EECS E6690 hw2

Chong Hu ch3467

Sep 22, 2019

P1

(a) For Ridge regression optimization problem:

$$\begin{aligned}\hat{\beta}_1, \hat{\beta}_2 &= \arg\min_{\beta_1,\beta_2} \mathbf{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2 \\ &= \arg\min_{\beta_1,\beta_2} (y_1 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_1 x_{21} - \beta_2 x_{22})^2 + \lambda\beta_1^2 + \lambda\beta_2^2 \\ &= \arg\min_{\beta_1,\beta_2} f(\beta_1, \beta_2)\end{aligned}$$

f is concave w.r.t β_1 and β_2. Therefore, to minimize f, we need to set

$$\frac{\partial f}{\partial \beta_1} = -2x_{11}(y_1 - \beta_1 x_{11} - \beta_2 x_{12}) - 2x_{21}(y_2 - \beta_1 x_{21} - \beta_2 x_{22}) + 2\lambda\beta_1 = 0 \quad (1)$$

$$\frac{\partial f}{\partial \beta_2} = -2x_{12}(y_1 - \beta_1 x_{11} - \beta_2 x_{12}) - 2x_{22}(y_2 - \beta_1 x_{21} - \beta_2 x_{22}) + 2\lambda\beta_2 = 0 \quad (2)$$

(b) For convenience, simply note $x_1 = x_{11} = x_{12}$ and $x_2 = x_{21} = x_{22}$.
From (1):

$$(x_1^2 + x_2^2 + \lambda)\beta_1 + (x_1^2 + x_2^2)\beta_2 = x_1 y_1 + x_2 y_2$$

From (2):

$$(x_1^2 + x_2^2)\beta_1 + (x_1^2 + x_2^2 + \lambda)\beta_2 = x_1 y_1 + x_2 y_2$$

Therefore,

$$\begin{gathered}(x_1^2 + x_2^2 + \lambda)\beta_1 + (x_1^2 + x_2^2)\beta_2 = (x_1^2 + x_2^2)\beta_1 + (x_1^2 + x_2^2 + \lambda)\beta_2 \\ \lambda\beta_1 = \lambda\beta_2\end{gathered}$$

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

(c) For Lasso regression optimization problem:

$$\begin{aligned}\hat{\beta}_1, \hat{\beta}_2 &= \arg\min_{\beta_1,\beta_2} \mathbf{RSS} + \lambda \sum_{j=1}^{p} |\beta_j| \\ &= \arg\min_{\beta_1,\beta_2} (y_1 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_1 x_{21} - \beta_2 x_{22})^2 + \lambda|\beta_1| + \lambda|\beta_2| \\ &= \arg\min_{\beta_1,\beta_2} g(\beta_1, \beta_2)\end{aligned}$$

(d) g is concave w.r.t β_1 and β_2. Therefore, to minimize f, we need to set

$$\frac{\partial g}{\partial \beta_1} = 0 \quad (3)$$

$$\frac{\partial g}{\partial \beta_2} = 0 \quad (4)$$

Based on the sign of β_1 and β_2, the partial derivative of g will be different. Here I simply use $\pm\lambda$ to represent those cases. In (3), it depends on β_1; in (4), it depends on β_2. When $\beta > 0$, it takes $+$; when $\beta < 0$, it takes $-$.

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

For convenience, simply note $x_1 = x_{11} = x_{12}$ and $x_2 = x_{21} = x_{22}$.

$$\begin{aligned}(x_1^2 + x_2^2)\beta_1 + (x_1^2 + x_2^2)\beta_2 = x_1 y_1 + x_2 y_2 \pm \lambda \\ (x_1^2 + x_2^2)\beta_1 + (x_1^2 + x_2^2)\beta_2 = x_1 y_1 + x_2 y_2 \pm \lambda\end{aligned}$$

which have solutions when both $\beta_1 > 0$ and $\beta_2 > 0$ or both $\beta_1 < 0$ and $\beta_2 < 0$. When the equations have solutions, there multiple solutions since two equations provide same constrain. When both $\beta_1 > 0$ and $\beta_2 > 0$, solutions are:

$$(x_1^2 + x_2^2)\beta_1 + (x_1^2 + x_2^2)\beta_2 = x_1 y_1 + x_2 y_2 - \lambda$$

When both $\beta_1 < 0$ and $\beta_2 < 0$, solutions are:

$$(x_1^2 + x_2^2)\beta_1 + (x_1^2 + x_2^2)\beta_2 = x_1 y_1 + x_2 y_2 - \lambda$$

P2

(a)

With $p = 1$, the loss will become $\left((y_1 - \beta_1)^2 + \lambda\beta_1^2\right)$. I plot this term with respect to β_1 for different values of y_1 and λ. From the plot we can clear see that, the β such that this term takes minimum is exactly the value of $\hat{\beta}_1^R$.

```
## [1] "for y_1 0.000000 lambda 0.100000: "
## [1] "beta ridge estimate : 0.000000"
## [1] "beta ridge from graph  : 0.005025"
## [1] "for y_1 0.000000 lambda 0.500000: "
## [1] "beta ridge estimate : 0.000000"
## [1] "beta ridge from graph  : 0.005025"
## [1] "for y_1 0.000000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.000000"
## [1] "beta ridge from graph  : 0.005025"
## [1] "for y_1 0.200000 lambda 0.100000: "
## [1] "beta ridge estimate : 0.181818"
## [1] "beta ridge from graph  : 0.180905"
## [1] "for y_1 0.200000 lambda 0.500000: "
## [1] "beta ridge estimate : 0.133333"
## [1] "beta ridge from graph  : 0.130653"
## [1] "for y_1 0.200000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.100000"
## [1] "beta ridge from graph  : 0.105528"
## [1] "for y_1 0.500000 lambda 0.100000: "
## [1] "beta ridge estimate : 0.454545"
## [1] "beta ridge from graph  : 0.457286"
## [1] "for y_1 0.500000 lambda 0.500000: "
## [1] "beta ridge estimate : 0.333333"
## [1] "beta ridge from graph  : 0.331658"
## [1] "for y_1 0.500000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.250000"
## [1] "beta ridge from graph  : 0.243719"
## [1] "for y_1 0.800000 lambda 0.100000: "
## [1] "beta ridge estimate : 0.727273"
## [1] "beta ridge from graph  : 0.721106"
## [1] "for y_1 0.800000 lambda 0.500000: "
## [1] "beta ridge estimate : 0.533333"
## [1] "beta ridge from graph  : 0.532663"
## [1] "for y_1 0.800000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.400000"
## [1] "beta ridge from graph  : 0.394472"
```

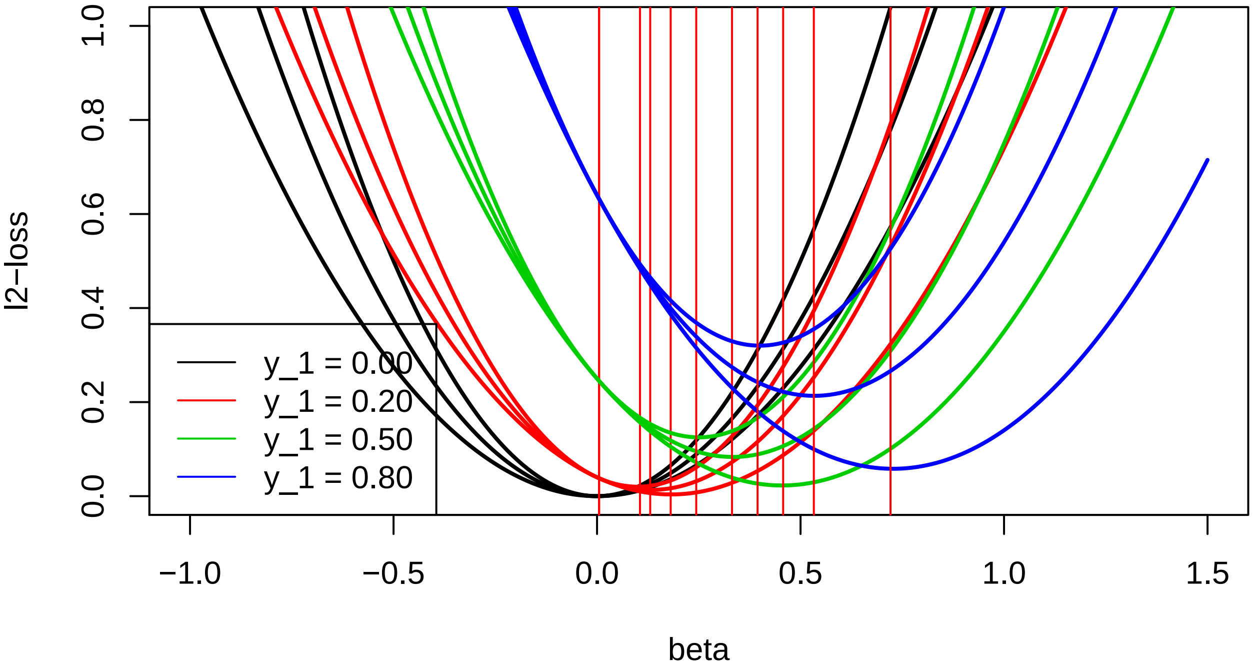

Figure 1: l2-Loss vs. β_1 for different y_1 and λ

(b)

