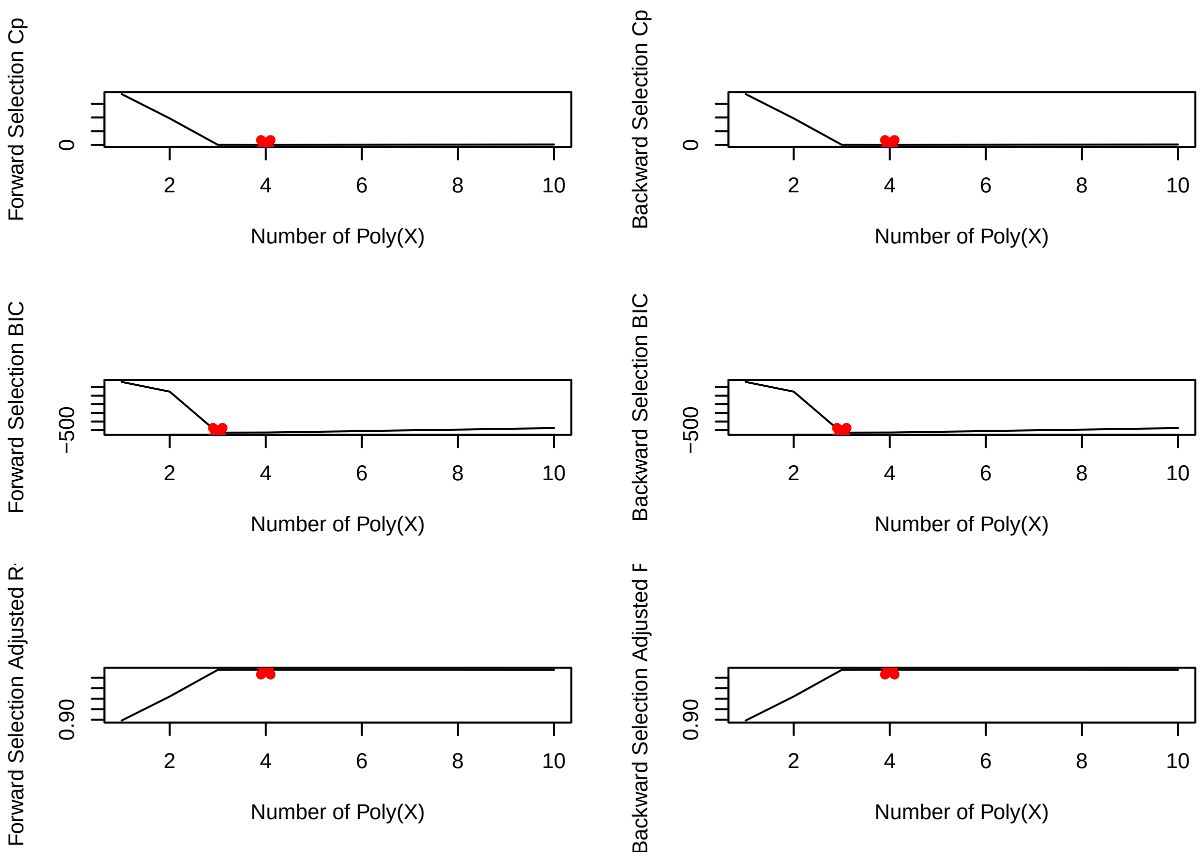
```r
# backward selection
regfit.bwd <- regsubsets(Y~poly(X,10,raw=T), data=data.frame(Y,X), nvmax=10)
(bwd.summary <- summary(regfit.bwd))
```

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

```
## 2  ( 1 )  "*"                  "*"                  " "
## 3  ( 1 )  "*"                  "*"                  "*"
## 4  ( 1 )  "*"                  "*"                  "*"
## 5  ( 1 )  "*"                  "*"                  "*"
## 6  ( 1 )  "*"                  "*"                  "*"
## 7  ( 1 )  "*"                  "*"                  "*"
## 8  ( 1 )  "*"                  "*"                  "*"
## 9  ( 1 )  "*"                  "*"                  "*"
## 10  ( 1 ) "*"                  "*"                  "*"
##           poly(X, 10, raw = T)4 poly(X, 10, raw = T)5 poly(X, 10, raw = T)6
## 1  ( 1 )  " "                  " "                  " "
## 2  ( 1 )  " "                  " "                  " "
## 3  ( 1 )  " "                  " "                  " "
## 4  ( 1 )  " "                  "*"                  " "
## 5  ( 1 )  " "                  "*"                  "*"
## 6  ( 1 )  " "                  " "                  " "
## 7  ( 1 )  " "                  "*"                  "*"
## 8  ( 1 )  "*"                  " "                  "*"
## 9  ( 1 )  "*"                  "*"                  "*"
## 10  ( 1 ) "*"                  "*"                  "*"
##           poly(X, 10, raw = T)7 poly(X, 10, raw = T)8 poly(X, 10, raw = T)9
## 1  ( 1 )  " "                  " "                  " "
## 2  ( 1 )  " "                  " "                  " "
## 3  ( 1 )  " "                  " "                  " "
## 4  ( 1 )  " "                  " "                  " "
## 5  ( 1 )  " "                  " "                  " "
## 6  ( 1 )  "*"                  "*"                  "*"
## 7  ( 1 )  " "                  "*"                  " "
## 8  ( 1 )  " "                  "*"                  "*"
## 9  ( 1 )  " "                  "*"                  "*"
## 10  ( 1 ) "*"                  "*"                  "*"
##           poly(X, 10, raw = T)10
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
## 5  ( 1 )  " "
## 6  ( 1 )  " "
## 7  ( 1 )  "*"
## 8  ( 1 )  "*"
## 9  ( 1 )  "*"
## 10  ( 1 ) "*"
```

```r
par(mfrow=c(3,2))

min.cp <- which.min(fwd.summary$cp)
plot(fwd.summary$cp, xlab="Number of Poly(X)", ylab="Forward Selection Cp", type="l")
points(min.cp, fwd.summary$cp[min.cp], col="red", pch=4, lwd=5)

min.cp <- which.min(bwd.summary$cp)
plot(bwd.summary$cp, xlab="Number of Poly(X)", ylab="Backward Selection Cp", type="l")
points(min.cp, bwd.summary$cp[min.cp], col="red", pch=4, lwd=5)

min.bic <- which.min(fwd.summary$bic)
```

```
plot(fwd.summary$bic, xlab="Number of Poly(X)", ylab="Forward Selection BIC", type="l")
points(min.bic, fwd.summary$bic[min.bic], col="red", pch=4, lwd=5)

min.bic <- which.min(bwd.summary$bic)
plot(bwd.summary$bic, xlab="Number of Poly(X)", ylab="Backward Selection BIC", type="l")
points(min.bic, bwd.summary$bic[min.bic], col="red", pch=4, lwd=5)

min.adjr2 <- which.max(fwd.summary$adjr2)
plot(fwd.summary$adjr2, xlab="Number of Poly(X)", ylab="Forward Selection Adjusted R^2", type="l")
points(min.adjr2, fwd.summary$adjr2[min.adjr2], col="red", pch=4, lwd=5)

min.adjr2 <- which.max(bwd.summary$adjr2)
plot(bwd.summary$adjr2, xlab="Number of Poly(X)", ylab="Backward Selection Adjusted R^2", type="l")
points(min.adjr2, bwd.summary$adjr2[min.adjr2], col="red", pch=4, lwd=5)
```




```
# coefficients of selected models
coef(regfit.fwd, which.min(fwd.summary$cp))
```

```
##           (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##            5.07200775           10.38745596           -3.15424359
## poly(X, 10, raw = T)3 poly(X, 10, raw = T)5
##            1.55797426            0.08072292
```

```
coef(regfit.bwd, which.min(bwd.summary$cp))
```

```
##            (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##             5.07200775           10.38745596           -3.15424359
## poly(X, 10, raw = T)3 poly(X, 10, raw = T)5
##             1.55797426            0.08072292
```

```
coef(regfit.fwd, which.min(fwd.summary$bic))
```

```
##            (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##               5.061507              9.975280             -3.123791
## poly(X, 10, raw = T)3
##               2.017639
```

```
coef(regfit.bwd, which.min(bwd.summary$bic))
```

```
##            (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##               5.061507              9.975280             -3.123791
## poly(X, 10, raw = T)3
##               2.017639
```

