Homework 2

Yi Yang (yy3089)

October 24, 2021

1 P1

1.1 a

In this question, the RSS for the linear regression is:

$$\begin{aligned} RSS &= \sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_{ij}\right)^2 \\ &= \left(\hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{12}\right)^2 + \left(\hat{\beta}_0 + \hat{\beta}_1 x_{21} + \hat{\beta}_2 x_{22}\right)^2 \end{aligned} \tag{1.1}$$

So the Ridge Regression Optimization is:

$$\begin{aligned} \min RSS + \lambda \sum_{j=1}^{p}\beta_j^2 &= \sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p}\hat{\beta}_j{}^2 \\ &= \left(\hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{12}\right)^2 + \left(\hat{\beta}_0 + \hat{\beta}_1 x_{21} + \hat{\beta}_2 x_{22}\right)^2 + \lambda\left(\hat{\beta}_1{}^2 + \hat{\beta}_2{}^2\right) \end{aligned} \tag{1.2}$$

1.2 b

For the optimization result:

$$\begin{aligned} \frac{\partial f}{\partial \beta_1} &= -2x_{11}\left(y_1 - \beta_1 x_{11} - \beta_2 x_{12}\right) - 2x_{21}\left(y_2 - \beta_1 x_{21} - \beta_2 x_{22}\right) + 2\lambda\beta_1 = 0 \\ \frac{\partial f}{\partial \beta_2} &= -2x_{12}\left(y_1 - \beta_1 x_{11} - \beta_2 x_{12}\right) - 2x_{22}\left(y_2 - \beta_1 x_{21} - \beta_2 x_{22}\right) + 2\lambda\beta_2 = 0 \end{aligned} \tag{1.3}$$

Denote that $x_1 = x_{11} = x_{21}$, $x_2 = x_{12} = x_{22}$. To solve the two equations, we need to make some transform:

$$\begin{aligned} \left(x_1^2 + x_2^2 + \lambda\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2\right)\hat{\beta}_2 &= x_1 y_1 + x_2 y_2 \\ \left(x_1^2 + x_2^2\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2 + \lambda\right)\hat{\beta}_2 &= x_1 y_1 + x_2 y_2 \\ \left(x_1^2 + x_2^2 + \lambda\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2\right)\hat{\beta}_2 &= \left(x_1^2 + x_2^2\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2 + \lambda\right)\hat{\beta}_2 \\ \left(x_1^2 + x_2^2 + \lambda\right)\left(\hat{\beta}_1 - \hat{\beta}_2\right) + \left(x_1^2 + x_2^2\right)\left(\hat{\beta}_2 - \hat{\beta}_1\right) &= 0 \\ \lambda\hat{\beta}_1 &= \lambda\hat{\beta}_2 \end{aligned} \tag{1.4}$$

Since $\lambda \neq 0$, $\hat{\beta}_1 = \hat{\beta}_2$

1.3 c

LASSO Regression Optimization is:

$$\begin{aligned} \min RSS + \lambda \sum_{j=1}^{p} |\beta_j| &= \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\hat{\beta}_j| \\ &= \left(\hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{12}\right)^2 + \left(\hat{\beta}_0 + \hat{\beta}_1 x_{21} + \hat{\beta}_2 x_{22}\right)^2 + \lambda\left(|\hat{\beta}_1| + |\hat{\beta}_2|\right) \end{aligned} \tag{1.5}$$

1.4 d

For the optimization result:

$$\begin{aligned} \frac{\partial f}{\partial \beta_1} &= -2x_{11}\left(y_1 - \beta_1 x_{11} - \beta_2 x_{12}\right) - 2x_{21}\left(y_2 - \beta_1 x_{21} - \beta_2 x_{22}\right) \pm \lambda = 0 \\ \frac{\partial f}{\partial \beta_2} &= -2x_{12}\left(y_1 - \beta_1 x_{11} - \beta_2 x_{12}\right) - 2x_{22}\left(y_2 - \beta_1 x_{21} - \beta_2 x_{22}\right) \pm \lambda = 0 \end{aligned} \tag{1.6}$$

Denote that $x_1 = x_{11} = x_{21}$, $x_2 = x_{12} = x_{22}$. To solve the two equations, we need to make some transform:

$$\begin{aligned} \left(x_1^2 + x_2^2\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2\right)\hat{\beta}_2 &= x_1 y_1 + x_2 y_2 \pm \lambda \\ \left(x_1^2 + x_2^2\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2\right)\hat{\beta}_2 &= x_1 y_1 + x_2 y_2 \pm \lambda \end{aligned} \tag{1.7}$$

Only when $\hat{\beta}_1 > 0$ and $\hat{\beta}_2 > 0$ or $\hat{\beta}_1 < 0$ and $\hat{\beta}_2 < 0$, there will be solutions for this problem. Due to the two equations are same in form, there will be more than one solution for this problem. if $\hat{\beta}_1 > 0$ and $\hat{\beta}_2 > 0$, the equation will be:

$$\left(x_1^2 + x_2^2\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2\right)\hat{\beta}_2 = x_1 y_1 + x_2 y_2 - \lambda \tag{1.8}$$

Otherwise, the equation will be:

$$\left(x_1^2 + x_2^2\right)\hat{\beta}_1 + \left(x_1^2 + x_2^2\right)\hat{\beta}_2 = x_1 y_1 + x_2 y_2 + \lambda \tag{1.9}$$

2 P2

2.1 a

When $p = 1$, the optimization function is $(y-\beta_1)^2 + \lambda\beta_1^2 = (1+\lambda)\beta_1^2 - 2y\beta_1 + y^2$. To find the β_1 that minimize the y, we need to calculate $2(1+\lambda)\beta_1 - 2y = 0$
We can plot the image:

```
y = 2
lambda = 1
beta = seq(-5,5,0.01)

plot(beta,(y-beta)^2 + lambda*beta^2,type="l",xlab = "beta"
,ylab = "Optimization function",xlim=c(-5,5))

min_x = y/(1+lambda)
min_y = (y-min_x)^2 + lambda*min_x^2
points(min_x,min_y,col="red")

title("Optimization Result for Ridge Regression")
```

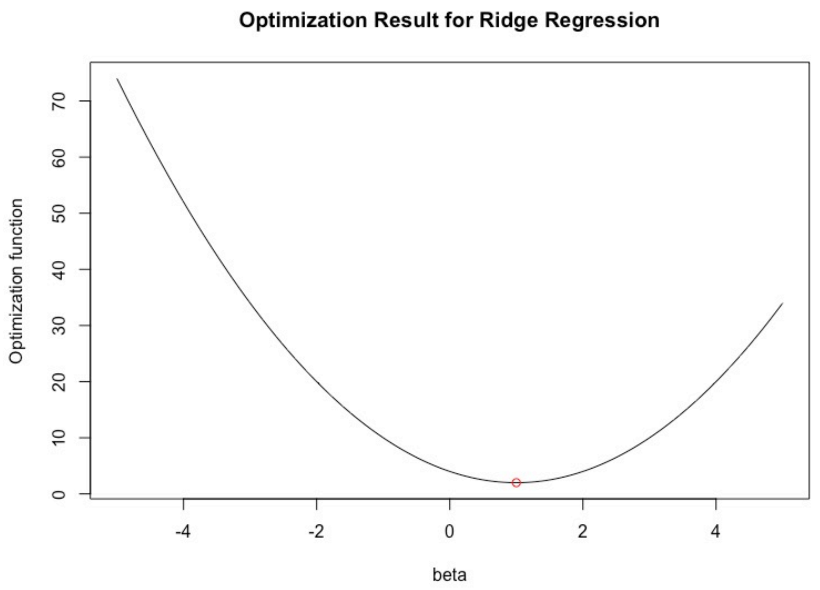


Figure 2.1: Screenshot for *'Ridge Regression'*

2.2 b

When $p = 1$, the optimization function is $(y-\beta_1)^2 + \lambda|\beta_1| = \beta_1^2 - (2y \pm \lambda)\beta_1 + y^2$. To find the β_1 that minimize the y, we need to calculate $2\beta_1 - (2y \pm \lambda) = 0$. If $\beta_1 \geqslant 0$, that's $2\beta_1 - (2y - \lambda) = 0$. Otherwise, that's $2\beta_1 - (2y + \lambda) = 0$.
We can plot the image for all three conditions:

```
y = 2
lambda = 3
beta = seq(-5,5,0.01)
```

```
plot(beta,(y-beta)^2 + lambda*abs(beta),type="l",xlab = "beta"
,ylab = "Optimization␣function",xlim=c(-5,5))

if(y > lambda/2) {
  min_x1 = y - lambda/2
} else {
  if(y < -lambda/2){
    min_x1 = y + lambda/2
  } else {
    min_x1 = 0
  }
}

min_y1 = (y-min_x1)^2 + lambda*abs(min_x1)
points(min_x1,min_y1,col="red")

title("Optimization␣Result␣for␣LASSO␣Regression")
```

