HW2

Yihang Ding

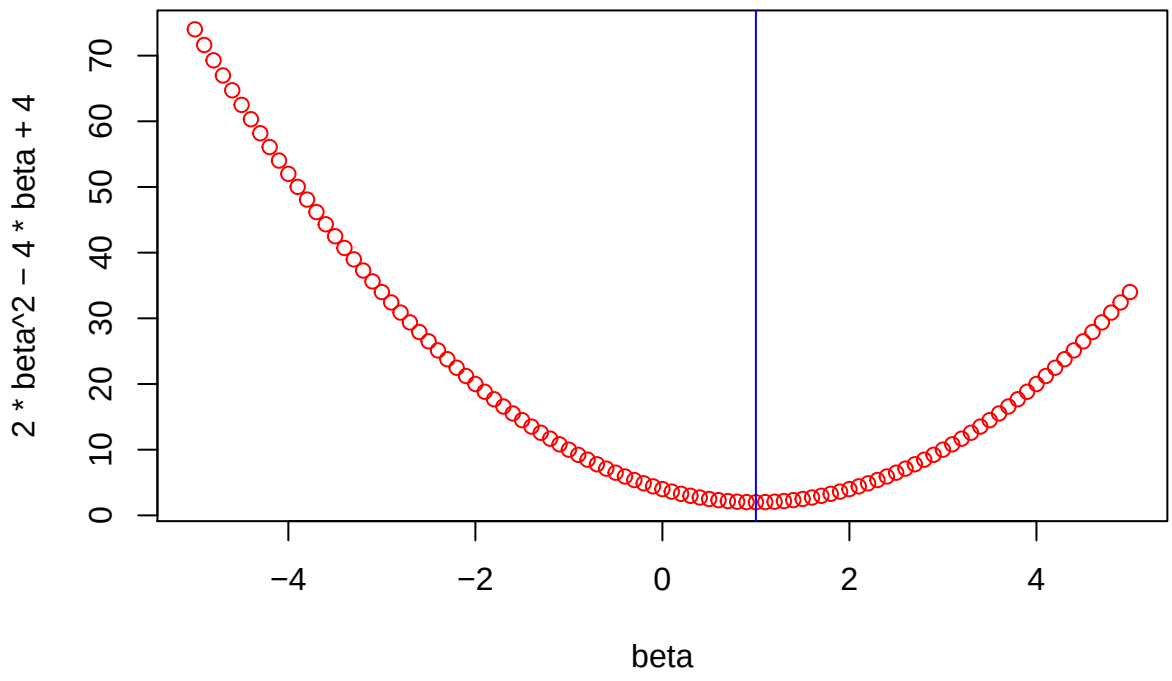
10/10/2019

```
library(lasso2)
```

```
## R Package to solve regression problems while imposing
##   an L1 constraint on the parameters. Based on S-plus Release 2.1
## Copyright (C) 1998, 1999
## Justin Lokhorst   <jlokhors@stats.adelaide.edu.au>
## Berwin A. Turlach <bturlach@stats.adelaide.edu.au>
## Bill Venables     <wvenable@stats.adelaide.edu.au>
##
## Copyright (C) 2002
## Martin Maechler <maechler@stat.math.ethz.ch>
```