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

```
##            (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##             5.07200775           10.38745596           -3.15424359
## poly(X, 10, raw = T)3 poly(X, 10, raw = T)5
##             1.55797426            0.08072292
```

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

```
##            (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##             5.07200775           10.38745596           -3.15424359
## poly(X, 10, raw = T)3 poly(X, 10, raw = T)5
##             1.55797426            0.08072292
```

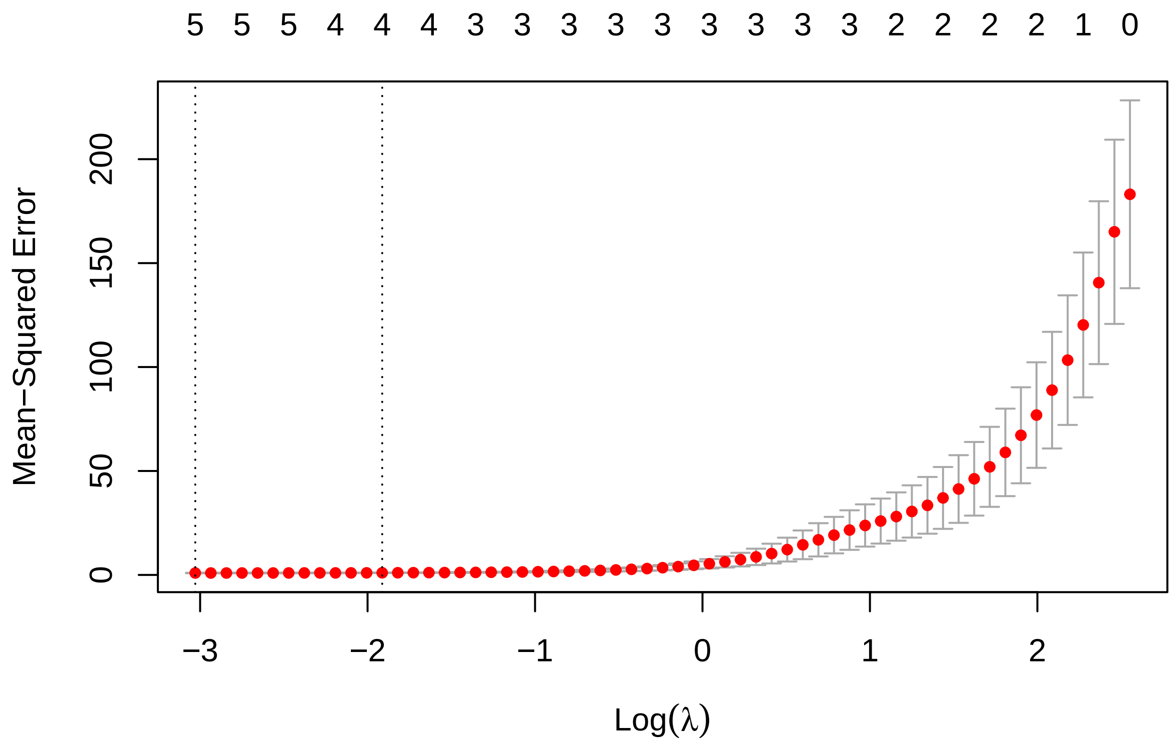
Best subset, foward selection and backward selection all resulted in the same best models

Part e)

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

```
## [1] 0.04835977
```

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




```
predict(lasso.mod, s=lambda, type="coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)            5.032793082
## poly(X, 10, raw = T)1  10.211994502
## poly(X, 10, raw = T)2  -3.092418697
## poly(X, 10, raw = T)3   1.708546779
## poly(X, 10, raw = T)4   .
## poly(X, 10, raw = T)5   0.049655684
## poly(X, 10, raw = T)6   .
## poly(X, 10, raw = T)7   0.000459821
## poly(X, 10, raw = T)8   .
## poly(X, 10, raw = T)9   .
## poly(X, 10, raw = T)10  .
```

Lasso regression selects the correct predictors: X, X^2 and X^3

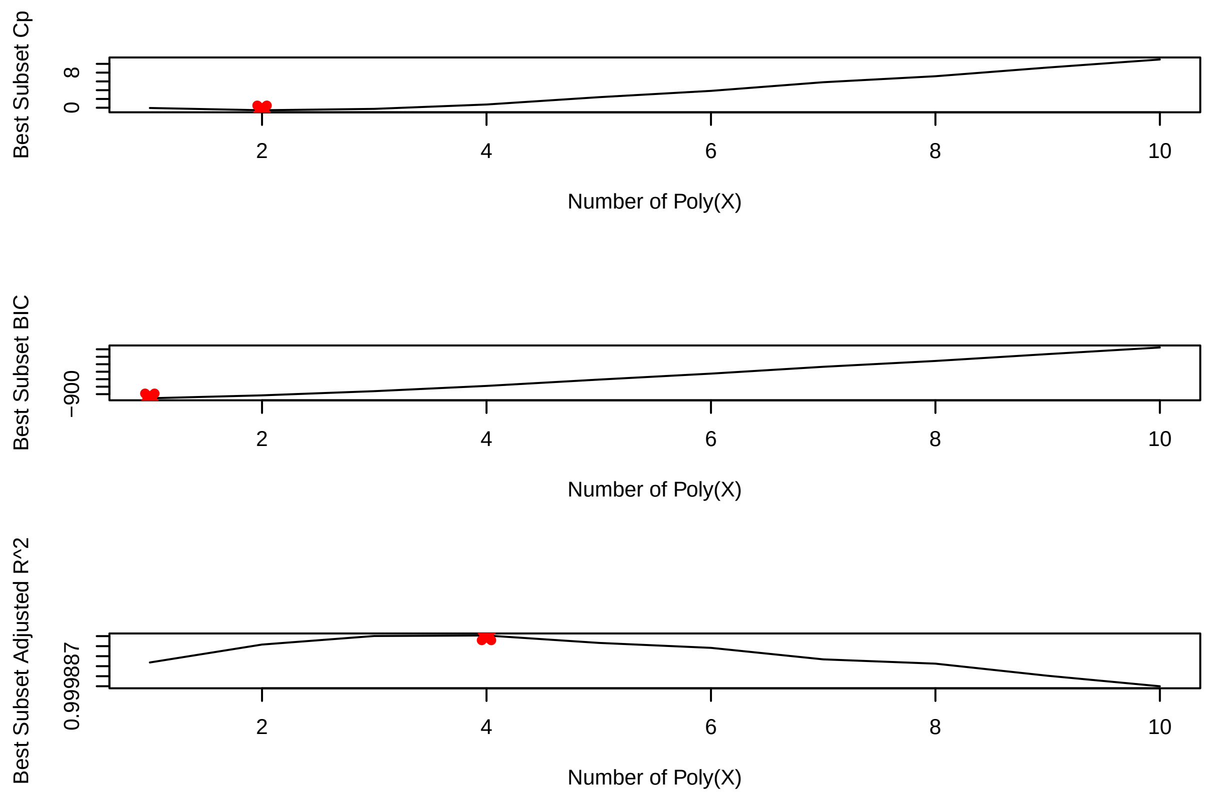
Part f)

```
Y2 <- 5 + 1.5*X^7 + eps

# best subset model selection
regfit.full <- regsubsets(Y2~poly(X,10,raw=T), data=data.frame(Y,X), nvmax=10)
par(mfrow=c(3,1))
(reg.summary <- summary(regfit.full))
```