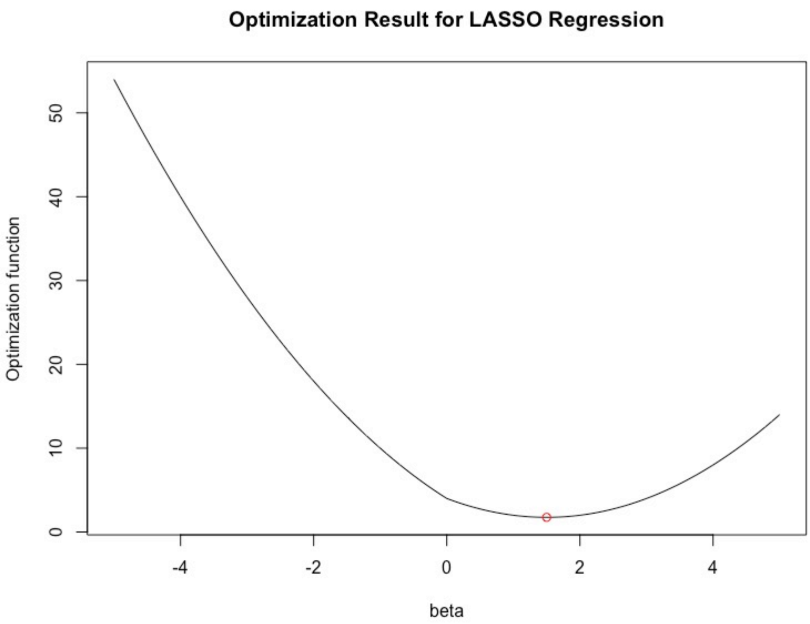


Figure 2.2: Screenshot for *'LASSO Regression'*

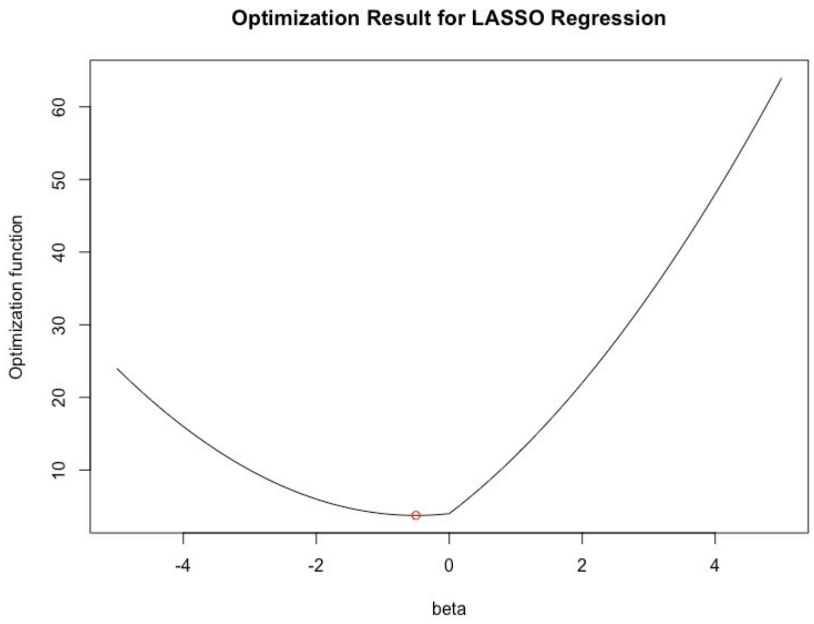


Figure 2.3: Screenshot for *'LASSO Regression'*

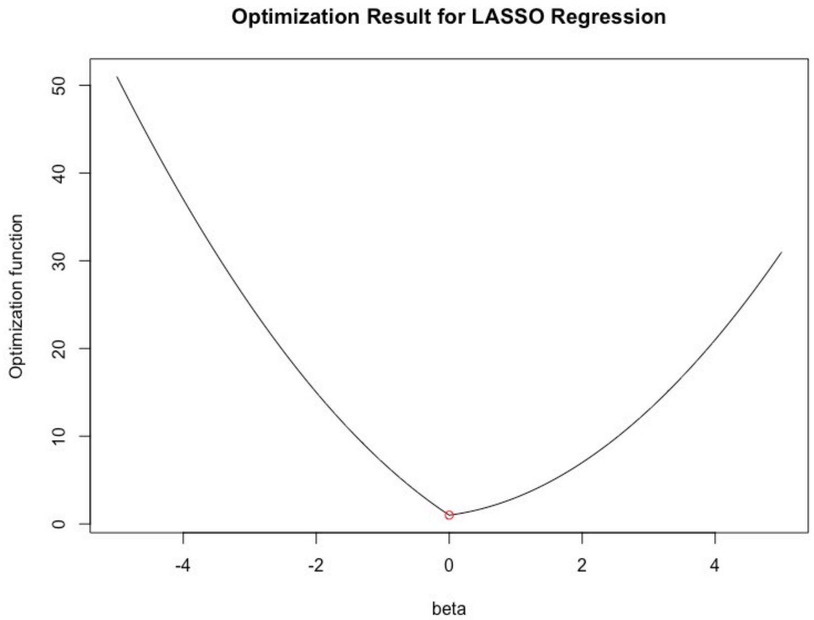


Figure 2.4: Screenshot for *'LASSO Regression'*

3 P3

3.1 a

We know that $y_i \sim \mathcal{N}(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j, \sigma^2)$, so the likelihood function is:

$$\begin{aligned}\mathscr{L}(X,\beta,\sigma \mid Y) &= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right)\right)^2}{2\sigma^2}} \\ &= \frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right)\right)^2}\end{aligned} \tag{3.1}$$

3.2 b

We know that posterior $\propto \mathscr{L} \times$ prior, and prior$(\beta) = \frac{1}{2b} e^{(-\frac{\sum_{j=1}^{p}|\beta_j|}{b})}$, thus:

$$\begin{aligned}\text{posterior} &\propto \frac{1}{(\sigma\sqrt{2\pi})^n} e^{\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2\right)} * \frac{1}{2b} e^{(-\frac{\sum_{j=1}^{p}|\beta_j|}{b})} \\ &\propto \frac{1}{2b} * \frac{1}{(\sigma\sqrt{2\pi})^n} e^{\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{\sum_{j=1}^{p}|\beta_j|}{b}\right)}\end{aligned} \tag{3.2}$$

3.3 c

To prove the argument, we need to prove that the maximum of the posterior is given by the LASSO regression with λ:

$$\log(\text{posterior}) = \log(\frac{1}{2b} * \frac{1}{(\sigma\sqrt{2\pi})^n}) + \left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{\sum_{j=1}^{p}|\beta_j|}{b}\right) \tag{3.3}$$

To find the maximum for the posterior, we need to find the minimum for the $\left(\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{\sum_{j=1}^{p}|\beta_j|}{b}\right)$:

$$\begin{aligned}\arg\min_{\beta}\left(\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{\sum_{j=1}^{p}|\beta_j|}{b}\right) &= \arg\min_{\beta}\left(\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{2\sigma^2\sum_{j=1}^{p}|\beta_j|}{b}\right) \\ &= \arg\min_{\beta}\left(\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{2\sigma^2}{b}\sum_{j=1}^{p}|\beta_j|\right)\end{aligned} \tag{3.4}$$

Let $\lambda = \frac{2\sigma^2}{b}$, that's the same form with LASSO Regression.

3.4 d

$$\begin{aligned}\text{prior}(\beta) &= \prod_{j=1}^{p} \mathbf{P}(\beta_j) \\ &= \prod_{j=1}^{p} \frac{1}{\sqrt{2\pi c}} e^{\left(-\frac{\beta_j^2}{2c}\right)} \\ &= \frac{1}{(\sqrt{2\pi c})^p} e^{\left(-\frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right)}\end{aligned} \tag{3.5}$$

$$
\begin{aligned}
\text{posterior} &\propto \mathscr{L} \times \text{prior} \\
&\propto \frac{1}{(\sigma\sqrt{2\pi})^n} e^{\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2\right)} * \frac{1}{(\sqrt{2\pi c})^p} e^{\left(-\frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right)} \\
&\propto \frac{1}{(\sigma\sqrt{2\pi})^n} * \frac{1}{(\sqrt{2\pi c})^p} e^{\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right)}
\end{aligned} \tag{3.6}
$$

3.5 e

To prove the argument, we need to prove that the maximum of the posterior is given by the Ridge Regression with λ:

$$
\log(\text{posterior}) = \log(\frac{1}{2b} * \frac{1}{(\sigma\sqrt{2\pi})^n}) + \left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right) \tag{3.7}
$$

To find the maximum for the posterior, we need to find the minimum for the $\left(\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right)$:

$$
\begin{aligned}
\arg\min_{\beta}\left(\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right) &= \arg\min_{\beta}\left(\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{2\sigma^2\sum_{j=1}^{p}|\beta_j|}{2c}\right) \\
&= \arg\min_{\beta}\left(\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{\sigma^2}{c}\sum_{j=1}^{p}|\beta_j|\right)
\end{aligned} \tag{3.8}
$$

Let $\lambda = \frac{\sigma^2}{c}$, that's the same form with Ridge Regression.

4 P4

4.1 a

```r
set.seed(1)
x = rnorm(100)
eps = rnorm(100)
```

4.2 b

```r
beta = c(1, -1, 2, 0.5)
y = beta[1] + beta[2]*x + beta[3]*x^2 + beta[4]*x^3 + eps
```

4.3 c

```r
library(leaps)
ans.df = data.frame(y = y, x = x)
ans.model = regsubsets(y ~ poly(x, 10, raw = T),
                       data = ans.df, nvmax = 10)
ans.summary = summary(ans.model)

min_p = which.min(ans.summary$cp)
plot(ans.summary$cp, xlab = "Subset size", ylab = "Cp",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
title("Best Subset with Cp")
coefficients(ans.model, id = which.min(ans.summary$cp))

min_p = which.min(ans.summary$bic)
plot(ans.summary$bic, xlab = "Subset size", ylab = "BIC",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
title("Best Subset with BIC")
coefficients(ans.model, id = which.min(ans.summary$bic))

max_p = which.max(ans.summary$adjr2)
plot(ans.summary$adjr2, xlab = "Subset size", ylab = "AdjR2",
     col = "blue", pch = 20, type = "o")
points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
title("Best Subset with AdjR2")
coefficients(ans.model, id = which.max(ans.summary$adjr2))
```

The best model uses a X^5 term. $|\beta_0 - \hat{\beta}_0| = 0.072$, $|\beta_1 - \hat{\beta}_1| = 0.445$, $|\beta_2 - \hat{\beta}_2| = 0.157$

