Statistical Learning for Biological and Information Systems Problem Set #2

Chenye Yang cy2540@columbia.edu

October 11, 2019

P1

(a)

Ans:
Ridge regression optimization problem:

$$\begin{aligned} \min \quad & RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \\ &= \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \\ &= (y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2 + \lambda\beta_1^2 + \lambda\beta_2^2 \end{aligned} \tag{1}$$

(b)

Ans:
From the statement, we know that $x_{11} = x_{12} = a,\ x_{21} = x_{22} = b,\ a + b = 0,\ \beta_0 = 0.$
The equation (1) write as:

$$\begin{aligned} \min \quad & [y_1 - (\beta_1 + \beta_2)a]^2 + [y_2 - (\beta_1 + \beta_2)b]^2 + \lambda(\beta_1^2 + \beta_2^2) \\ =& y_1^2 + y_2^2 - (\beta_1 + \beta_2)(2y_1 a + 2y_2 b) + (a^2 + b^2)(\beta_1 + \beta_2)^2 + \lambda(\beta_1^2 + \beta_2^2) \\ =& f(\beta_1, \beta_2) \end{aligned} \tag{2}$$

To solve this optimization problem, let:

$$\begin{aligned} \frac{\partial f(\beta_1, \beta_2)}{\partial \beta_1} = 0 \\ \frac{\partial f(\beta_1, \beta_2)}{\partial \beta_2} = 0 \end{aligned} \tag{3}$$

We have:

$$\begin{aligned} -(2y_1 a + 2y_2 b) + 2(a^2 + b^2)(\hat{\beta}_1 + \hat{\beta}_2) + 2\lambda\hat{\beta}_1 = 0 \\ -(2y_1 a + 2y_2 b) + 2(a^2 + b^2)(\hat{\beta}_1 + \hat{\beta}_2) + 2\lambda\hat{\beta}_2 = 0 \end{aligned} \tag{4}$$

Therefore,

$$2\lambda\hat{\beta}_1 = 2\lambda\hat{\beta}_2 \tag{5}$$

Because in ridge regression, $\lambda > 0$. Thus:

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

(c)

Ans:
Lasso optimization problem:

$$\begin{aligned} \min \quad & RSS + \lambda\sum_{j=1}^{p}|\beta_j| \\ &= \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| \\ &= (y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2 + \lambda|\beta_1| + \lambda|\beta_2| \end{aligned} \tag{6}$$

(d)

Ans:
Same as **P1 (b)**, we write the lasso optimization equation (6) as:

$$\begin{aligned} \min \quad & y_1^2 + y_2^2 - (\beta_1+\beta_2)(2y_1a + 2y_2b) + (a^2+b^2)(\beta_1+\beta_2)^2 + \lambda(|\beta_1| + |\beta_2|) \\ =& g(\beta_1, \beta_2) \end{aligned} \tag{7}$$

Then

$$\begin{aligned} \frac{\partial g(\beta_1,\beta_2)}{\partial\beta_1} &= \begin{cases} -(2y_1a + 2y_2b) + 2(a^2+b^2)(\hat{\beta}_1+\hat{\beta}_2) + \lambda, & \hat{\beta}_1 > 0 \\ -(2y_1a + 2y_2b) + 2(a^2+b^2)(\hat{\beta}_1+\hat{\beta}_2) - \lambda, & \hat{\beta}_1 < 0 \end{cases} \\ \frac{\partial g(\beta_1,\beta_2)}{\partial\beta_2} &= \begin{cases} -(2y_1a + 2y_2b) + 2(a^2+b^2)(\hat{\beta}_1+\hat{\beta}_2) + \lambda, & \hat{\beta}_2 > 0 \\ -(2y_1a + 2y_2b) + 2(a^2+b^2)(\hat{\beta}_1+\hat{\beta}_2) - \lambda, & \hat{\beta}_2 < 0 \end{cases} \end{aligned} \tag{8}$$

(1) When $\hat{\beta}_1, \hat{\beta}_2 > 0$

$$\begin{cases} \dfrac{\partial g(\beta_1,\beta_2)}{\partial\beta_1} = 0 \\ \dfrac{\partial g(\beta_1,\beta_2)}{\partial\beta_2} = 0 \end{cases}$$

$\Rightarrow$

$$-(2y_1a + 2y_2b) + 2(a^2+b^2)(\hat{\beta}_1+\hat{\beta}_2) + \lambda = 0$$

$\Rightarrow$

$$\begin{aligned} \hat{\beta}_1 + \hat{\beta}_2 &= \frac{(2y_1a + 2y_2b) - \lambda}{2(a^2+b^2)} \\ &= \frac{y_1a + y_2b - \lambda/2}{(a+b)^2 - 2ab} \\ &= -\frac{y_1}{2b} - \frac{y_2}{2a} + \frac{\lambda}{4ab} \end{aligned} \tag{9}$$

Therefore, any $\hat{\beta}_1, \hat{\beta}_2$ satisfies equation (9) and $\hat{\beta}_1, \hat{\beta}_2 > 0$ will be a solution to the lasso optimization problem in **P1 (c)**.

(2) Similarly, when $\hat{\beta}_1, \hat{\beta}_2 < 0$

$$\begin{cases} \dfrac{\partial g(\beta_1, \beta_2)}{\partial \beta_1} = 0 \\ \dfrac{\partial g(\beta_1, \beta_2)}{\partial \beta_2} = 0 \end{cases}$$

$\Rightarrow$

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

$\Rightarrow$

$$\hat{\beta}_1 + \hat{\beta}_2 = -\frac{y_1}{2b} - \frac{y_2}{2a} - \frac{\lambda}{4ab} \tag{10}$$

Therefore, any $\hat{\beta}_1, \hat{\beta}_2$ satisfies equation (10) and $\hat{\beta}_1, \hat{\beta}_2 < 0$ will be a solution to the lasso optimization problem in **P1 (c)**.

(3) When $\hat{\beta}_1 < 0,\ \hat{\beta}_2 > 0$ or $\hat{\beta}_1 > 0,\ \hat{\beta}_2 < 0$

$$\begin{cases} \dfrac{\partial g(\beta_1, \beta_2)}{\partial \beta_1} = 0 \\ \dfrac{\partial g(\beta_1, \beta_2)}{\partial \beta_2} = 0 \end{cases}$$

$\Rightarrow$

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

$\Rightarrow \lambda = 0$, which is not Lasso.
(4) When $\hat{\beta}_1 = 0$
Equation (7) changes to

$$\min \quad y_1^2 + y_2^2 - \beta_2(2y_1 a + 2y_2 b) + (a^2 + b^2)\beta_2^2 + \lambda(|\beta_2|) = g(\beta_2)$$

$$\frac{dg(\beta_2)}{d\beta_2} = -(2y_1 a + 2y_2 b) + 2(a^2 + b^2)\beta_2 \pm \lambda = 0$$

$\Rightarrow$

$$\hat{\beta}_2 = -\frac{y_1}{2b} - \frac{y_2}{2a} \pm \frac{\lambda}{4ab}, \quad \hat{\beta}_2 \gtrless 0 \tag{11}$$

Therefore, $\hat{\beta}_1 = 0$ and $\hat{\beta}_2$ satisfies equation (11) will be a solution to the lasso optimization problem in **P1 (c)**.

(5) When $\hat{\beta}_2 = 0$
Equation (7) changes to

$$\min \quad y_1^2 + y_2^2 - \beta_1(2y_1 a + 2y_2 b) + (a^2 + b^2)\beta_1^2 + \lambda(|\beta_1|) = g(\beta_1)$$

$$\frac{dg(\beta_1)}{d\beta_1} = -(2y_1 a + 2y_2 b) + 2(a^2 + b^2)\beta_1 \pm \lambda = 0$$

$\Rightarrow$

$$\hat{\beta}_1 = -\frac{y_1}{2b} - \frac{y_2}{2a} \pm \frac{\lambda}{4ab}, \quad \hat{\beta}_1 \gtrless 0 \tag{12}$$

Therefore, $\hat{\beta}_2 = 0$ and $\hat{\beta}_1$ satisfies equation (12) will be a solution to the lasso optimization problem in **P1 (c)**.

(6) When $\hat{\beta}_1 = \hat{\beta}_2 = 0$
Equation (7) changes to $\min y_1^2 + y_2^2$, which means the regression function is $y = 0$.

In summary, in this setting, the lasso coefficients $\hat{\beta}_1, \hat{\beta}_2$ are not unique.

P2

(a)

Ans:
When $p = 1$, equation (1) in the question changes to:

$$\begin{aligned} &\min \sum_{j=1}^{1} (y_j - \beta_j)^2 + \lambda \sum_{j=1}^{1} \beta_j^2 \\ =&(y_1 - \beta_1)^2 + \lambda \beta_1^2 = f(\beta_1) \end{aligned}$$

Let $y_1 = 1, \lambda = 0.5$, plot the ridge regression optimization function as black line in Figure 1. Also plot the ridge regression estimates equation (3) in question, $\hat{\beta}_1^R = y_1/(1+\lambda)$, as a vertical red line in same figure. Two lines intersect at the small red circle, which is exactly on the lowest point of curve. Considering the lowest point of curve means the solve of ridge regression optimization problem, the estimated coefficient point. Therefore, the plot confirms that (1) is solved by (3)
Result:

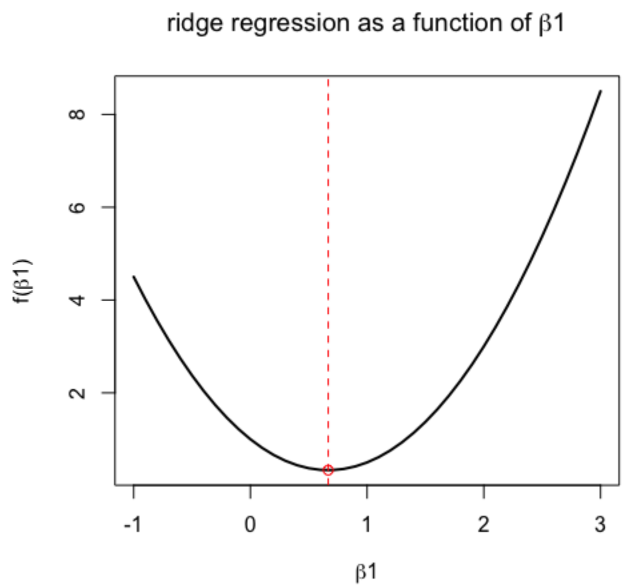


Figure 1: Ridge regression optimization function over one coefficient

Code:

```
# (a)
y1 = 1
lambda = 0.5
beta1 = seq(from=-1, to=3, by=0.1)
f = (y1 - beta1)^2 + lambda*(beta1^2)
```

```
plot(beta1, f, type="l", lwd=2, xlab=expression(paste(beta, '1')),
    ylab=expression(paste('f(',beta, '1)')),main=expression(paste("
    ridge regression as a function of ",beta, '1')))
abline(v=y1/(1+lambda), lty=2, col="red")
beta1 = y1/(1+lambda)
f = (y1 - beta1)^2 + lambda*(beta1^2)
points(beta1, f, col="red")
```

(b)

Ans:
When $p = 1$, equation (2) in the question changes to:

$$\begin{aligned} &\min \sum_{j=1}^{1}(y_j - \beta_j)^2 + \lambda \sum_{j=1}^{1} |\beta_j| \\ =&(y_1 - \beta_1)^2 + \lambda|\beta_1| = g(\beta_1) \end{aligned}$$

For $y_1 > \lambda/2$, let $y_1 = 1, \lambda = 0.5$, plot the lasso optimization function as black line in Fig.2. Also plot the lasso estimates equation (4) in question, $\hat{\beta}_1^L = y_j - \lambda/2$, as a vertical red line in same figure.

For $y_1 < -\lambda/2$, let $y_1 = -1, \lambda = 0.5$, plot the lasso optimization function as black line in Fig.3. Also plot the lasso estimates equation (4) in question, $\hat{\beta}_1^L = y_j + \lambda/2$, as a vertical red line in same figure.

