HW5

Yue Zhang

2025-10-19

```
setwd("/Users/yuezhang/Documents/Biostat/PH1976")
getwd()
library(ISLR2)
library(ggplot2)
library(tidyr)
library(dtplyr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggcorrplot)
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
library(class)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-10
```

```r
library(leaps)
library(pls)
```

```
##
## Attaching package: 'pls'
##
## The following object is masked from 'package:stats':
##
##     loadings
```

3.

(a). As s increases from 0, the training RSS will steadily decrease as the model becomes more flexible.

(b). For test RSS, it will decrease initially and then eventually starts increasing in a U shape because if we fit too many independent variables, the model might be overfitting.

(c). The variance will steadily increase as more variables are included, the model becomes complex and sensitive, it will change drastically between different datasets.

(d). The bias will steadily decrease because we include more variable and more noises, the difference between the average prediction of the model and the true value will be smaller.

(e). The irreducible error will remain constant because it is caused by inherent randomness or noise in the data and it is unavoidable.

5.

(a).

$$L(\beta_1, \beta_2) = \sum_{i=1}^{2} (y_i - (\beta_1 + \beta_2)x_i - \beta_0)^2 + \lambda(\beta_1^2 + \beta_2^2).$$

As $x_{11} = x_{12}$, $x_{21} = x_{22}$, we can say that $x_1 = x_{11} = x_{12}$, $x_2 = x_{21} = x_{22}$. As $x_{11} + x_{21} = 0$, $x_{21} + x_{22} = 0$, $x_1 = -x_2$. Therefore, $\hat{\beta}_0 = 0$, we need to minimize:

$$L(\beta_1, \beta_2) = \sum_{i=1}^{2} (y_i - (\beta_1 + \beta_2)x_i)^2 + \lambda(\beta_1^2 + \beta_2^2).$$

(b).

$$\frac{\partial L}{\partial \beta_1} = -2\sum_{i=1}^{2} x_i(y_i - (\beta_1 + \beta_2)x_i) + 2\lambda\beta_1,$$

$$\frac{\partial L}{\partial \beta_2} = -2\sum_{i=1}^{2} x_i(y_i - (\beta_1 + \beta_2)x_i) + 2\lambda\beta_2.$$

To minimize these equations, we need to set them as zero

$$\hat{\beta}_1 = \hat{\beta}_2 = \frac{\sum_{i=1}^{2} x_i(y_i - (\beta_1 + \beta_2)x_i)}{\lambda}$$

(c).

$$L(\beta_1, \beta_2) = \sum_{i=1}^{2} (y_i - (\beta_1 + \beta_2)x_i)^2 + \lambda(|\beta_1| + |\beta_2|).$$

(d). Assumes that there are some α value that satisfies $\beta_1 + \beta_2 = \alpha$, $|\beta_1| + |\beta_2| \geq |alpha|$ Therefore,

$$L(\beta_1, \beta_2) \geq \sum_{i=1}^{2} (y_i - \alpha x_i)^2 + \lambda\alpha.$$

There are a lot of (β_1, β_2) pairs that satisfies $\beta_1 + \beta_2 = \alpha$ and $\beta_1\hat{\alpha}$ 0, $\beta_2\hat{\alpha}$ 0 will minimize the equation.

8.

(a).

```
set.seed(1215)
X = rnorm(100)
epsilon = rnorm(100)
```

(b).

```
Y = 8 + 2 * X + 3 * X^2 + 4 * X^3 + epsilon
```

(c).

```
df = data.frame(X, Y)
best_sub = regsubsets(data = df, Y ~ poly(X, 10, raw = TRUE), nvmax = 10)
summary1 = summary(best_sub)
summary1
```

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

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

```r
min_Cp = which.min(summary1$cp)
min_BIC = which.min(summary1$bic)
max_adjR = which.max(summary1$adjr2)

cat(
  "Min Mallow's CP:", min_Cp, "\n",
  "Min BIC:", min_BIC, "\n",
  "Max Adjusted R-squared:", max_adjR, sep = "")
```

```
## Min Mallow's CP:3
## Min BIC:3
## Max Adjusted R-squared:3
```

```r
coef(best_sub, min_Cp)
```

```
##               (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                  8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                  4.038280
```

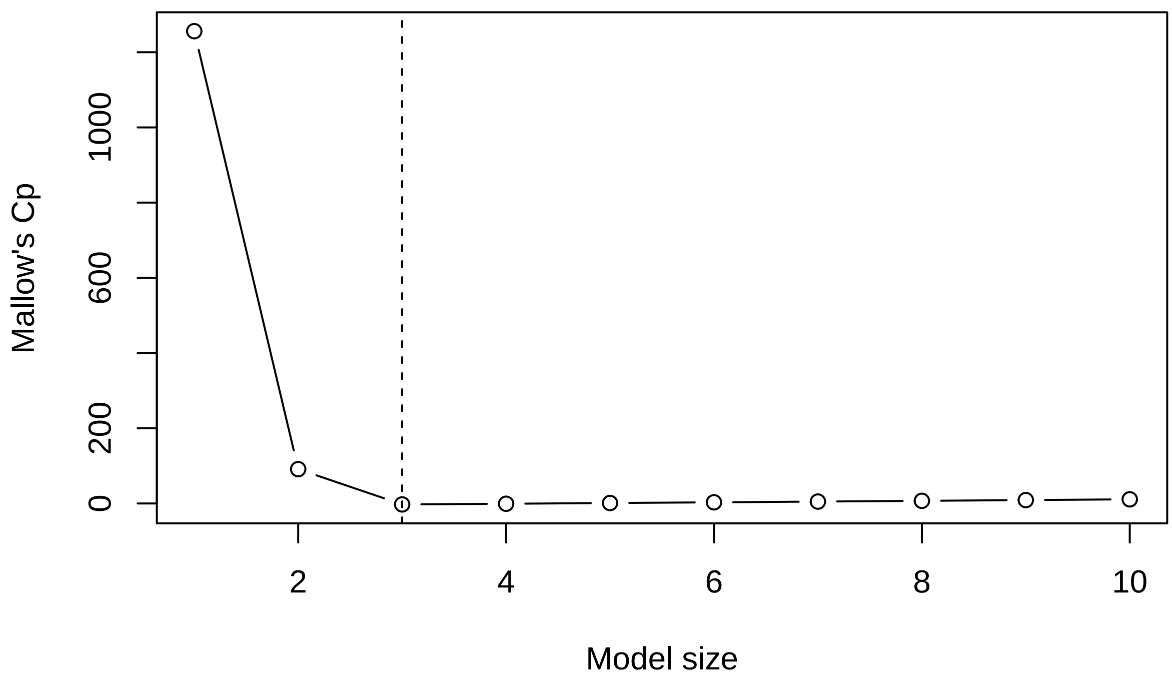
```r
coef(best_sub, min_BIC)
```

```
##             (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                4.038280
```

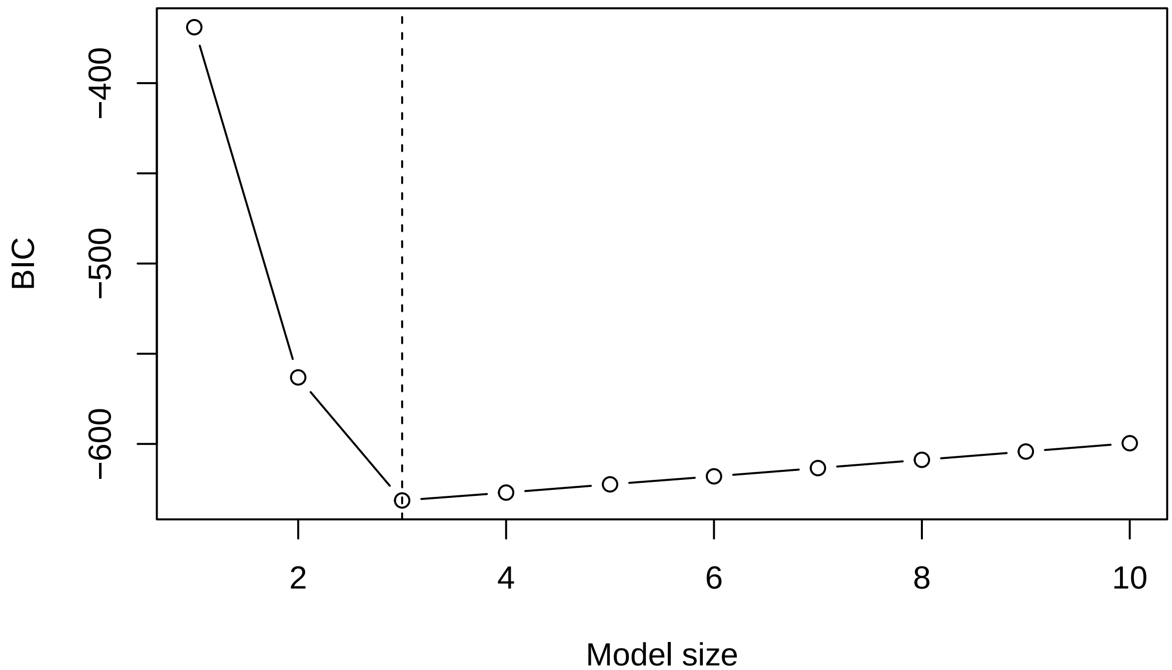
```r
coef(best_sub, max_adjR)
```

```
##             (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                4.038280
```