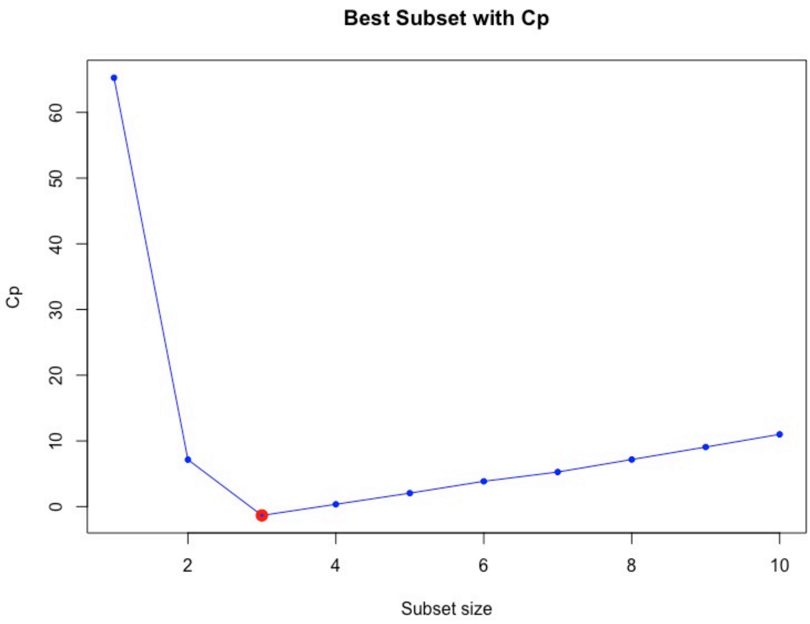


Figure 4.1: Screenshot for *'Best Subset with C_p'*

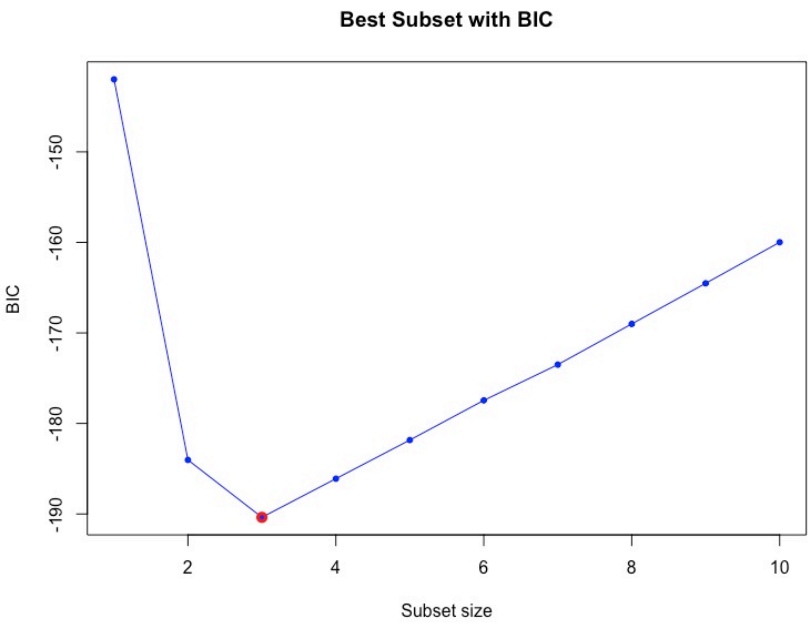


Figure 4.2: Screenshot for *'Best Subset with BIC'*

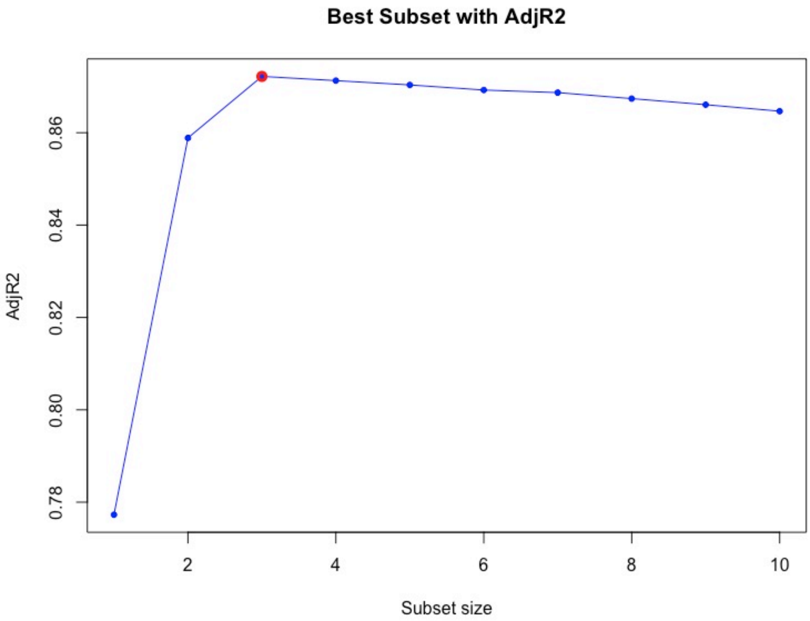
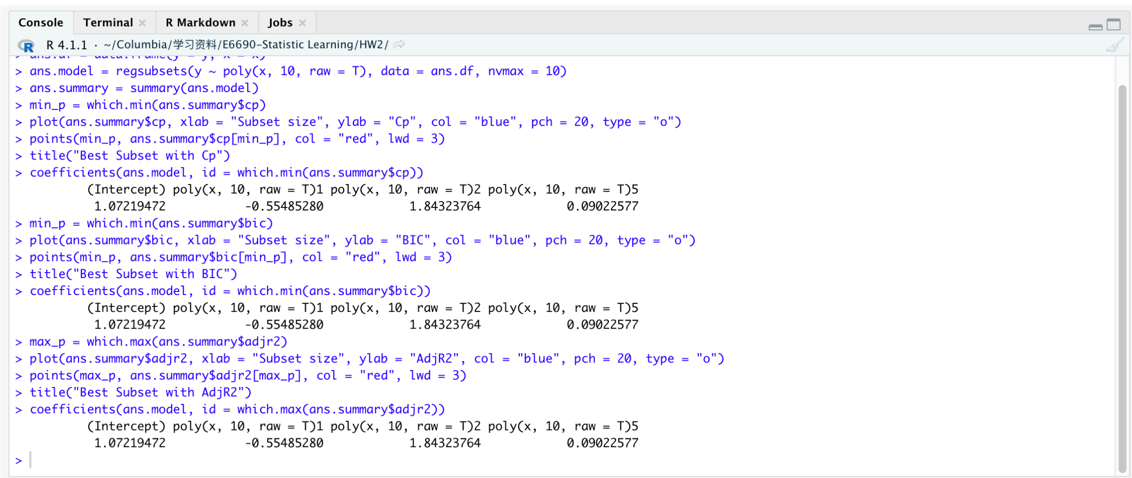


Figure 4.3: Screenshot for *'Best Subset with Adjust R^2'*



```
Console  Terminal  R Markdown  Jobs
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
> ans.model = regsubsets(y ~ poly(x, 10, raw = T), data = ans.df, nvmax = 10)
> ans.summary = summary(ans.model)
> min_p = which.min(ans.summary$cp)
> plot(ans.summary$cp, xlab = "Subset size", ylab = "Cp", col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
> title("Best Subset with Cp")
> coefficients(ans.model, id = which.min(ans.summary$cp))
         (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5
          1.07219472           -0.55485280            1.84323764            0.09022577
> min_p = which.min(ans.summary$bic)
> plot(ans.summary$bic, xlab = "Subset size", ylab = "BIC", col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
> title("Best Subset with BIC")
> coefficients(ans.model, id = which.min(ans.summary$bic))
         (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5
          1.07219472           -0.55485280            1.84323764            0.09022577
> max_p = which.max(ans.summary$adjr2)
> plot(ans.summary$adjr2, xlab = "Subset size", ylab = "AdjR2", col = "blue", pch = 20, type = "o")
> points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
> title("Best Subset with AdjR2")
> coefficients(ans.model, id = which.max(ans.summary$adjr2))
         (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5
          1.07219472           -0.55485280            1.84323764            0.09022577
>
```

Figure 4.4: Screenshot for *'Coefficients'*

4.4 d

If we use forward selection:

```
library(leaps)
ans.df = data.frame(y = y, x = x)
ans.model = regsubsets(y ~ poly(x, 10, raw = T),
                         data = ans.df, nvmax = 10, method = "forward")
ans.summary = summary(ans.model)

min_p = which.min(ans.summary$cp)
plot(ans.summary$cp, xlab = "Subset size", ylab = "Cp",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
title("Best Subset with Cp using 'forward'")
coefficients(ans.model, id = which.min(ans.summary$cp))

min_p = which.min(ans.summary$bic)
plot(ans.summary$bic, xlab = "Subset size", ylab = "BIC",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
title("Best Subset with BIC using 'forward'")
coefficients(ans.model, id = which.min(ans.summary$bic))

max_p = which.max(ans.summary$adjr2)
plot(ans.summary$adjr2, xlab = "Subset size", ylab = "AdjR2",
     col = "blue", pch = 20, type = "o")
points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
title("Best Subset with AdjR2 using 'forward'")
coefficients(ans.model, id = which.max(ans.summary$adjr2))
```

Forward stepwise uses a X^5 and a X^7 term. $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$ do not change a lot.

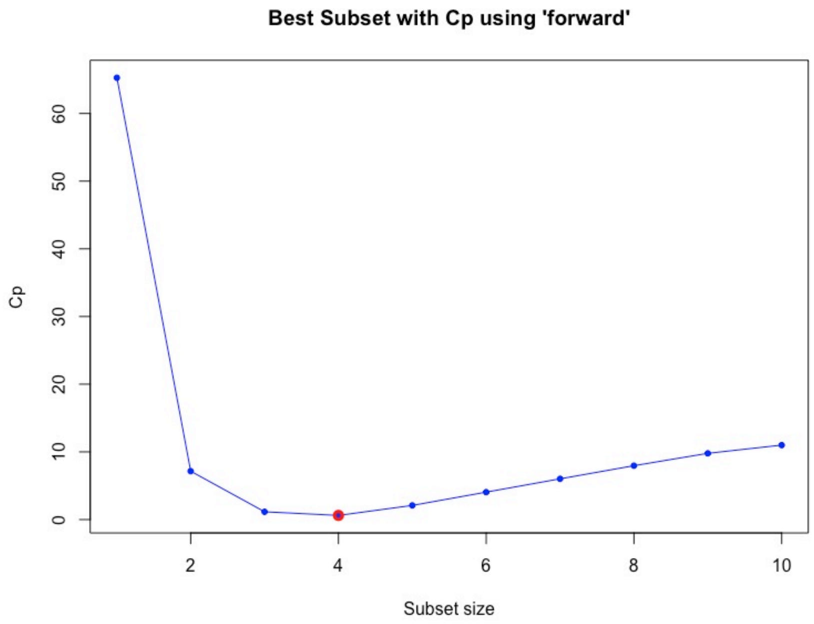


Figure 4.5: Screenshot for *'Best Subset with C_p using 'forward' '*

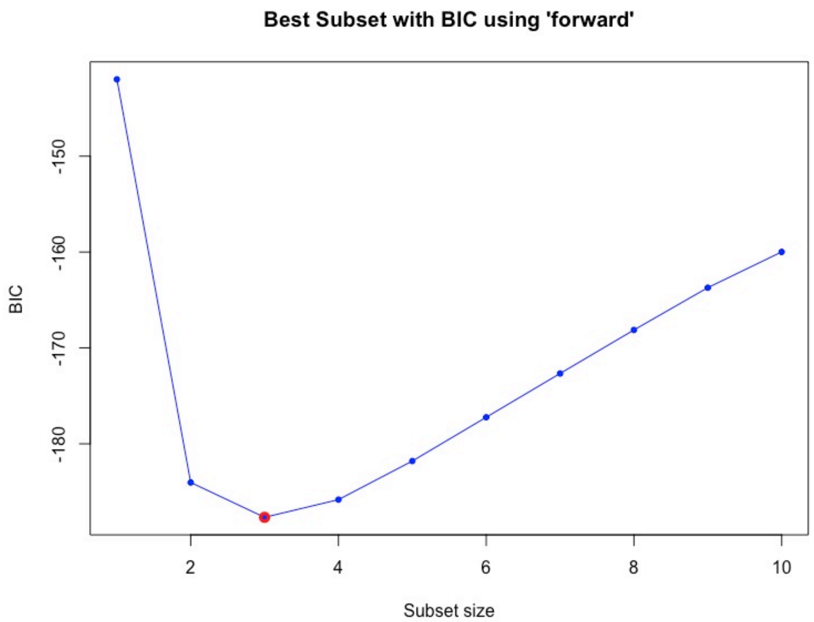


Figure 4.6: Screenshot for *'Best Subset with BIC using 'forward' '*

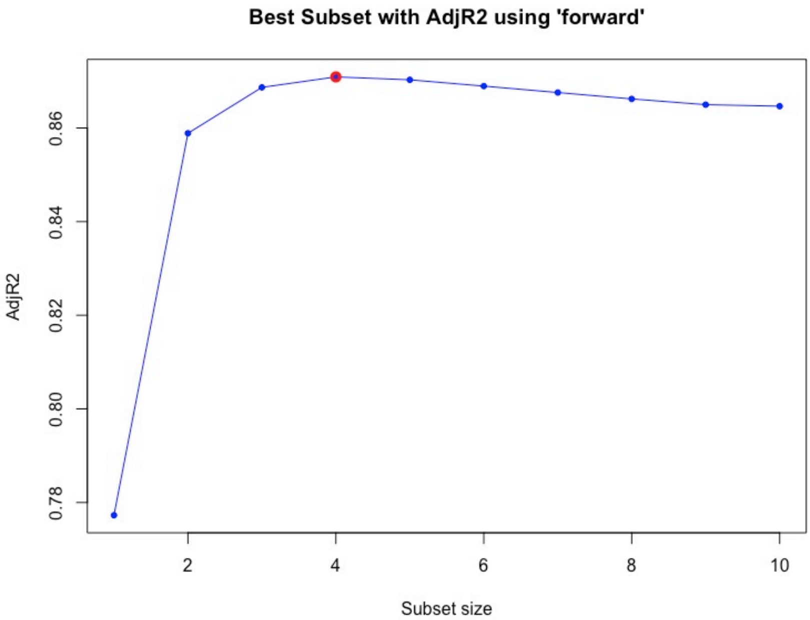


Figure 4.7: Screenshot for *'Best Subset with Adjust R^2 using 'forward''*

```
Console   Terminal ×   R Markdown ×   Jobs ×
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
> plot(ans.summary$cp, xlab = "Subset size", ylab = "Cp",
+      col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
> title("Best Subset with Cp using 'forward'")
> coefficients(ans.model, id = which.min(ans.summary$cp))
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5 poly(x, 10, raw = T)7
        1.070684876          -0.581441777           1.847950163           0.104710260          -0.002672665
> min_p = which.min(ans.summary$bic)
> plot(ans.summary$bic, xlab = "Subset size", ylab = "BIC",
+      col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
> title("Best Subset with BIC using 'forward'")
> coefficients(ans.model, id = which.min(ans.summary$bic))
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
         1.07636566           -0.36989702            1.82056631            0.01610623
> max_p = which.max(ans.summary$adjr2)
> plot(ans.summary$adjr2, xlab = "Subset size", ylab = "AdjR2",
+      col = "blue", pch = 20, type = "o")
> points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
> title("Best Subset with AdjR2 using 'forward'")
> coefficients(ans.model, id = which.max(ans.summary$adjr2))
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5 poly(x, 10, raw = T)7
        1.070684876          -0.581441777           1.847950163           0.104710260          -0.002672665
> |
```