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

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

```
min.cp <- which.min(reg.summary$cp)
plot(reg.summary$cp, xlab="Number of Poly(X)", ylab="Best Subset Cp", type="l")
points(min.cp, reg.summary$cp[min.cp], col="red", pch=4, lwd=5)
min.bic <- which.min(reg.summary$bic)
plot(reg.summary$bic, xlab="Number of Poly(X)", ylab="Best Subset BIC", type="l")
points(min.bic, reg.summary$bic[min.bic], col="red", pch=4, lwd=5)
min.adjr2 <- which.max(reg.summary$adjr2)
plot(reg.summary$adjr2, xlab="Number of Poly(X)", ylab="Best Subset Adjusted R^2", type="l")
points(min.adjr2, reg.summary$adjr2[min.adjr2], col="red", pch=4, lwd=5)
```




```
coef(regfit.full, min.cp)
```

```
##          (Intercept) poly(X, 10, raw = T)2 poly(X, 10, raw = T)7
##            5.0704904            -0.1417084             1.5015552
```

```r
coef(regfit.full, min.bic)
```

```
##           (Intercept) poly(X, 10, raw = T)7
##               4.95894               1.50077
```

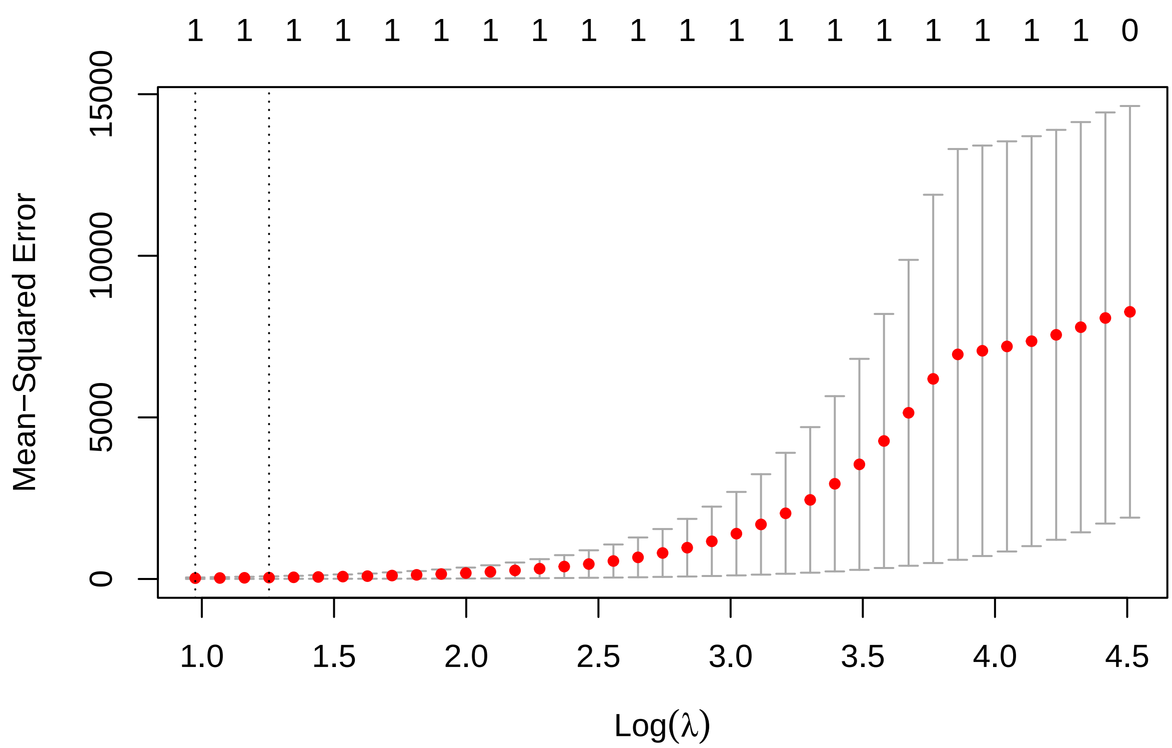
```r
coef(regfit.full, min.adjr2)
```

```
##           (Intercept) poly(X, 10, raw = T)1 poly(X, 10, raw = T)2
##             5.0762524             0.2914016            -0.1617671
## poly(X, 10, raw = T)3 poly(X, 10, raw = T)7
##            -0.2526527             1.5091338
```

```r
# lasso regression
xmat <- model.matrix(Y2~poly(X,10,raw=T))[,-1]
lasso.mod <- cv.glmnet(xmat, Y2, alpha=1)
(lambda <- lasso.mod$lambda.min)
```

```
## [1] 2.651535
```

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



```r
predict(lasso.mod, s=lambda, type="coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)            5.143574
## 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  1.457022
## poly(X, 10, raw = T)8  .
## poly(X, 10, raw = T)9  .
## poly(X, 10, raw = T)10 .
```

Lasso selects the correct model but best subset diagnostics indicate using 1 to 4 predictors

---

EXERCISE 9:

Part a)

```r
require(ISLR2)
data(College)
set.seed(1)
trainid <- sample(1:nrow(College), nrow(College)/2)
train <- College[trainid,]
test <- College[-trainid,]
str(College)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

Part b)

```
fit.lm <- lm(Apps~., data=train)
pred.lm <- predict(fit.lm, test)
(err.lm <- mean((test$Apps - pred.lm)^2))  # test error
```

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

Part c)

```
require(glmnet)
xmat.train <- model.matrix(Apps~., data=train)[,-1]
xmat.test <- model.matrix(Apps~., data=test)[,-1]
fit.ridge <- cv.glmnet(xmat.train, train$Apps, alpha=0)
(lambda <- fit.ridge$lambda.min)  # optimal lambda
```

```
## [1] 405.8404
```

```
pred.ridge <- predict(fit.ridge, s=lambda, newx=xmat.test)
(err.ridge <- mean((test$Apps - pred.ridge)^2))  # test error
```

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

Part d)

```
require(glmnet)
xmat.train <- model.matrix(Apps~., data=train)[,-1]
xmat.test <- model.matrix(Apps~., data=test)[,-1]
fit.lasso <- cv.glmnet(xmat.train, train$Apps, alpha=1)
(lambda <- fit.lasso$lambda.min)  # optimal lambda
```

```
## [1] 1.97344
```

```
pred.lasso <- predict(fit.lasso, s=lambda, newx=xmat.test)
(err.lasso <- mean((test$Apps - pred.lasso)^2))  # test error
```

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