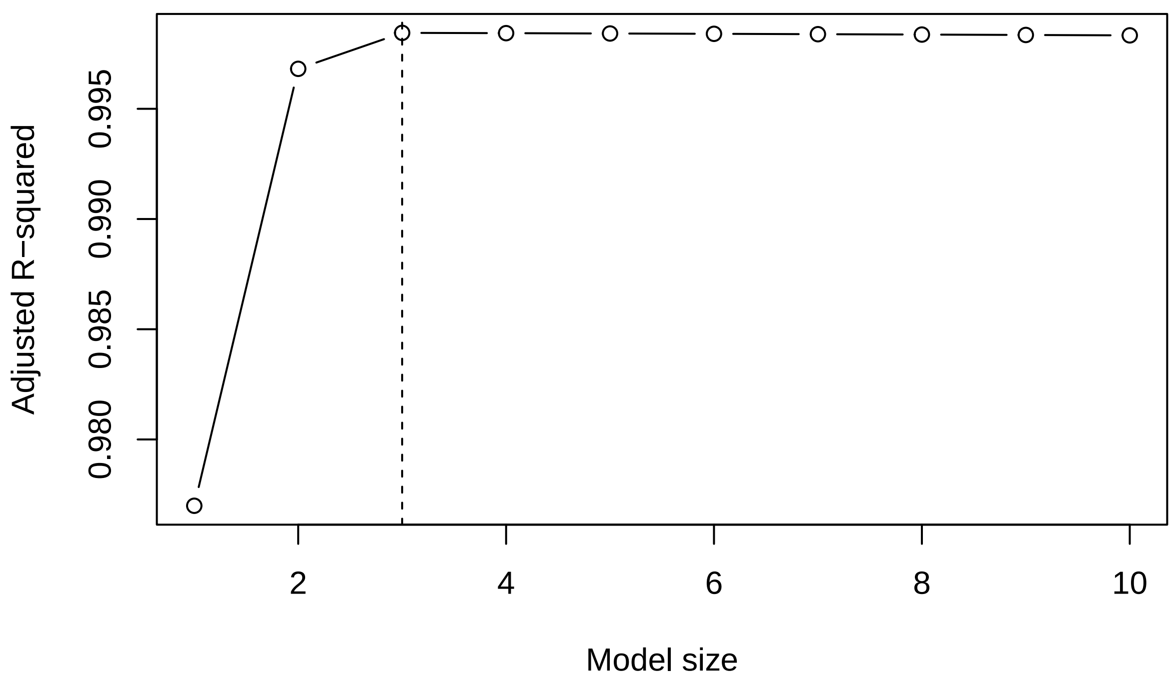
```r
par(mfrow = c(1,1))
plot(1:10, summary1$cp, type = "b", xlab = "Model size", ylab = "Mallow's Cp")
abline(v = min_Cp, lty = 2)
```




```r
plot(1:10, summary1$bic, type = "b", xlab = "Model size", ylab = "BIC")
abline(v = min_BIC, lty = 2)
```




```
plot(1:10, summary1$adjr2, type = "b", xlab = "Model size", ylab = "Adjusted R-squared")
abline(v = max_adjR, lty = 2)
```



Based on the results, the best model is: $Y = 8.0770 + 1.9653X + 2.9228X^2 + 4.0383X^3$.

(d). #Forward

```
forward_sub = regsubsets(data = df, Y ~ poly(X, 10, raw = TRUE), nvmax = 10, method = "forward")
summary2 = summary(forward_sub)
summary2
```

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

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

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

```r
min_Cp2 = which.min(summary2$cp)
min_BIC2 = which.min(summary2$bic)
max_adjR2 = which.max(summary2$adjr2)

cat(
  "Min Mallow's CP:", min_Cp2, "\n",
  "Min BIC:", min_BIC2, "\n",
  "Max Adjusted R-squared:", max_adjR2, sep = "")
```

```
## Min Mallow's CP:3
## Min BIC:3
## Max Adjusted R-squared:3
```

```r
coef(forward_sub, min_Cp2)
```

```
##              (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                 8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                 4.038280
```

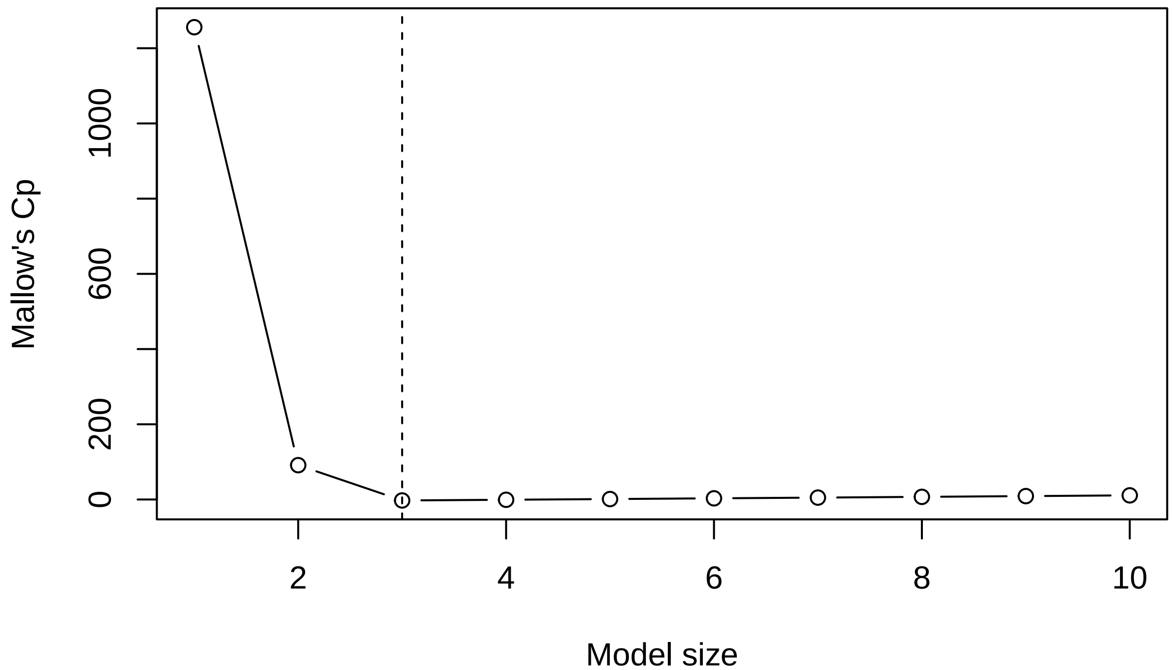
```r
coef(forward_sub, min_BIC2)
```

```
##              (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                 8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                 4.038280
```

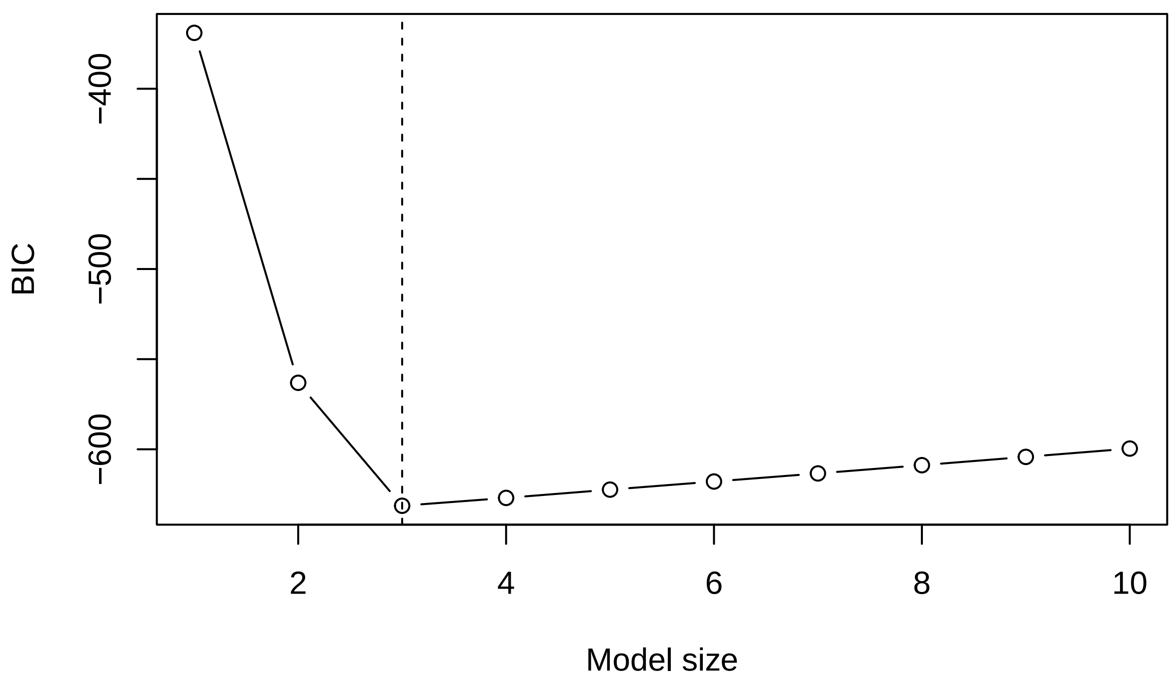
```r
coef(forward_sub, max_adjR2)
```

```
##              (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                 8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                 4.038280
```