Figure 4.8: Screenshot for *'Coefficients using 'forward''*

If we use backward selection:

```r
library(leaps)
ans.df = data.frame(y = y, x = x)
ans.model = regsubsets(y ~ poly(x, 10, raw = T),
                       data = ans.df, nvmax = 10, method = "backward")
ans.summary = summary(ans.model)

min_p = which.min(ans.summary$cp)
plot(ans.summary$cp, xlab = "Subset size", ylab = "Cp",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
title("Best Subset with Cp using 'backward'")
coefficients(ans.model, id = which.min(ans.summary$cp))

min_p = which.min(ans.summary$bic)
plot(ans.summary$bic, xlab = "Subset size", ylab = "BIC",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
title("Best Subset with BIC using 'backward'")
coefficients(ans.model, id = which.min(ans.summary$bic))

max_p = which.max(ans.summary$adjr2)
plot(ans.summary$adjr2, xlab = "Subset size", ylab = "AdjR2",
     col = "blue", pch = 20, type = "o")
points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
title("Best Subset with AdjR2 using 'backward'")
coefficients(ans.model, id = which.max(ans.summary$adjr2))
```

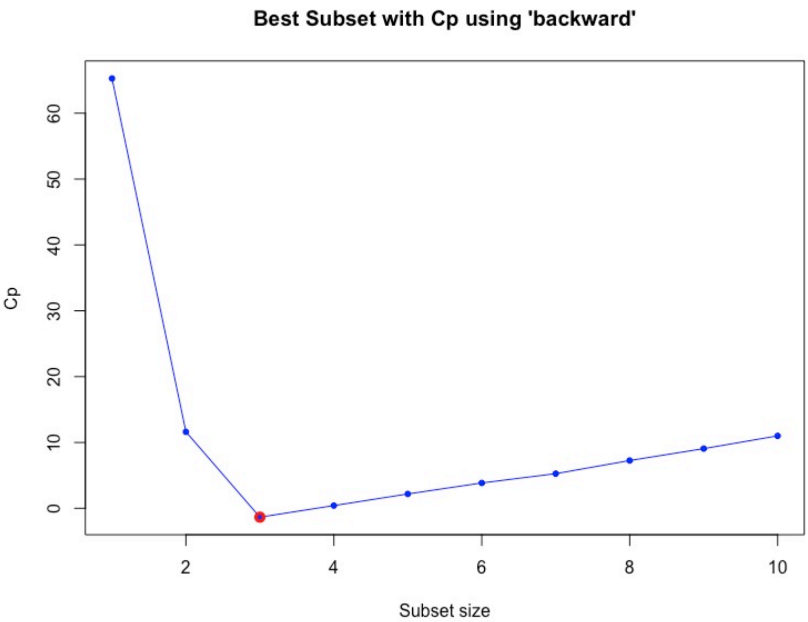


Figure 4.9: Screenshot for *'Best Subset with C_p using 'backward''*

Backward stepwise uses a X^5 term. $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$ do not change a lot.

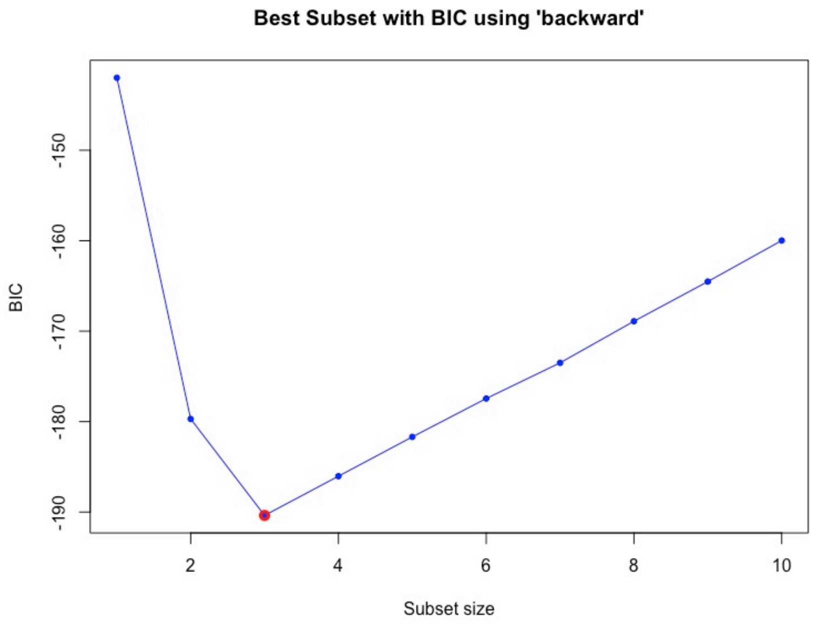


Figure 4.10: Screenshot for *'Best Subset with BIC using 'backward' '*

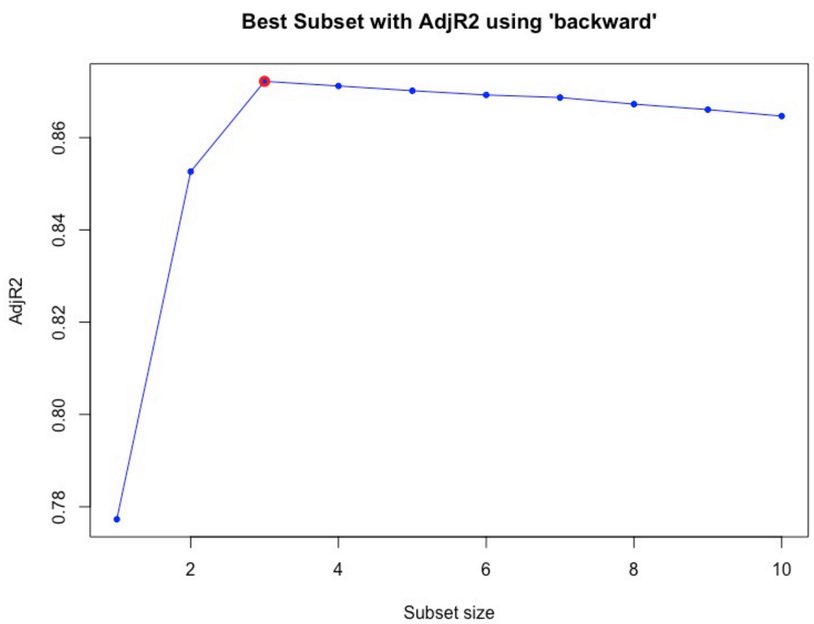
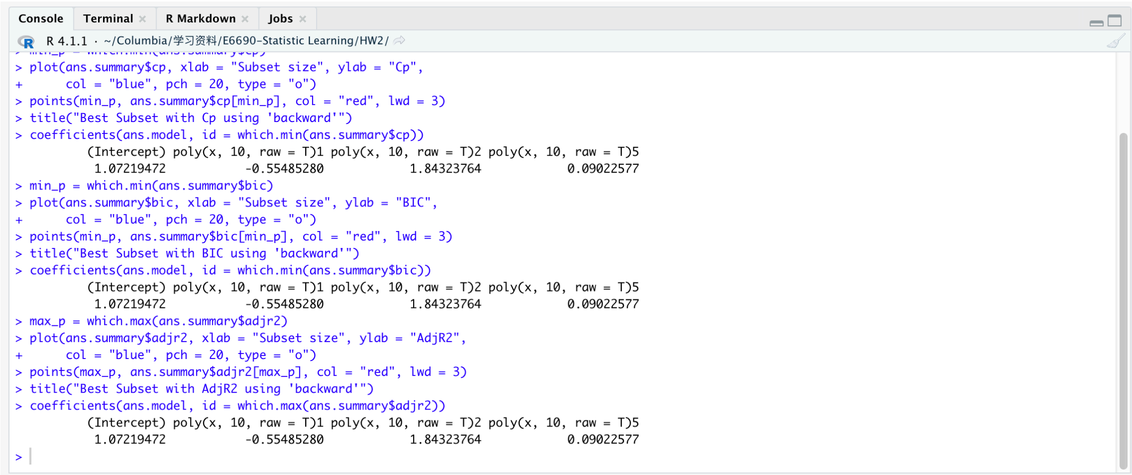


Figure 4.11: Screenshot for *'Best Subset with Adjust R^2 using 'backward' '*



```
Console  Terminal  R Markdown  Jobs
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
> plot(ans.summary$cp, xlab = "Subset size", ylab = "Cp",
+      col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
> title("Best Subset with Cp using 'backward'")
> coefficients(ans.model, id = which.min(ans.summary$cp))
         (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5
          1.07219472           -0.55485280            1.84323764            0.09022577
> min_p = which.min(ans.summary$bic)
> plot(ans.summary$bic, xlab = "Subset size", ylab = "BIC",
+      col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
> title("Best Subset with BIC using 'backward'")
> coefficients(ans.model, id = which.min(ans.summary$bic))
         (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5
          1.07219472           -0.55485280            1.84323764            0.09022577
> max_p = which.max(ans.summary$adjr2)
> plot(ans.summary$adjr2, xlab = "Subset size", ylab = "AdjR2",
+      col = "blue", pch = 20, type = "o")
> points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
> title("Best Subset with AdjR2 using 'backward'")
> coefficients(ans.model, id = which.max(ans.summary$adjr2))
         (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)5
          1.07219472           -0.55485280            1.84323764            0.09022577
>
```

Figure 4.12: Screenshot for *'Coefficients using 'backward''*

4.5 e

```r
library(glmnet)
xmat = model.matrix(y ~ poly(x, 10, raw = T), data = ans.df)[, -1]
ans.lasso = cv.glmnet(xmat, y, alpha = 1)
ans.lambda = ans.lasso$lambda.min
ans.lambda
plot(ans.lasso)

ans.lasso = glmnet(xmat, y, alpha = 1)
predict(ans.lasso, s = ans.lambda, type = "coefficients")
```