With $p = 1$, the loss will become $\left((y_1 - \beta_1)^2 + \lambda|\beta_1|\right)$. I plot this term with respect to β_1 for different values of y_1 and λ. From the plot we can clear see that, the β such that this term takes minimum is exactly the value of $\hat{\beta}_1^L$.

```
## [1] "for y_1 -0.500000 lambda 0.100000: "
## [1] "beta ridge estimate : -0.450000"
## [1] "beta ridge from graph  : -0.447236"
## [1] "for y_1 -0.500000 lambda 0.500000: "
## [1] "beta ridge estimate : -0.250000"
## [1] "beta ridge from graph  : -0.246231"
## [1] "for y_1 -0.500000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.000000"
## [1] "beta ridge from graph  : -0.007538"
## [1] "for y_1 -0.200000 lambda 0.100000: "
## [1] "beta ridge estimate : -0.150000"
## [1] "beta ridge from graph  : -0.145729"
## [1] "for y_1 -0.200000 lambda 0.500000: "
## [1] "beta ridge estimate : 0.000000"
## [1] "beta ridge from graph  : -0.007538"
## [1] "for y_1 -0.200000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.000000"
## [1] "beta ridge from graph  : -0.007538"
## [1] "for y_1 0.500000 lambda 0.100000: "
## [1] "beta ridge estimate : 0.450000"
## [1] "beta ridge from graph  : 0.444724"
## [1] "for y_1 0.500000 lambda 0.500000: "
## [1] "beta ridge estimate : 0.250000"
## [1] "beta ridge from graph  : 0.243719"
## [1] "for y_1 0.500000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.000000"
## [1] "beta ridge from graph  : 0.005025"
## [1] "for y_1 0.800000 lambda 0.100000: "
## [1] "beta ridge estimate : 0.750000"
## [1] "beta ridge from graph  : 0.746231"
## [1] "for y_1 0.800000 lambda 0.500000: "
## [1] "beta ridge estimate : 0.550000"
## [1] "beta ridge from graph  : 0.545226"
## [1] "for y_1 0.800000 lambda 1.000000: "
## [1] "beta ridge estimate : 0.300000"
## [1] "beta ridge from graph  : 0.293970"
```

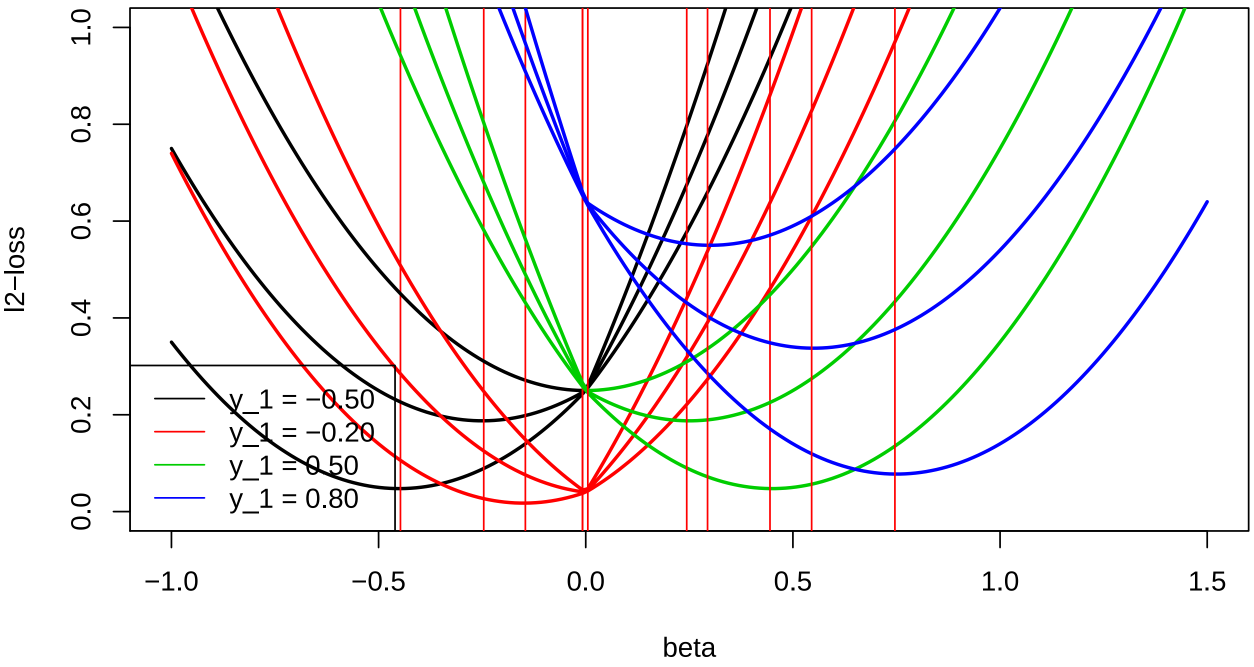

Figure 2: l1-Loss vs. β_1 for different y_1 and λ

P3

(a)

$$\begin{aligned}\text{likelihood} = \mathcal{L}(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma^2) &= \prod_{i=1}^{n} \mathbf{P}(y_i \mid \mathbf{x}_i, \boldsymbol{\beta}, \sigma^2) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Bigg(- \frac{1}{2\sigma^2}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 \Bigg) \\ &= (2\pi\sigma^2)^{-n/2} \exp\Bigg(- \frac{1}{2\sigma^2} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 \Bigg)\end{aligned}$$

(b)

$$\text{prior}(\boldsymbol{\beta}) = \frac{1}{2b} \exp{(-\frac{\sum_{j=1}^{p}|\beta_j|}{b})}$$

$$\begin{aligned}\text{posterior} &\propto \mathcal{L} \times \text{prior} \\ &\propto (2\pi\sigma^2)^{-n/2} \exp\Bigg(- \frac{1}{2\sigma^2} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 \Bigg) \times \frac{1}{2b} \exp{(-\frac{\sum_{j=1}^{p}|\beta_j|}{b})} \\ &\propto \frac{(2\pi\sigma^2)^{-n/2}}{2b} \exp\Bigg(- \frac{1}{2\sigma^2} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{\sum_{j=1}^{p}|\beta_j|}{b} \Bigg) \qquad (*)\end{aligned}$$

(c) To find the mode of the posterior of $\boldsymbol{\beta}$, we need to find the maximum of $(*)$. Since $\exp{(x)}$ is an increasing function, we only need to find the minimum of

$$\Bigg(\frac{1}{2\sigma^2} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{\sum_{j=1}^{p}|\beta_j|}{b} \Bigg)$$

which means

$$\hat{\boldsymbol{\beta}}_{\text{Lasso from Bayesian}} = \arg\min_{\boldsymbol{\beta}} \Bigg(\frac{1}{2\sigma^2} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{\sum_{j=1}^{p}|\beta_j|}{b} \Bigg)$$

We also have

$$\tilde{\boldsymbol{\beta}}_{\text{Lasso}} = \arg\min_{\boldsymbol{\beta}} \Bigg(\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \lambda \sum_{j=1}^{p}|\beta_j| \Bigg)$$

Set $\lambda = \frac{2\sigma^2}{b}$, we can get:

$$\tilde{\boldsymbol{\beta}}_{\text{Lasso}} = \hat{\boldsymbol{\beta}}_{\text{Lasso from Bayesian}}$$

Hence, the lasso estimate is the mode for β under this posterior distribution.