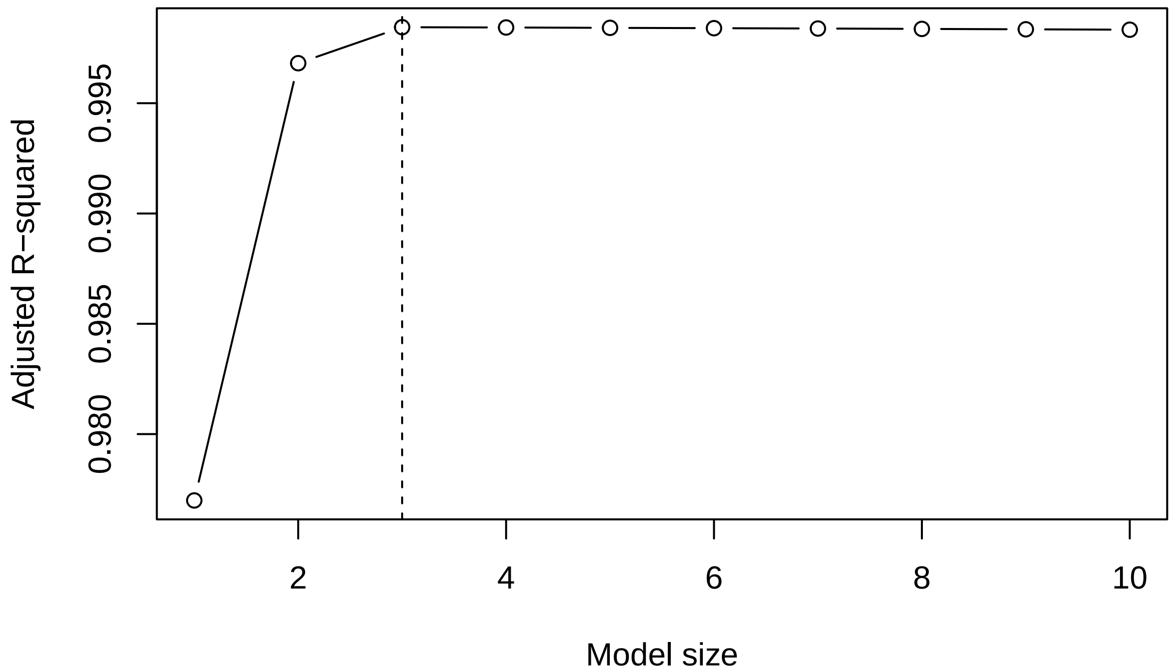
```r
par(mfrow = c(1,1))
plot(1:10, summary2$cp, type = "b", xlab = "Model size", ylab = "Mallow's Cp")
abline(v = min_Cp2, lty = 2)
```




```r
plot(1:10, summary2$bic, type = "b", xlab = "Model size", ylab = "BIC")
abline(v = min_BIC2, lty = 2)
```




```r
plot(1:10, summary2$adjr2, type = "b", xlab = "Model size", ylab = "Adjusted R-squared")
abline(v = max_adjR2, lty = 2)
```



The best forward subset model is: $Y = 8.0770 + 1.9653X + 2.9228X^2 + 4.0383X^3$ which is the same as part c.

#Backward

```
backward_sub = regsubsets(data = df, Y ~ poly(X, 10, raw = TRUE), nvmax = 10, method = "backward")
summary3 = summary(backward_sub)
summary3
```

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

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

```r
min_Cp3 = which.min(summary3$cp)
min_BIC3 = which.min(summary3$bic)
max_adjR3 = which.max(summary3$adjr2)

cat(
  "Min Mallow's CP:", min_Cp3, "\n",
  "Min BIC:", min_BIC3, "\n",
```

```r
  "Max Adjusted R-squared:", max_adjR3, sep = "")
```

```
## Min Mallow's CP:3
## Min BIC:3
## Max Adjusted R-squared:3
```

```r
coef(backward_sub, min_Cp3)
```

```
##             (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                4.038280
```

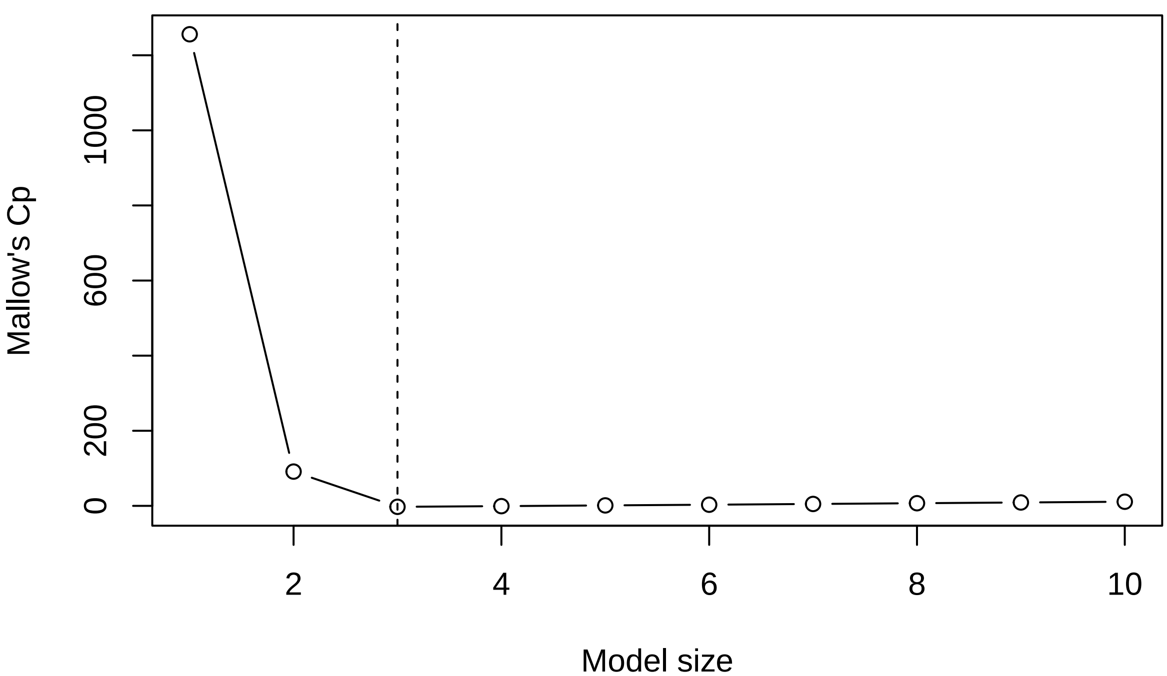
```r
coef(backward_sub, min_BIC3)
```

```
##             (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                4.038280
```

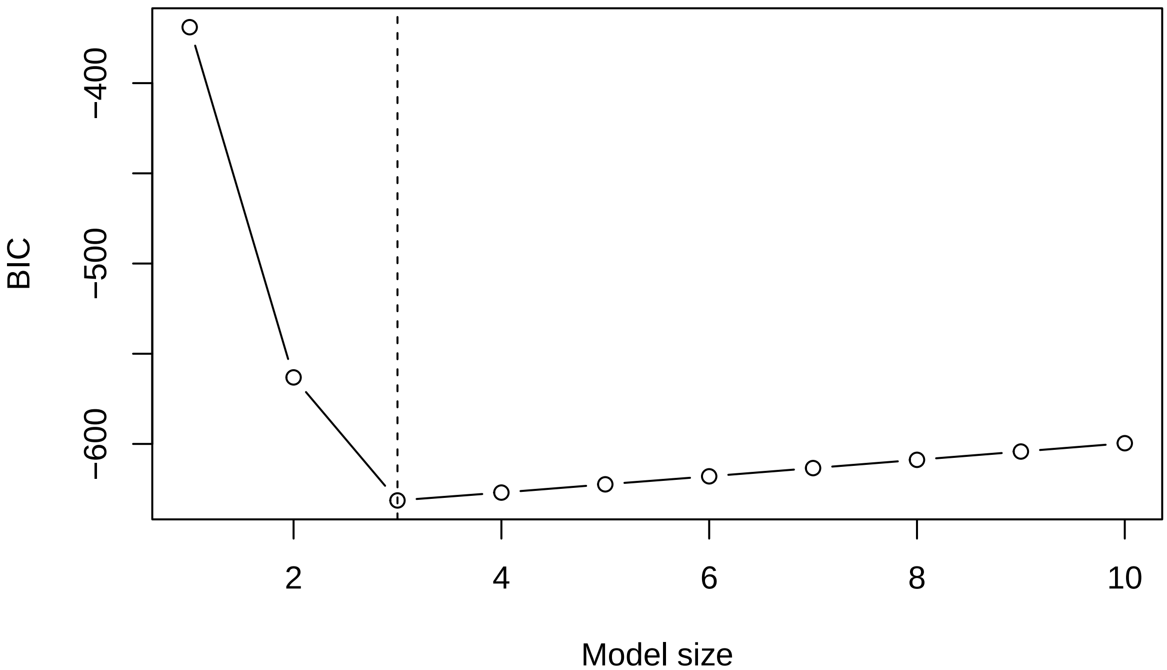
```r
coef(backward_sub, max_adjR3)
```

```
##             (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2
##                8.077001                 1.965296                 2.922793
## poly(X, 10, raw = TRUE)3
##                4.038280
```