```
coef.lasso <- predict(fit.lasso, type="coefficients", s=lambda)[1:ncol(College),]
coef.lasso[coef.lasso != 0]
```

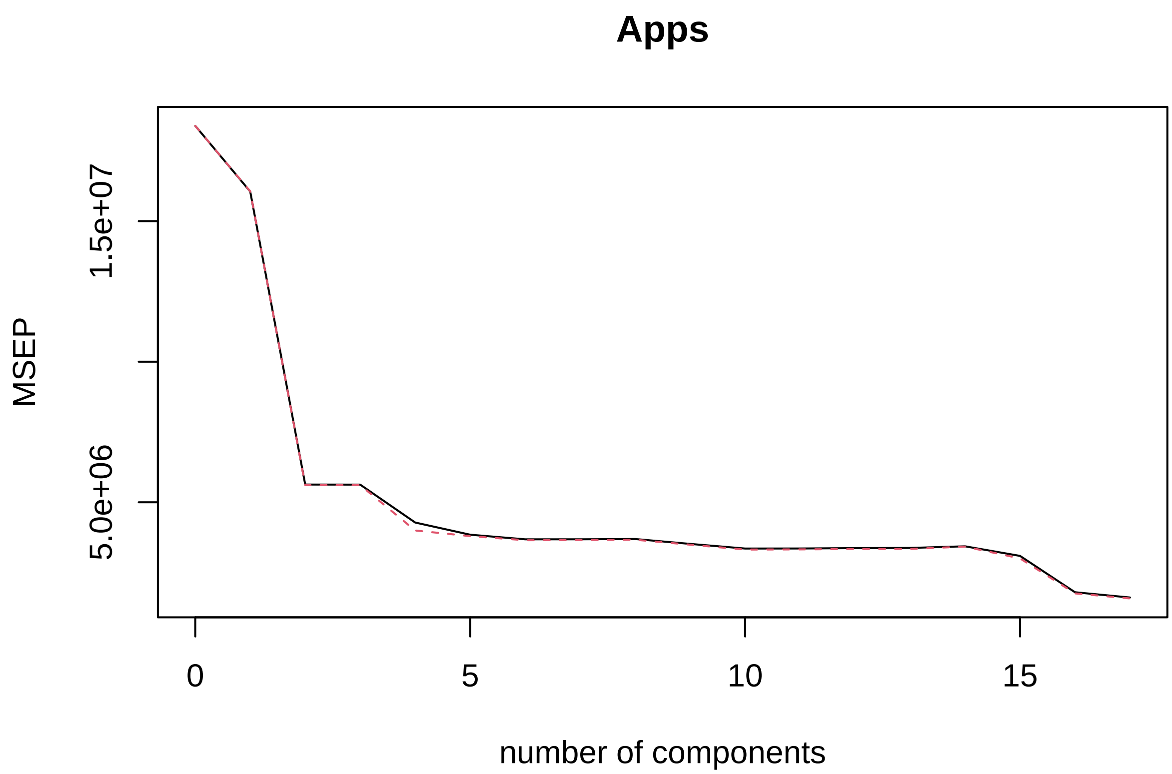
```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -7.688896e+02 -3.127034e+02  1.762718e+00 -1.318195e+00  6.482356e+01
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
## -2.081406e+01  7.119149e-02  1.246161e-02 -1.049091e-01  2.088305e-01
##         Books      Personal           PhD      Terminal     S.F.Ratio
##  2.926466e-01  3.955068e-03 -1.455463e+01  5.395858e+00  2.171398e+01
##   perc.alumni        Expend     Grad.Rate
##  5.088260e-01  4.824455e-02  7.036148e+00
```

```r
length(coef.lasso[coef.lasso != 0])
```

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

Part e)

```r
require(pls)
set.seed(1)
fit.pcr <- pcr(Apps~., data=train, scale=TRUE, validation="CV")
validationplot(fit.pcr, val.type="MSEP")
```




```r
summary(fit.pcr)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4288     4006     2373     2372     2069     1961     1919
## adjCV         4288     4007     2368     2369     1999     1948     1911
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
```

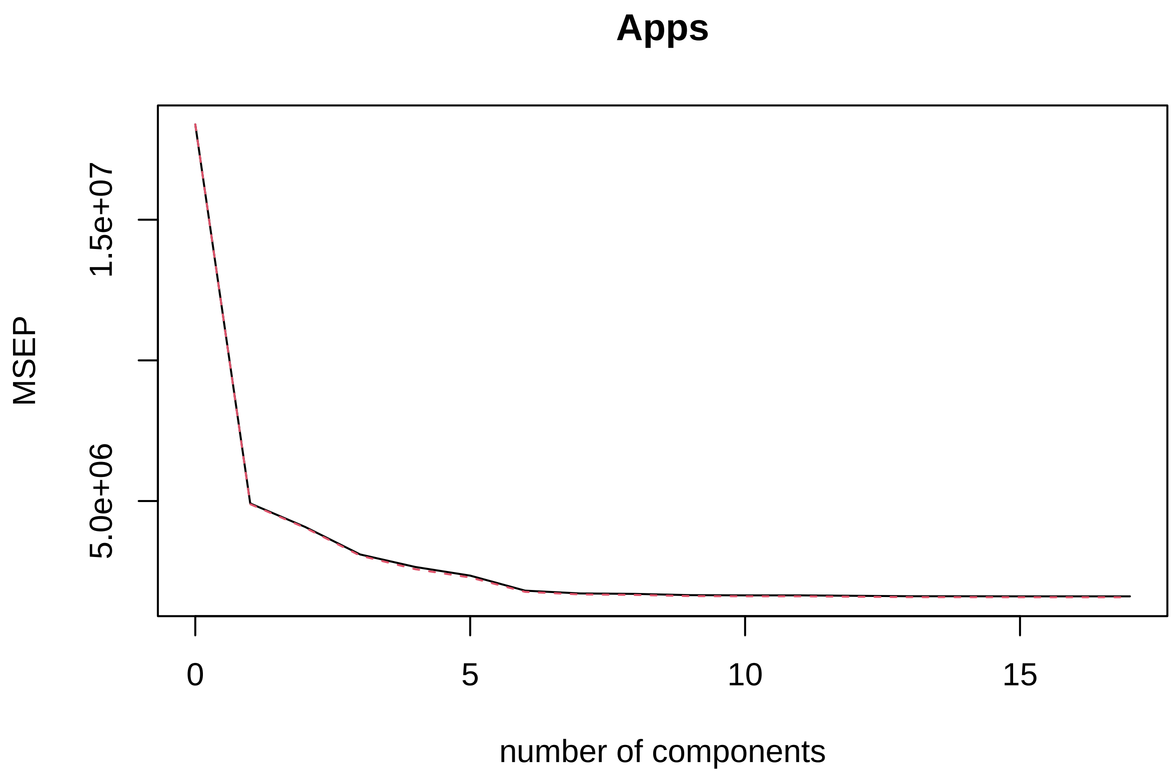
```
## CV         1919      1921      1876       1832       1832       1836       1837
## adjCV      1912      1915      1868       1821       1823       1827       1827
##        14 comps  15 comps  16 comps  17 comps
## CV          1853      1759      1341      1270
## adjCV       1850      1733      1326      1257
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       32.20    57.78    65.31    70.99    76.37    81.27     84.8    87.85
## Apps    13.44    70.93    71.07    79.87    81.15    82.25     82.3    82.33
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.62     92.91     94.98     96.74     97.79     98.72     99.42
## Apps    83.38     84.76     84.80     84.84     85.11     85.14     90.55
##       16 comps  17 comps
## X        99.88    100.00
## Apps     93.42     93.89
```

```
pred.pcr <- predict(fit.pcr, test, ncomp=16)  # min Cv at M=16
(err.pcr <- mean((test$Apps - pred.pcr)^2))  # test error
```

```
## [1] 1137877
```

Part f)

```
require(pls)
set.seed(1)
fit.pls <- plsr(Apps~., data=train, scale=TRUE, validation="CV")
validationplot(fit.pls, val.type="MSEP")
```