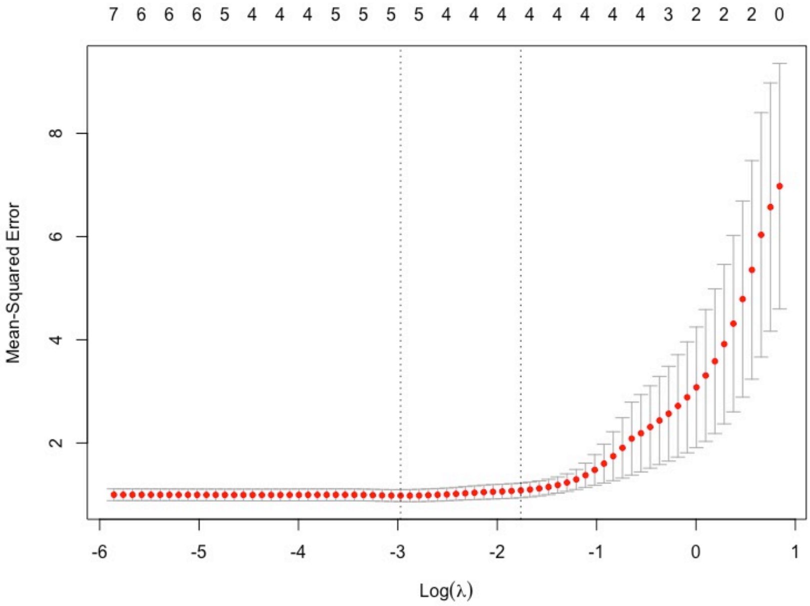
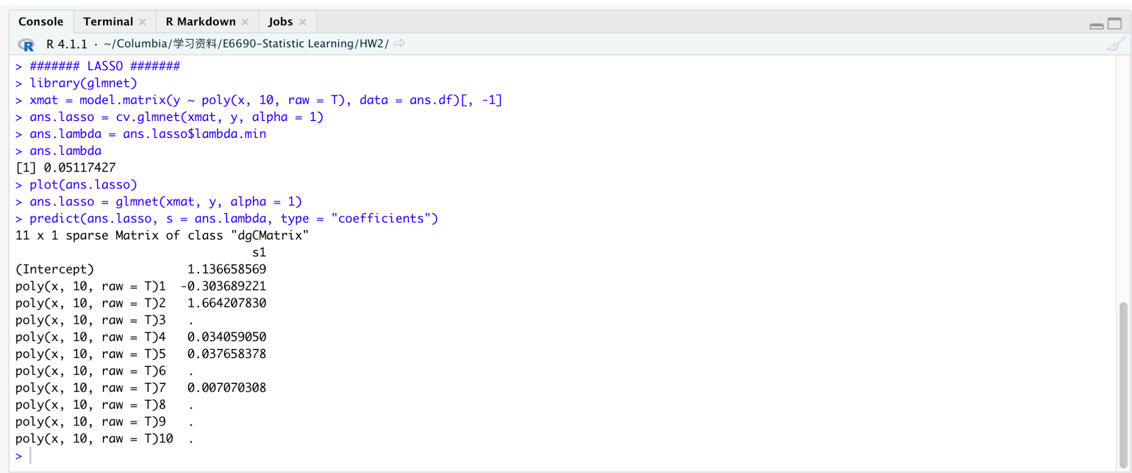


Figure 4.13: Screenshot for *'LASSO Cross-validation'*



```
Console  Terminal  R Markdown  Jobs
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
> ####### LASSO #######
> library(glmnet)
> xmat = model.matrix(y ~ poly(x, 10, raw = T), data = ans.df)[, -1]
> ans.lasso = cv.glmnet(xmat, y, alpha = 1)
> ans.lambda = ans.lasso$lambda.min
> ans.lambda
[1] 0.05117427
> plot(ans.lasso)
> ans.lasso = glmnet(xmat, y, alpha = 1)
> predict(ans.lasso, s = ans.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                                s1
(Intercept)             1.136658569
poly(x, 10, raw = T)1  -0.303689221
poly(x, 10, raw = T)2   1.664207830
poly(x, 10, raw = T)3   .
poly(x, 10, raw = T)4   0.034059050
poly(x, 10, raw = T)5   0.037658378
poly(x, 10, raw = T)6   .
poly(x, 10, raw = T)7   0.007070308
poly(x, 10, raw = T)8   .
poly(x, 10, raw = T)9   .
poly(x, 10, raw = T)10  .
>
```

Figure 4.14: Screenshot for *'Coefficients'*

LASSO uses a X^4 and X^5 term. $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$ do not change a lot.

4.6 f

```
beta_7 = 3
yy = beta[1] * beta_7*x^7 + eps
ans.df = data.frame(y = yy, x = x)
ans.model = regsubsets(y ~ poly(x, 10, raw = T),
                       data = ans.df, nvmax = 10)
ans.summary = summary(ans.model)

min_p = which.min(ans.summary$cp)
plot(ans.summary$cp, xlab = "Subset␣size", ylab = "Cp",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
title("Best␣Subset␣with␣Cp")
coefficients(ans.model, id = which.min(ans.summary$cp))

min_p = which.min(ans.summary$bic)
plot(ans.summary$bic, xlab = "Subset␣size", ylab = "BIC",
     col = "blue", pch = 20, type = "o")
points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
title("Best␣Subset␣with␣BIC")
coefficients(ans.model, id = which.min(ans.summary$bic))

max_p = which.max(ans.summary$adjr2)
plot(ans.summary$adjr2, xlab = "Subset␣size", ylab = "AdjR2",
     col = "blue", pch = 20, type = "o")
points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
title("Best␣Subset␣with␣AdjR2")
coefficients(ans.model, id = which.max(ans.summary$adjr2))
```

Among all three criterion, BIC picks up the best estimation of the true model.

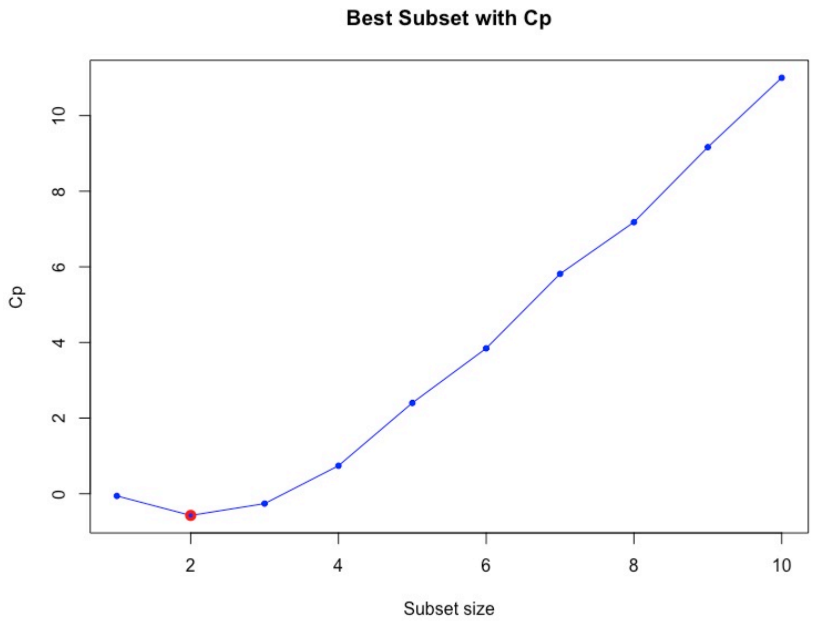


Figure 4.15: Screenshot for *'Best Subset with C_p'*

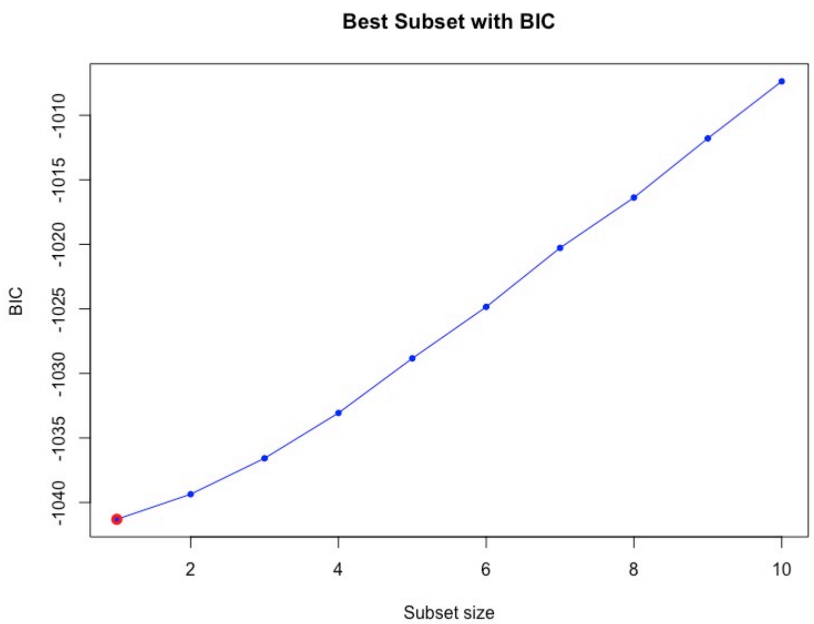


Figure 4.16: Screenshot for *'Best Subset with BIC'*

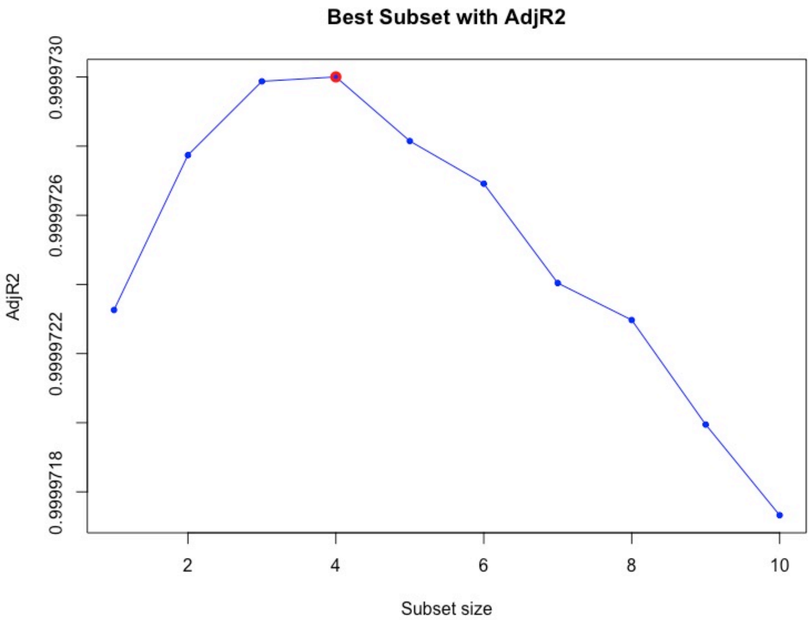
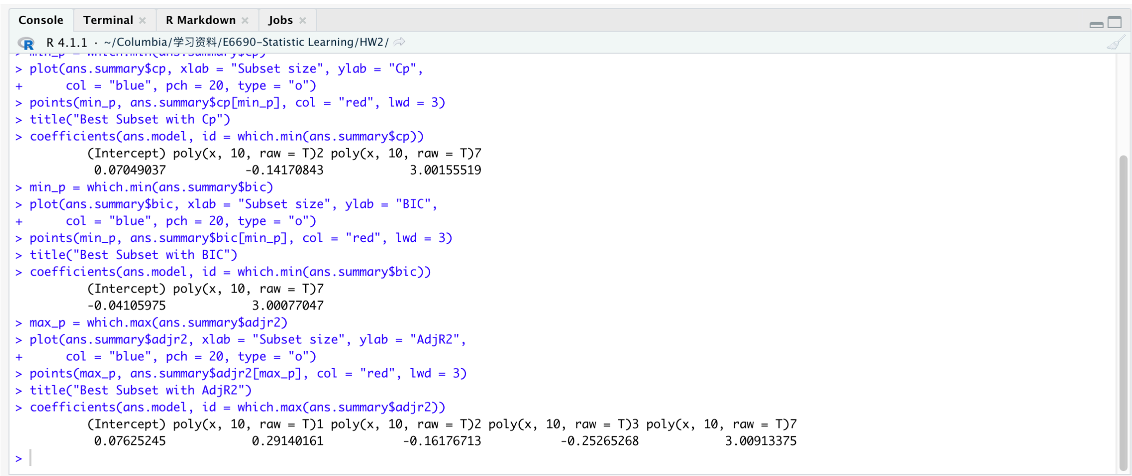