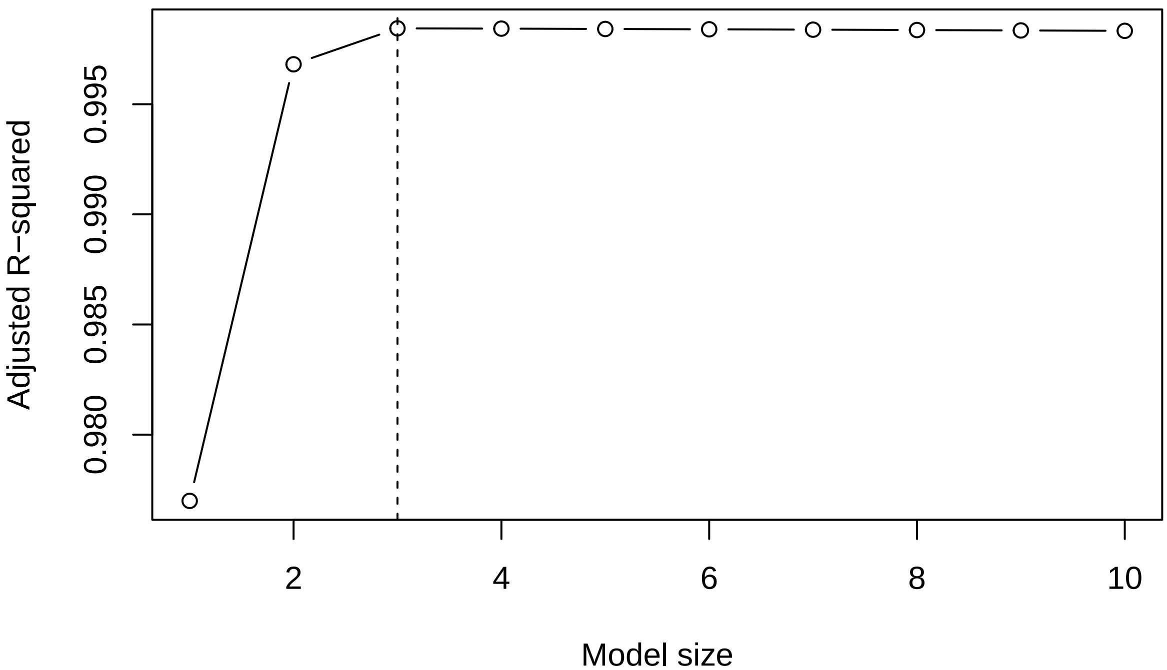
```r
par(mfrow = c(1,1))
plot(1:10, summary3$cp, type = "b", xlab = "Model size", ylab = "Mallow's Cp")
abline(v = min_Cp3, lty = 2)
```




```r
plot(1:10, summary3$bic, type = "b", xlab = "Model size", ylab = "BIC")
abline(v = min_BIC3, lty = 2)
```




```
plot(1:10, summary3$adjr2, type = "b", xlab = "Model size", ylab = "Adjusted R-squared")
abline(v = max_adjR3, lty = 2)
```



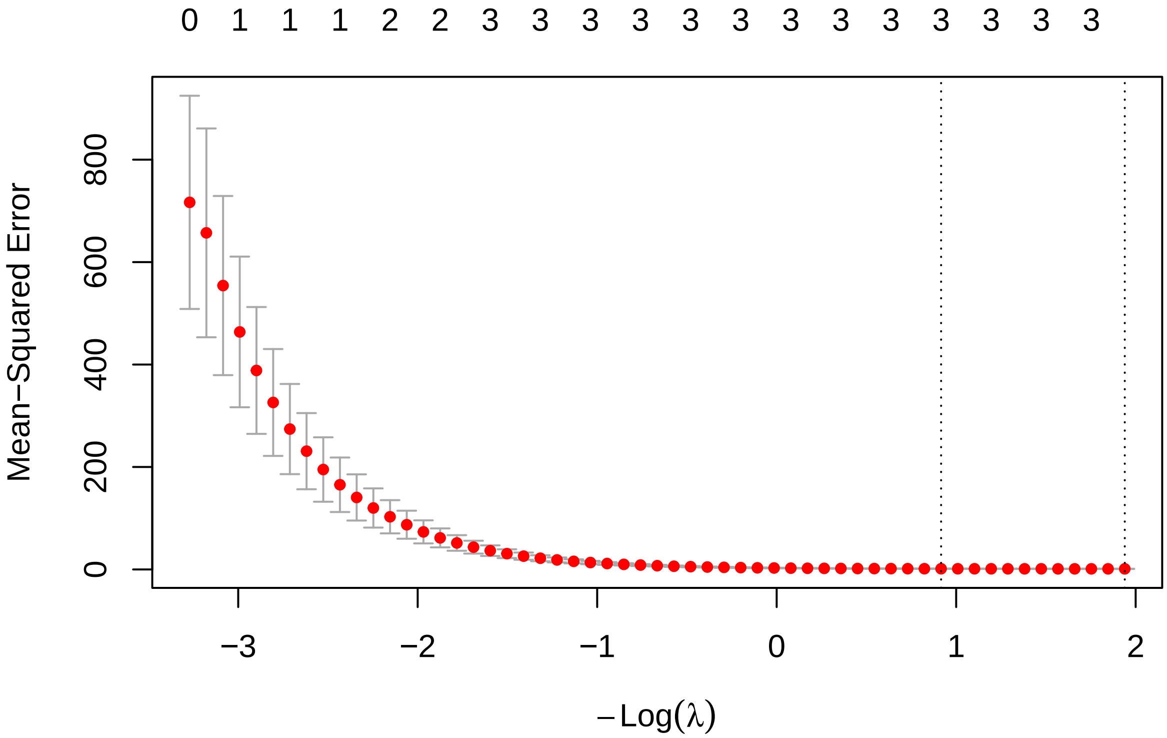
The best backward subset model is: $Y = 8.0770 + 1.9653X + 2.9228X^2 + 4.0383X^3$ which is the same as part c.

(e).

```
lasso = cv.glmnet(poly(df$X, 10, raw = TRUE), df$Y, alpha = 1, nfolds = 10, standardize = TRUE)
best_lambda = lasso$lambda.min
best_lambda
```

```
## [1] 0.1437779
```

```r
plot(lasso)
```




```r
coef_min = coef(lasso, s = "lambda.min")
coef_min
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##               lambda.min
## (Intercept)     8.171832
## 1               1.816136
## 2               2.852118
## 3               4.052946
## 4               .
## 5               .
## 6               .
## 7               .
## 8               .
## 9               .
## 10              .
```

The best backward subset model is: $Y = 8.1718 + 1.8161X + 2.8521X^2 + 4.0529X^3$ which is slightly different as part c.