```
summary(fit.pls)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4288     2217     2019     1761     1630     1533     1347
## adjCV         4288     2211     2012     1749     1605     1510     1331
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1309     1303     1286      1283      1283      1277      1271
## adjCV     1296     1289     1273      1270      1270      1264      1258
##        14 comps  15 comps  16 comps  17 comps
## CV         1270      1270      1270      1270
## adjCV      1258      1257      1257      1257
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       27.21    50.73    63.06    65.52    70.20    74.20    78.62    80.81
## Apps    75.39    81.24    86.97    91.14    92.62    93.43    93.56    93.68
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.29     87.17     89.15     91.37     92.58     94.42     96.98
```

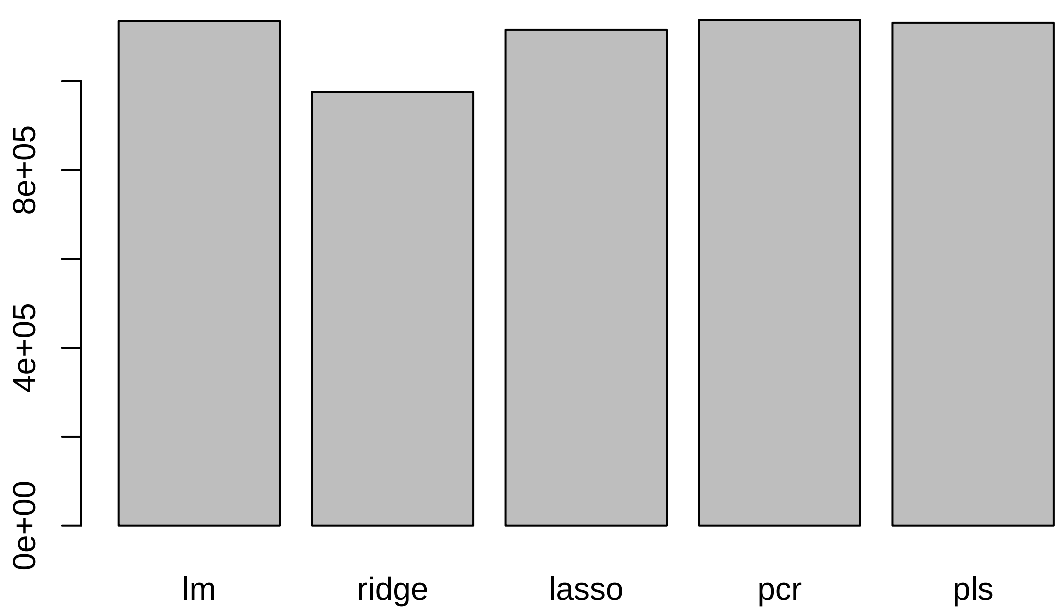
```
## Apps    93.76     93.79     93.83     93.86     93.88     93.89     93.89
##       16 comps  17 comps
## X        98.78    100.00
## Apps     93.89     93.89
```

```r
pred.pls <- predict(fit.pls, test, ncomp=10)  # min Cv at M=10
(err.pls <- mean((test$Apps - pred.pls)^2))  # test error
```

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

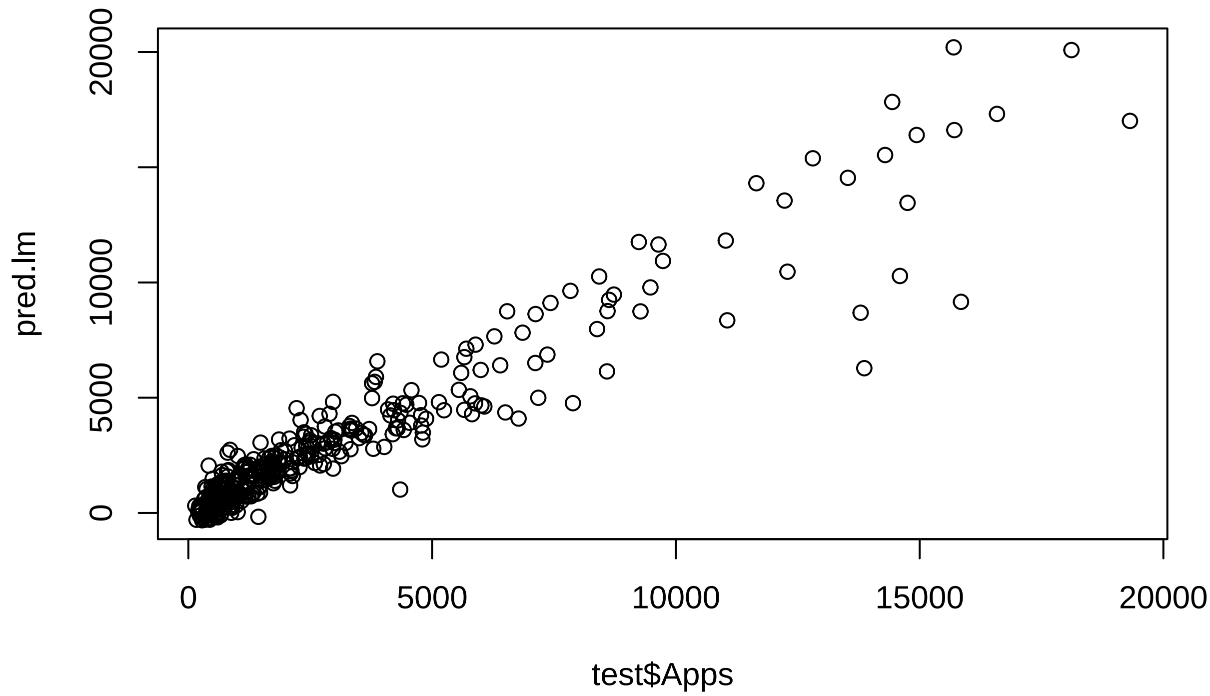
Part g)

```r
err.all <- c(err.lm, err.ridge, err.lasso, err.pcr, err.pls)
names(err.all) <- c("lm", "ridge", "lasso", "pcr", "pls")
barplot(err.all )
```



The test errors aren't much different. The ridge and lasso seem to perform slightly better while the PCR/PLS don't show any improvement from the full linear regression model.

```r
plot(test$Apps, pred.lm)
```



EXERCISE 10:

Part a)

```
set.seed(1)
eps <- rnorm(1000)
xmat <- matrix(rnorm(1000*20), ncol=20)
betas <- sample(-5:5, 20, replace=TRUE)
betas[c(3,6,7,10,13,17)] <- 0
betas
```

```
##  [1] -3  5  0 -5  2  0  0  4  5  0  4  1  0  5 -3  4  0  4 -2  1
```

```
y <- xmat %*% betas + eps
```

Part b)

```
set.seed(1)
trainid <- sample(1:1000, 100, replace=FALSE)
xmat.train <- xmat[trainid,]
xmat.test <- xmat[-trainid,]
y.train <- y[trainid,]
```

```
y.test <- y[-trainid,]
train <- data.frame(y=y.train, xmat.train)
test <- data.frame(y=y.test, xmat.test)
```

Part c)

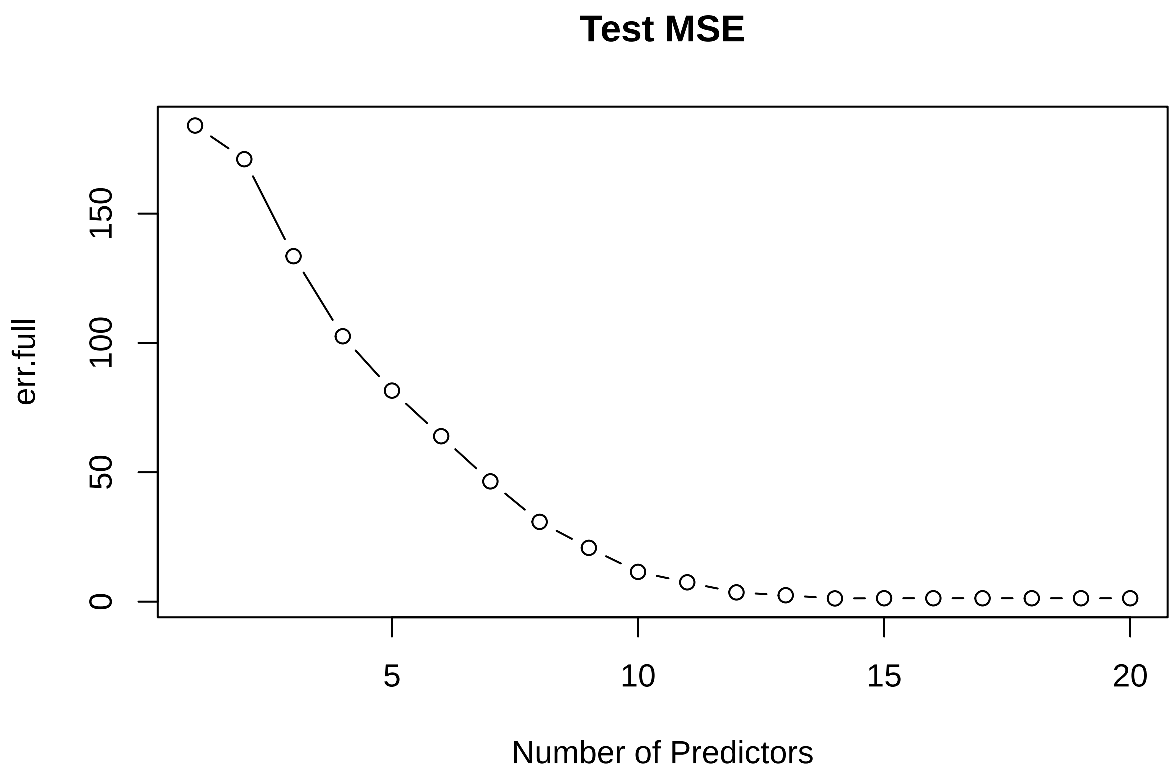
```
require(leaps)