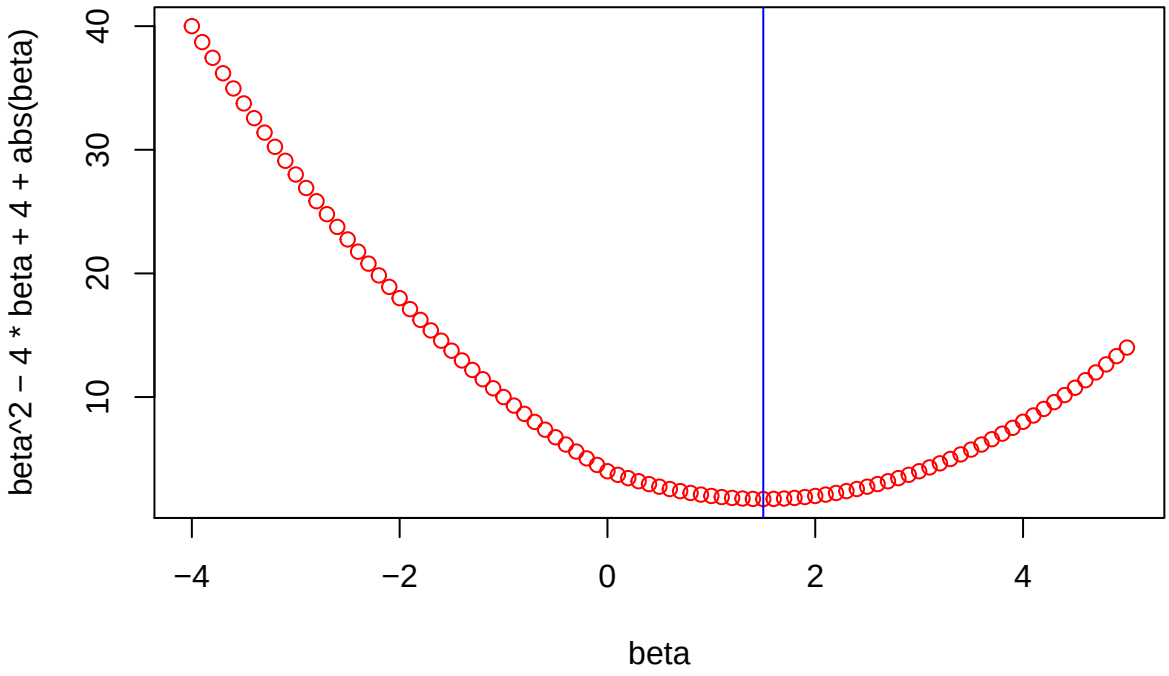
P2 (a) With $y_1 = 2$ and $\lambda = 1$, (1) equals to $(2 - \beta_1)^2 + \beta_1^2$, i.e. $2\beta_1^2$-$4\beta_1$+4 Under which in (3), $\hat{\beta}_1^R = 2/(1+\lambda) = 1$ So the min value of plot should be where x = 1, which can be proved by the plot.

```
beta = seq(from = -5, to = 5, by = 0.1)
plot(beta, 2*beta^2-4*beta+4, col='red', type = "b")
abline(v=1, col='blue')
```



(b) With $y_1 = 2$ and $\lambda = 1$, (2) equals to $(2 - \beta_1)^2 + |\beta_1|$, i.e. β_1^2-4β_1+4+$|\beta_1|$ Under which in (4), $\hat{\beta_1^R}$ = 2-1/2 = 1.5 So the min value of plot should be where x = 1.5.

```
beta = seq(from = -4, to = 5, by = 0.1)
plot(beta, beta^2-4*beta+4+abs(beta), col='red', type = "b")
abline(v=1.5, col='blue')
```



P4 a)

```
set.seed(1)
x = rnorm(100, 0, 1)
eps = rnorm(100, 0, 0.25)
```

b)

```
b0 = 1
b1 = 2
b2 = 3
b3 = 4
y = b0 + b1*x + b2*I(x^2) + b3*I(x^3) + eps
```

c)

```r
library(leaps)
library(ISLR)
df = data.frame(y, x)
fit = regsubsets(y ~ poly(x, 10), data = df, nvmax = 10)
fit_sum = summary(fit)
which.min(fit_sum$cp)
```

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

```r
which.min(fit_sum$bic)
```