(f). #Best Subset

```r
Y2 = 8 + 7 * X^7 + epsilon
df2 = data.frame(X, Y2)

best_sub2 = regsubsets(data = df2, Y2 ~ poly(X, 10, raw = TRUE), nvmax = 10)
summary4 = summary(best_sub2)
summary4
```

```
## Subset selection object
```

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

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

```
min_Cp4 = which.min(summary4$cp)
min_BIC4 = which.min(summary4$bic)
max_adjR4 = which.max(summary4$adjr2)

cat(
  "Min Mallow's CP:", min_Cp4, "\n",
  "Min BIC:", min_BIC4, "\n",
  "Max Adjusted R-squared:", max_adjR4, sep = "")
```

```
## Min Mallow's CP:1
## Min BIC:1
## Max Adjusted R-squared:2
```

```
coef(best_sub2, min_Cp4)
```

```
##              (Intercept) poly(X, 10, raw = TRUE)7
##                 8.009195                 7.000194
```

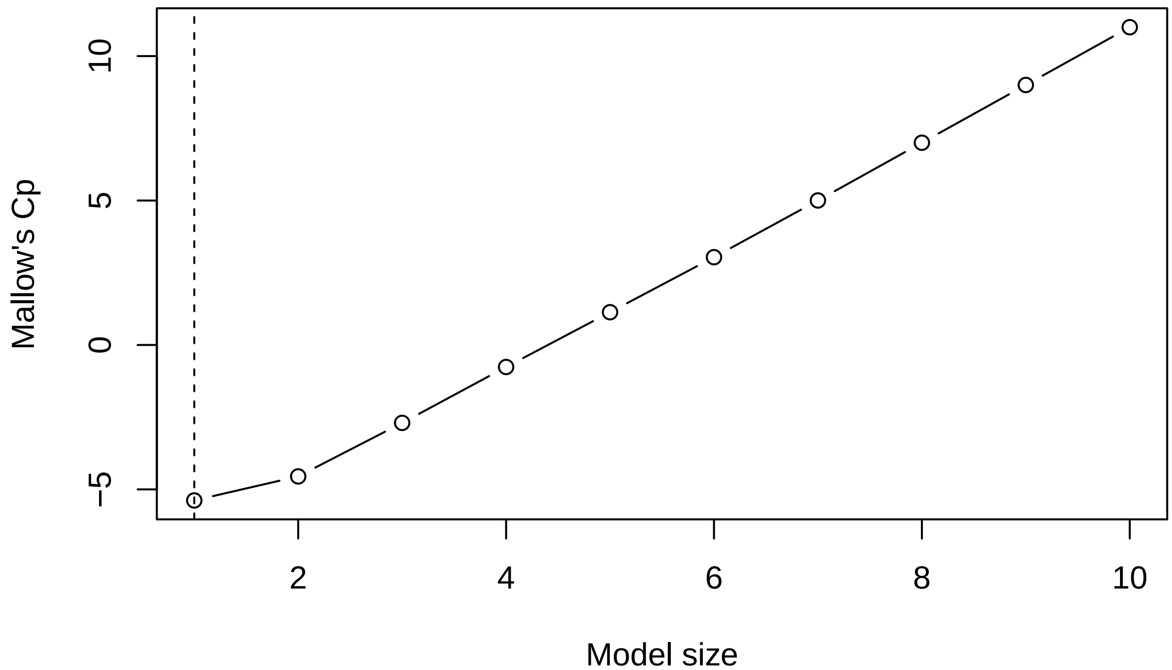
```
coef(best_sub2, min_BIC4)
```

```
##              (Intercept) poly(X, 10, raw = TRUE)7
##                 8.009195                 7.000194
```

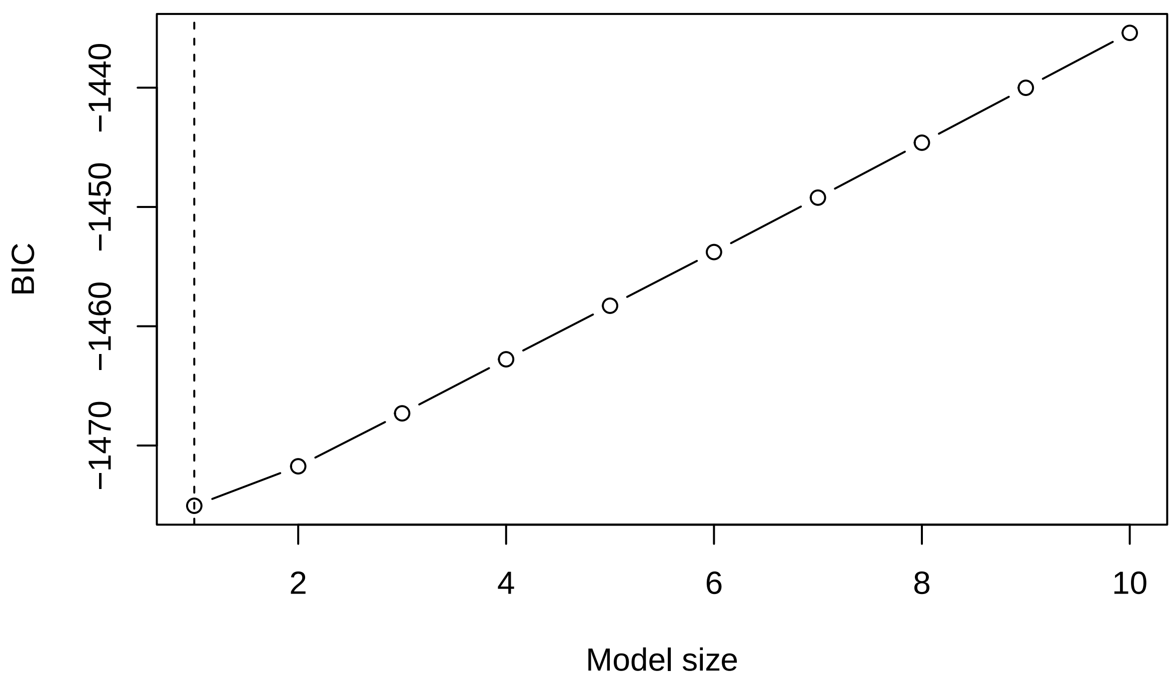
```
coef(best_sub2, max_adjR4)
```

```
##              (Intercept) poly(X, 10, raw = TRUE)7 poly(X, 10, raw = TRUE)8
##             8.0315273441             7.0012992425            -0.0004521743
```

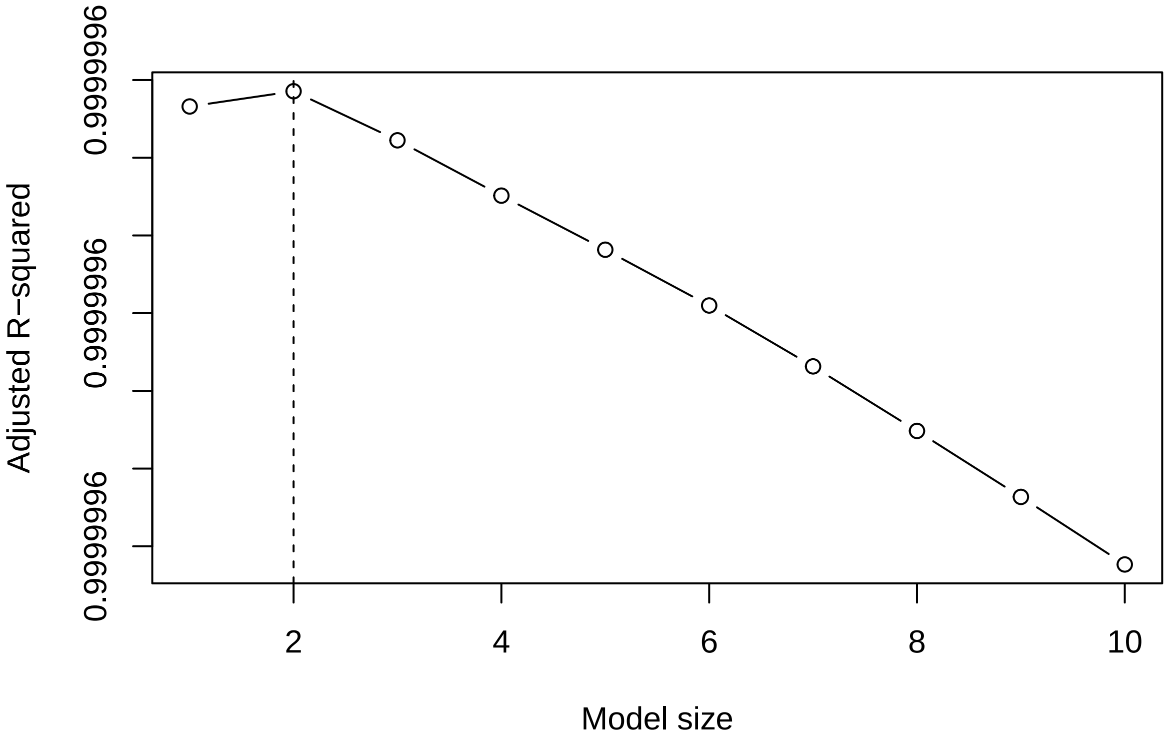
```
par(mfrow = c(1,1))
plot(1:10, summary4$cp, type = "b", xlab = "Model size", ylab = "Mallow's Cp")
abline(v = min_Cp4, lty = 2)
```




```
plot(1:10, summary4$bic, type = "b", xlab = "Model size", ylab = "BIC")
abline(v = min_BIC4, lty = 2)
```




```
plot(1:10, summary4$adjr2, type = "b", xlab = "Model size", ylab = "Adjusted R-squared")
abline(v = max_adjR4, lty = 2)
```