# predict function from chapter 6 labs
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.full <- regsubsets(y~., data=train, nvmax=20)
err.full <- rep(NA, 20)
for(i in 1:20) {
  pred.full <- predict(regfit.full, train, id=i)
  err.full[i] <- mean((train$y - pred.full)^2)
}
plot(1:20, err.full, type="b", main="Training MSE", xlab="Number of Predictors")
```

```r
which.min(err.full)  # min for train error should be at max pred count
```

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

Part d)

```r
err.full <- rep(NA, 20)
for(i in 1:20) {
  pred.full <- predict(regfit.full, test, id=i)
  err.full[i] <- mean((test$y - pred.full)^2)
}
plot(1:20, err.full, type="b", main="Test MSE", xlab="Number of Predictors")
```



Part e)

```r
which.min(err.full)  # optimal number of predictors from best subset
```

```
## [1] 14
```

Part f)

```r
(coef.best <- coef(regfit.full, id=which.min(err.full)))
```

```
##   (Intercept)            X1            X2            X4            X5            X8
##   0.002830717 -2.971273645   5.016860247 -5.032930569   2.011190672   4.043424774
##            X9           X11           X12           X14           X15           X16
##   5.166187040   4.002964100   1.038961952   4.950846519 -2.815120305   3.816000415
##           X18           X19           X20
##   4.146759315 -1.943471497   1.199621119
```

```r
betas[betas != 0]
```

```
##  [1] -3  5 -5  2  4  5  4  1  5 -3  4  4 -2  1
```

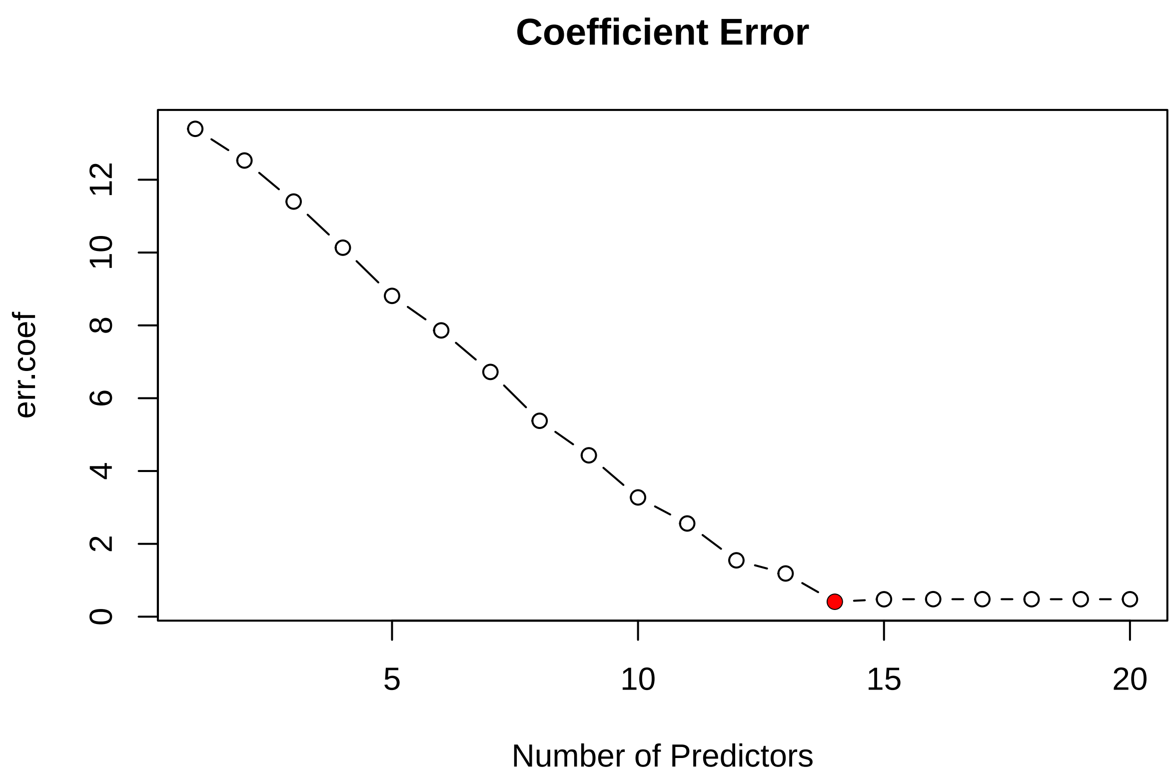
```r
names(betas) <- paste0("X", 1:20)
merge(data.frame(beta=names(betas),betas), data.frame(beta=names(coef.best),coef.best), all.x=T, sort=F
```

```
##    beta betas coef.best
## 1    X1    -3 -2.971274
## 2    X2     5  5.016860
## 3    X4    -5 -5.032931
## 4    X5     2  2.011191
## 5    X8     4  4.043425
## 6    X9     5  5.166187
## 7   X11     4  4.002964
## 8   X12     1  1.038962
## 9   X14     5  4.950847
## 10  X15    -3 -2.815120
## 11  X16     4  3.816000
## 12  X18     4  4.146759
## 13  X19    -2 -1.943471
## 14  X20     1  1.199621
## 15  X13     0        NA
## 16   X6     0        NA
## 17   X3     0        NA
## 18  X17     0        NA
## 19  X10     0        NA
## 20   X7     0        NA
```

The best subset model selected all the correct predictors

Part g)

```r
err.coef <- rep(NA, 20)
for(i in 1:20) {
  coef.i <- coef(regfit.full, id=i)
  df.err <- merge(data.frame(beta=names(betas),betas), data.frame(beta=names(coef.i),coef.i), all.x=T)
  df.err[is.na(df.err[,3]),3] <- 0
  err.coef[i] <- sqrt(sum((df.err[,2] - df.err[,3])^2))
}
plot(1:20, err.coef, type="b", main="Coefficient Error", xlab="Number of Predictors")
points(which.min(err.coef), err.coef[which.min(err.coef)], col="red", pch=16)
```



The coefficient error plot shows a very similar plot to the test error plot

---

EXERCISE 11:

Part a)

```
require(leaps)   # forward and backward selection
require(glmnet)  # ridge and lasso
require(MASS)    # Boston data set
data(Boston)