(d)

$$\begin{aligned}\text{prior}(\boldsymbol{\beta}) &= \prod_{j=1}^{p} \mathbf{P}(\beta_j) \\ &= \prod_{j=1}^{p} \frac{1}{\sqrt{2\pi c}} \exp\Bigg(- \frac{\beta_j^2}{2c} \Bigg) \\ &= (2\pi c)^{-n/2} \exp\Bigg(- \frac{1}{2c} \sum_{j=1}^{p} \beta_j^2 \Bigg)\end{aligned}$$

$$\begin{aligned}\text{posterior} &\propto \mathcal{L} \times \text{prior}\\ &\propto (2\pi\sigma^2)^{-n/2} \exp\Bigg(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2\Bigg) \times (2\pi c)^{-n/2} \exp\Bigg(-\frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\Bigg)\\ &= (2\pi\sigma^2)^{-n/2}(2\pi c)^{-n/2} \exp\Bigg(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\Bigg)\end{aligned}$$

(e) Ignoring the multiplicative constant, we can have

$$\text{posterior} \propto \exp\Bigg(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\Bigg) \qquad (**)$$

From $(**)$, we can clear see the posterior of β is in the form of normal distribution, since for each β_j, there is a square term in exponential term which could be rearranged into $(\beta_j - \mu_{\beta_j})^2$. The multiplicative residual terms in could be treated as variance and others can be treated as constant. The mode and the mean for $\boldsymbol{\beta}$ under this posterior distribution are same value. In order to find the mode, we need to minimum

$$\Bigg(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 - \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\Bigg)$$

Hence,

$$\hat{\boldsymbol{\beta}}_{\text{Ridge from Bayesian}} = \arg\min_{\boldsymbol{\beta}}\Bigg(\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\Bigg)$$

We also have

$$\tilde{\boldsymbol{\beta}}_{\text{Ridge}} = \arg\min_{\boldsymbol{\beta}}\Bigg(\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \beta_0)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\Bigg)$$

Set $\lambda = \frac{\sigma^2}{c}$, we can get:

$$\tilde{\boldsymbol{\beta}}_{\text{Ridge}} = \hat{\boldsymbol{\beta}}_{\text{Ridge from Bayesian}}$$

Hence, the ridge regression estimate is both the mode and the mean for $\boldsymbol{\beta}$ under this posterior distribution

P4

(a)

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

(b)

Here I take $\beta_0 = 0.2$, $\beta_1 = 0.6$, $\beta_2 = -0.1$ and $\beta_3 = 0.1$ and generate response vector Y.

```r
beta_0 = 0.2
beta_1 = 0.6
beta_2 = -0.1
beta_3 = 0.1
y = beta_0 + beta_1 * x + beta_2 * x^2 + beta_3 * x^3
```

(c)

```
library(leaps)
t.df <- data.frame(y = y, x = x)
t.full<-regsubsets(y ~ poly(x, 10, raw = T), data=t.df, nvmax = 10)
t.summary = summary(t.full)
```

```
# find model for best Cp
which.min(t.summary$cp)
```

```
## [1] 5
```

```
plot(t.summary$cp, xlab = "Subset Size", ylab = "Cp", col = "blue", pch = 2, type = "l")
points(which.min(t.summary$cp), t.summary$cp[which.min(t.summary$cp)],
       pch = 2, col = "red", lwd = 2)
```

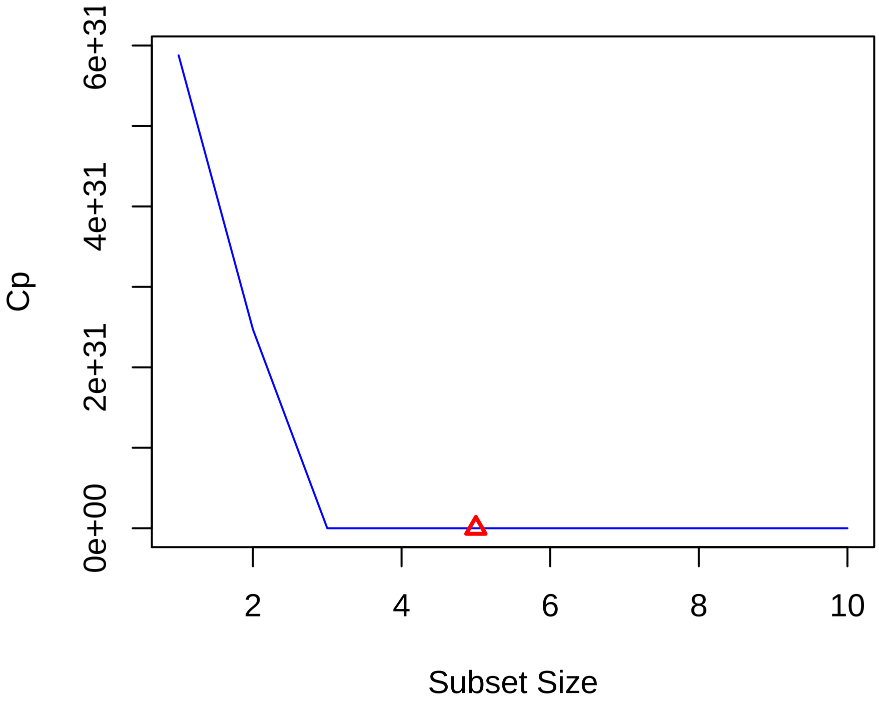

Figure 3: C_p vs. Subset Size

```
coefficients(t.full, id = which.min(t.summary$cp))
```

```
##           (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##          2.000000e-01           6.000000e-01          -1.000000e-01
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)8 poly(x, 10, raw = T)10
##          1.000000e-01          -4.336809e-18           7.047314e-19
```

According to C_p, the best model contains X, X^2, X^3, X^8, X^{10}. Coefficients corresponding each terms are shown before.

```
# find model for best BIC
which.min(t.summary$bic)
```

```
## [1] 5
```

```
plot(t.summary$bic, xlab = "Subset Size", ylab = "BIC", col = "blue", pch = 2, type = "l")
points(which.min(t.summary$bic), t.summary$bic[which.min(t.summary$bic)],
       pch = 2, col = "red", lwd = 2)
```

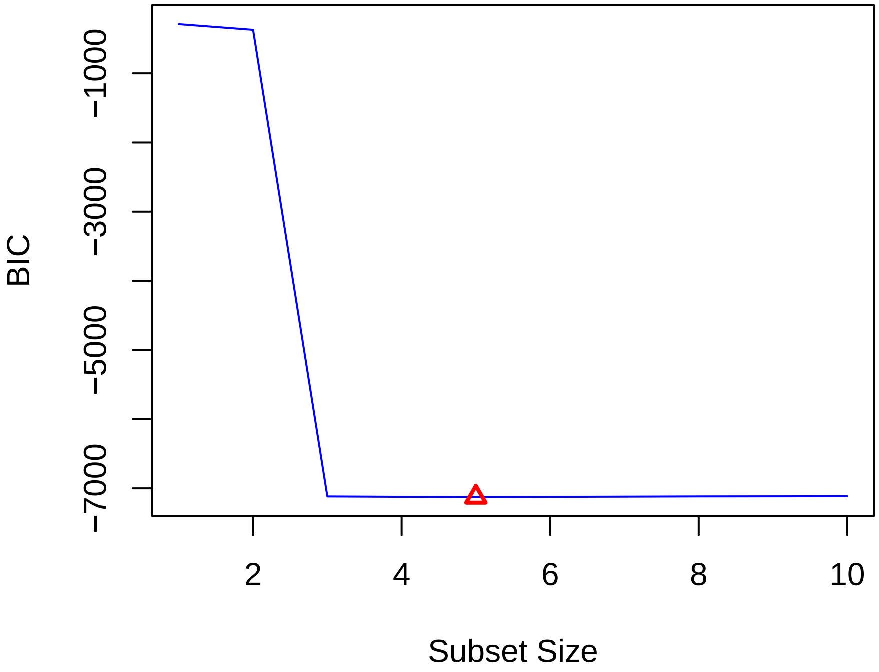

Figure 4: ***BIC*** vs. Subset Size

```
coefficients(t.full, id = which.min(t.summary$bic))
```

```
##            (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##           2.000000e-01           6.000000e-01          -1.000000e-01
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)8 poly(x, 10, raw = T)10
##           1.000000e-01          -4.336809e-18           7.047314e-19
```

According to BIC, the best model contains X, X^2, X^3, X^8, X^{10}. Coefficients corresponding each terms are shown before.

```
# find model for best Adjusted R
which.max(t.summary$adjr2)
```

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

```
plot(t.summary$adjr2, xlab = "Subset Size", ylab = "adj R2", col = "blue", pch = 2, type = "l")
points(which.max(t.summary$adjr2), t.summary$adjr2[which.max(t.summary$adjr2)],
       pch = 2, col = "red", lwd = 2)
```