For $|y_1| \leq \lambda/2$, let $y_1 = 0.1, \lambda = 0.5$, plot the lasso optimization function as black line in Fig.4. Also plot the lasso estimates equation (4) in question, $\hat{\beta}_1^L = 0$, as a vertical red line in same figure.

In these figures, two lines intersect at the small red circle, which is exactly on the lowest point of curve. Considering the lowest point of curve means the solve of lasso optimization problem, the estimated coefficient point. In summary, the plot confirms that (2) is solved by (4)
Code:

```
# (b)
# y1 > lambda/2
y1 = 1
lambda = 0.5
beta1 = seq(from=-1, to=3, by=0.1)
g = (y1 - beta1)^2 + lambda*(abs(beta1))
plot(beta1, g, type="l", lwd=2, xlab=expression(paste(beta, '1')),
    ylab=expression(paste('g(',beta, '1)')),main=expression(paste("
    lasso as a function of ",beta, '1')))
abline(v=y1-lambda/2, lty=2, col="red")
beta1 = y1-lambda/2
g = (y1 - beta1)^2 + lambda*(abs(beta1))
points(beta1, g, col="red")
# y1 < -lambda/2
```

```
y1 = -1
lambda = 0.5
beta1 = seq(from=-3, to=1, by=0.1)
g = (y1 - beta1)^2 + lambda*(abs(beta1))
plot(beta1, g, type="l", lwd=2, xlab=expression(paste(beta, '1')),
    ylab=expression(paste('g(',beta, '1)')),main=expression(paste("
    lasso as a function of ",beta, '1')))
abline(v=y1+lambda/2, lty=2, col="red")
beta1 = y1+lambda/2
g = (y1 - beta1)^2 + lambda*(abs(beta1))
points(beta1, g, col="red")
# |y1| <= lambda/2
y1 = 0.1
lambda = 0.5
beta1 = seq(from=-2, to=2, by=0.1)
g = (y1 - beta1)^2 + lambda*(abs(beta1))
plot(beta1, g, type="l", lwd=2, xlab=expression(paste(beta, '1')),
    ylab=expression(paste('g(',beta, '1)')),main=expression(paste("
    lasso as a function of ",beta, '1')))
abline(v=0, lty=2, col="red")
beta1 = 0
g = (y1 - beta1)^2 + lambda*(abs(beta1))
points(beta1, g, col="red")
```

Result:

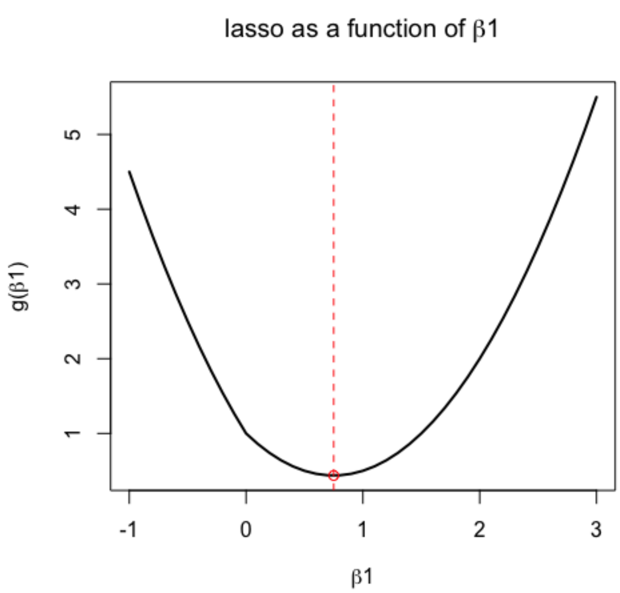


Figure 2: Lasso optimization function over one coefficient, $y_1 > \lambda/2$

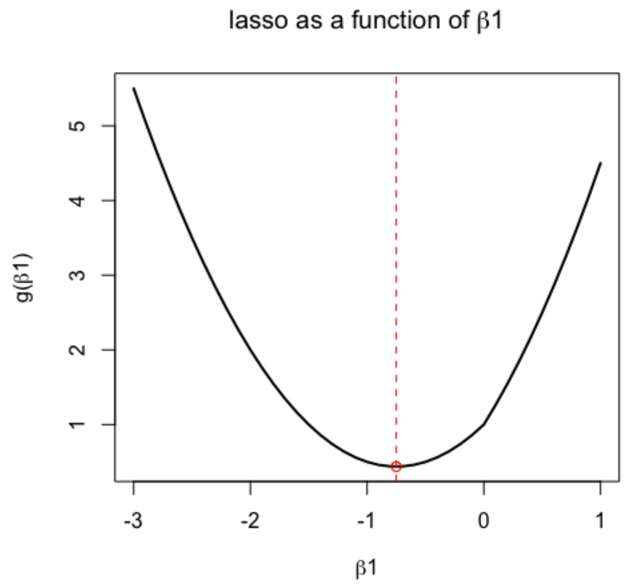


Figure 3: Lasso optimization function over one coefficient, $y_1 < -\lambda/2$

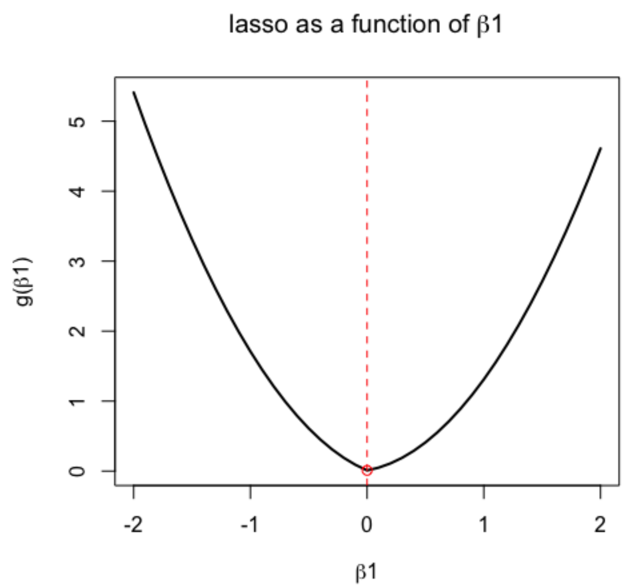


Figure 4: Lasso optimization function over one coefficient, $|y_1| \leq \lambda/2$

P3

(a)

Ans:
Because $\epsilon_1, \epsilon_2, ..., \epsilon_n$ are independent and identically distributed from a $N(0, \sigma^2)$ distribution, i.e., the data points are normally distributed about the regression line, the likelihood of the data can be write as follow:

$$\mathcal{L}(\boldsymbol{\beta}|\boldsymbol{y}) \coloneqq P(\boldsymbol{y}|\boldsymbol{\beta}) = \prod_{i=1}^{n} P_Y(y_i|\boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_i-(\beta_0+\sum_{j=1}^{p} x_{ij}\beta_j)]^2}{2\sigma^2}} \tag{13}$$

(b)

Ans:
Because $\beta_1, \beta_2, ..., \beta_p$ are independent and identically distributed according to a double-exponential distribution, and $p(\boldsymbol{\beta}) = \frac{1}{2b} exp(-|\boldsymbol{\beta}|/b),\ |\boldsymbol{\beta}| = \sum_{j=1}^{p} |\beta_j|$, the posterior for $\boldsymbol{\beta}$ in this setting is:

$$\begin{aligned} P(\boldsymbol{\beta}|\boldsymbol{y}) &= \frac{P(\boldsymbol{y}|\boldsymbol{\beta})P(\boldsymbol{\beta})}{P(\boldsymbol{y})} \\ &= \frac{1}{\prod_{i=1}^{n} P(y_i)} \times \frac{1}{2b} e^{-\sum_{j=1}^{p} |\beta_j|/b} \times \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_i-(\beta_0+\sum_{j=1}^{p} x_{ij}\beta_j)]^2}{2\sigma^2}} \end{aligned} \tag{14}$$

(c)

Ans:
From the view of maximum a posterior probability estimate (usually abbreviated by MAP), we should find the maximum of the posterior in equation (14) in order to find the best estimate of predictors $\boldsymbol{\beta}$:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{\text{MAP}} &= \arg\max_{\boldsymbol{\beta}} P(\boldsymbol{\beta}|\boldsymbol{y}) \\ &= \arg\max_{\boldsymbol{\beta}} \frac{1}{\prod_{i=1}^{n} P(y_i)} \times \frac{1}{2b} e^{-\sum_{j=1}^{p} |\beta_j|/b} \times \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_i-(\beta_0+\sum_{j=1}^{p} x_{ij}\beta_j)]^2}{2\sigma^2}} \\ &= \arg\max_{\boldsymbol{\beta}} [\log \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_i-(\beta_0+\sum_{j=1}^{p} x_{ij}\beta_j)]^2}{2\sigma^2}} + \log \frac{1}{2b} e^{-\sum_{j=1}^{p} |\beta_j|/b}] \\ &= \arg\max_{\boldsymbol{\beta}} [\sum_{i=1}^{n} -\frac{[y_i-(\beta_0+\sum_{j=1}^{p} x_{ij}\beta_j)]^2}{2\sigma^2} - \sum_{j=1}^{p} |\beta_j|/b] \\ &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^2} [\sum_{i=1}^{n} [y_i - (\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j)]^2 + \frac{2\sigma^2}{b} \sum_{j=1}^{p} |\beta_j|] \\ &= \arg\min_{\boldsymbol{\beta}} [\sum_{i=1}^{n} [y_i - (\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j)]^2 + \lambda \sum_{j=1}^{p} |\beta_j|], \quad \lambda = \frac{2\sigma^2}{b} \end{aligned} \tag{15}$$

The above equation (15) is exactly the form of Lasso estimation. When b is small, meaning the density of double-exponential distribution has a sharp increase at its mean, β_j is more likely to have the mean value 0. So some of the coefficient of predictors can be 0, which is the variable selection. When b is large, β_j is more possible to have any value, just like the least squared estimation. Therefore, the lasso estimate is the mode for β under this posterior distribution.

(d)

Ans:
Because $\beta_1, \beta_2, ..., \beta_p$ are independent and identically distributed according to a normal distribution with mean 0 and variance c, the posterior for $\boldsymbol{\beta}$ in this setting is:

$$
\begin{aligned}
P(\boldsymbol{\beta}|\boldsymbol{y}) &= \frac{P(\boldsymbol{y}|\boldsymbol{\beta})P(\boldsymbol{\beta})}{P(\boldsymbol{y})} \\
&= \frac{1}{\prod_{i=1}^n P(y_i)} \times \prod_{j=1}^p \frac{1}{\sqrt{c}\sqrt{2\pi}} e^{-\frac{\beta_j^2}{2c}} \times \prod_{i=1}^n \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_i-(\beta_0+\sum_{j=1}^p x_{ij}\beta_j)]^2}{2\sigma^2}}
\end{aligned} \tag{16}
$$

(e)

Ans: From the view of maximum a posterior probability estimate (usually abbreviated by MAP), we should find the maximum of the posterior in equation (16) in order to find the best estimate of predictors $\boldsymbol{\beta}$:

$$
\begin{aligned}
\hat{\boldsymbol{\beta}}_{\mathrm{MAP}} &= \arg\max_{\boldsymbol{\beta}} P(\boldsymbol{\beta}|\boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{\beta}} \frac{1}{\prod_{i=1}^n P(y_i)} \times \prod_{j=1}^p \frac{1}{\sqrt{c}\sqrt{2\pi}} e^{-\frac{\beta_j^2}{2c}} \times \prod_{i=1}^n \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_i-(\beta_0+\sum_{j=1}^p x_{ij}\beta_j)]^2}{2\sigma^2}} \\
&= \arg\max_{\boldsymbol{\beta}} [\log \prod_{i=1}^n \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_i-(\beta_0+\sum_{j=1}^p x_{ij}\beta_j)]^2}{2\sigma^2}} + \log \prod_{j=1}^p \frac{1}{\sqrt{c}\sqrt{2\pi}} e^{-\frac{\beta_j^2}{2c}}] \\
&= \arg\max_{\boldsymbol{\beta}} [\sum_{i=1}^n -\frac{[y_i-(\beta_0+\sum_{j=1}^p x_{ij}\beta_j)]^2}{2\sigma^2} - \sum_{j=1}^p \frac{\beta_j^2}{2c}] \\
&= \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^2} [\sum_{i=1}^n [y_i - (\beta_0 + \sum_{j=1}^p x_{ij}\beta_j)]^2 + \frac{\sigma^2}{c} \sum_{j=1}^p \beta_j^2] \\
&= \arg\min_{\boldsymbol{\beta}} [\sum_{i=1}^n [y_i - (\beta_0 + \sum_{j=1}^p x_{ij}\beta_j)]^2 + \lambda \sum_{j=1}^p \beta_j^2], \quad \lambda = \frac{\sigma^2}{c}
\end{aligned} \tag{17}
$$

The above equation (17) is exactly the form of ridge regression estimation. When c is small, meaning the density is more shrunken to its mean value 0, β_j is more likely to have a value near 0. So the coefficient of predictors are restricted small. When c is large, β_j is more possible to have any value, just like the least squared estimation. Therefore, the ridge regression estimate is both the mode and the mean for β under this posterior distribution.