Figure 4.17: Screenshot for *’Best Subset with Adjust* R^2*’*



```
Console  Terminal  R Markdown  Jobs
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
> plot(ans.summary$cp, xlab = "Subset size", ylab = "Cp",
+      col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$cp[min_p], col = "red", lwd = 3)
> title("Best Subset with Cp")
> coefficients(ans.model, id = which.min(ans.summary$cp))
        (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
         0.07049037           -0.14170843            3.00155519
> min_p = which.min(ans.summary$bic)
> plot(ans.summary$bic, xlab = "Subset size", ylab = "BIC",
+      col = "blue", pch = 20, type = "o")
> points(min_p, ans.summary$bic[min_p], col = "red", lwd = 3)
> title("Best Subset with BIC")
> coefficients(ans.model, id = which.min(ans.summary$bic))
        (Intercept) poly(x, 10, raw = T)7
        -0.04105975            3.00077047
> max_p = which.max(ans.summary$adjr2)
> plot(ans.summary$adjr2, xlab = "Subset size", ylab = "AdjR2",
+      col = "blue", pch = 20, type = "o")
> points(max_p, ans.summary$adjr2[max_p], col = "red", lwd = 3)
> title("Best Subset with AdjR2")
> coefficients(ans.model, id = which.max(ans.summary$adjr2))
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
         0.07625245            0.29140161           -0.16176713           -0.25265268            3.00913375
>
```

Figure 4.18: Screenshot for *’Coefficients’*

```r
xmat = model.matrix(y ~ poly(x, 10, raw = T), data = ans.df)[, -1]
ans.lasso = cv.glmnet(xmat, y, alpha = 1)
ans.lambda = ans.lasso$lambda.min
ans.lambda
plot(ans.lasso)

ans.lasso = glmnet(xmat, y, alpha = 1)
predict(ans.lasso, s = ans.lambda, type = "coefficients")
```

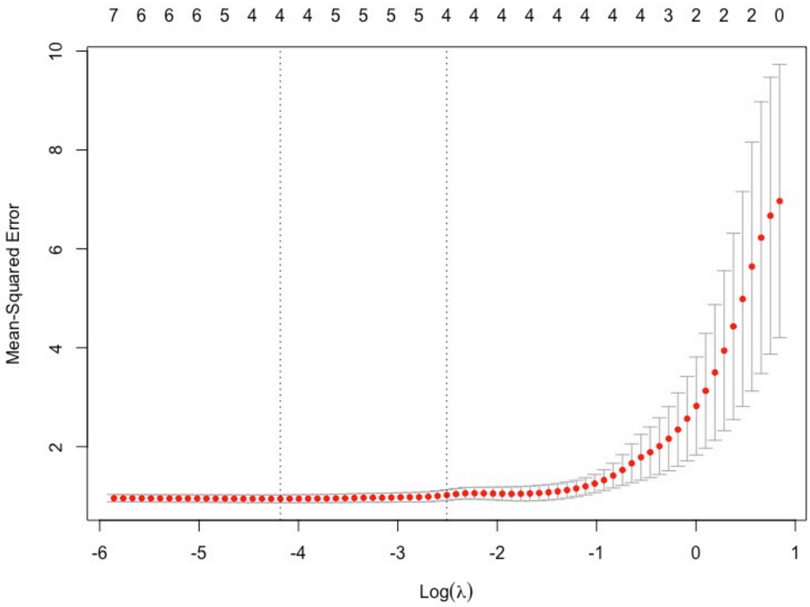
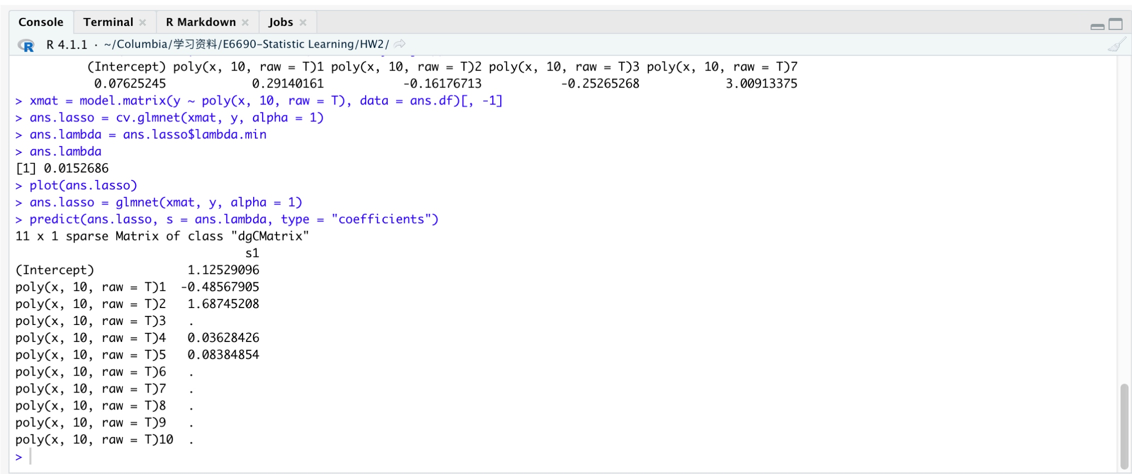


Figure 4.19: Screenshot for *'LASSO Cross-validation'*



```
Console  Terminal  R Markdown  Jobs
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
         0.07625245            0.29140161           -0.16176713           -0.25265268            3.00913375
> xmat = model.matrix(y ~ poly(x, 10, raw = T), data = ans.df)[, -1]
> ans.lasso = cv.glmnet(xmat, y, alpha = 1)
> ans.lambda = ans.lasso$lambda.min
> ans.lambda
[1] 0.0152686
> plot(ans.lasso)
> ans.lasso = glmnet(xmat, y, alpha = 1)
> predict(ans.lasso, s = ans.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                               s1
(Intercept)            1.12529096
poly(x, 10, raw = T)1 -0.48567905
poly(x, 10, raw = T)2  1.68745208
poly(x, 10, raw = T)3  .
poly(x, 10, raw = T)4  0.03628426
poly(x, 10, raw = T)5  0.08384854
poly(x, 10, raw = T)6  .
poly(x, 10, raw = T)7  .
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  .
poly(x, 10, raw = T)10 .
>
```

Figure 4.20: Screenshot for *'Coefficients'*

LASSO gives a best estimation of the true model. Only one variable besides the Intercept is given in the estimation.

5 P5

5.1 a

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

5.2 b

```
College.lm = lm(Apps~., data = College.train)
summary(College.lm)

College.pred = predict(College.lm, College.test)
College.error = mean((College.test$Apps - College.pred)^2)
```

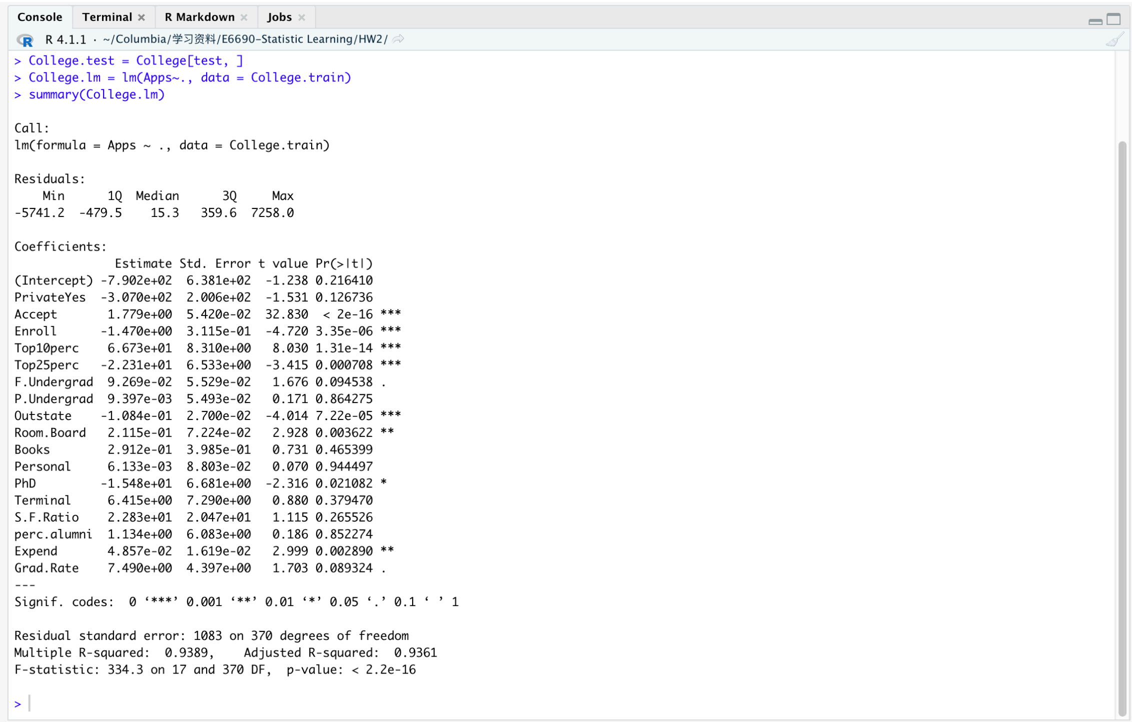


Figure 5.1: Screenshot for *'Model summary'*

The error for test dataset is 1135758.

5.3 c

```
train_mat = model.matrix(Apps ~ ., data = College.train)
test_mat = model.matrix(Apps ~ ., data = College.test)
College.ridge = cv.glmnet(train_mat, College.train$Apps, alpha = 0)
plot(College.ridge)
```

```
College.lambda = College.ridge$lambda.min
College.pred = predict(College.ridge, newx=test_mat, s=College.lambda)
College.error = mean((College.test$Apps - College.pred)^2)
```

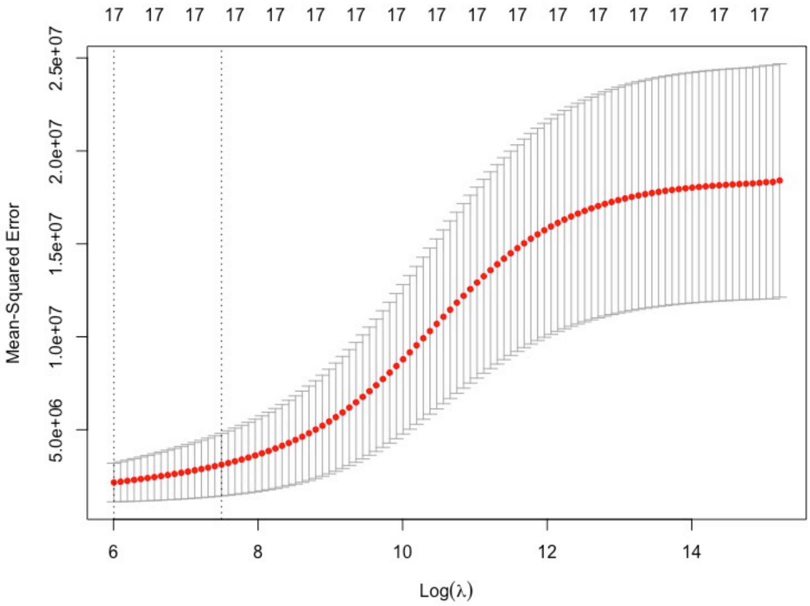


Figure 5.2: Screenshot for *'Ridge Regression'*

The error for test dataset is 976261.5. Ridge Regression get a better result than Linear Regression.