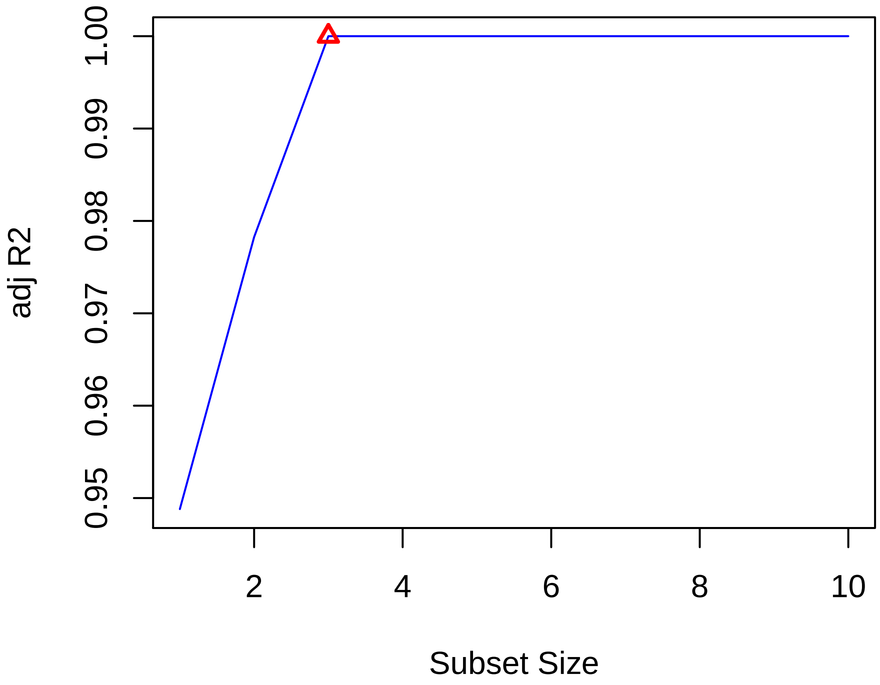

Figure 5: *adjusted* R^2 vs. Subset Size

```r
coefficients(t.full, id = which.max(t.summary$adjr2))
```

```
##          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##                  0.2                   0.6                  -0.1
## poly(x, 10, raw = T)3
##                  0.1
```

According to *Adjusted* R^2, the best model contains X, X^2, X^3. Coefficients corresponding each terms are shown before.

(d)

```r
library(leaps)
t.fwd<-regsubsets(y ~ poly(x, 10, raw = T), data=t.df, nvmax = 10,
                  method = "forward")
t.bwd<-regsubsets(y ~ poly(x, 10, raw = T), data=t.df, nvmax = 10,
                  method = "backward")
t.fwd.summary = summary(t.fwd)
t.bwd.summary = summary(t.bwd)
```

```r
# find model for best Cp
which.min(t.fwd.summary$cp)
```

```
## [1] 10
```

```r
plot(t.fwd.summary$cp, xlab = "Subset Size", ylab = "Cp", col = "blue", pch = 2, type = "l")
points(which.min(t.fwd.summary$cp), t.fwd.summary$cp[which.min(t.fwd.summary$cp)],
       pch = 2, col = "red", lwd = 2)
```

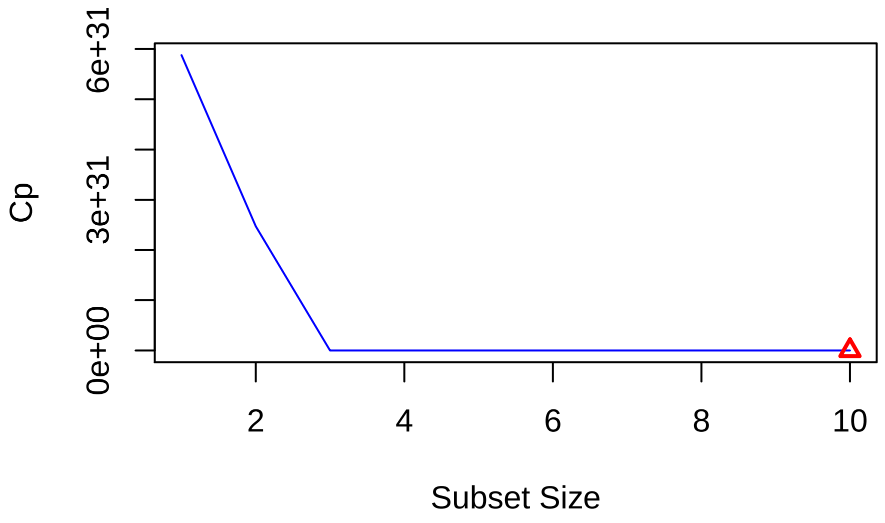

Figure 6: $\boldsymbol{C_p}$ vs. Subset Size for forward setpwise selection

```
coefficients(t.fwd, id = which.min(t.fwd.summary$cp))
```

```
##            (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2 
##           2.000000e-01           6.000000e-01          -1.000000e-01 
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)4  poly(x, 10, raw = T)5 
##           1.000000e-01          -1.087528e-15          -4.896425e-16 
##  poly(x, 10, raw = T)6  poly(x, 10, raw = T)7  poly(x, 10, raw = T)8 
##           7.360585e-16           1.748269e-16          -1.995471e-16 
##  poly(x, 10, raw = T)9 poly(x, 10, raw = T)10 
##          -1.875436e-17           1.800680e-17
```

For forward stepwise selection, according to C_p, the best model contains $X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}$. Coefficients corresponding each terms are shown before.

```
# find model for best Cp
which.min(t.bwd.summary$cp)
```

```
## [1] 5
```

```
plot(t.bwd.summary$cp, xlab = "Subset Size", ylab = "Cp", col = "blue", pch = 2, type = "l")
points(which.min(t.bwd.summary$cp), t.bwd.summary$cp[which.min(t.bwd.summary$cp)],
       pch = 2, col = "red", lwd = 2)
```

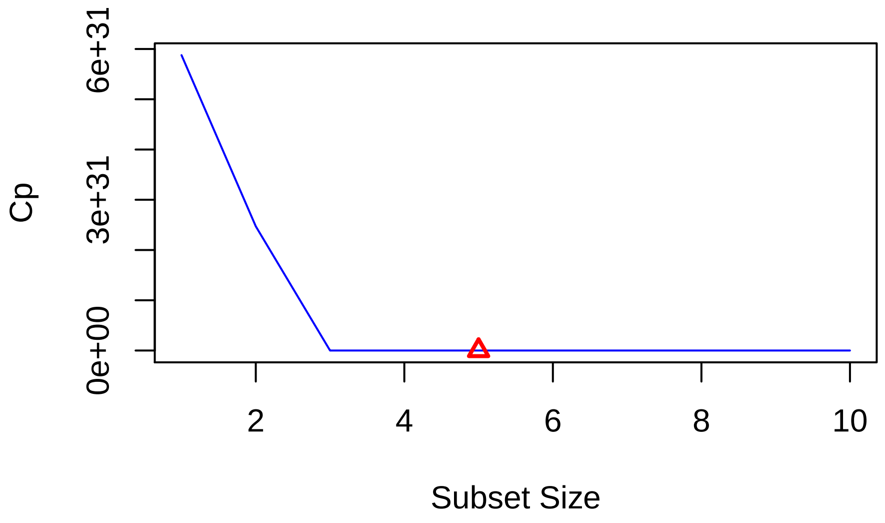

Figure 7: $\boldsymbol{C_p}$ vs. Subset Size for backward setpwise selection

```
coefficients(t.bwd, id = which.min(t.bwd.summary$cp))
```

```
##             (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##            2.000000e-01           6.000000e-01          -1.000000e-01
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)8 poly(x, 10, raw = T)10
##            1.000000e-01          -9.300273e-18           1.492010e-18
```

For backward stepwise selection, according to C_p, the best model contains X, X^2, X^3, X^8, X^{10}. Coefficients corresponding each terms are shown before.