P4

(a)

Code:

```
# (a)
set.seed(1)
X = rnorm(100, mean = 0, sd = 1)
e = rnorm(100, mean = 0, sd = 1)
```

Result:

```
> summary(X)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.2147  -0.4942   0.1139   0.1089   0.6915   2.4016
> summary(e)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.91436  -0.65105  -0.17722  -0.03781   0.50090   2.30798
```

(b)

Code:

```
# (b)
beta0 = 1
beta1 = 2
beta2 = 3
beta3 = 5
Y = beta0 + beta1*X + beta2*X^2 + beta3*X^3 + e
```

Result:

```
> summary(Y)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-43.6798   -0.3189    1.9192    4.6648    5.7107   92.7915
```

(c)

Code:

```
# (c)
# create a single data set containing both X and Y
simulatedData = data.frame(
  response = Y,
  X_1 = X,
  X_2 = X^2,
  X_3 = X^3,
  X_4 = X^4,
```

```
  X_5 = X^5,
  X_6 = X^6,
  X_7 = X^7,
  X_8 = X^8,
  X_9 = X^9,
  X_10 = X^10,
  stringsAsFactors = FALSE
)
str(simulatedData) # Get the structure of the data frame.
library(leaps)
# perform best subset selection, using 10 predictors
regfit.full = regsubsets(response~.,simulatedData,nvmax = 10)
reg.summary = summary(regfit.full)
# RSS
plot(reg.summary$rss,main="Best subset selection",xlab="Number of
    Variables",ylab="RSS",type = "l")
# Adjusted RSq
plot(reg.summary$adjr2,main="Best subset selection",xlab="Number of
    Variables",ylab="Adjusted RSq",type = "l")
abline(v=which.max(reg.summary$adjr2), lty=2, col="red")
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.
    summary$adjr2)],col="red")
coef(regfit.full,which.max(reg.summary$adjr2))
# Cp
plot(reg.summary$cp,main="Best subset selection",xlab="Number of
    Variables",ylab="Cp",type = "l")
abline(v=which.min(reg.summary$cp), lty=2, col="red")
points(which.min(reg.summary$cp),reg.summary$cp[which.min(reg.summary$
    cp)],col="red")
coef(regfit.full,which.min(reg.summary$cp))
# BIC
plot(reg.summary$bic,main="Best subset selection",xlab="Number of
    Variables",ylab="BIC",type = "l")
abline(v=which.min(reg.summary$bic), lty=2, col="red")
points(which.min(reg.summary$bic),reg.summary$bic[which.min(reg.
    summary$bic)],col="red")
coef(regfit.full,which.min(reg.summary$bic))
```

Result:

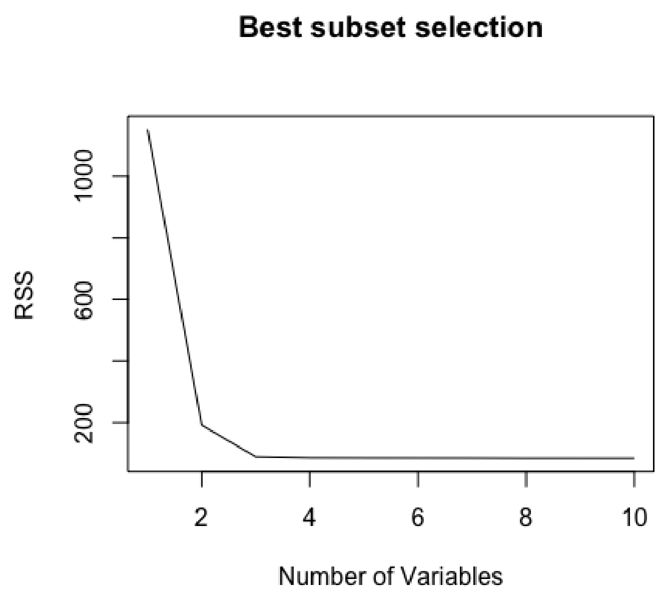


Figure 5: RSS in best subset selection

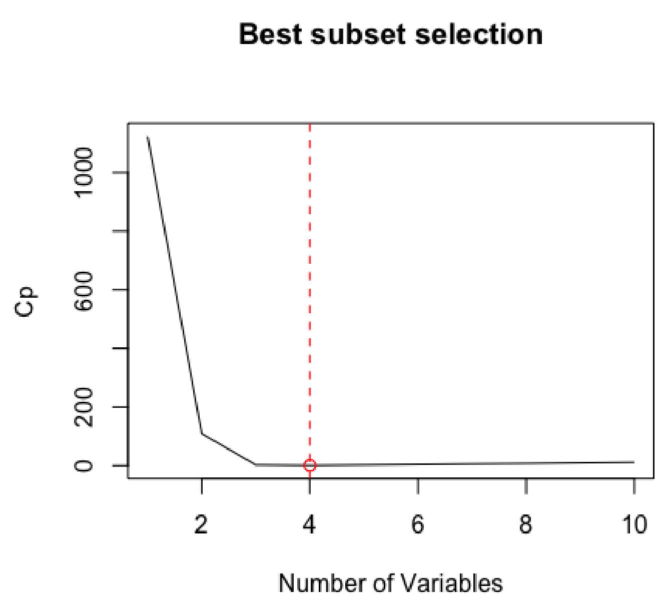


Figure 6: Best model obtained according to C_p

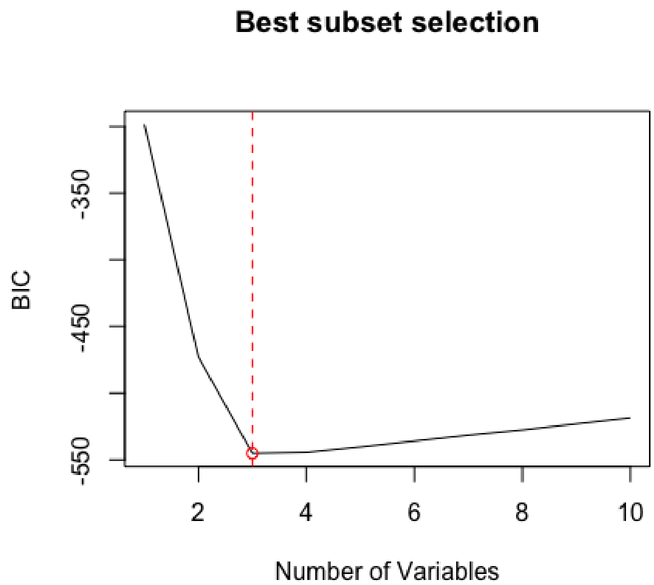


Figure 7: Best model obtained according to BIC

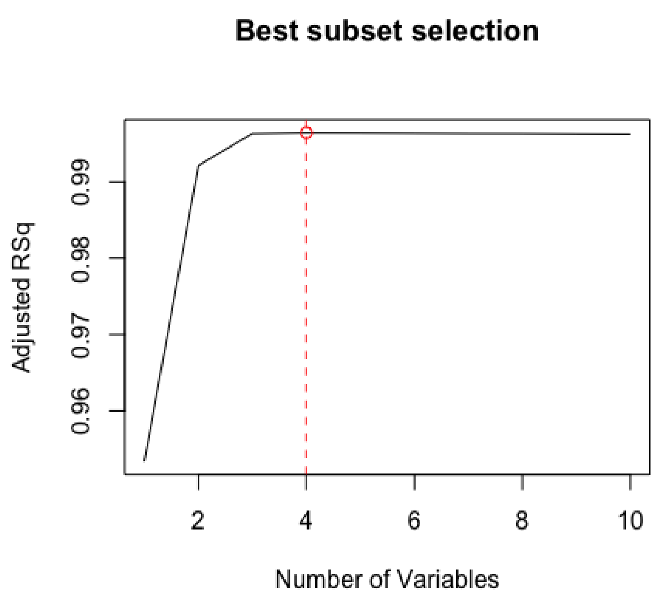


Figure 8: Best model obtained according to $Adjusted\ R^2$

The training data:

```
> str(simulatedData) # Get the structure of the data frame.
'data.frame':   100 obs. of  11 variables:
 $ response: num  -0.925 1.542 -2.405 32.283 1.509 ...
 $ X_1     : num  -0.626 0.184 -0.836 1.595 0.33 ...
 $ X_2     : num  0.3924 0.0337 0.6983 2.5449 0.1086 ...
 $ X_3     : num  -0.24585 0.00619 -0.5835 4.05986 0.03578 ...
 $ X_4     : num  0.15401 0.00114 0.48759 6.47662 0.01179 ...
 $ X_5     : num  -0.096482 0.000209 -0.407443 10.332031 0.003884 ...
 $ X_6     : num  6.04e-02 3.84e-05 3.40e-01 1.65e+01 1.28e-03 ...
 $ X_7     : num  -3.79e-02 7.04e-06 -2.85e-01 2.63e+01 4.22e-04 ...
 $ X_8     : num  2.37e-02 1.29e-06 2.38e-01 4.19e+01 1.39e-04 ...
 $ X_9     : num  -1.49e-02 2.38e-07 -1.99e-01 6.69e+01 4.58e-05 ...
 $ X_10    : num  9.31e-03 4.36e-08 1.66e-01 1.07e+02 1.51e-05 ...
```

The coefficients of the best model obtained according to C_p:

```
> coef(regfit.full,which.min(reg.summary$cp))
(Intercept)          X_1          X_2          X_3          X_5
 1.07200775   2.38745596   2.84575641   4.55797426   0.08072292
```

The coefficients of the best model obtained according to BIC:

```
> coef(regfit.full,which.min(reg.summary$bic))
(Intercept)          X_1          X_2          X_3
   1.061507     1.975280     2.876209     5.017639
```

The coefficients of the best model obtained according to *Adjusted* R^2:

```
> coef(regfit.full,which.max(reg.summary$adjr2))
(Intercept)          X_1          X_2          X_3          X_5
 1.07200775   2.38745596   2.84575641   4.55797426   0.08072292
```

Ans:
From Figure 6, the best model using best subset selection method according to C_p contains 4 predictors. The best model obtained is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_5 X^5 \\ &= 1.072 + 2.387X + 2.846X^2 + 4.558X^3 + 0.081X^5\end{aligned}$$

From Figure 7, the best model using best subset selection method according to BIC contains 3 predictors. The best model obtained is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 \\ &= 1.062 + 1.975X + 2.876X^2 + 5.018X^3\end{aligned}$$

From Figure 8, the best model using best subset selection method according to *Adjusted* R^2 contains 4 predictors. The best model obtained is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_5 X^5 \\ &= 1.072 + 2.387X + 2.846X^2 + 4.558X^3 + 0.081X^5\end{aligned}$$

The model from C_p and *Adjusted* R^2 still have a noise variable, although whose coefficient is small. The estimated coefficients of signal variables are close to the true value.