# split data into training and test sets
set.seed(1)
trainid <- sample(1:nrow(Boston), nrow(Boston)/2)
train <- Boston[trainid,]
test <- Boston[-trainid,]
xmat.train <- model.matrix(crim~., data=train)[,-1]
xmat.test <- model.matrix(crim~., data=test)[,-1]
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
```

```
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```r
# ridge regression model
fit.ridge <- cv.glmnet(xmat.train, train$crim, alpha=0)
(lambda <- fit.ridge$lambda.min)  # optimal lambda
```

```
## [1] 0.5919159
```

```r
pred.ridge <- predict(fit.ridge, s=lambda, newx=xmat.test)
(err.ridge <- mean((test$crim - pred.ridge)^2))  # test error
```

```
## [1] 40.92777
```

```r
predict(fit.ridge, s=lambda, type="coefficients")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 14.702068319
## zn           0.035283661
## indus       -0.119976460
## chas        -0.616052143
## nox         -5.629356997
## rm           0.228001209
## age         -0.004314219
## dis         -0.768474303
## rad          0.434236779
## tax          0.003139323
## ptratio     -0.298647697
## black       -0.013823430
## lstat        0.262179351
## medv        -0.146028474
```

```r
# lasso regression model
fit.lasso <- cv.glmnet(xmat.train, train$crim, alpha=1)
(lambda <- fit.lasso$lambda.min)  # optimal lambda
```

```
## [1] 0.06805595
```

```r
pred.lasso <- predict(fit.lasso, s=lambda, newx=xmat.test)
(err.lasso <- mean((test$crim - pred.lasso)^2))  # test error
```

```
## [1] 40.90173
```

```r
predict(fit.lasso, s=lambda, type="coefficients")
```

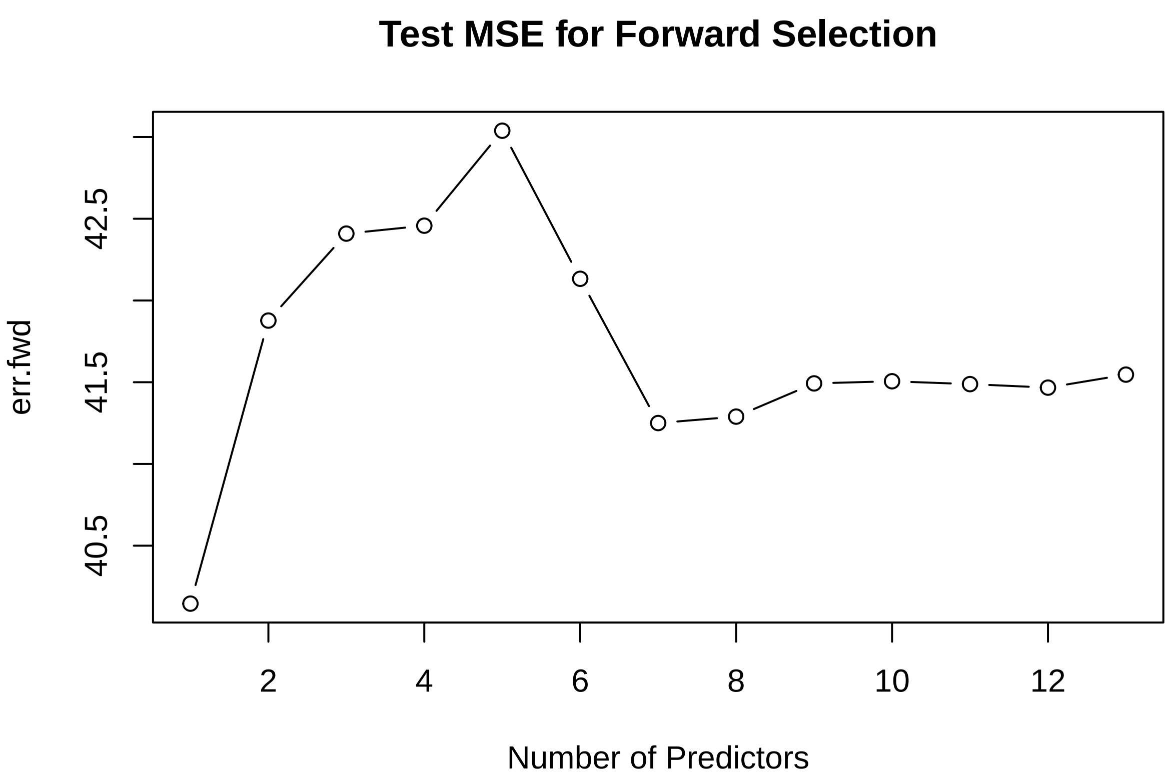
```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 17.65005513
## zn           0.03516255
## indus       -0.11838293
## chas        -0.43135144
## nox         -7.19578180
## rm           0.04271112
## age          .
## dis         -0.76801501
## rad          0.52430211
## tax          .
## ptratio     -0.35072332
## black       -0.01307754
## lstat        0.25559458
## medv        -0.14805010
```

```r
# predict function from chapter 6 labs
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
}

# forward selection
fit.fwd <- regsubsets(crim~., data=train, nvmax=ncol(Boston)-1)
(fwd.summary <- summary(fit.fwd))
```

```
## Subset selection object
## Call: regsubsets.formula(crim ~ ., data = train, nvmax = ncol(Boston) -
##     1)
## 13 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 13
```

```
## Selection Algorithm: exhaustive
##           zn  indus chas nox rm  age dis rad tax ptratio black lstat medv
## 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 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"   "*"
```

```
err.fwd <- rep(NA, ncol(Boston)-1)
for(i in 1:(ncol(Boston)-1)) {
  pred.fwd <- predict(fit.fwd, test, id=i)
  err.fwd[i] <- mean((test$crim - pred.fwd)^2)
}
plot(err.fwd, type="b", main="Test MSE for Forward Selection", xlab="Number of Predictors")
```



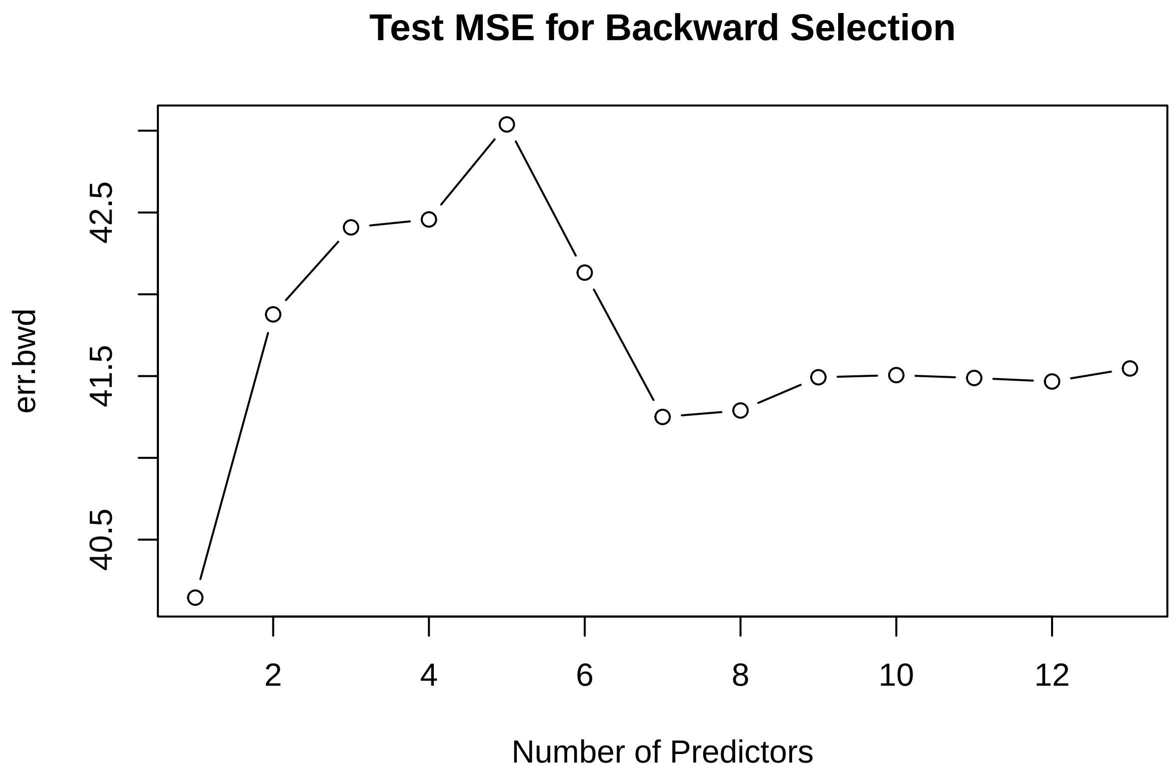