FOr the forward stepwise selection, it has one more terms X^4, X^5, X^6, X^7, X^9, while for the backward stepwise selection, it is basically same as the result in (c).

```
# find model for best BIC
which.min(t.fwd.summary$bic)
```

```
## [1] 5
```

```
plot(t.fwd.summary$bic, xlab = "Subset Size", ylab = "BIC", col = "blue", pch = 2, type = "l")
points(which.min(t.fwd.summary$bic), t.fwd.summary$bic[which.min(t.fwd.summary$bic)],
       pch = 2, col = "red", lwd = 2)
```

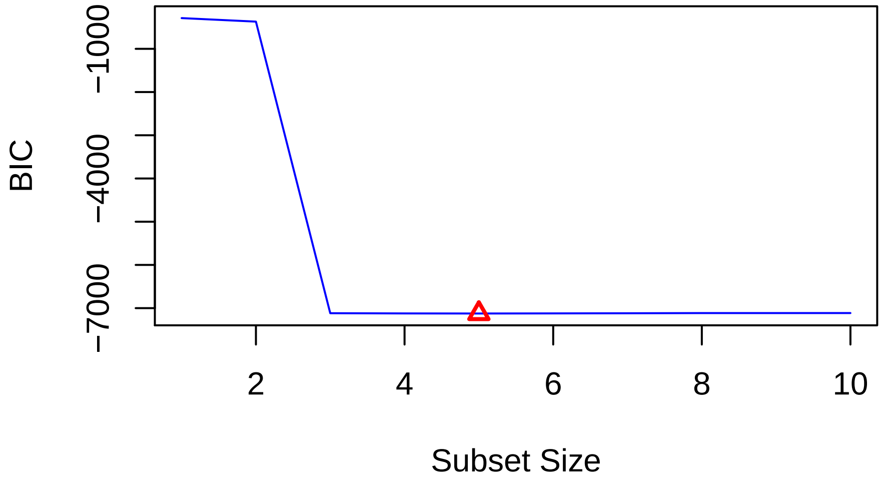

Figure 8: $\boldsymbol{BIC}$ vs. Subset Size forward stepwise selection

```
coefficients(t.fwd, id = which.min(t.fwd.summary$bic))
```

```
##           (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##          2.000000e-01           6.000000e-01          -1.000000e-01
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)4 poly(x, 10, raw = T)10
##          1.000000e-01          -1.067972e-16           2.820157e-19
```

For forward stepwise selection, according to BIC, the best model contains X, X^2, X^3, X^4X^{10}. Coefficients corresponding each terms are shown before.

```
# find model for best BIC
which.min(t.bwd.summary$bic)
```

```
## [1] 5
```

```
plot(t.bwd.summary$bic, xlab = "Subset Size", ylab = "BIC", col = "blue", pch = 2, type = "l")
points(which.min(t.bwd.summary$bic), t.bwd.summary$bic[which.min(t.bwd.summary$bic)],
       pch = 2, col = "red", lwd = 2)
```

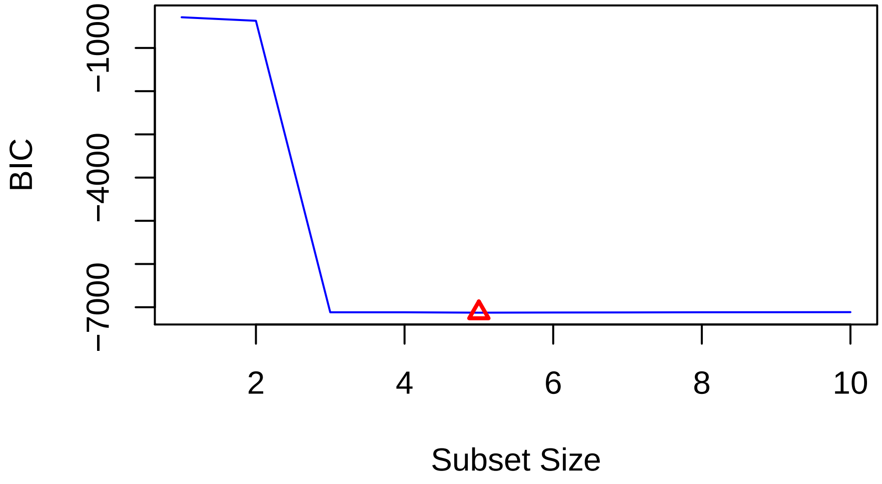

Figure 9: $\boldsymbol{BIC}$ vs. Subset Size backward stepwise selection

```r
coefficients(t.bwd, id = which.min(t.bwd.summary$bic))
```

```
##           (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##          2.000000e-01           6.000000e-01          -1.000000e-01
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)8 poly(x, 10, raw = T)10
##          1.000000e-01          -9.300273e-18           1.492010e-18
```

For backward stepwise selection, according to BIC, the best model contains X, X^2, X^3, X^8, X^{10}. Coefficients corresponding each terms are shown before.

Compared with result in (c), the forward stepwise selection provides one more term X^4 and without term X^8, while for the backward stepwise selection, it is basically same as the result in (c).

```r
# find model for best Adjusted R
which.max(t.fwd.summary$adjr2)
```

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

```r
plot(t.fwd.summary$adjr2, xlab = "Subset Size", ylab = "adj R2", col = "blue", pch = 2, type = "l")
points(which.max(t.fwd.summary$adjr2), t.fwd.summary$adjr2[which.max(t.fwd.summary$adjr2)],
       pch = 2, col = "red", lwd = 2)
```

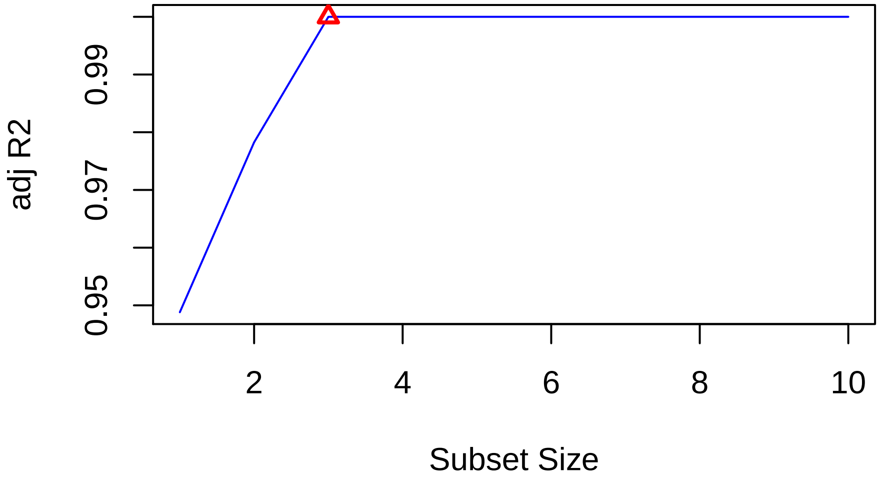

Figure 10: $adjusted\ R^2$ vs. Subset Size for forward stepwise selection

```r
coefficients(t.fwd, id = which.max(t.fwd.summary$adjr2))
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##                   0.2                   0.6                  -0.1
## poly(x, 10, raw = T)3
##                   0.1
```

For forward stepwise selection, according to $Adjusted\ R^2$, the best model contains X, X^2, X^3. Coefficients corresponding each terms are shown before.

```r
# find model for best Adjusted R
which.max(t.bwd.summary$adjr2)
```

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

```r
plot(t.bwd.summary$adjr2, xlab = "Subset Size", ylab = "adj R2", col = "blue", pch = 2, type = "l")
points(which.max(t.bwd.summary$adjr2), t.bwd.summary$adjr2[which.max(t.bwd.summary$adjr2)],
       pch = 2, col = "red", lwd = 2)
```

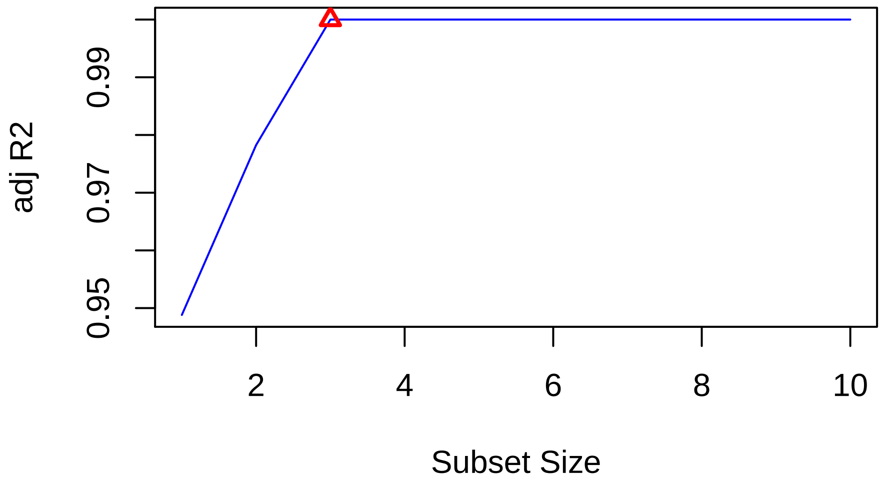

Figure 11: *adjusted* R^2 vs. Subset Size for backward stepwise selection

```
coefficients(t.bwd, id = which.max(t.bwd.summary$adjr2))
```

```
##            (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##                    0.2                   0.6                  -0.1
## poly(x, 10, raw = T)3
##                    0.1
```

For backward stepwise selection, according to *Adjusted* R^2, the best model contains X, X^2, X^3. Coefficients corresponding each terms are shown before.

The forward stepwise selection model and backward selection model are same as the previous result in (c).