(d)

Code:

```
# (d)
# perform forward stepwise selection, using 10 predictors
regfit.fwd = regsubsets(response~.,simulatedData,nvmax = 10,method = "
    forward")
reg.summary = summary(regfit.fwd)
# RSS
plot(reg.summary$rss,main="Forward stepwise selection",xlab="Number of
    Variables",ylab="RSS",type = "l")
# Adjusted RSq
plot(reg.summary$adjr2,main="Forward stepwise selection",xlab="Number
    of Variables",ylab="Adjusted RSq",type = "l")
abline(v=which.max(reg.summary$adjr2), lty=2, col="red")
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.
    summary$adjr2)],col="red")
coef(regfit.fwd,which.max(reg.summary$adjr2))
# Cp
plot(reg.summary$cp,main="Forward stepwise selection",xlab="Number of
    Variables",ylab="Cp",type = "l")
abline(v=which.min(reg.summary$cp), lty=2, col="red")
points(which.min(reg.summary$cp),reg.summary$cp[which.min(reg.summary$
    cp)],col="red")
coef(regfit.fwd,which.min(reg.summary$cp))
# BIC
plot(reg.summary$bic,main="Forward stepwise selection",xlab="Number of
    Variables",ylab="BIC",type = "l")
abline(v=which.min(reg.summary$bic), lty=2, col="red")
points(which.min(reg.summary$bic),reg.summary$bic[which.min(reg.
    summary$bic)],col="red")
coef(regfit.fwd,which.min(reg.summary$bic))
# perform backward stepwise selection, using 10 predictors
regfit.bwd = regsubsets(response~.,simulatedData,nvmax = 10,method = "
    backward")
reg.summary = summary(regfit.bwd)
# RSS
plot(reg.summary$rss,main="Backward stepwise selection",xlab="Number
    of Variables",ylab="RSS",type = "l")
# Adjusted RSq
plot(reg.summary$adjr2,main="Backward stepwise selection",xlab="Number
    of Variables",ylab="Adjusted RSq",type = "l")
abline(v=which.max(reg.summary$adjr2), lty=2, col="red")
```

```
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.
    summary$adjr2)],col="red")
coef(regfit.bwd,which.max(reg.summary$adjr2))
# Cp
plot(reg.summary$cp,main="Backward stepwise selection",xlab="Number of
    Variables",ylab="Cp",type = "l")
abline(v=which.min(reg.summary$cp), lty=2, col="red")
points(which.min(reg.summary$cp),reg.summary$cp[which.min(reg.summary$
    cp)],col="red")
coef(regfit.bwd,which.min(reg.summary$cp))
# BIC
plot(reg.summary$bic,main="Backward stepwise selection",xlab="Number
    of Variables",ylab="BIC",type = "l")
abline(v=which.min(reg.summary$bic), lty=2, col="red")
points(which.min(reg.summary$bic),reg.summary$bic[which.min(reg.
    summary$bic)],col="red")
coef(regfit.bwd,which.min(reg.summary$bic))
```

Result:

The coefficients of the forward stepwise selection obtained according to C_p:

```
> coef(regfit.fwd,which.min(reg.summary$cp))
(Intercept)        X_1        X_2        X_3        X_5
 1.07200775 2.38745596 2.84575641 4.55797426 0.08072292
```

The coefficients of the forward stepwise selection obtained according to BIC:

```
> coef(regfit.fwd,which.min(reg.summary$bic))
(Intercept)        X_1        X_2        X_3
   1.061507   1.975280   2.876209   5.017639
```

The coefficients of the forward stepwise selection obtained according to *Adjusted* R^2:

```
> coef(regfit.fwd,which.max(reg.summary$adjr2))
(Intercept)        X_1        X_2        X_3        X_5
 1.07200775 2.38745596 2.84575641 4.55797426 0.08072292
```

The coefficients of the backward stepwise selection obtained according to C_p:

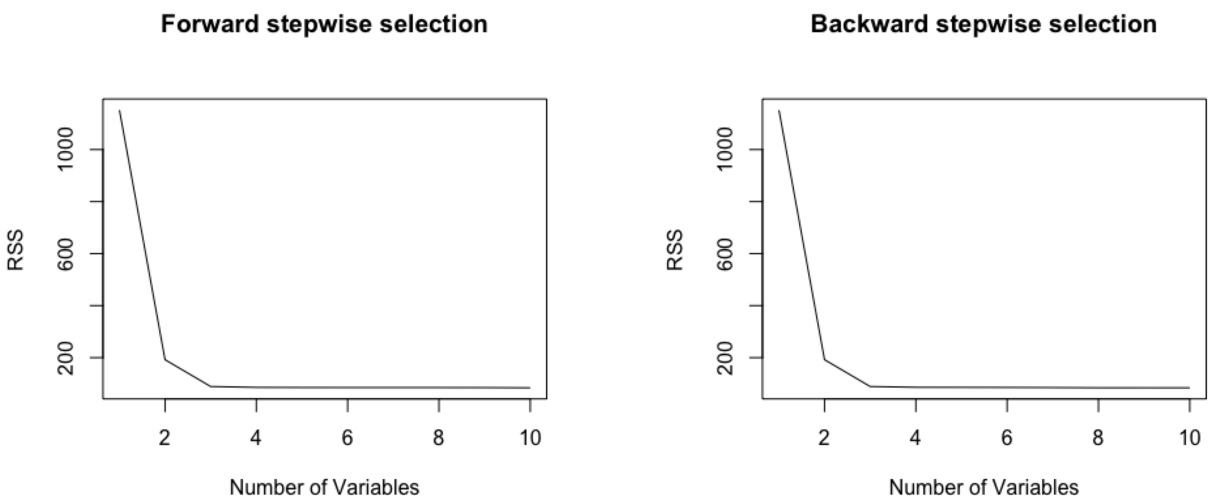
```
> coef(regfit.bwd,which.min(reg.summary$cp))
(Intercept)         X_1         X_2         X_3         X_9
1.079236362 2.231905828 2.833494180 4.819555807 0.001290827
```

The coefficients of the backward stepwise selection obtained according to BIC:

```
> coef(regfit.bwd,which.min(reg.summary$bic))
(Intercept)        X_1        X_2        X_3
   1.061507   1.975280   2.876209   5.017639
```

The coefficients of the backward stepwise selection obtained according to *Adjusted* R^2:

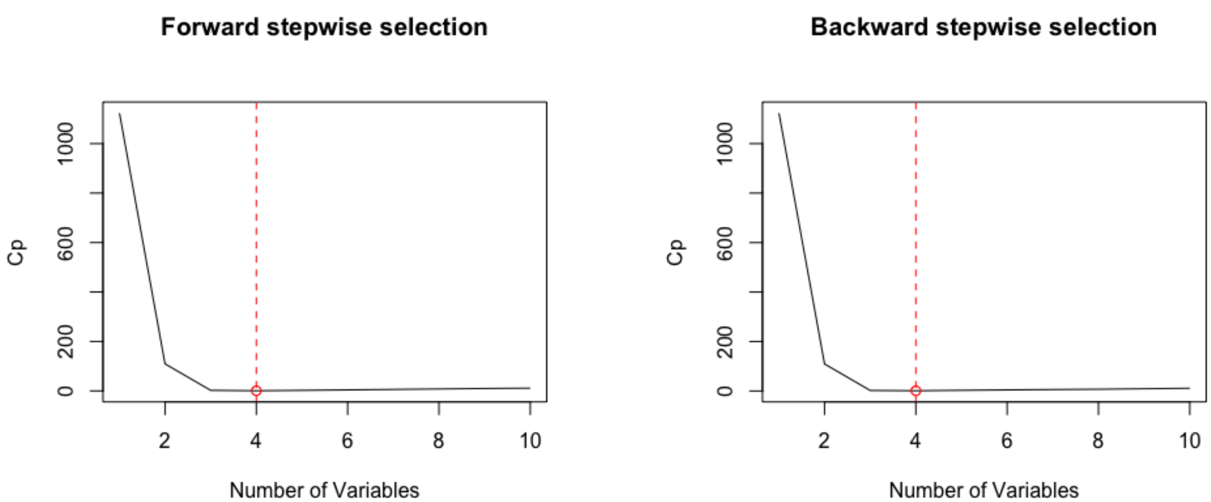
```
> coef(regfit.bwd,which.max(reg.summary$adjr2))
(Intercept)          X_1          X_2          X_3          X_9
1.079236362  2.231905828  2.833494180  4.819555807  0.001290827
```



(a) Forward stepwise selection

(b) Backward stepwise selection

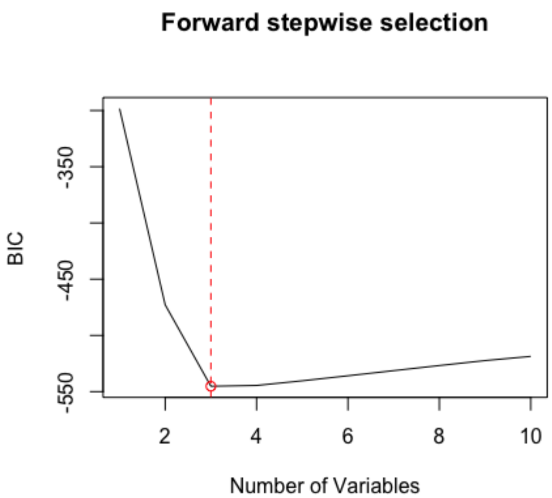
Figure 9: RSS



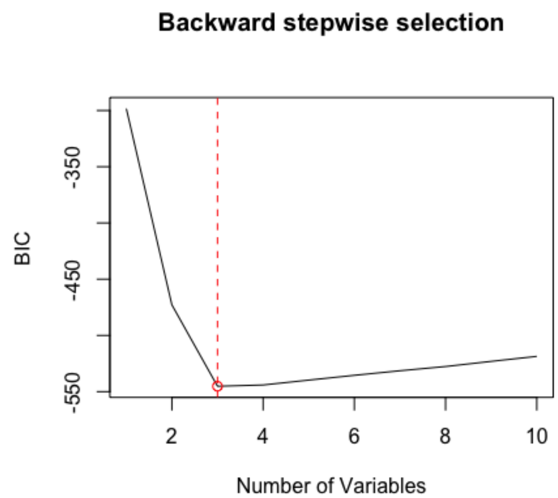
(a) Forward stepwise selection

(b) Backward stepwise selection

Figure 10: Best model obtained according to C_p

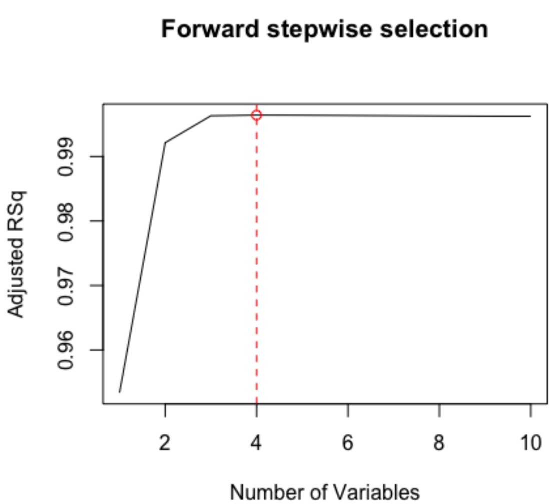


(a) Forward stepwise selection

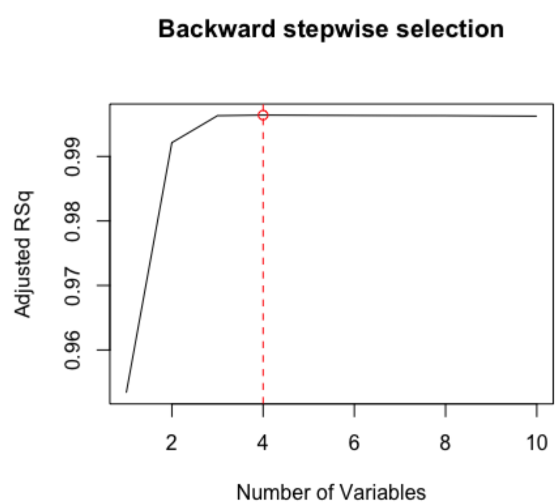


(b) Backward stepwise selection

Figure 11: Best model obtained according to BIC



(a) Forward stepwise selection



(b) Backward stepwise selection

Figure 12: Best model obtained according to $Adjusted\ R^2$

Ans:

From Figure 10, the best model using forward and backward stepwise selection method according to C_p contains 4 predictors, which is the same number with best subset selection. The best model obtained by forward stepwise selection is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_5 X^5 \\ &= 1.072 + 2.387X + 2.846X^2 + 4.558X^3 + 0.081X^5\end{aligned}$$

The best model obtained by backward stepwise selection is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_9 X^9 \\ &= 1.079 + 2.232X + 2.833X^2 + 4.820X^3 + 0.001X^9\end{aligned}$$

From Figure 11, the best model using forward stepwise selection method according to BIC contains 3 predictors, which is the same number with best subset selection. The best model obtained by forward stepwise selection is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 \\ &= 1.062 + 1.975X + 2.876X^2 + 5.018X^3\end{aligned}$$

The best model obtained by backward stepwise selection is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 \\ &= 1.062 + 1.975X + 2.876X^2 + 5.018X^3\end{aligned}$$

From Figure 12, the best model using forward stepwise selection method according to $Adjusted\ R^2$ contains 4 predictors, which is the same number with best subset selection. The best model obtained by forward stepwise selection is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_5 X^5 \\ &= 1.072 + 2.387X + 2.846X^2 + 4.558X^3 + 0.081X^5\end{aligned}$$

The best model obtained by backward stepwise selection is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_9 X^9 \\ &= 1.079 + 2.232X + 2.833X^2 + 4.820X^3 + 0.001X^9\end{aligned}$$

The model from C_p and $Adjusted\ R^2$ still have a noise variable, although whose coefficient is small. The estimated coefficients of signal variables are close to the true value.