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

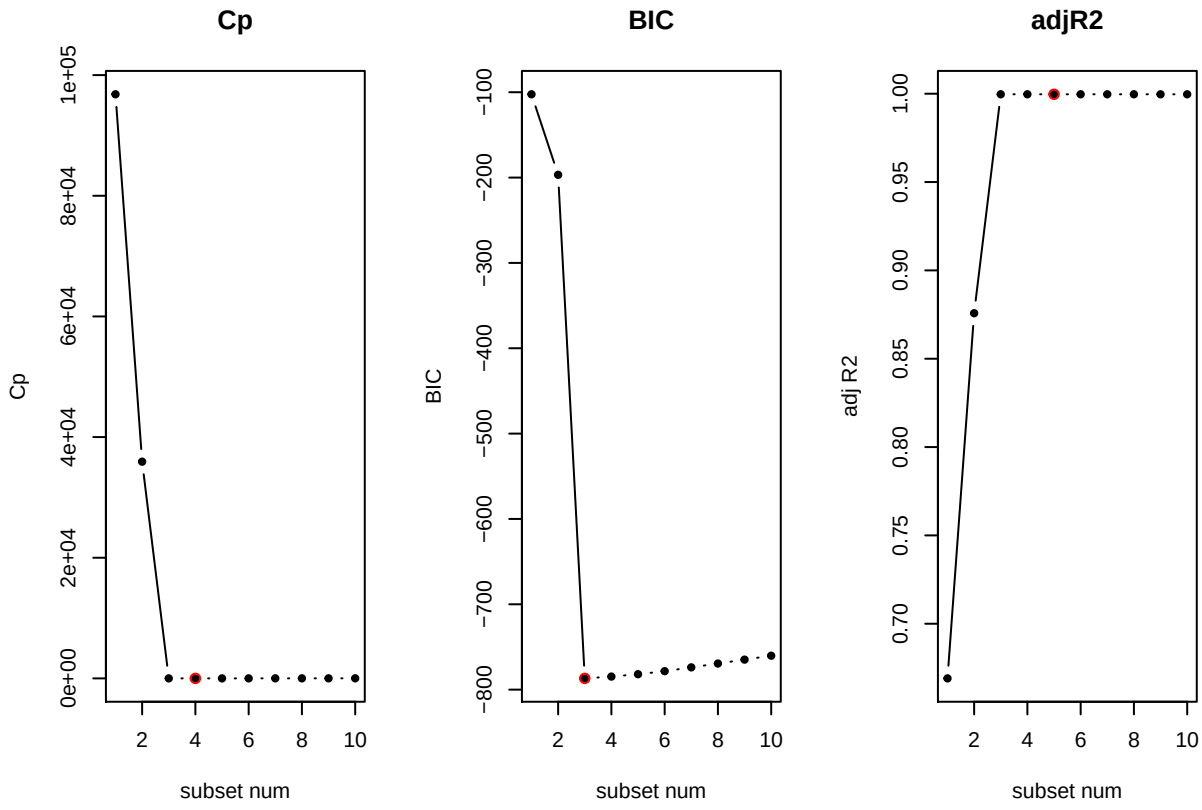
```r
which.max(fit_sum$adjr2)
```

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

```r
par(mfrow = c(1,3))
plot(fit_sum$cp, xlab = "subset num", ylab = "Cp", type = "b", pch = 20)
title("Cp")
points(4, fit_sum$cp[4], col = 'red')

plot(fit_sum$bic, xlab = "subset num", ylab = "BIC", type = "b", pch = 20)
title("BIC")
points(3, fit_sum$bic[3], col = "red")

plot(fit_sum$adjr2, xlab = "subset num", ylab = "adj R2", type = "b", pch = 20)
title("adjR2")
points(5, fit_sum$adjr2[5], col = "red")
```




```
coefficients(fit, id = 3)
```

```
##  (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3
##     4.482519   108.370705    46.051860    59.958781
```

(d) forward

```
fit_fwd = regsubsets(y ~ poly(x, 10), data = df, nvmax = 10, method = "forward")
fwd_sum = summary(fit_fwd)
which.min(fwd_sum$cp)
```

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

```
which.min(fwd_sum$bic)
```