(e)

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-18
```

```
x_mat <- model.matrix(y ~ poly(x, 10, raw = T), data = t.df)[, -1]
t.cv <-cv.glmnet(x_mat, y, alpha = 1)
```

```
plot(t.cv)
```

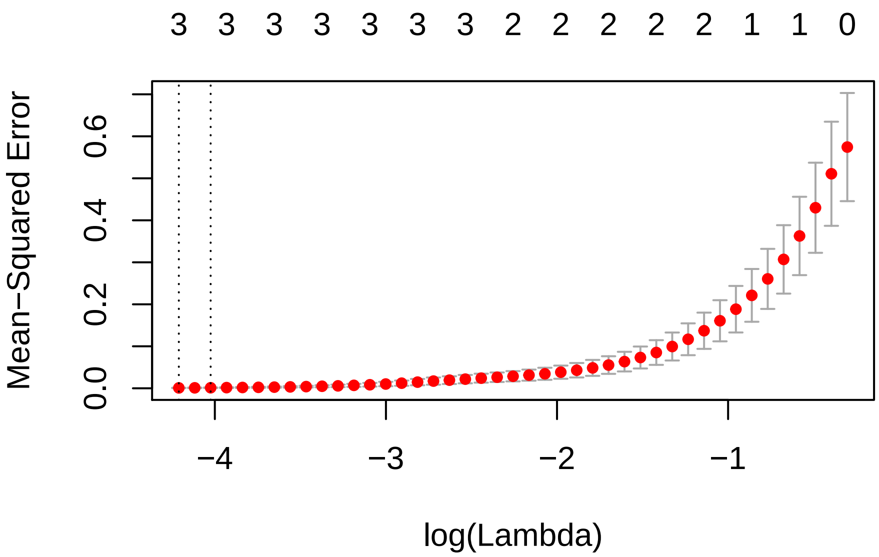

Figure 12: Cross-Validation Error vs. $log(\lambda)$

```
t.cv$lambda.min
```

```
## [1] 0.01483824
```

```
coef(t.cv, t.cv$lambda.min)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                   1
## (Intercept)              0.18848641
## poly(x, 10, raw = T)1    0.59467375
## poly(x, 10, raw = T)2   -0.08360996
## poly(x, 10, raw = T)3    0.09434374
## poly(x, 10, raw = T)4    .
## poly(x, 10, raw = T)5    .
## 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   .
```

```
t.cv$lambda.1se
```

```
## [1] 0.01787271
```

```
coef(t.cv, t.cv$lambda.1se)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                   1
## (Intercept)              0.18615213
## poly(x, 10, raw = T)1    0.59344128
## poly(x, 10, raw = T)2   -0.08027458
## poly(x, 10, raw = T)3    0.09322801
## poly(x, 10, raw = T)4    .
## poly(x, 10, raw = T)5    .
## 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  .
```

Here I find λ that gives minimum mean cross-validated error is 0.0148382 and λ that gives the most regularized model such that error is within one standard error of the minimum is 0.0178727. While those two λs are a litte different, coefficients are similar. The models have intercept and three terms X, X^2, X^3. However, there are a litte difference with the true values.

(f)

```
beta_0 = 0.5
beta_7 = 0.3
y = beta_0 + beta_7 * x^7 + epsilon

t2.df <- data.frame(y = y, x = x)
t2.full<-regsubsets(y ~ poly(x, 10, raw = T), data=t2.df, nvmax = 10)
t2.summary = summary(t2.full)
```

```
which.min(t2.summary$cp)
```

```
## [1] 2
```

```
coefficients(t2.full, id = which.min(t2.summary$cp))
```

```
##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##             0.5704904            -0.1417084             0.3015552
```

```
which.min(t2.summary$bic)
```

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

```
coefficients(t2.full, id = which.min(t2.summary$bic))
```

```
##           (Intercept) poly(x, 10, raw = T)7
##             0.4589402             0.3007705
```

```
which.max(t2.summary$adjr2)
```

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

```
coefficients(t2.full, id = which.min(t2.summary$adjr2))
```

```
##            (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##             0.67282867             0.51409233            -1.13146007
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)4  poly(x, 10, raw = T)5
##            -0.93113515             1.90382807             0.55109577
##  poly(x, 10, raw = T)6  poly(x, 10, raw = T)7  poly(x, 10, raw = T)8
##            -1.26499408             0.14430680             0.31986888
##  poly(x, 10, raw = T)9 poly(x, 10, raw = T)10
##             0.01627747            -0.02690171
```

From three standards, C_p, BIC and $Adjusted\ R^2$, only BIC could provid best model which has the same terms with true formula. The coefficients that model provides are not far away from true model.

```
x_mat2 <- model.matrix(y ~ poly(x, 10, raw = T), data = t2.df)[, -1]
t2.cv <-cv.glmnet(x_mat2, y, alpha = 1)
t2.cv$lambda.min
```

```
## [1] 0.09072803
```

```
plot(t2.cv)
```

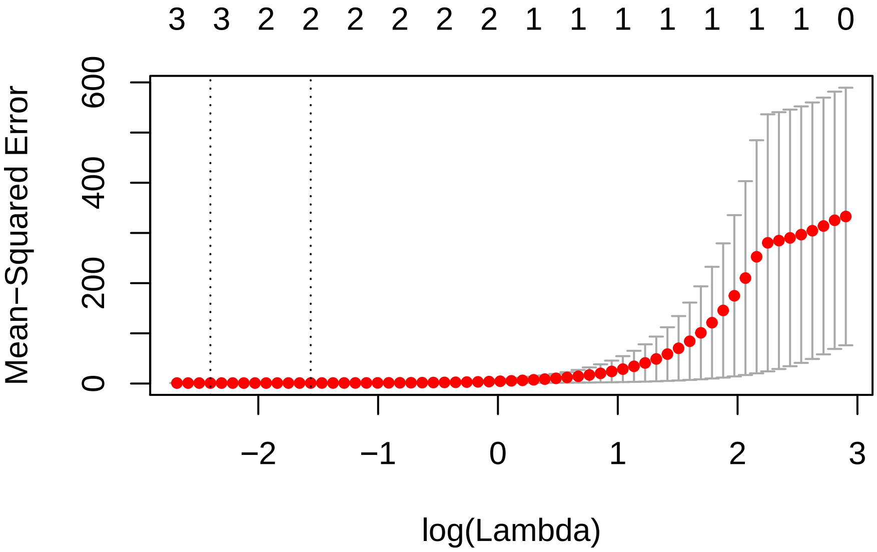

Figure 13: Cross-Validation Error vs. $log(\lambda)$

```
t2.full.mod = glmnet(x_mat2, y, alpha = 1)
predict(t2.full.mod, s = t2.cv$lambda.min, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                   1
## (Intercept)             0.494509159
## poly(x, 10, raw = T)1   .
## poly(x, 10, raw = T)2  -0.039663778
## poly(x, 10, raw = T)3   .
## poly(x, 10, raw = T)4   .
## poly(x, 10, raw = T)5   .
## poly(x, 10, raw = T)6   .
## poly(x, 10, raw = T)7   0.293996164
## poly(x, 10, raw = T)8   .
## poly(x, 10, raw = T)9   0.001059253
## poly(x, 10, raw = T)10  .
```

The lasso model will give 0.090728 as λ. Given that λ, the model will intercept and three terms X^2, X^7, X^9. While the intercept is not far away from true intercept and $\hat{\beta}_7$ is close to the true value, there are small uncessary coefficients β_2, β_9. Compared with lasso model, the predicted intercept for best subset selection is not quite close to the true value.

P5

(a)

```
# load College data set.
College.df <- read.csv("College.csv", header = T, row.names=1)
# check na
sum(is.na(College.df))
```

```
## [1] 0
```

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

(b)

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

```
##
## Call:
## lm(formula = Apps ~ ., data = College.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5741.2  -479.5    15.3   359.6  7258.0
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.902e+02  6.381e+02  -1.238 0.216410
## PrivateYes  -3.070e+02  2.006e+02  -1.531 0.126736
## Accept       1.779e+00  5.420e-02  32.830  < 2e-16 ***
## Enroll      -1.470e+00  3.115e-01  -4.720 3.35e-06 ***
## Top10perc    6.673e+01  8.310e+00   8.030 1.31e-14 ***
## Top25perc   -2.231e+01  6.533e+00  -3.415 0.000708 ***
## F.Undergrad  9.269e-02  5.529e-02   1.676 0.094538 .
## P.Undergrad  9.397e-03  5.493e-02   0.171 0.864275
## Outstate    -1.084e-01  2.700e-02  -4.014 7.22e-05 ***
## Room.Board   2.115e-01  7.224e-02   2.928 0.003622 **
## Books        2.912e-01  3.985e-01   0.731 0.465399
## Personal     6.133e-03  8.803e-02   0.070 0.944497
## PhD         -1.548e+01  6.681e+00  -2.316 0.021082 *
## Terminal     6.415e+00  7.290e+00   0.880 0.379470
## S.F.Ratio    2.283e+01  2.047e+01   1.115 0.265526
## perc.alumni  1.134e+00  6.083e+00   0.186 0.852274
## Expend       4.857e-02  1.619e-02   2.999 0.002890 **
## Grad.Rate    7.490e+00  4.397e+00   1.703 0.089324 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1083 on 370 degrees of freedom
## Multiple R-squared:  0.9389, Adjusted R-squared:  0.9361
## F-statistic: 334.3 on 17 and 370 DF,  p-value: < 2.2e-16
```

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

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

Therefore, the test error is 1.1357583×10^6.

(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)
```