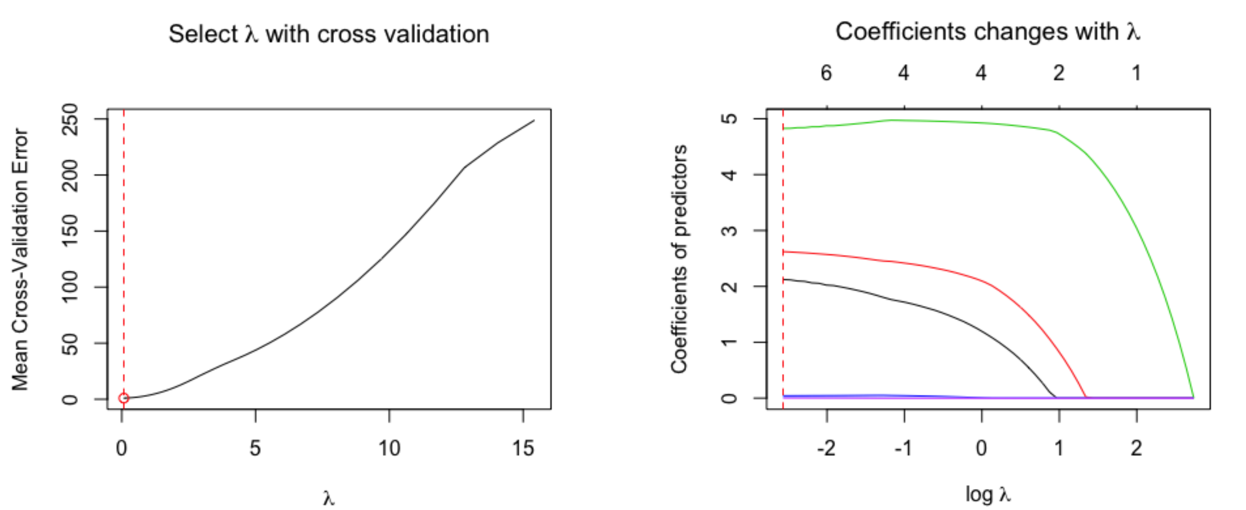
In summary, in this problem setting, the forward stepwise selection gets the same models with best subset selection, no matter what criterion C_p or BIC or $Adjusted\ R^2$ is used. However, the backward stepwise selection gets the different models with best subset selection, when C_p or $Adjusted\ R^2$ is used. Meanwhile, the backward stepwise selection gets the same model with best subset selection when BIC is used.

(e)

Code:

```
library("glmnet")
x_lasso = model.matrix(response~., simulatedData)[,-1]
y_lasso = simulatedData$response
cv.lasso <- cv.glmnet(x_lasso, y_lasso, nfolds = 10, parallel=TRUE,
    standardize=TRUE, type.measure='mse')
# plot(cv.lasso)
plot(cv.lasso$lambda, cv.lasso$cvm, main=expression(paste("Select ",
    lambda, ' with cross validation')),xlab=expression(lambda),ylab="
    Mean Cross-Validation Error", type = 'l')
abline(v=cv.lasso$lambda.min, lty=2, col="red")
points(cv.lasso$lambda.min, cv.lasso$cvm[which(cv.lasso$lambda==cv.
    lasso$lambda.min)],col="red")
plot(cv.lasso$glmnet.fit, xvar="lambda", label=TRUE)
abline(v=log(cv.lasso$lambda.min), lty=2, col="red")
cv.lasso$lambda.min
# cv.lasso$lambda.1se
coef(cv.lasso, s=cv.lasso$lambda.min)
```

Result:



(a) Ten-fold cross-validation MSE for the lasso (b) The corresponding lasso coefficient estimates

Figure 13: Lasso model with cross-validation to select optimal λ

```
> cv.lasso$lambda.min
[1] 0.07676824
> coef(cv.lasso, s=cv.lasso$lambda.min)
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) 1.183697416
X_1         2.123302063
X_2         2.621261843
X_3         4.826107793
X_4         0.042987976
X_5         0.010370603
X_6         .
X_7         0.003844961
X_8         .
X_9         .
X_10        .
```

Ans:

Using ten-fold cross-validation, the λ in lasso is selected as $\lambda = 0.07676824$. The small λ means the lasso will have similar performance with least squared estimation. The model obtained by lasso is:

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_4 X^4 + \hat{\beta}_5 X^5 + \hat{\beta}_7 X^7 \\ &= 1.184 + 2.123X + 2.621X^2 + 4.826X^3 + 0.043X^4 + 0.010X^5 + 0.004X^7\end{aligned}$$

This model shows variable selection but still includes some noise variables, although whose coefficients are small. The estimated coefficients of signal variables are close to their true value.

(f)

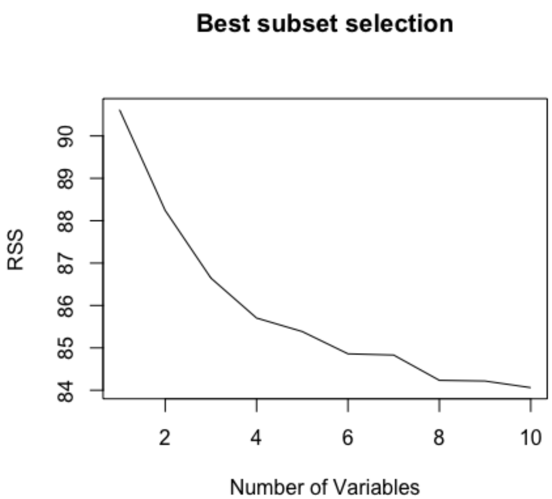
Code:

```
# (f)
beta0 = 1
beta7 = 13
Y = beta0 + beta7*X^7 + e
# best subset selection
# create a single data set containing both X and Y
simulatedData = data.frame(
  response = Y,
  X_1 = X,
  X_2 = X^2,
  X_3 = X^3,
  X_4 = X^4,
  X_5 = X^5,
  X_6 = X^6,
  X_7 = X^7,
  X_8 = X^8,
  X_9 = X^9,
  X_10 = X^10,
  stringsAsFactors = FALSE
```

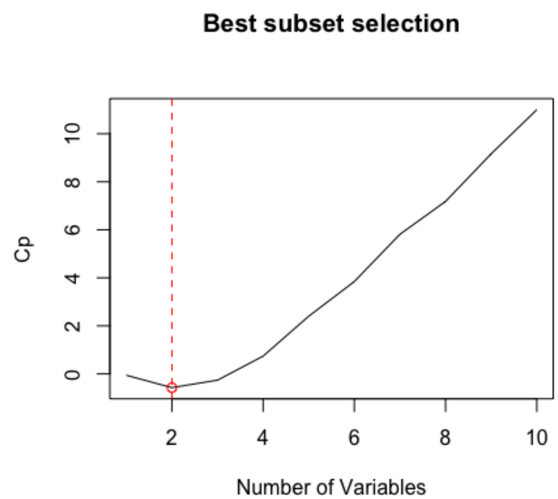
```r
)
str(simulatedData) # Get the structure of the data frame.
library(leaps)
# perform best subset selection, using 10 predictors
regfit.full = regsubsets(response~.,simulatedData,nvmax = 10)
reg.summary = summary(regfit.full)
# RSS
plot(reg.summary$rss,main="Best subset selection",xlab="Number of 
    Variables",ylab="RSS",type = "l")
# Adjusted RSq
plot(reg.summary$adjr2,main="Best subset selection",xlab="Number of 
    Variables",ylab="Adjusted RSq",type = "l")
abline(v=which.max(reg.summary$adjr2), lty=2, col="red")
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.
    summary$adjr2)],col="red")
coef(regfit.full,which.max(reg.summary$adjr2))
# Cp
plot(reg.summary$cp,main="Best subset selection",xlab="Number of 
    Variables",ylab="Cp",type = "l")
abline(v=which.min(reg.summary$cp), lty=2, col="red")
points(which.min(reg.summary$cp),reg.summary$cp[which.min(reg.summary$
    cp)],col="red")
coef(regfit.full,which.min(reg.summary$cp))
# BIC
plot(reg.summary$bic,main="Best subset selection",xlab="Number of 
    Variables",ylab="BIC",type = "l")
abline(v=which.min(reg.summary$bic), lty=2, col="red")
points(which.min(reg.summary$bic),reg.summary$bic[which.min(reg.
    summary$bic)],col="red")
coef(regfit.full,which.min(reg.summary$bic))

# lasso
library("glmnet")
x_lasso = model.matrix(response~., simulatedData)[,-1]
y_lasso = simulatedData$response
cv.lasso <- cv.glmnet(x_lasso, y_lasso, nfolds = 10, parallel=TRUE,
    standardize=TRUE, type.measure='mse')
# plot(cv.lasso)
plot(cv.lasso$lambda, cv.lasso$cvm, main=expression(paste("Select ",
    lambda, ' with cross validation')),xlab=expression(lambda),ylab="
    Mean Cross-Validation Error", type = 'l')
abline(v=cv.lasso$lambda.min, lty=2, col="red")
points(cv.lasso$lambda.min, cv.lasso$cvm[which(cv.lasso$lambda==cv.
    lasso$lambda.min)],col="red")
plot(cv.lasso$glmnet.fit, xvar="lambda", main=expression(paste("
    Coefficients changes with ",lambda)),xlab=expression(paste('log ',
    lambda)),ylab="Coefficients of predictors")
```

```
abline(v=log(cv.lasso$lambda.min), lty=2, col="red")
cv.lasso$lambda.min
# cv.lasso$lambda.1se
coef(cv.lasso, s=cv.lasso$lambda.min)
```

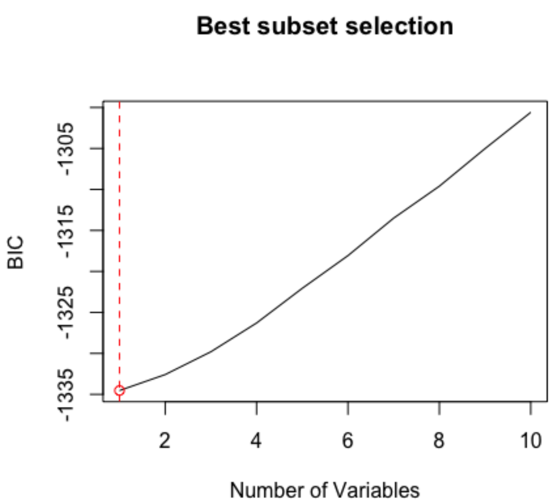
Result:



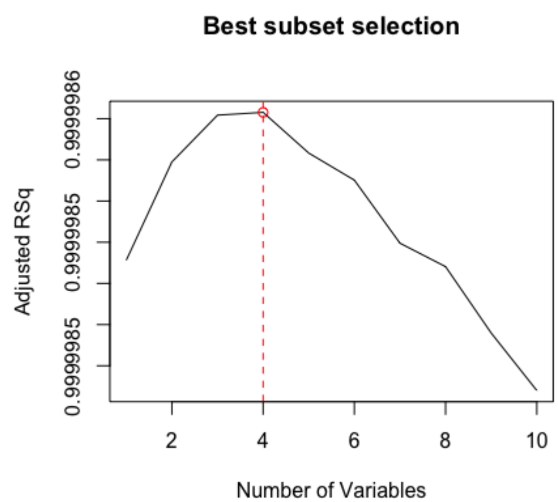
(a) *RSS*



(b) C_p



(c) *BIC*



(d) *Adjusted* R^2

Figure 14: Best subset selection

```
> str(simulatedData) # Get the structure of the data frame.
'data.frame':   100 obs. of  11 variables:
 $ response: num  -0.113 1.042 -3.61 342.983 0.351 ...
 $ X_1     : num  -0.626 0.184 -0.836 1.595 0.33 ...
 $ X_2     : num  0.3924 0.0337 0.6983 2.5449 0.1086 ...
 $ X_3     : num  -0.24585 0.00619 -0.5835 4.05986 0.03578 ...
 $ X_4     : num  0.15401 0.00114 0.48759 6.47662 0.01179 ...
 $ X_5     : num  -0.096482 0.000209 -0.407443 10.332031 0.003884 ...
 $ X_6     : num  6.04e-02 3.84e-05 3.40e-01 1.65e+01 1.28e-03 ...
 $ X_7     : num  -3.79e-02 7.04e-06 -2.85e-01 2.63e+01 4.22e-04 ...
 $ X_8     : num  2.37e-02 1.29e-06 2.38e-01 4.19e+01 1.39e-04 ...
 $ X_9     : num  -1.49e-02 2.38e-07 -1.99e-01 6.69e+01 4.58e-05 ...
 $ X_10    : num  9.31e-03 4.36e-08 1.66e-01 1.07e+02 1.51e-05 ...
```

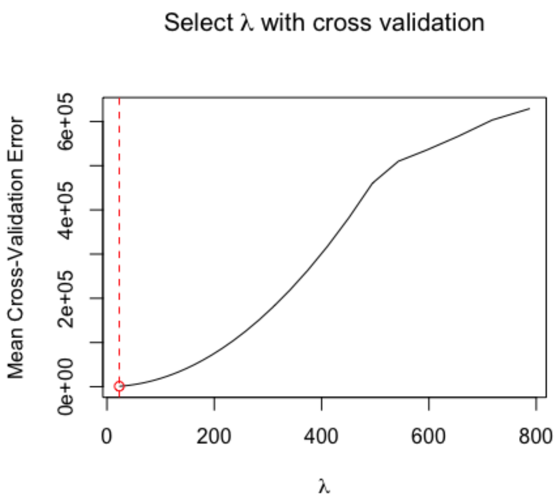
```
> coef(regfit.full,which.max(reg.summary$adjr2))
(Intercept)         X_1         X_2         X_3         X_7
  1.0762524   0.2914016  -0.1617671  -0.2526527  13.0091338
```

```
> coef(regfit.full,which.min(reg.summary$cp))
(Intercept)         X_2         X_7
  1.0704904  -0.1417084  13.0015552
```

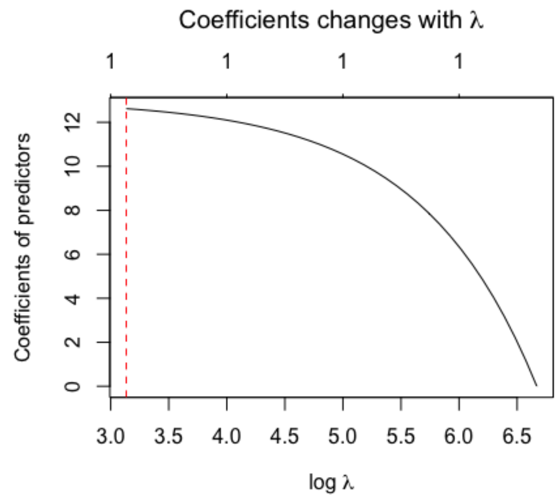
```
> coef(regfit.full,which.min(reg.summary$bic))
(Intercept)         X_7
  0.9589402  13.0007705
```

```
> cv.lasso$lambda.min
[1] 22.96953
```

```
> coef(cv.lasso, s=cv.lasso$lambda.min)
11 x 1 sparse Matrix of class "dgCMatrix"
                    1
(Intercept)  2.558369
X_1          .
X_2          .
X_3          .
X_4          .
X_5          .
X_6          .
X_7         12.621791
X_8          .
X_9          .
X_10         .
```



(a) Ten-fold cross-validation MSE



(b) The corresponding lasso coefficient estimates

Figure 15: Lasso

Ans:
Best subset selection and lasso both do the variable selection.
Best subset selection will get different models when different criteria are used:

$$\begin{aligned}
Adjusted\ R^2 :\ \hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_7 X^7 \\
&= 1.076 + 0.291X - 0.162X^2 - 0.253X^3 + 13.009X^7 \\
C_p :\ \hat{Y} &= \hat{\beta}_0 + \hat{\beta}_2 X^2 + \hat{\beta}_7 X^7 \\
&= 1.070 - 0.142X^2 + 13.002X^7 \\
BIC :\ \hat{Y} &= \hat{\beta}_0 + \hat{\beta}_7 X^7 \\
&= 0.959 + 13.001X^7
\end{aligned}$$

Adjusted R^2 and C_p still include some noise variables, whose coefficients are very small. BIC doesn't include noise variable.

Lasso gets a model without noise variables as:

$$\begin{aligned}
\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_7 X^7 \\
&= 2.558 + 12.622X^7
\end{aligned}$$

However the estimated coefficients of the model are not as close to their true value as best subset selection, because the regularization modification $\lambda \sum_{j=1}^{p} |\beta_j|$ will affect the estimation of coefficients.

P5

(a)

Code:

```
# (a)
rm(list=ls())
set.seed(1)
College = read.csv("/Users/yangchenye/Downloads/College.csv",header=T,
   na.strings="?")
dim(College)
College=na.omit(College) # remove incomplete cases
dim(College)
names(College)

# 75% of the sample size
smp_size = floor(0.75 * nrow(College))
train_ind = sample(seq_len(nrow(College)), size = smp_size)
# select train data and test data, drop the name 'X' of College
train = subset(College[train_ind, ],select = -X)
test = subset(College[-train_ind, ],select = -X)
```

Result:

```
> dim(College)
[1] 777  19
> names(College)
 [1] "X"           "Private"     "Apps"        "Accept"
 [5] "Enroll"      "Top10perc"   "Top25perc"   "F.Undergrad"
 [9] "P.Undergrad" "Outstate"    "Room.Board"  "Books"
[13] "Personal"    "PhD"         "Terminal"    "S.F.Ratio"
[17] "perc.alumni" "Expend"      "Grad.Rate"
> dim(test)
[1] 195  18
> dim(train)
[1] 582  18
```

(b)

Code:

```
# (b)
train_fit = lm(train$Grad.Rate~., data = train)
summary(train_fit)
test_lm_predict = predict(train_fit, test[-18]) # predict
test_lm_MSE = mean((test$Grad.Rate - test_lm_predict) ^ 2) # test MSE
```

Result:

```
> summary(train_fit)

Call:
lm(formula = train$Grad.Rate ~ ., data = train)

Residuals:
    Min       1Q   Median       3Q      Max
-50.657   -7.049   -0.529    7.117   51.438

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 26.8066318  5.8737195   4.564 6.17e-06 ***
PrivateYes   4.7305052  1.9991510   2.366 0.018306 *
Apps         0.0010352  0.0005326   1.944 0.052435 .
Accept      -0.0007207  0.0010828  -0.666 0.505961
Enroll       0.0024664  0.0028023   0.880 0.379161
Top10perc    0.0922866  0.0830377   1.111 0.266877
Top25perc    0.0811958  0.0621955   1.305 0.192257
F.Undergrad -0.0002828  0.0004785  -0.591 0.554695
P.Undergrad -0.0016878  0.0004434  -3.806 0.000157 ***
Outstate     0.0008544  0.0002648   3.227 0.001324 **
Room.Board   0.0021872  0.0006795   3.219 0.001362 **
Books       -0.0015732  0.0033449  -0.470 0.638309
Personal    -0.0016231  0.0008667  -1.873 0.061621 .
PhD          0.0964858  0.0654935   1.473 0.141252
Terminal    -0.0477951  0.0724298  -0.660 0.509599
S.F.Ratio    0.2824858  0.1955257   1.445 0.149083
perc.alumni  0.3194714  0.0572827   5.577 3.80e-08 ***
Expend      -0.0003060  0.0001671  -1.831 0.067599 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.8 on 564 degrees of freedom
Multiple R-squared:  0.4723,    Adjusted R-squared:  0.4564
F-statistic:  29.7 on 17 and 564 DF,  p-value: < 2.2e-16
```

```
> test_lm_MSE
[1] 163.6218
```

Ans:
The test mean squared error obtained is 163.6218.

(c)

Code:

```
# (c) ridge
```

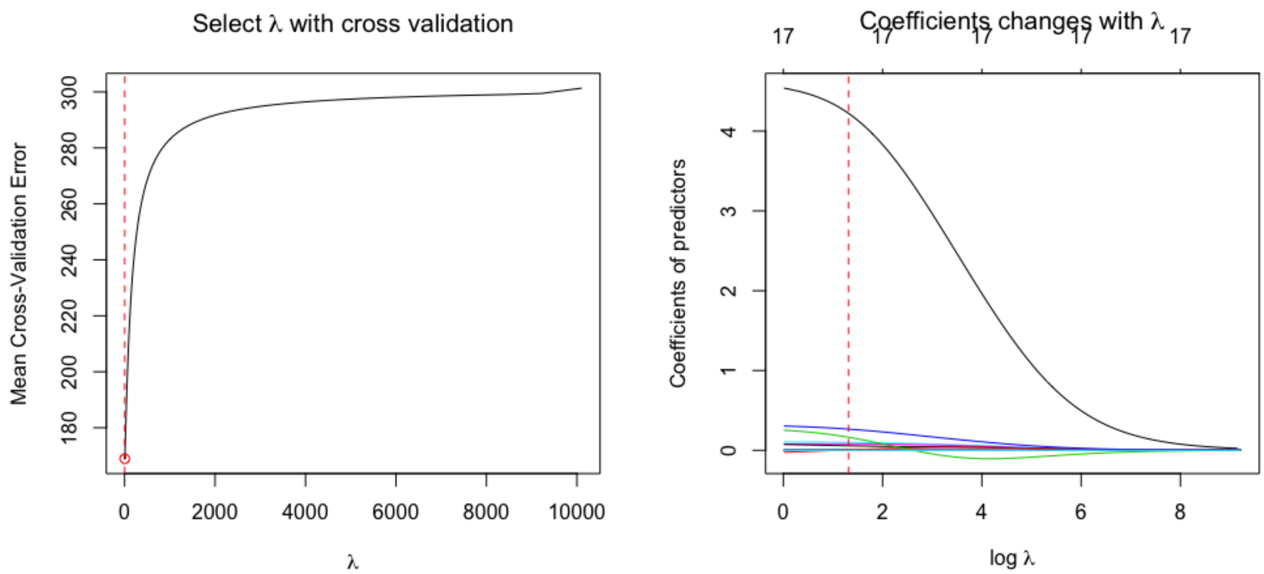
```
library("glmnet")
x_ridge = model.matrix(train$Grad.Rate~., data = train)[,-1]
y_ridge = train$Grad.Rate
# Ridge: Alpha = 0
cv.ridge = cv.glmnet(x_ridge, y_ridge, alpha=0, nfolds = 10, parallel=
    TRUE, standardize=TRUE, type.measure='mse')
# CVE~lambda
plot(cv.ridge$lambda, cv.ridge$cvm, main=expression(paste("Select ",
    lambda, ' with cross validation')),xlab=expression(lambda),ylab="
    Mean Cross-Validation Error", type = 'l')
abline(v=cv.ridge$lambda.min, lty=2, col="red")
points(cv.ridge$lambda.min, cv.ridge$cvm[which(cv.ridge$lambda==cv.
    ridge$lambda.min)],col="red")
plot(cv.ridge$glmnet.fit, xvar="lambda", main=expression(paste("
    Coefficients changes with ",lambda)),xlab=expression(paste('log ',
    lambda)),ylab="Coefficients of predictors")
abline(v=log(cv.ridge$lambda.min), lty=2, col="red")
# best lambda
cv.ridge$lambda.min
coef(cv.ridge, s=cv.ridge$lambda.min)
# best ridge regression with best lambda
ridge_best = glmnet(x_ridge, y_ridge, alpha=0, lambda = cv.ridge$
    lambda.min, standardize=TRUE)
test_ridge_predict = predict(ridge_best, model.matrix(test$Grad.Rate~
    ., data = test)[,-1], s="lambda.min") # predict
test_ridge_MSE = mean((test$Grad.Rate - test_ridge_predict) ^ 2) #
    test MSE
```