5.4 d

```
College.lasso = cv.glmnet(train_mat, College.train$Apps, alpha = 1)
plot(College.lasso)
College.lambda = College.lasso$lambda.min
College.pred = predict(College.lasso, newx=test_mat, s=College.lambda)
College.error = mean((College.test$Apps - College.pred)^2)
```

The error for test dataset is 1115901. LASSO Regression get a better result than Linear Regression but not as good as Ridge Regression.

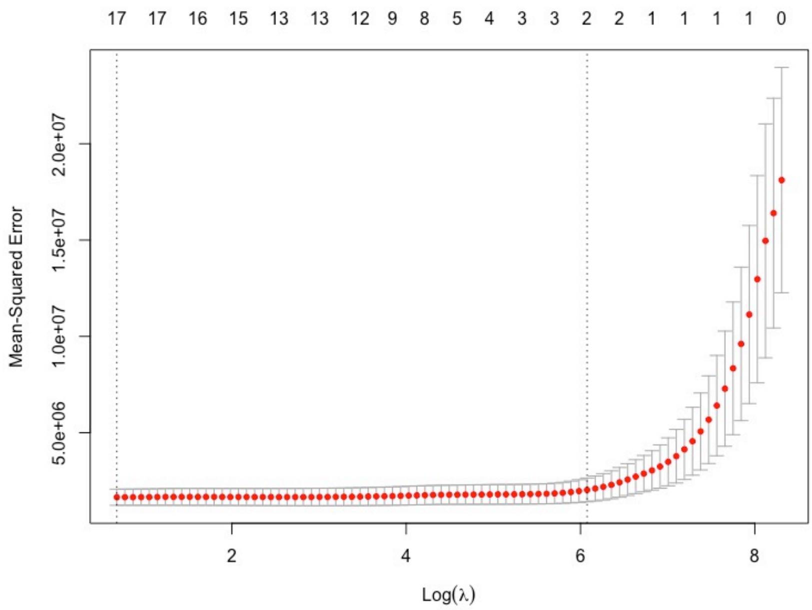


Figure 5.3: Screenshot for *'LASSO Regression'*

6 P6

6.1 a

```
set.seed(1)
n = 1000
p = 20

xmat = matrix(rnorm(n*p), n, p)
eps = rnorm(n)

beta = rnorm(p)
beta[3] = 0
beta[6] = 0
beta[9] = 0
beta[12] = 0
beta[15] = 0
beta[18] = 0


y = xmat %*% beta + eps
```

I set $\beta_3, \beta_6, \beta_9, \beta_{12}, \beta_{15}, \beta_{18}$ to be zero.

6.2 b

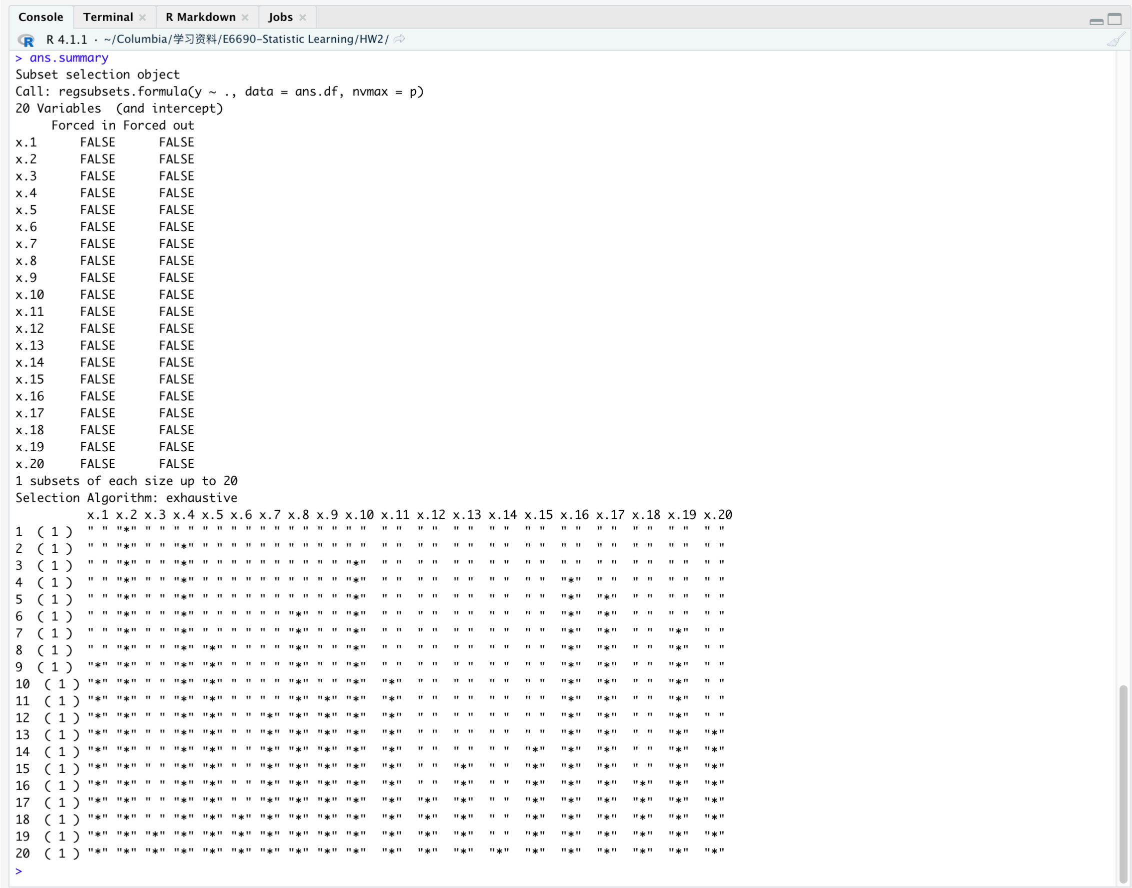
```
train = sample(1: n, 100)
test = -train
xmat.train = xmat[train, ]
xmat.test = xmat[test, ]
y.train = y[train, ]
y.test = y[test, ]
```

6.3 c

```
ans.df = data.frame(x = xmat.train, y = y.train)
ans.model = regsubsets(y ~ ., data = ans.df, nvmax = p)
ans.summary = summary(ans.model)
ans.summary

ans.train_mse = rep(NA, p)
ans.xcols = colnames(xmat, do.NULL = F, prefix = "x.")
for(i in 1:p){
  c_i = coef(ans.model, id = i)
  if(i > 1){
    y.train.pred = as.matrix(xmat.train[, ans.xcols %in% names(c_i)] %*%
                             c_i[names(c_i) %in% ans.xcols])
  }
  else
  {
    y.train.pred = as.matrix(xmat.train[, ans.xcols %in% names(c_i)] *
                             c_i[names(c_i) %in% ans.xcols])
  }
```

```r
  ans.train_mse[i] = mean((y.train-y.train.pred)^2)
}
plot(ans.train_mse, ylab = "Training␣MSE",xlab = "Subset␣Size",
      pch = 20, type = "o", col = "blue")
title("MSE-Subset␣size␣Curve")
```



```
Console  Terminal  R Markdown  Jobs
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
> ans.summary
Subset selection object
Call: regsubsets.formula(y ~ ., data = ans.df, nvmax = p)
20 Variables  (and intercept)
     Forced in Forced out
x.1      FALSE      FALSE
x.2      FALSE      FALSE
x.3      FALSE      FALSE
x.4      FALSE      FALSE
x.5      FALSE      FALSE
x.6      FALSE      FALSE
x.7      FALSE      FALSE
x.8      FALSE      FALSE
x.9      FALSE      FALSE
x.10     FALSE      FALSE
x.11     FALSE      FALSE
x.12     FALSE      FALSE
x.13     FALSE      FALSE
x.14     FALSE      FALSE
x.15     FALSE      FALSE
x.16     FALSE      FALSE
x.17     FALSE      FALSE
x.18     FALSE      FALSE
x.19     FALSE      FALSE
x.20     FALSE      FALSE
1 subsets of each size up to 20
Selection Algorithm: exhaustive
          x.1 x.2 x.3 x.4 x.5 x.6 x.7 x.8 x.9 x.10 x.11 x.12 x.13 x.14 x.15 x.16 x.17 x.18 x.19 x.20
1  ( 1 )  " " "*" " " " " " " " " " " " " " " " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
2  ( 1 )  " " "*" " " "*" " " " " " " " " " " " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
3  ( 1 )  " " "*" " " "*" " " " " " " " " " " "*"  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
4  ( 1 )  " " "*" " " "*" " " " " " " " " " " "*"  " "  " "  " "  " "  " "  "*"  " "  " "  " "  " "
5  ( 1 )  " " "*" " " "*" " " " " " " " " " " "*"  " "  " "  " "  " "  " "  "*"  "*"  " "  " "  " "
6  ( 1 )  " " "*" " " "*" " " " " " " "*" " " "*"  " "  " "  " "  " "  " "  "*"  "*"  " "  " "  " "
7  ( 1 )  " " "*" " " "*" " " " " " " "*" " " "*"  " "  " "  " "  " "  " "  "*"  "*"  " "  "*"  " "
8  ( 1 )  " " "*" " " "*" "*" " " " " "*" " " "*"  " "  " "  " "  " "  " "  "*"  "*"  " "  "*"  " "
9  ( 1 )  "*" "*" " " "*" "*" " " " " "*" " " "*"  " "  " "  " "  " "  " "  "*"  "*"  " "  "*"  " "
10  ( 1 ) "*" "*" " " "*" "*" " " " " "*" " " "*"  "*"  " "  " "  " "  " "  "*"  "*"  " "  "*"  " "
11  ( 1 ) "*" "*" " " "*" "*" " " " " "*" "*" "*"  "*"  " "  " "  " "  " "  "*"  "*"  " "  "*"  " "
12  ( 1 ) "*" "*" " " "*" "*" " " "*" "*" "*" "*"  "*"  " "  " "  " "  " "  "*"  "*"  " "  "*"  " "
13  ( 1 ) "*" "*" " " "*" "*" " " "*" "*" "*" "*"  "*"  " "  " "  " "  " "  "*"  "*"  " "  "*"  "*"
14  ( 1 ) "*" "*" " " "*" "*" " " "*" "*" "*" "*"  "*"  " "  " "  " "  "*"  "*"  "*"  " "  "*"  "*"
15  ( 1 ) "*" "*" " " "*" "*" " " "*" "*" "*" "*"  "*"  " "  "*"  " "  "*"  "*"  "*"  " "  "*"  "*"
16  ( 1 ) "*" "*" " " "*" "*" " " "*" "*" "*" "*"  "*"  " "  "*"  " "  "*"  "*"  "*"  "*"  "*"  "*"
17  ( 1 ) "*" "*" " " "*" "*" " " "*" "*" "*" "*"  "*"  "*"  "*"  " "  "*"  "*"  "*"  "*"  "*"  "*"
18  ( 1 ) "*" "*" " " "*" "*" "*" "*" "*" "*" "*"  "*"  "*"  "*"  " "  "*"  "*"  "*"  "*"  "*"  "*"
19  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"  "*"  "*"  "*"  " "  "*"  "*"  "*"  "*"  "*"  "*"
20  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"
>
```