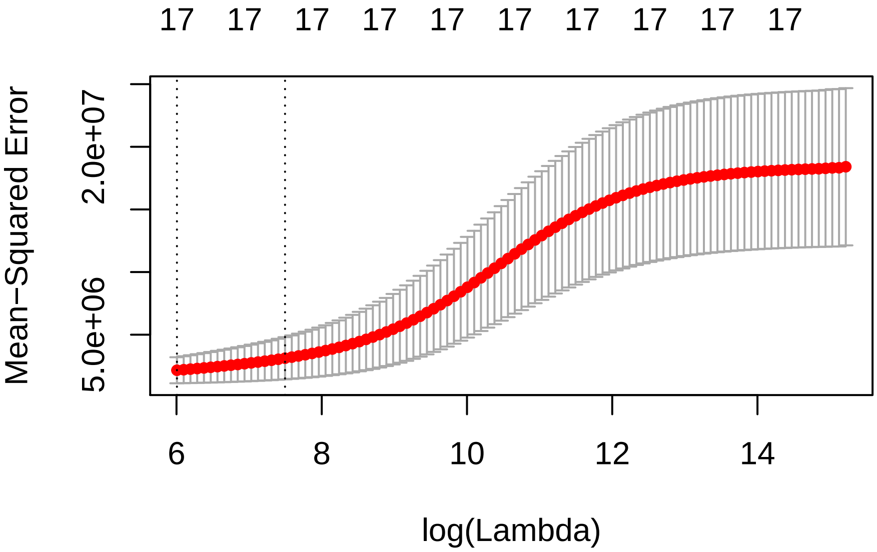

Figure 14: Cross-Validation Error vs. $log(\lambda)$ for Ridge regression on College data

```
College.ridge$lambda.min
```

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

```
coef(College.ridge, College.ridge$lambda.min )
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -2.028493e+03
## (Intercept)  .
## PrivateYes  -2.878966e+02
## Accept       1.131098e+00
## Enroll       3.781109e-01
## Top10perc    3.051156e+01
## Top25perc   -3.566151e-01
## F.Undergrad  5.588155e-02
## P.Undergrad  2.056636e-02
## Outstate    -3.903449e-02
## Room.Board   2.627548e-01
## Books        4.148845e-01
## Personal    -3.207653e-02
## PhD         -7.852690e+00
## Terminal    -1.014368e+00
## S.F.Ratio    2.774094e+01
## perc.alumni -5.371397e+00
## Expend       5.883054e-02
## Grad.Rate    8.645503e+00
```

Through Cross-Validation, we get $\lambda = 405.8403596$.

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

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

On the test data, the MSE is 9.762615×10^5. MSE is slightly smaller than linear regression.

(d)

```
College.lasso = cv.glmnet(train_mat, College.train$Apps, alpha = 1)
plot(College.lasso)
```

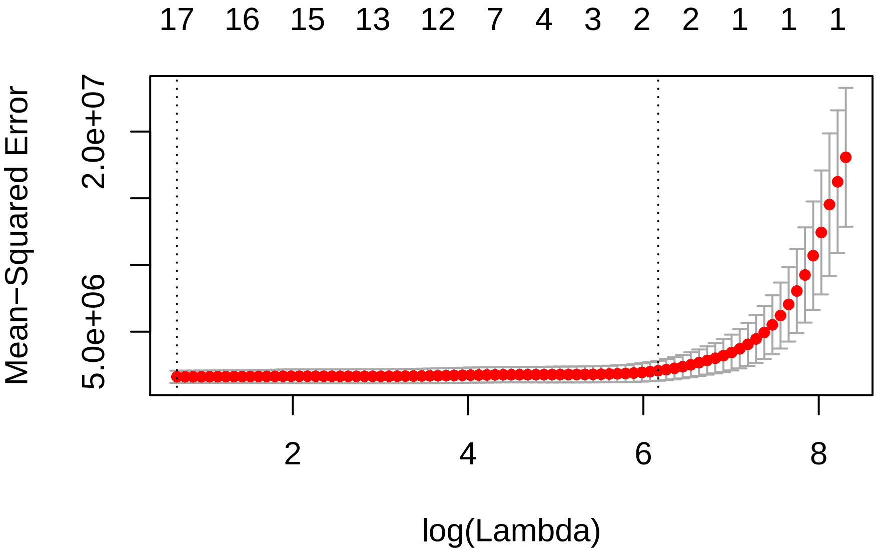

Figure 15: Cross-Validation Error vs. $log(\lambda)$ for Lasso regression on College data

```
College.lasso$lambda.min
```

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

```
coef(College.lasso, College.lasso$lambda.min )
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -7.688896e+02
## (Intercept)  .
## PrivateYes  -3.127034e+02
## Accept       1.762718e+00
## Enroll      -1.318195e+00
## Top10perc    6.482356e+01
## Top25perc   -2.081406e+01
## F.Undergrad  7.119149e-02
## P.Undergrad  1.246161e-02
## Outstate    -1.049091e-01
## Room.Board   2.088305e-01
## Books        2.926466e-01
## Personal     3.955068e-03
## PhD         -1.455463e+01
## Terminal     5.395858e+00
## S.F.Ratio    2.171398e+01
## perc.alumni  5.088260e-01
## Expend       4.824455e-02
## Grad.Rate    7.036148e+00
```

Through Cross-Validation, we get $\lambda = 1.97344$.

```
lasso.pred = predict(College.lasso, newx=test_mat, s=College.lasso$lambda.min)
mean((College.test[, "Apps"] - lasso.pred)^2)
```

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

On the test data, the MSE is 1.1159006×10^6, which is slightly smaller than linear regression and slightly bigger than ridge regression. The coefficients are shown before. All coefficients are non-zero and the number of non-zero coefficients is 17 if we don't consider intercept.

P6

(a)

```
set.seed(1)
n = 1000
p = 20
X_mat =  matrix(rnorm(n * p), n, p)
Beta = rnorm(p)
Beta[3]  = 0
Beta[5]  = 0
Beta[7]  = 0
Beta[13]  = 0
Beta[17]  = 0
epsilon = rnorm(n)
Y = X_mat %*% Beta + epsilon
```

Here I set some elements exactly equal to zero.

(b)

```
set.seed(1)
train = sample(1: 1000, 100)
test = (-train)
X_mat.train = X_mat[train, ]
X_mat.test = X_mat[test, ]
Y.train = Y[train, ]
Y.test = Y[test, ]
```

(c)

```
d.full = regsubsets(y ~ ., data = data.frame(x = X_mat.train, y = Y.train),
                    nvmax = p)
d.summary = summary(d.full)
d.summary
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data.frame(x = X_mat.train, 
##     y = Y.train), 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
## 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 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"  "*"  "*"  "*"  "*" 
##           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 ) "*"  "*"  "*"  "*"  "*"  "*"
```

```r
mse.train = rep(NA, p)
x_cols = colnames(X_mat, do.NULL = FALSE, prefix = "x.")
for(i in 1:p){
  c_i = coef(d.full, id = i)
  if(i > 1){
    Y.train.pred = as.matrix(X_mat.train[, x_cols %in% names(c_i)] %*%
                              c_i[names(c_i) %in% x_cols])
  }
  else
  {
    Y.train.pred = as.matrix(X_mat.train[, x_cols %in% names(c_i)] *
                              c_i[names(c_i) %in% x_cols])
  }
  mse.train[i] = mean((Y.train - Y.train.pred)^2)
}
```

```r
plot(mse.train, ylab = "Training MSE",xlab = "Subset Size", pch = 16, type = "b", col = "blue")
```