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

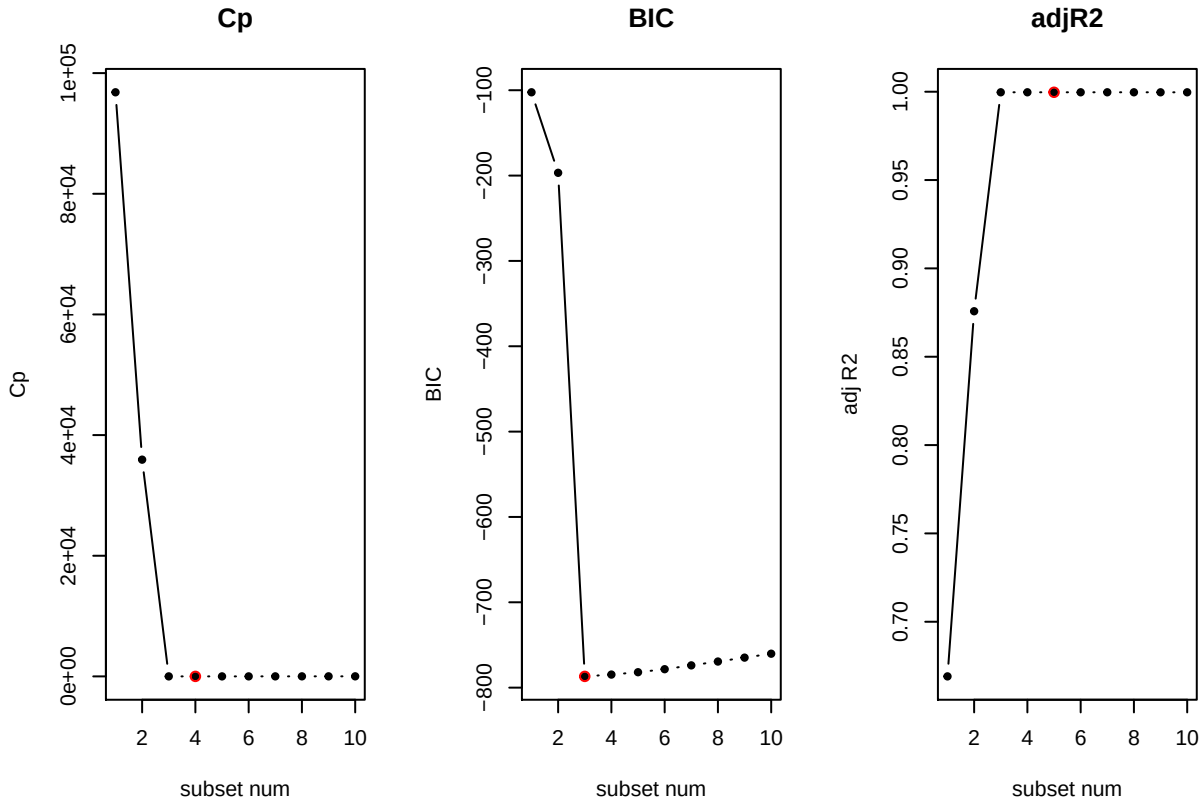
```
which.max(fwd_sum$adjr2)
```

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

```r
par(mfrow = c(1,3))
plot(fwd_sum$cp, xlab = "subset num", ylab = "Cp", type = "b", pch = 20)
title("Cp")
points(4, fwd_sum$cp[4], col = 'red')

plot(fwd_sum$bic, xlab = "subset num", ylab = "BIC", type = "b", pch = 20)
title("BIC")
points(3, fwd_sum$bic[3], col = "red")

plot(fwd_sum$adjr2, xlab = "subset num", ylab = "adj R2", type = "b", pch = 20)
title("adjR2")
points(5, fwd_sum$adjr2[5], col = "red")
```




```r
coefficients(fit_fwd, id = 3)
```

```
##  (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3
##     4.482519   108.370705    46.051860    59.958781
```

backward

```r
fit_bwd = regsubsets(y ~ poly(x, 10), data = df, nvmax = 10, method = "backward")
# backward
bwd_sum = summary(fit_fwd)
which.min(bwd_sum$cp)
```

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

```
which.min(bwd_sum$bic)
```