Result:

```
> cv.ridge$lambda.min
[1] 3.717836
> coef(cv.ridge, s=cv.ridge$lambda.min)
18 x 1 sparse Matrix of class "dgCMatrix"
                            1
(Intercept)   3.046609e+01
PrivateYes    4.222649e+00
Apps          3.903074e-04
Accept        3.282553e-04
Enroll        4.136841e-04
Top10perc     9.958251e-02
Top25perc     8.206982e-02
F.Undergrad  -6.921736e-05
P.Undergrad  -1.385642e-03
Outstate      6.635847e-04
Room.Board    1.878762e-03
Books        -1.775428e-03
Personal     -1.760206e-03
```

```
PhD           5.689299e-02
Terminal      5.003101e-03
S.F.Ratio     1.641541e-01
perc.alumni   2.657793e-01
Expend       -9.817824e-05
```

```
> test_ridge_MSE
[1] 164.5001
```



(a) Ten-fold cross-validation MSE

(b) The corresponding ridge regression coefficients

Figure 16: Ridge regression model

Ans:
The λ chosen in ridge regression is 3.717836. The test mean squared error obtained is 164.5001.

(d)

Code:

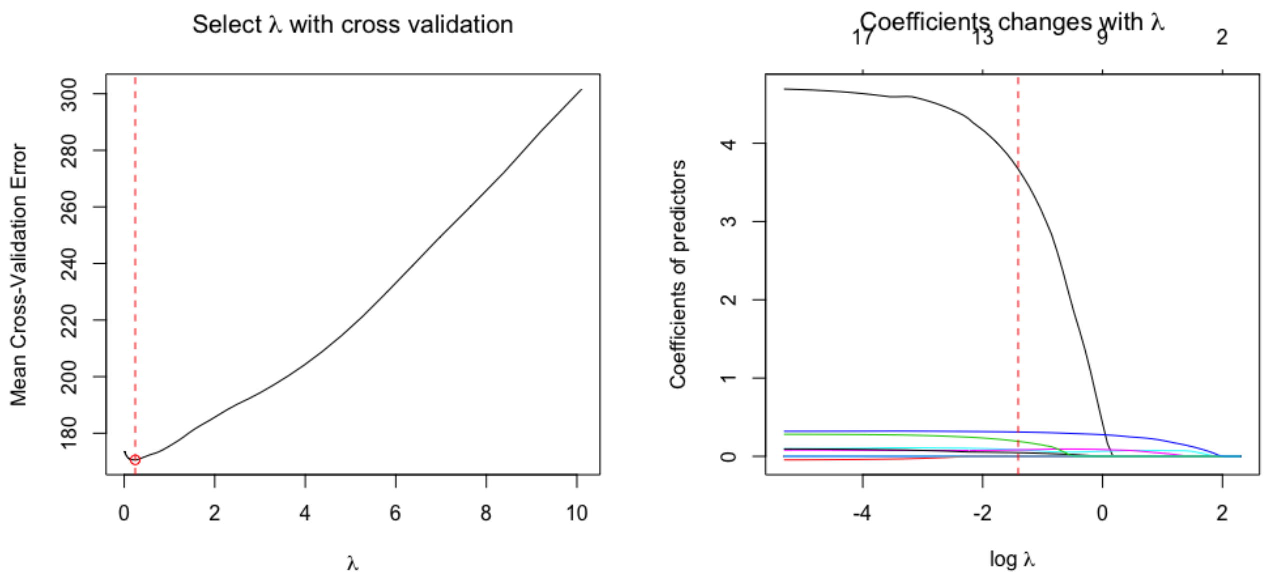
```
# (d) lasso
library("glmnet")
x_lasso = model.matrix(train$Grad.Rate~., data = train)[,-1]
y_lasso = train$Grad.Rate
# Lasso: Alpha = 1
cv.lasso = cv.glmnet(x_lasso, y_lasso, alpha=1, nfolds = 10, parallel=
    TRUE, standardize=TRUE, type.measure='mse')
# CVE~lambda
plot(cv.lasso$lambda, cv.lasso$cvm, main=expression(paste("Select ",
    lambda, ' with cross validation')),xlab=expression(lambda),ylab="
    Mean Cross-Validation Error", type = 'l')
```

```r
abline(v=cv.lasso$lambda.min, lty=2, col="red")
points(cv.lasso$lambda.min, cv.lasso$cvm[which(cv.lasso$lambda==cv.lasso$lambda.min)], col="red")
plot(cv.lasso$glmnet.fit, xvar="lambda", main=expression(paste("Coefficients changes with ",lambda)),xlab=expression(paste('log ',lambda)),ylab="Coefficients of predictors")
abline(v=log(cv.lasso$lambda.min), lty=2, col="red")
# best lambda
cv.lasso$lambda.min
cv.lasso$nzero[which(cv.lasso$lambda==cv.lasso$lambda.min)]
coef(cv.lasso, s=cv.lasso$lambda.min)
# best ridge regression with best lambda
lasso_best = glmnet(x_lasso, y_lasso, alpha=0, lambda = cv.lasso$lambda.min, standardize=TRUE)
test_lasso_predict = predict(lasso_best, model.matrix(test$Grad.Rate~., data = test)[,-1], s="lambda.min") # predict
test_lasso_MSE = mean((test$Grad.Rate - test_lasso_predict) ^ 2) # test MSE
```

Result:

```
> cv.lasso$lambda.min
[1] 0.2446088
> cv.lasso$nzero[which(cv.lasso$lambda==cv.lasso$lambda.min)]
s40
 13
> coef(cv.lasso, s=cv.lasso$lambda.min)
18 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept) 29.0167980959
PrivateYes   3.6706143690
Apps         0.0006514975
Accept       .
Enroll       .
Top10perc    0.0833713627
Top25perc    0.0821703158
F.Undergrad  .
P.Undergrad -0.0015119149
Outstate     0.0007874207
Room.Board   0.0019193219
Books       -0.0006307748
Personal    -0.0015104017
PhD          0.0465084115
Terminal     .
S.F.Ratio    0.1911923409
perc.alumni  0.3104190913
Expend      -0.0001341114
```

```
> test_lasso_MSE
[1] 163.8735
```



(a) Ten-fold cross-validation MSE

(b) The corresponding lasso coefficients

Figure 17: Lasso model

Ans:
The λ chosen in lasso is 0.2446088. The number of non-zero coefficient estimates is 13. The test mean squared error obtained is 163.8735.

P6

(a)

Code:

```
# (a)
rm(list=ls())
set.seed(1)
p = 20
n = 1000
X = rnorm(n, mean = 0, sd = 1)
e = rnorm(n, mean = 0, sd = 1)
c = seq(from = 1, to = 20, length.out = p)
beta = rep(0, p)
beta0 = 1
beta[1] = 2
beta[2] = 3
beta[3] = 5
Y = beta0 + beta[1]*X + beta[2]*X^2 + beta[3]*X^3 + e
```

(b)

Code:

```
# (b)
simulatedData = data.frame(
  response = Y,
  X_1 = X^c[1],
  X_2 = X^c[2],
  X_3 = X^c[3],
  X_4 = X^c[4],
  X_5 = X^c[5],
  X_6 = X^c[6],
  X_7 = X^c[7],
  X_8 = X^c[8],
  X_9 = X^c[9],
  X_10 = X^c[10],
  X_11 = X^c[11],
  X_12 = X^c[12],
  X_13 = X^c[13],
  X_14 = X^c[14],
  X_15 = X^c[15],
  X_16 = X^c[16],
  X_17 = X^c[17],
  X_18 = X^c[18],
  X_19 = X^c[19],
  X_20 = X^c[20],
```

```
  stringsAsFactors = FALSE
)
str(simulatedData) # Get the structure of the data frame.
# training set containing 100 observations, 1/10
smp_size = floor(0.1 * nrow(simulatedData))
train_ind = sample(seq_len(nrow(simulatedData)), size = smp_size)
# select train data and test data
train = simulatedData[train_ind, ]
# test = simulatedData[-train_ind, ]
test = model.matrix(~., data=simulatedData[-train_ind, ])
```

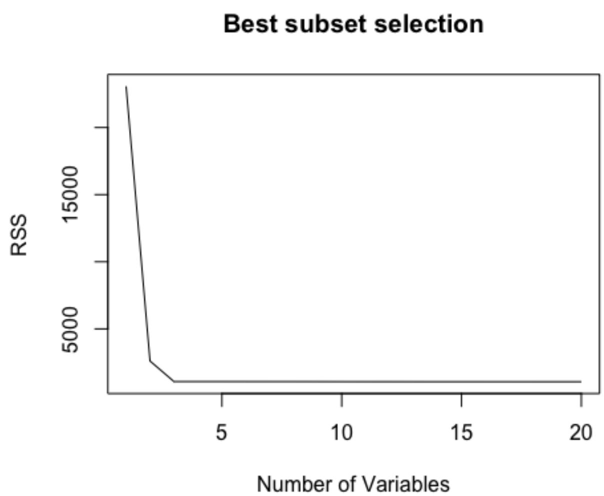
(c)

Code:

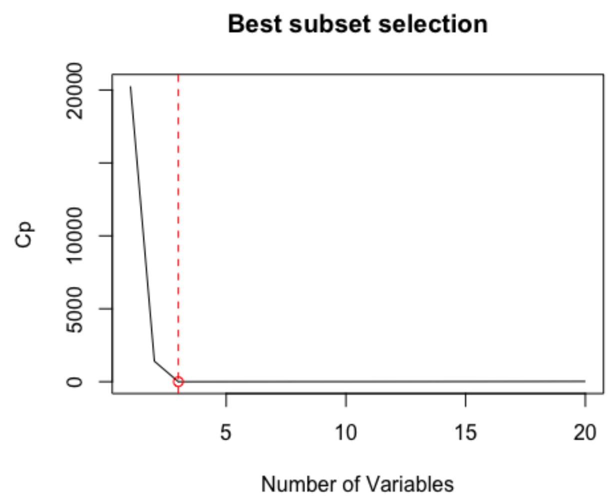
```
# (c)
library(leaps)
# perform best subset selection, using 20 predictors
regfit.full = regsubsets(response~.,simulatedData,nvmax = p)
reg.summary = summary(regfit.full)
# RSS
plot(reg.summary$rss,main="Best subset selection",xlab="Number of
    Variables",ylab="RSS",type = "l")
# Adjusted RSq
plot(reg.summary$adjr2,main="Best subset selection",xlab="Number of
    Variables",ylab="Adjusted RSq",type = "l")
abline(v=which.max(reg.summary$adjr2), lty=2, col="red")
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.
    summary$adjr2)],col="red")
coef(regfit.full,which.max(reg.summary$adjr2))
# Cp
plot(reg.summary$cp,main="Best subset selection",xlab="Number of
    Variables",ylab="Cp",type = "l")
abline(v=which.min(reg.summary$cp), lty=2, col="red")
points(which.min(reg.summary$cp),reg.summary$cp[which.min(reg.summary$
    cp)],col="red")
coef(regfit.full,which.min(reg.summary$cp))
# BIC
plot(reg.summary$bic,main="Best subset selection",xlab="Number of
    Variables",ylab="BIC",type = "l")
abline(v=which.min(reg.summary$bic), lty=2, col="red")
points(which.min(reg.summary$bic),reg.summary$bic[which.min(reg.
    summary$bic)],col="red")
coef(regfit.full,which.min(reg.summary$bic))

# the training set MSE associated with the best model of each size
```

```
plot(regfit.full$rss[-1]/n,main="Training MSE", xlab="Number of predictors", ylab = "MSE", col = "blue", type = "l", lty=2)
legend("topright", legend = "Training", col = "blue", lty=2)
which.min(regfit.full$rss[-1]/n)
```