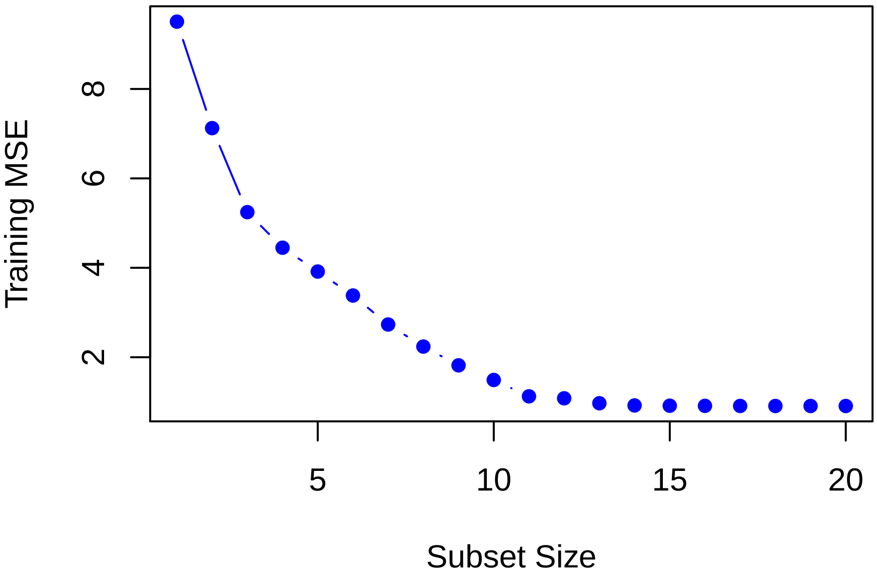

Figure 16: MSE on training set associated with the best model of each size

(d)

```
mse.test = rep(NA, p)
x_cols = colnames(X_mat, do.NULL = FALSE, prefix = "x.")
for(i in 1:p){
  c_i = coef(d.full, id = i)
  if(i > 1){
    Y.test.pred = as.matrix(X_mat.test[, x_cols %in% names(c_i)] %*%
                              c_i[names(c_i) %in% x_cols])
  }
  else
  {
    Y.test.pred = as.matrix(X_mat.test[, x_cols %in% names(c_i)] *
                              c_i[names(c_i) %in% x_cols])
  }
  mse.test[i] = mean((Y.test - Y.test.pred)^2)
}
```

```
plot(mse.test, ylab = "test MSE",xlab = "Subset Size", pch = 16, type = "b", col = "blue")
```

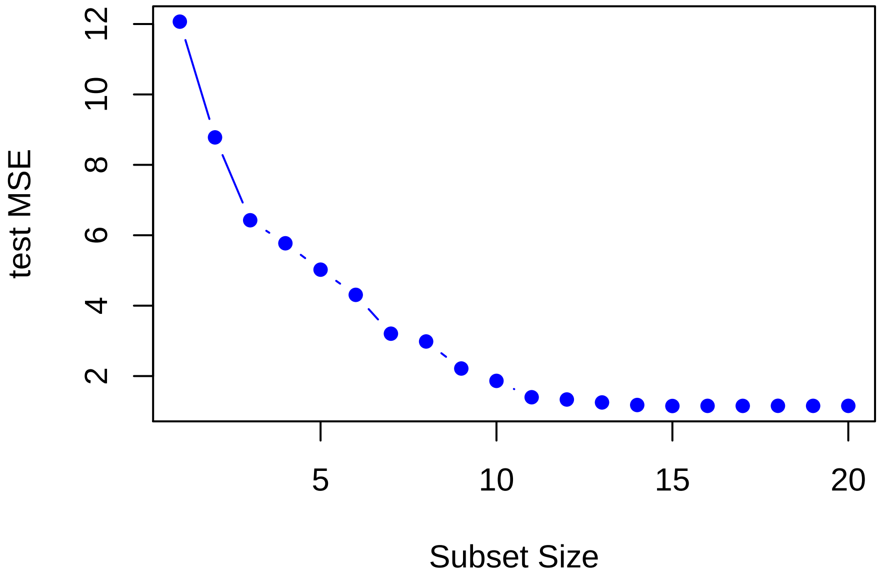

Figure 17: MSE on test set associated with the best model of each size

(e)

```r
which.min(mse.test)
```

```
## [1] 15
```

The model with 15 variables takes minimum MSE on test data set. Since the test set MSE is minimized for an intermediate model size, I don't need to re-generate data from step (a).

(f)

```r
coef(d.full, id = which.min(mse.test))
```

```
## (Intercept)         x.1         x.2         x.4         x.6         x.8 
##  0.19602793  0.08749005  0.27610304 -1.95181278 -0.28937028  0.75815397 
##         x.9        x.10        x.11        x.12        x.14        x.15 
##  2.09245387  0.71224778  0.81852484  0.73614855 -0.81017866 -0.68128008 
##        x.16        x.18        x.19        x.20 
## -0.35527038  1.61586315  0.93073165 -0.96895182
```

```r
Beta
```

```
##  [1]  0.2353485  0.2448250  0.0000000 -1.9348085  0.0000000 -0.2835501
##  [7]  0.0000000  0.7231804  2.0310355  0.7304903  0.8791534  0.5545564
## [13]  0.0000000 -0.6746580 -0.7154889 -0.2705279  0.0000000  1.6698068
## [19]  0.8922593 -1.0154889
```

Compared with true model, the model at which the test set MSE is minimized caught all zero β. But for other βs, the values are not quite close to the true value.

(g)

```
beta_err = rep(NA, p)
x_cols = colnames(X_mat, do.NULL = FALSE, prefix = "x.")
for(i in 1:p){
  c_i = coef(d.full, id = i)
  beta_err[i] = sqrt(sum((Beta[x_cols %in% names(c_i)] - c_i[names(c_i) %in% x_cols])^2))
}
plot(x = 1:p, y = beta_err, xlab = "Subset Size", ylab = "Error Between Estimated and True Coefficients
```

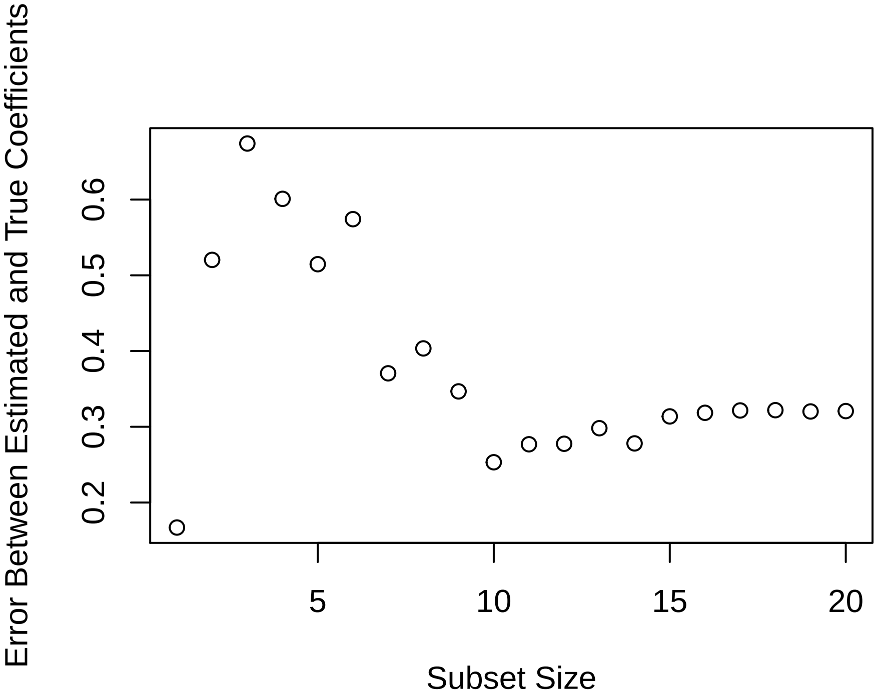

Figure 18: Error Between Estimated and True Coefficients vs. Subset Size

```
which.min(beta_err)
```

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

```
coef(d.full, id = which.min(beta_err))
```

```
## (Intercept)         x.9
##   -0.322370    1.864055
```

The model at which the β error is minimized is a model with one variable. This model is very different from the model from previous step. However, the curve we plot in this step is similar to the curve in previous step.

```
which(beta_err == sort(beta_err)[2])
```

```
## [1] 10
```

```
coef(d.full, id = which(beta_err == sort(beta_err)[2]))
```

```
## (Intercept)         x.4         x.8         x.9        x.10        x.11
##   0.2068073  -1.8341526   0.8159766   2.0596517   0.7125725   0.9141516
##        x.12        x.14        x.18        x.19        x.20
##   0.6512693  -0.7243549   1.5030793   0.9224435  -0.9659707
```

Then I check the model with second minimum β error, which is a model with 10 variables. This model is still not the model we got in previous step (f), but the amount variables is increased compared with previous model.