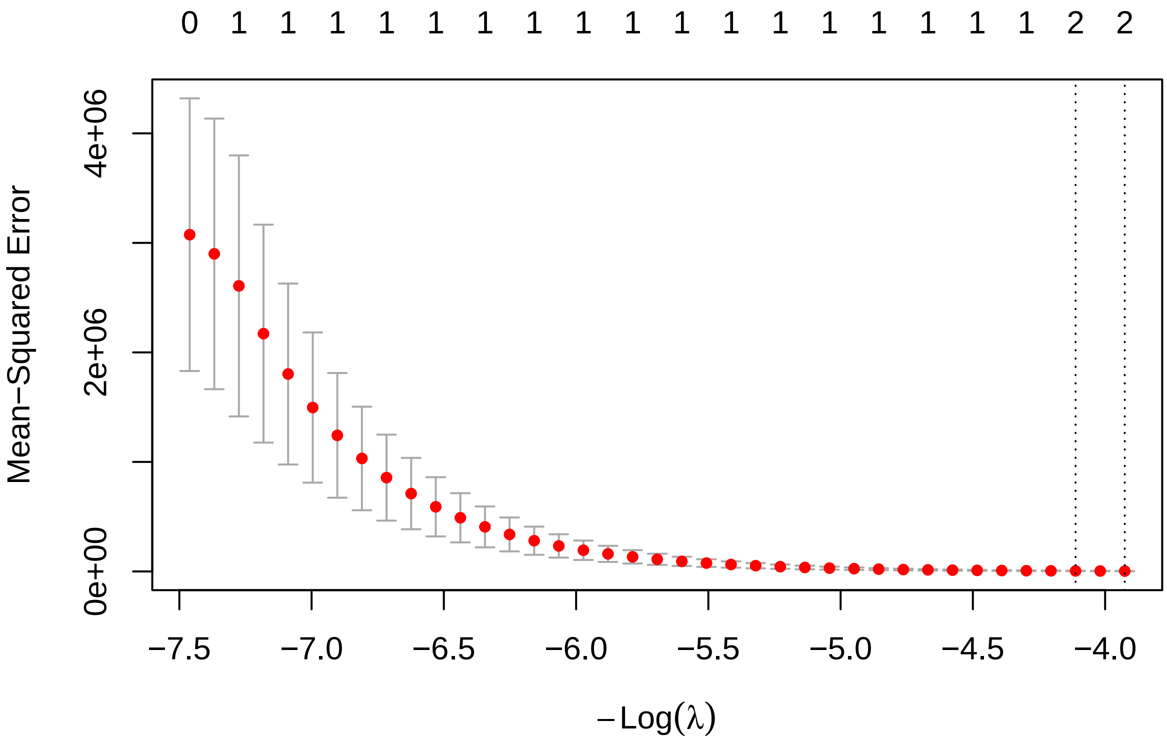
Based on the Mallow's Cp and BIC, the best subset model should be: $Y = 8.0092 + 7.0001X^7$, if based on the adjusted R-squared, the best subset model is: $Y = 8.0315 + 7.0013X^7 - 0.0005X^8$

#Lasso

```
lasso2 = cv.glmnet(poly(df2$X, 10, raw = TRUE), df2$Y2, alpha = 1, nfolds = 10, standardize = TRUE)
best_lambda2 = lasso2$lambda.min
best_lambda2
```

```
## [1] 50.68498
```

```
plot(lasso2)
```




```
coef_min2 = coef(lasso2, s = "lambda.min")
coef_min2
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##              lambda.min
## (Intercept) 19.2461374
## 1            .
## 2            .
## 3            .
## 4            .
## 5            0.2323746
## 6            .
## 7            6.7638456
## 8            .
## 9            .
## 10           .
```

The best Lasso model is: $Y = 19.2461 + 0.2324X^5 + 6.7638X^7$.

11.

(a).

```
boston = ISLR2::Boston
head(boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

```r
#We will use 10-fold CV RMSE to evaluate the models
y = boston$crim
X_mm = model.matrix(data = boston, crim ~ .)[, -1]
p = ncol(X_mm)

pred_regsubsets = function(object, newdata, id){
  form = as.formula(paste("crim ~", paste(names(coef(object, id))[-1], collapse = "+")))
  mm = model.matrix(form, newdata)
  drop(mm %*% coef(object, id))
}

rmse = function(y, yhat) sqrt(mean((y - yhat)^2))

K = 10
fold = sample(rep(1:K, length.out = nrow(boston)))
```

#Best subsets

```r
best_sub3 = regsubsets(data = boston, crim ~ ., nvmax = p)
summary5 = summary(best_sub3)
summary5
```

```
## Subset selection object
## Call: regsubsets.formula(data = boston, crim ~ ., nvmax = p)
## 12 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 12
## Selection Algorithm: exhaustive
##            zn  indus chas nox rm  age dis rad tax ptratio lstat medv
## 1  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     " "   " "
## 2  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     "*"   " "
## 3  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     "*"   "*"
## 4  ( 1 )  "*" " "   " "  " " " " " " "*" "*" " " " "     " "   "*"
## 5  ( 1 )  "*" "*"   " "  " " " " " " "*" "*" " " " "     " "   "*"
## 6  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " "*"     " "   "*"
## 7  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " "*"     "*"   "*"
## 8  ( 1 )  "*" "*"   " "  "*" " " " " "*" "*" " " "*"     "*"   "*"
## 9  ( 1 )  "*" "*"   " "  "*" "*" " " "*" "*" " " "*"     "*"   "*"
## 10  ( 1 ) "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 11  ( 1 ) "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 12  ( 1 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

```r
min_Cp5 = which.min(summary5$cp)
min_BIC5 = which.min(summary5$bic)
max_adjR5 = which.max(summary5$adjr2)

cat(
  "Min Mallow's CP:", min_Cp5, "\n",
  "Min BIC:", min_BIC5, "\n",
  "Max Adjusted R-squared:", max_adjR5, sep = "")
```

```
## Min Mallow's CP:7
## Min BIC:2
## Max Adjusted R-squared:9
```

```r
coef(best_sub3, min_Cp5)
```

```
## (Intercept)          zn         nox         dis         rad    ptratio
##  17.4668235   0.0449679 -12.4578238  -0.9425497   0.5615224  -0.3470306
##       lstat        medv
##   0.1147895  -0.1902559
```

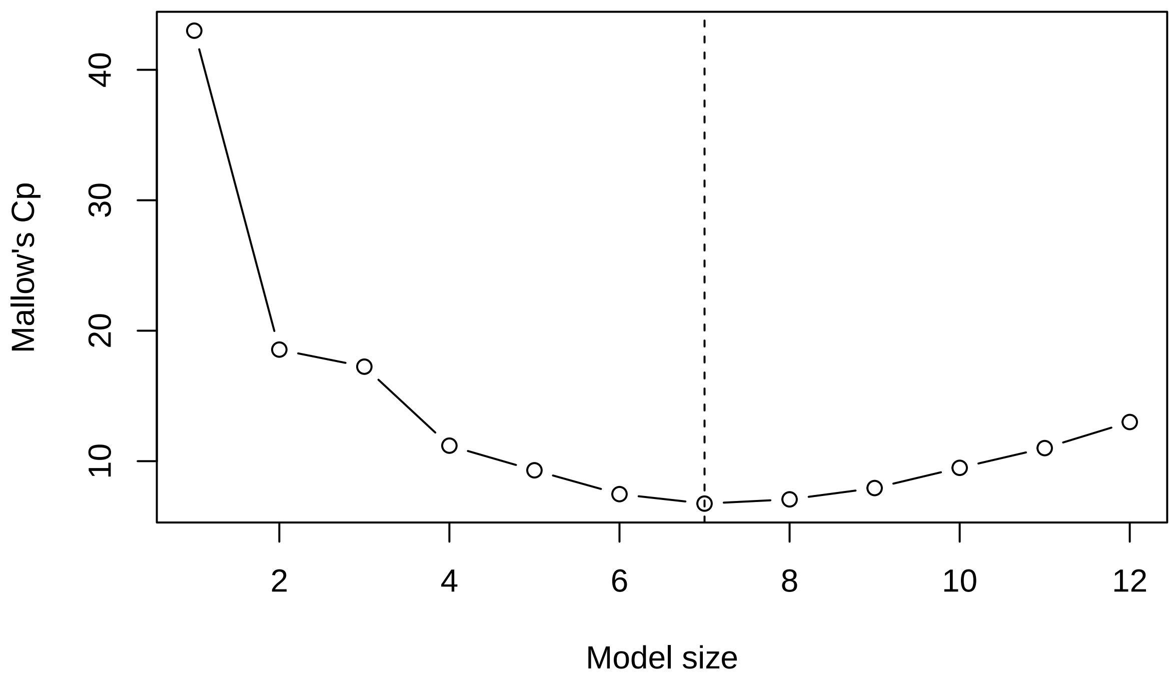
```r
coef(best_sub3, min_BIC5)
```

```
## (Intercept)         rad       lstat
##  -4.3814053   0.5228128   0.2372846
```