Figure 6.1: Screenshot for *'Model summary'*

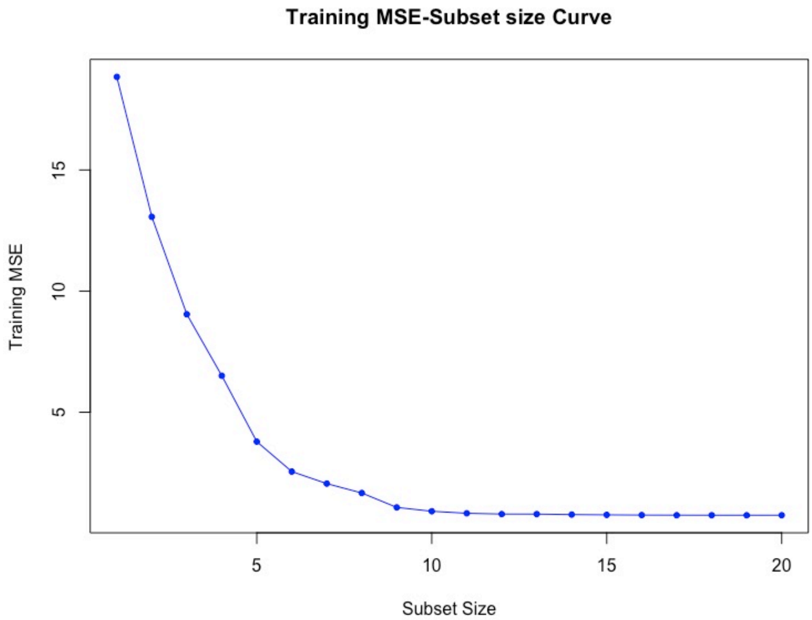


Figure 6.2: Screenshot for *'Training MSE-Subset size Curve'*

6.4 d

```
ans.test_mse = rep(NA, p)
ans.xcols = colnames(xmat, do.NULL = F, prefix = "x.")
for(i in 1:p){
  c_i = coef(ans.model, id = i)
  if(i > 1){
    y.test.pred = as.matrix(xmat.test[, ans.xcols %in% names(c_i)] %*%
                              c_i[names(c_i) %in% ans.xcols])
  }
  else
  {
    y.test.pred = as.matrix(xmat.test[, ans.xcols %in% names(c_i)] *
                              c_i[names(c_i) %in% ans.xcols])
  }
  ans.test_mse[i] = mean((y.test-y.test.pred)^2)
}
plot(ans.test_mse, ylab = "Testing MSE",xlab = "Subset Size",
     pch = 20, type = "o", col = "blue")
title("Testing MSE-Subset size Curve")
```

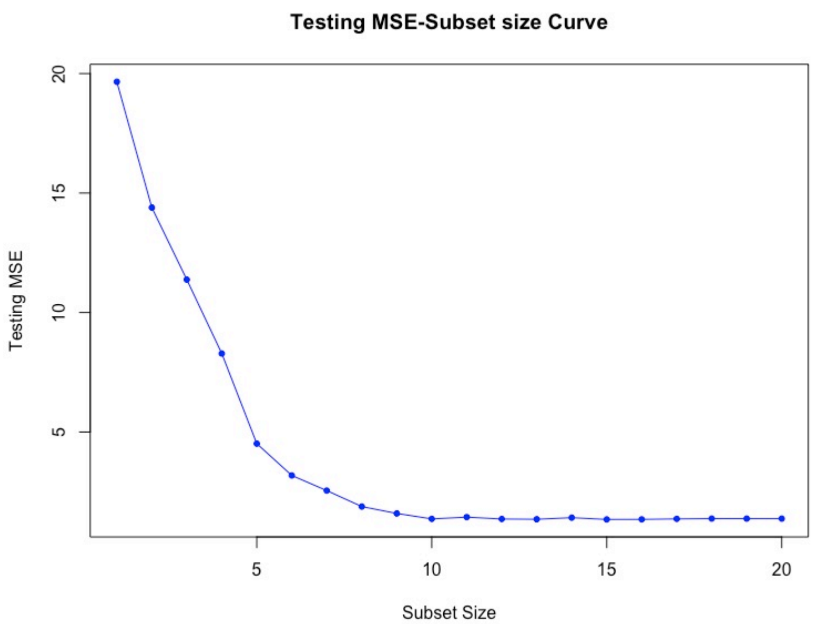


Figure 6.3: Screenshot for *'Testing MSE-Subset size Curve'*

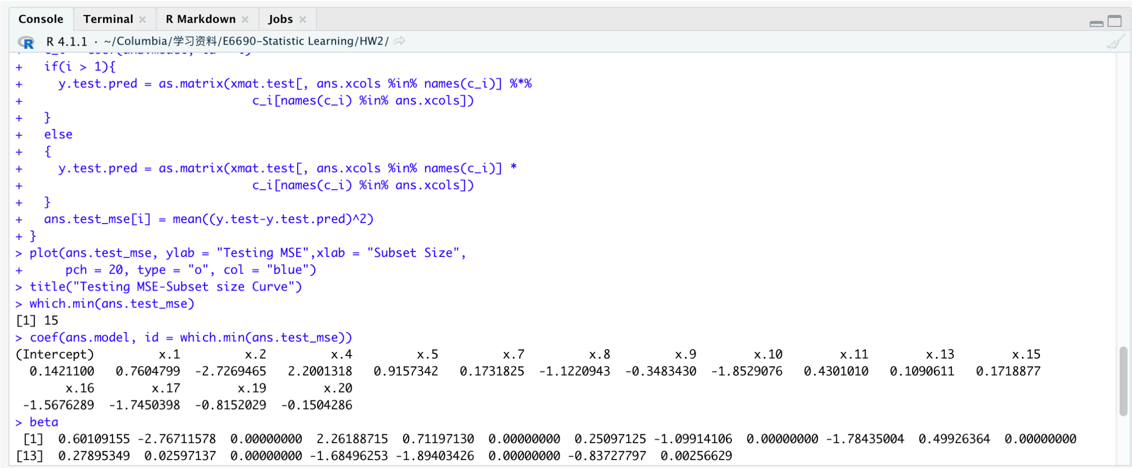
6.5 e

```
which.min(ans.test_mse)
```

The return of this function is 15, which means when there are 15 predictors in the estimated model, the model reaches the smallest test error.

6.6 f

```
coef(ans.model, id = which.min(ans.test_mse))
beta
```



```
Console  Terminal  R Markdown  Jobs
R 4.1.1 · ~/Columbia/学习资料/E6690-Statistic Learning/HW2/
+   if(i > 1){
+     y.test.pred = as.matrix(xmat.test[, ans.xcols %in% names(c_i)] %*%
+                                 c_i[names(c_i) %in% ans.xcols])
+   }
+   else
+   {
+     y.test.pred = as.matrix(xmat.test[, ans.xcols %in% names(c_i)] *
+                                 c_i[names(c_i) %in% ans.xcols])
+   }
+   ans.test_mse[i] = mean((y.test-y.test.pred)^2)
+ }
> plot(ans.test_mse, ylab = "Testing MSE",xlab = "Subset Size",
+      pch = 20, type = "o", col = "blue")
> title("Testing MSE-Subset size Curve")
> which.min(ans.test_mse)
[1] 15
> coef(ans.model, id = which.min(ans.test_mse))
(Intercept)         x.1         x.2         x.4         x.5         x.7         x.8         x.9        x.10        x.11        x.13        x.15 
  0.1421100   0.7604799  -2.7269465   2.2001318   0.9157342   0.1731825  -1.1220943  -0.3483430  -1.8529076   0.4301010   0.1090611   0.1718877 
       x.16        x.17        x.19        x.20 
 -1.5676289  -1.7450398  -0.8152029  -0.1504286 
> beta
 [1]  0.60109155 -2.76711578  0.00000000  2.26188715  0.71197130  0.00000000  0.25097125 -1.09914106  0.00000000 -1.78435004  0.49926364  0.00000000
[13]  0.27895349  0.02597137  0.00000000 -1.68496253 -1.89403426  0.00000000 -0.83727797  0.00256629
```

Figure 6.4: Screenshot for *'β value'*

The model finds that $\beta_3, \beta_6, \beta_{12}, \beta_{14}, \beta_{18}$ are zeros.

6.7 g

```
ans.beta_error = rep(NA, p)
ans.xcols = colnames(xmat, do.NULL = F, prefix = "x.")
for(i in 1:p){
  c_i = coef(ans.model, id = i)
  ans.beta_error[i] = sqrt(sum((beta[ans.xcols %in% names(c_i)] -
                                  c_i[names(c_i) %in% ans.xcols])^2) +
                              sum(beta[!(ans.xcols %in% names(c_i))]^2))
  }
plot(x = 1:p, ans.beta_error, xlab = "Coefficient #",
     ylab = "Beta error")
title("Beta error on each dimesion")
```

Compared with the result in (d), the model reaches the minimum beta error when there are 15 predictors. That's close to the result of 12 in (d). And when the number of predictors is 15, the β is also very small.

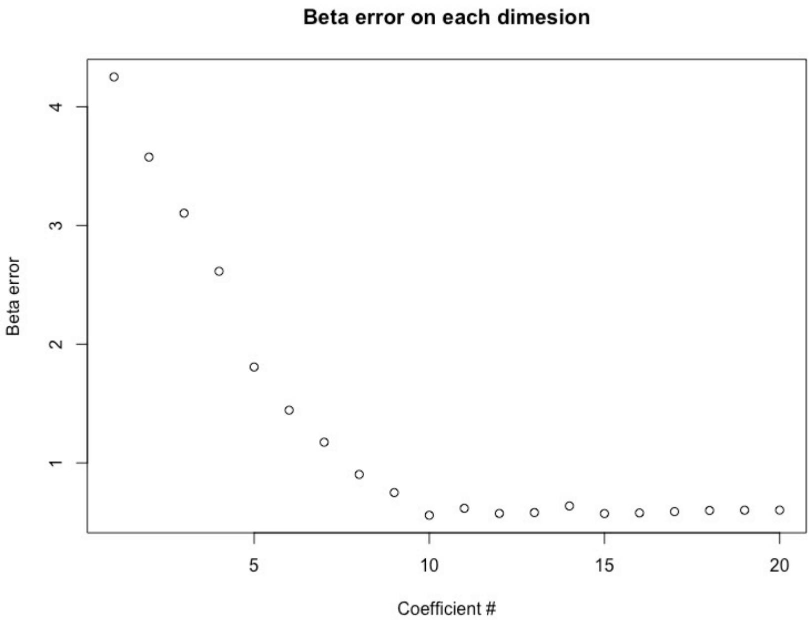


Figure 6.5: Screenshot for *'β error'*