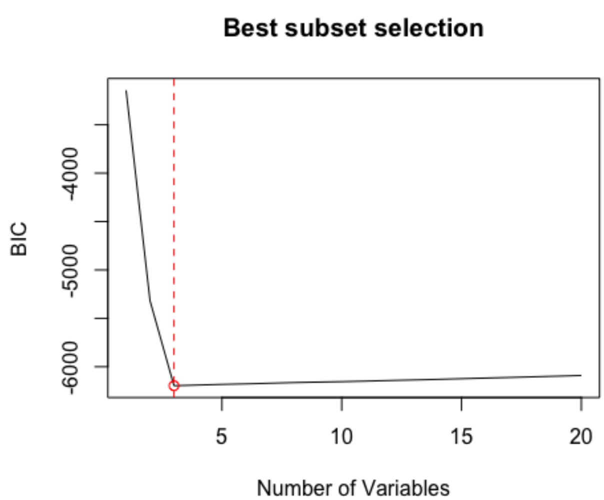
Result:



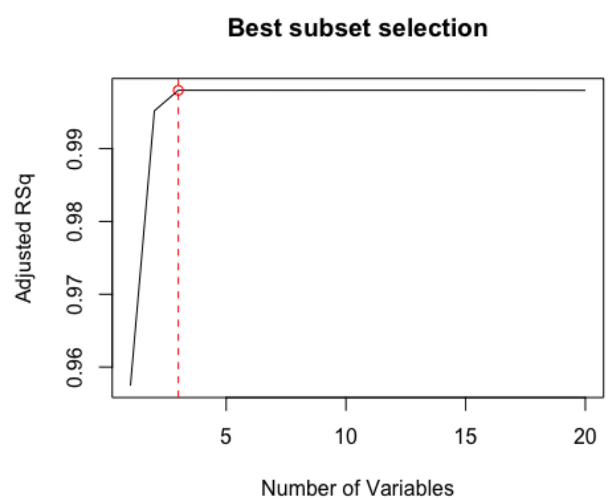
(a) *RSS*



(b) C_p



(c) *BIC*



(d) *Adjusted* R^2

Figure 18: Best subset selection

```
> coef(regfit.full,which.max(reg.summary$adjr2))
(Intercept)         X_1         X_2         X_3
  0.9648892   1.9270333   3.0181880   5.0249860
> coef(regfit.full,which.min(reg.summary$cp))
(Intercept)         X_1         X_2         X_3
  0.9648892   1.9270333   3.0181880   5.0249860
```

```
> coef(regfit.full,which.min(reg.summary$bic))
(Intercept)         X_1         X_2         X_3
  0.9648892   1.9270333   3.0181880   5.0249860
```

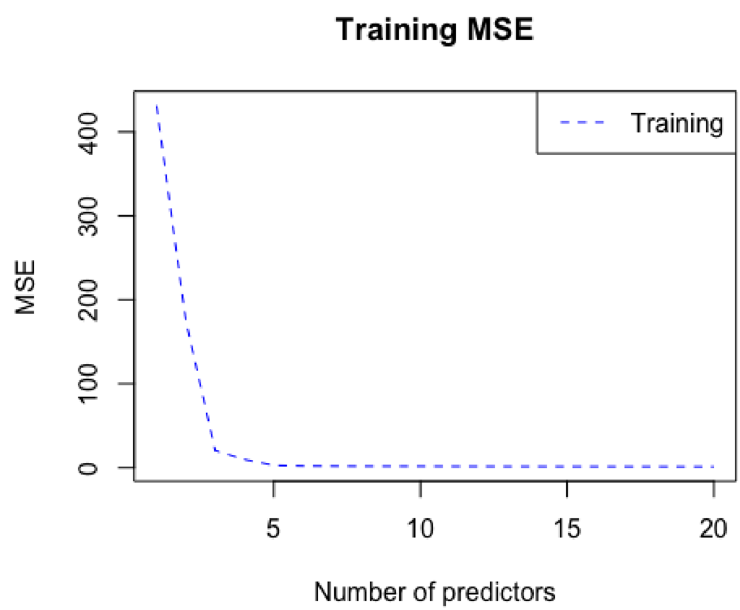


Figure 19: The training set MSE

```
> which.min(regfit.full$rss[-1]/n)
[1] 20
```

(d)

Code:

```
# (d)
# Plot the test set MSE associated with the best model of each size
test.full.MSE = rep(NA, p) # test set MSE
for (i in 1:p) {
  coefi = coef(regfit.full, id = i)
  pred = test[, names(coefi)] %*% coefi
  test.full.MSE[i] = mean((test[, "response"] - pred)^2)
}
plot(test.full.MSE, main="Testing MSE", xlab="Number of predictors",
    ylab = "MSE", col = "red", type = "l", lty=1)
legend("topright", legend = "Testing", col = "red", lty=1)
```

Result:

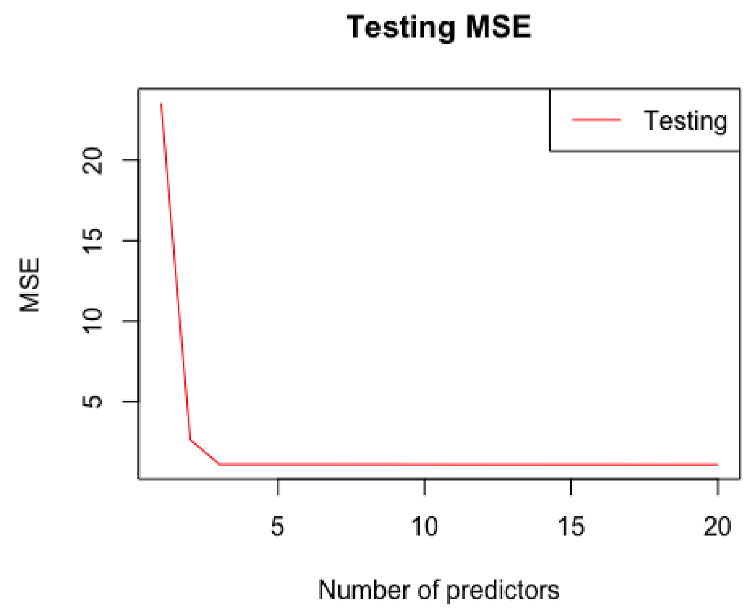


Figure 20: The testing set MSE

(e)

Code:

```
# (e)
which.min(test.full.MSE)
coef(regfit.full, id = which.min(test.full.MSE))
```

Result:

```
> which.min(test.full.MSE)
[1] 17
> coef(regfit.full, id = which.min(test.full.MSE))
  (Intercept)           X_1           X_2           X_3           X_6
 9.941718e-01  1.967462e+00  2.850958e+00  5.099776e+00  1.992884e-01
          X_7           X_9          X_10          X_11          X_12
-4.737564e-01  5.198375e-01 -1.351559e-01 -2.331000e-01  7.858723e-02
         X_13          X_14          X_15          X_16          X_17
 5.409708e-02 -1.980249e-02 -6.854977e-03  2.563927e-03  4.507476e-04
         X_18          X_19          X_20
-1.670087e-04 -1.204494e-05  4.337136e-06
```

Ans:
The test MSE takes its minimum value for the model with 17 predictors.

(f)

Ans:
The model at which test MSE minimized:

$$\begin{aligned}\hat{Y} =&\hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2 + \hat{\beta}_3 X^3 + \hat{\beta}_6 X^6 + \hat{\beta}_7 X^7 + \hat{\beta}_9 X^9 + \hat{\beta}_{10} X^{10} + \\ &\hat{\beta}_{11} X^{11} + \hat{\beta}_{12} X^{12} + \hat{\beta}_{13} X^{13} + \hat{\beta}_{14} X^{14} + \hat{\beta}_{15} X^{15} + \hat{\beta}_{16} X^{16} + \\ &\hat{\beta}_{17} X^{17} + \hat{\beta}_{18} X^{18} + \hat{\beta}_{19} X^{19} + \hat{\beta}_{20} X^{20} \\ =&0.994 + 1.967X + 2.851X^2 + 5.100X^3 + 0.199X^6 - 0.473X^7 + \\ &0.520X^9 - 0.135X^{10} - 0.233X^{11} + 0.079X^{12} + \\ &0.054X^{13} - 0.020X^{14} - 0.007X^{15} + 0.003X^{16} + \\ &5 \times 10^{-4} X^{17} - 2 \times 10^{-4} X^{18} - 1 \times 10^{-5} X^{19} + 4 \times 10^{-6} X^{20}\end{aligned}$$

The true model used to generate the data:

$$\begin{aligned}Y &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \\ &= 1 + 2X + 3X^2 + 5X^3\end{aligned}$$

The estimated model contains a lot of noise variables, comparing to the true model. The estimated coefficients of noise variables are small. The estimated coefficients of signal variables are close to their true value.

The estimated model with minimal test MSE has 17 predictors may because the train data and test data both have lots of noise, introduced by ϵ, and the model tries to fit the noise in training and predict the same noise in validation. Moreover, the true model only has first three non-zero coefficients. Therefore, the test MSE increases little when the degree of freedom increases to large, like Figure 20.

(g)

Code:

```
# (g)
y = rep(NA, p)
for (r in 1:p){
  temp = 0
  coefi = coef(regfit.full,id=r) # intercept, X_1, X_2,...
  for (j in names(coefi)[-1]){
    num = type.convert(substr(j,3,4), 'int')
    temp = temp + (beta[num] - coefi[[j]])^2
  }
  y[r] = temp
}
y = sqrt(y)

plot(y, xlab="Number of predictors", ylab = "y", col = "black", type =
    "l", lty=1)
```

Result:

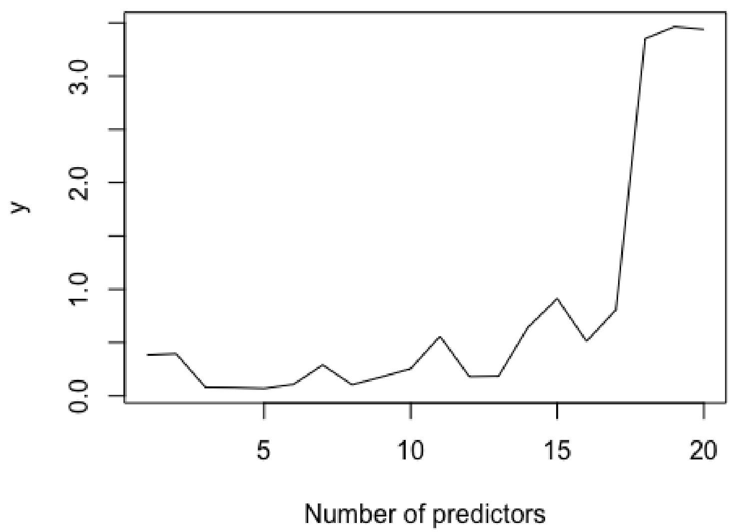


Figure 21: $y = \sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ for a range of values of r

Ans:
The plot displaying $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ for a range of values of r, number of coefficients, has a similar look with the regular test MSE plot. They both decrease first and then increase with the increasing of degree of freedom. But their are not exactly same because the $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ doesn't contain any form of X.

However, for this problem setting, they look different. Reasons:

The estimated model with minimal test MSE has 17 predictors may because the train data and test data both have lots of noise, introduced by ϵ, and the model tries to fit the noise in training and predict the same noise in validation. Moreover, the true model only has first three non-zero coefficients. Therefore, the test MSE increases little when the degree of freedom increases to large, like Figure 20.