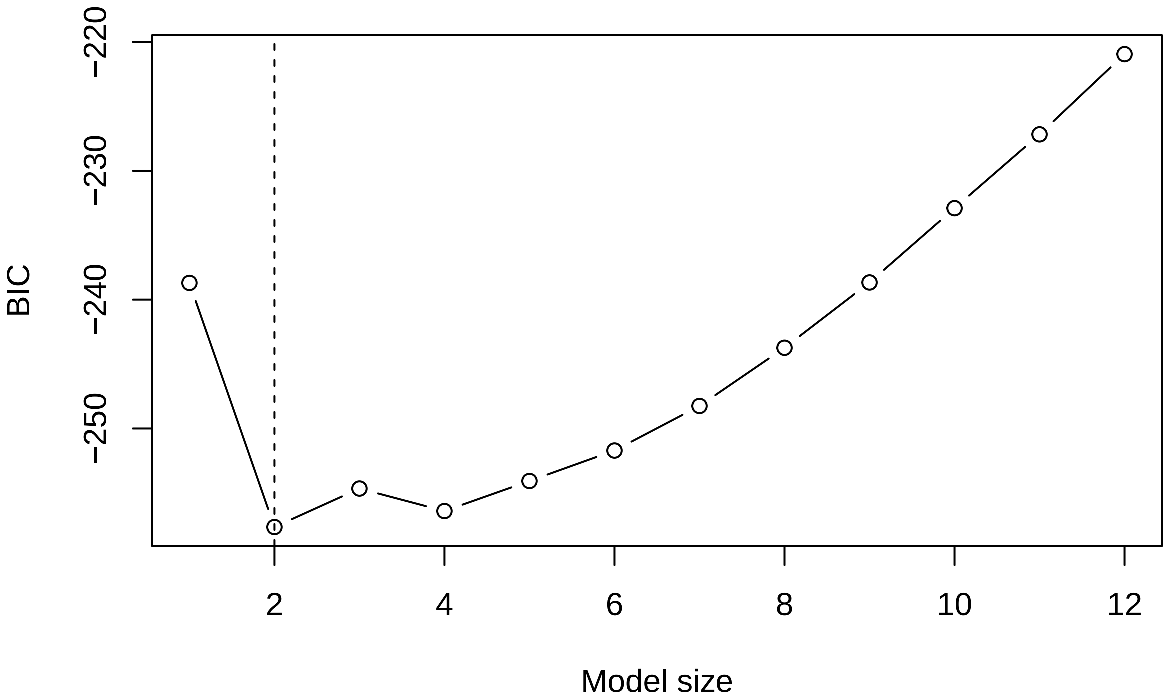
```r
coef(best_sub3, max_adjR5)
```

```
##  (Intercept)           zn        indus          nox           rm          dis
## 13.18247199   0.04305936  -0.08820604 -10.46823468   0.63510592  -1.00637216
##          rad      ptratio        lstat         medv
##   0.56096588  -0.30423849   0.14040855  -0.22012525
```

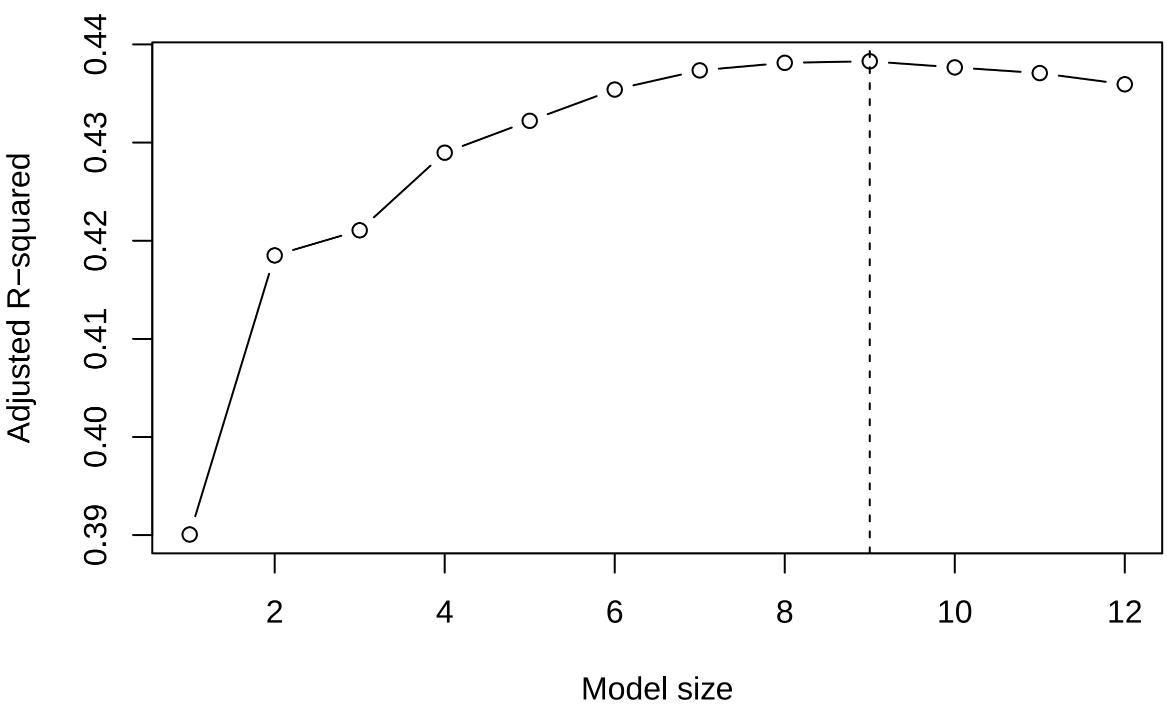
```r
par(mfrow = c(1,1))
plot(1:p, summary5$cp, type = "b", xlab = "Model size", ylab = "Mallow's Cp")
abline(v = min_Cp5, lty = 2)
```



```r
plot(1:p, summary5$bic, type = "b", xlab = "Model size", ylab = "BIC")
abline(v = min_BIC5, lty = 2)
```




```r
plot(1:p, summary5$adjr2, type = "b", xlab = "Model size", ylab = "Adjusted R-squared")
abline(v = max_adjR5, lty = 2)
```

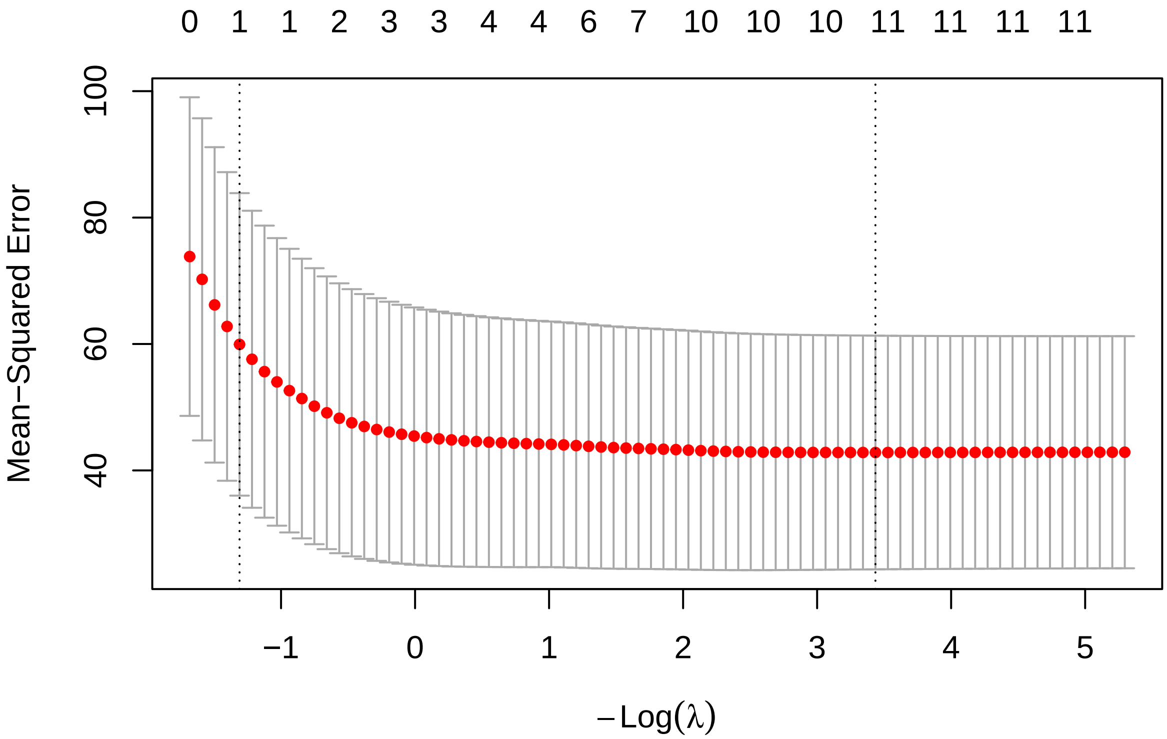


Based on the Mallow's Cp, the best subset model should be: $crim = 17.4668 + 0.04497zn - 12.4578nox - 0.9425dis + 0.5615rad - 0.3470ptratio + 0.1148lstat - 0.1903medv$. Based on the BIC, $crim = -4.3814 + 0.5228rad + 0.2373lstat$. For the adjusted R-squared, the best subset model is: $crim = 13.1825 + 0.0431zn - 0.0882indus - 10.4682nox + 0.6351rm - 1.0064dis + 0.5610rad -$