```
which.min(err.fwd)
```

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

```r
# backward selection
fit.bwd <- regsubsets(crim~., data=train, nvmax=ncol(Boston)-1)
(bwd.summary <- summary(fit.bwd))
```

```
## Subset selection object
## Call: regsubsets.formula(crim ~ ., data = train, nvmax = ncol(Boston) -
##     1)
## 13 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 13
## Selection Algorithm: exhaustive
##           zn  indus chas nox rm  age dis rad tax ptratio black lstat medv
## 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 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"   "*"
```

```r
err.bwd <- rep(NA, ncol(Boston)-1)
for(i in 1:(ncol(Boston)-1)) {
  pred.bwd <- predict(fit.bwd, test, id=i)
  err.bwd[i] <- mean((test$crim - pred.bwd)^2)
}
plot(err.bwd, type="b", main="Test MSE for Backward Selection", xlab="Number of Predictors")
```

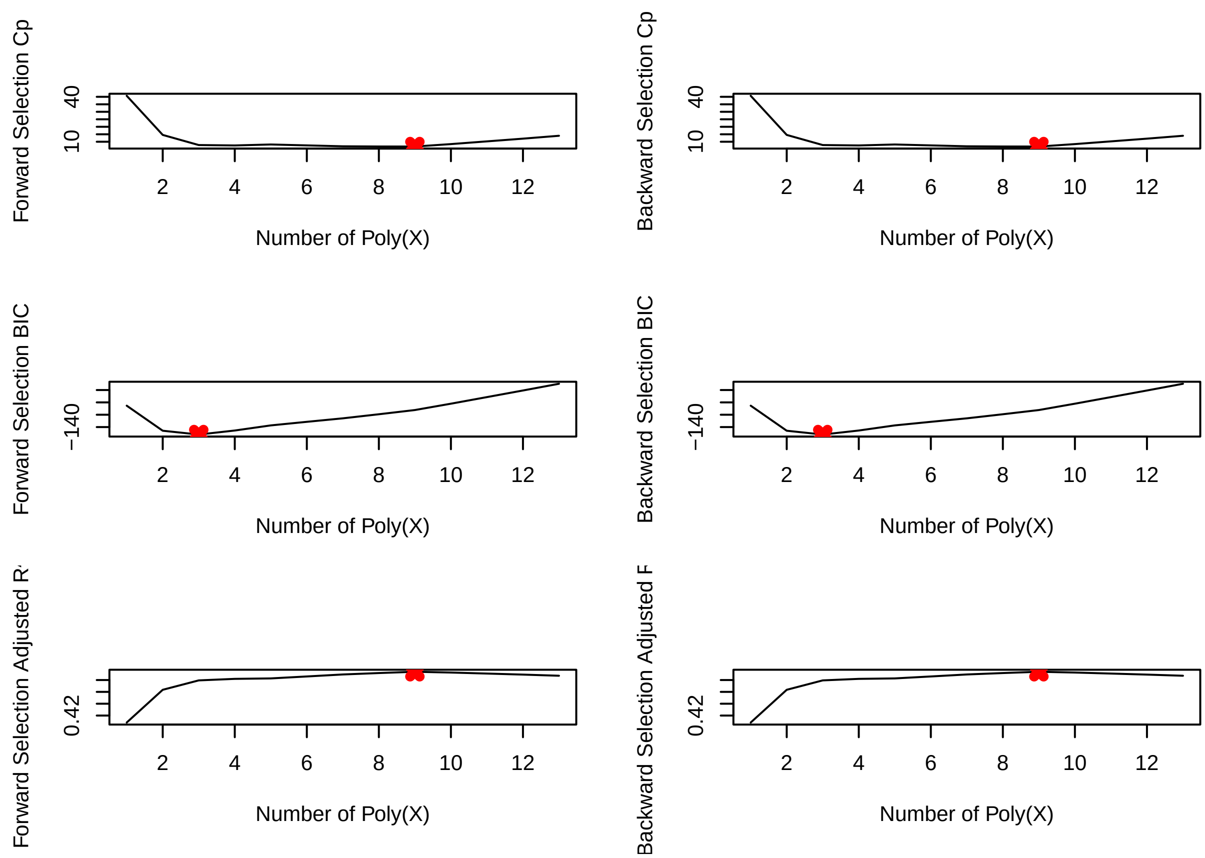



```
which.min(err.bwd)
```

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

```
par(mfrow=c(3,2))

min.cp <- which.min(fwd.summary$cp)
plot(fwd.summary$cp, xlab="Number of Poly(X)", ylab="Forward Selection Cp", type="l")
points(min.cp, fwd.summary$cp[min.cp], col="red", pch=4, lwd=5)

min.cp <- which.min(bwd.summary$cp)
plot(bwd.summary$cp, xlab="Number of Poly(X)", ylab="Backward Selection Cp", type="l")
points(min.cp, bwd.summary$cp[min.cp], col="red", pch=4, lwd=5)

min.bic <- which.min(fwd.summary$bic)
plot(fwd.summary$bic, xlab="Number of Poly(X)", ylab="Forward Selection BIC", type="l")
points(min.bic, fwd.summary$bic[min.bic], col="red", pch=4, lwd=5)

min.bic <- which.min(bwd.summary$bic)
plot(bwd.summary$bic, xlab="Number of Poly(X)", ylab="Backward Selection BIC", type="l")
points(min.bic, bwd.summary$bic[min.bic], col="red", pch=4, lwd=5)

min.adjr2 <- which.max(fwd.summary$adjr2)
plot(fwd.summary$adjr2, xlab="Number of Poly(X)", ylab="Forward Selection Adjusted R^2", type="l")
points(min.adjr2, fwd.summary$adjr2[min.adjr2], col="red", pch=4, lwd=5)
```

```r
min.adjr2 <- which.max(bwd.summary$adjr2)
plot(bwd.summary$adjr2, xlab="Number of Poly(X)", ylab="Backward Selection Adjusted R^2", type="l")
points(min.adjr2, bwd.summary$adjr2[min.adjr2], col="red", pch=4, lwd=5)
```



Part b)

```r
err.ridge
```

```
## [1] 40.92777
```

```r
err.lasso
```

```
## [1] 40.90173
```

```r
err.fwd
```

```
##  [1] 40.14557 41.87706 42.40901 42.45745 43.03836 42.13258 41.25016 41.28957
##  [9] 41.49271 41.50577 41.48839 41.46692 41.54639
```

```r
err.bwd
```

```
##  [1] 40.14557 41.87706 42.40901 42.45745 43.03836 42.13258 41.25016 41.28957
##  [9] 41.49271 41.50577 41.48839 41.46692 41.54639
```

Probably choose the lasso model because its test MSE is close to best and eliminates some predictors to reduce model complexity

Part c)

No because not all the predictors add much value to the model