$0.3042ptratio + 0.1404lstat - 0.2201medv$

#Lasso

```
cv_lasso = cv.glmnet(X_mm, y, alpha = 1, nfolds = 10)
plot(cv_lasso)
```



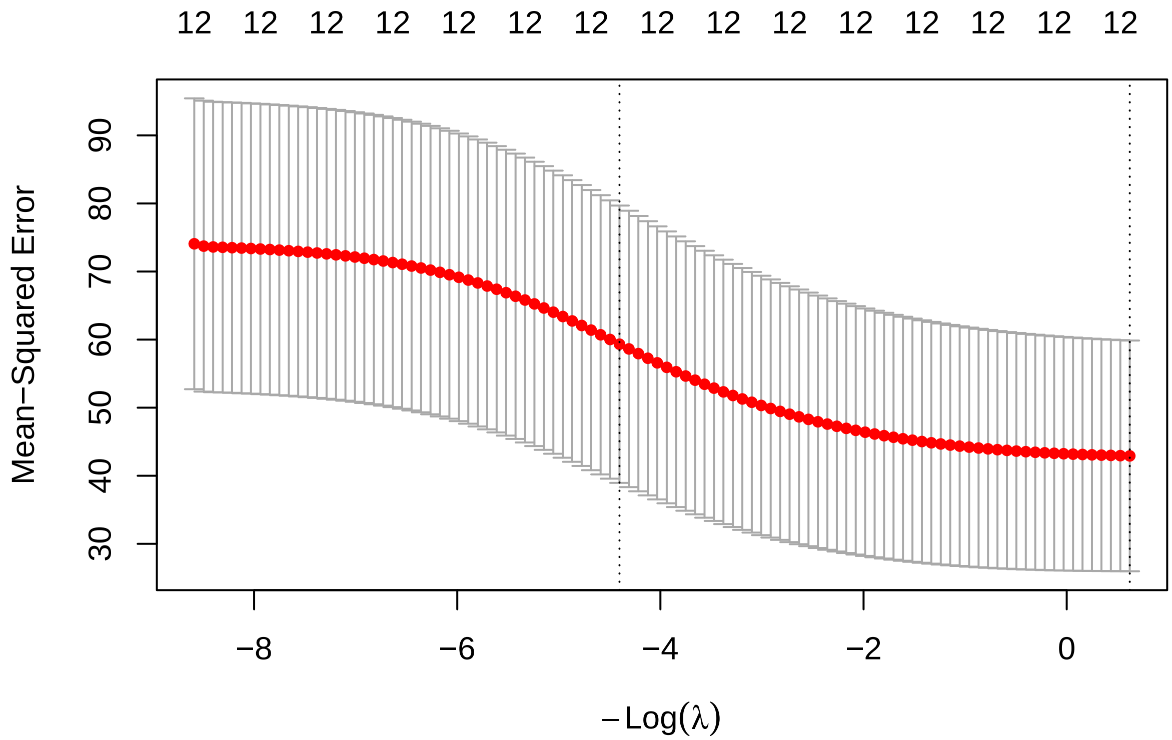

```
coef(cv_lasso, s = "lambda.min")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                  lambda.min
## (Intercept) 10.9830172958
## zn           0.0400473944
## indus       -0.0662084982
## chas        -0.7160278330
## nox         -8.0426443608
## rm           0.4946556538
## age          .
## dis         -0.8790279219
## rad          0.5592234693
## tax         -0.0008828103
## ptratio     -0.2541193818
## lstat        0.1364191296
## medv        -0.1937324075
```

Based on the best λ, the best subset model should be: $crim = 10.9830 + 0.0400zn - 0.0662indus - 0.7160chas - 8.0426nox + 0.4947rm - 0.8790dis + 0.5592rad - 0.0009tax - 0.2541ptratio + 0.1364lstat - 0.1937medv$.

#Ridge

```
cv_ridge = cv.glmnet(X_mm, y, alpha = 0, nfolds = 10)
plot(cv_ridge)
```




```
coef(cv_ridge, s = "lambda.min")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                 lambda.min
## (Intercept)  5.222257e+00
## zn           3.375080e-02
## indus       -7.773854e-02
## chas        -8.254911e-01
## nox         -4.905342e+00
## rm           5.165813e-01
## age          9.537272e-05
## dis         -7.217088e-01
## rad          4.433556e-01
## tax          3.737247e-03
## ptratio     -1.642366e-01
## lstat        1.563538e-01
## medv        -1.584061e-01
```

The best ridge model is:$crim = 5.2223 + 0.0338zn - 0.0777indus - 0.8255chas - 4.9053nox + 0.5166rm + 0.0000954age - 0.7217dis + 0.4434rad + 0.00374tax - 0.1642ptratio + 0.1564lstat - 0.1584medv$.

(b).

```
#Best subsets
cv_bs = matrix(NA_real_, K, p)
for(k in 1:K){
  fit = regsubsets(crim ~ ., data = boston[fold != k, ], nvmax = p)
  for(m in 1:p){
    yhat = pred_regsubsets(fit, boston[fold == k, ], id = m)
    cv_bs[k, m] = sqrt(mean( (boston$crim[fold == k] - yhat)^2 ))
  }
}
```

```r
rmse_bs = colMeans(cv_bs)
size_bs = which.min(rmse_bs)
rmse_bs_min = min(rmse_bs)
coef_bs = coef(regsubsets(crim ~ ., data = boston, nvmax = p), size_bs)
coef_bs
```

```
##   (Intercept)            zn         indus          chas           nox
##  13.801555544   0.045818028  -0.058345869  -0.828283847 -10.022404078
##            rm           dis           rad           tax       ptratio
##   0.623650191  -1.008539667   0.612822152  -0.003782956  -0.304784434
##         lstat          medv
##   0.137698956  -0.220092318
```

```r
#Lasso
rmse_lasso = numeric(K)
lambda_lass = numeric(K)

for(k in 1:K){
  tr = fold != k; te = !tr
  fit_cv = cv.glmnet(X_mm[tr, ], y[tr], alpha = 1)
  lambda_lass[k] = fit_cv$lambda.min
  pred = as.numeric(predict(fit_cv, newx = X_mm[te, ], s = "lambda.min"))
  rmse_lasso[k] = rmse(y[te], pred)
}

rmse_lasso_mean = mean(rmse_lasso)

lasso_full = cv.glmnet(X_mm, y, alpha = 1)
coef_lasso = coef(lasso_full, s = "lambda.min")
coef_lasso
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##              lambda.min
## (Intercept)  9.76734200
## zn           0.03781780
## indus       -0.06715706
## chas        -0.67616141
## nox         -7.08411010
## rm           0.43340686
## age          .
## dis         -0.81982181
## rad          0.54122756
## tax          .
## ptratio     -0.23042121
## lstat        0.13565977
## medv        -0.18214662
```

```r
#Ridge
rmse_ridge = numeric(K)
lambda_rid = numeric(K)

for(k in 1:K){
  tr = fold != k; te <- !tr
  fit_cv = cv.glmnet(X_mm[tr, ], y[tr], alpha = 0)
  lambda_rid[k] = fit_cv$lambda.min
```

```r
  pred = as.numeric(predict(fit_cv, newx = X_mm[te, ], s = "lambda.min"))
  rmse_ridge[k] = rmse(y[te], pred)
}

rmse_ridge_mean = mean(rmse_ridge)
ridge_full = cv.glmnet(X_mm, y, alpha = 0)
coef_ridge = coef(ridge_full, s = "lambda.min")
coef_ridge
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                lambda.min
## (Intercept)  5.222257e+00
## zn           3.375080e-02
## indus       -7.773854e-02
## chas        -8.254911e-01
## nox         -4.905342e+00
## rm           5.165813e-01
## age          9.537272e-05
## dis         -7.217088e-01
## rad          4.433556e-01
## tax          3.737247e-03
## ptratio     -1.642366e-01
## lstat        1.563538e-01
## medv        -1.584061e-01
```

```r
c(
  RMSE_best_subset = rmse_bs_min,
  RMSE_lasso       = rmse_lasso_mean,
  RMSE_ridge       = rmse_ridge_mean
)
```

```
## RMSE_best_subset       RMSE_lasso       RMSE_ridge
##         5.707388         5.690106         5.669860
```

Based on the result, we will choose Ridge as the best model as it has the least RMSE. The best model is: $crim = 5.2223 + 0.0338zn - 0.0777indus - 0.8255chas - 4.9053nox + 0.5166rm + 0.0000954age - 0.7217dis + 0.4434rad + 0.00374tax - 0.1642ptratio + 0.1564lstat - 0.1584medv$.

(c). The ridge model features all the variables. Ridge will shrinks coefficients toward 0 but does not set them exactly to 0.