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

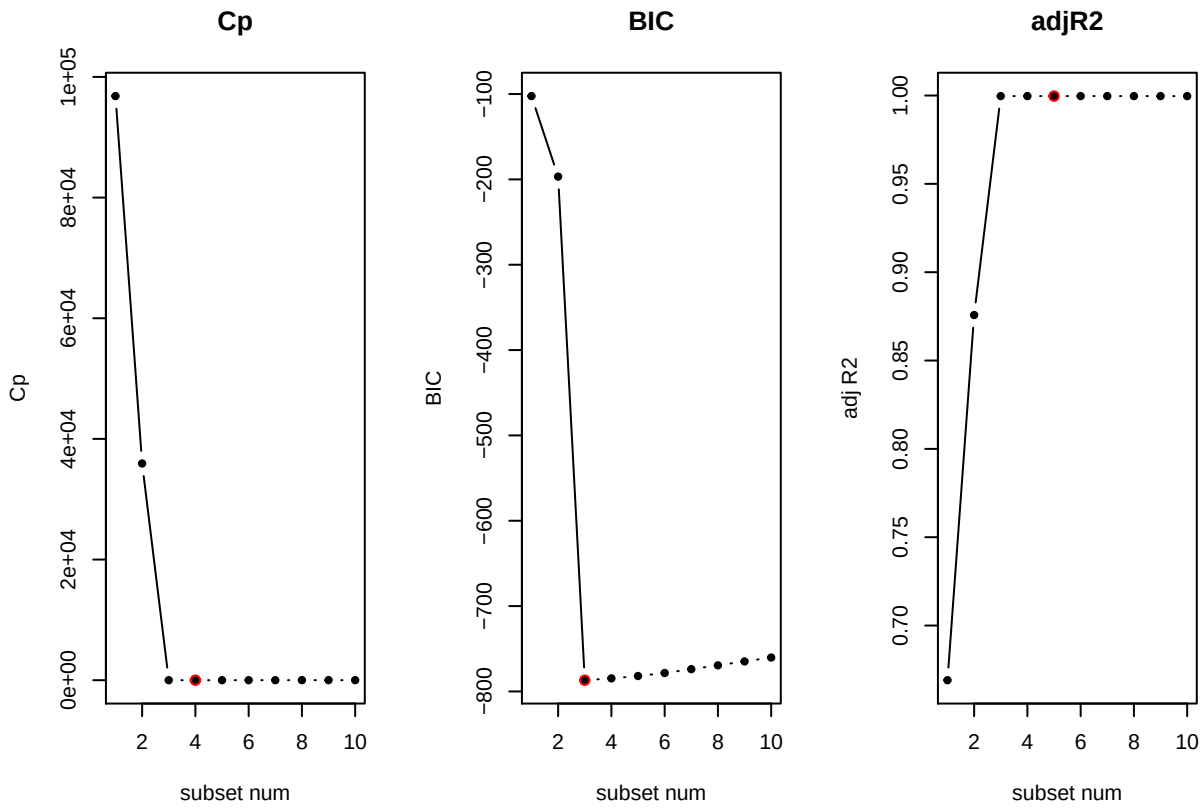
```
which.max(bwd_sum$adjr2)
```

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

```
par(mfrow = c(1,3))
plot(bwd_sum$cp, xlab = "subset num", ylab = "Cp", type = "b", pch = 20)
title("Cp")
points(4, bwd_sum$cp[4], col = 'red')

plot(bwd_sum$bic, xlab = "subset num", ylab = "BIC", type = "b", pch = 20)
title("BIC")
points(3, bwd_sum$bic[3], col = "red")

plot(bwd_sum$adjr2, xlab = "subset num", ylab = "adj R2", type = "b", pch = 20)
title("adjR2")
points(5, bwd_sum$adjr2[5], col = "red")
```



```
coefficients(fit_bwd, id = 3)
```

```
##  (Intercept) poly(x, 10)1 poly(x, 10)2 poly(x, 10)3
##     4.482519   108.370705    46.051860    59.958781
```

Both forward and backword method pick X^1, X^2, X^2

(e)

```
library(glmnet)
```

```
## Loading required package: Matrix
```

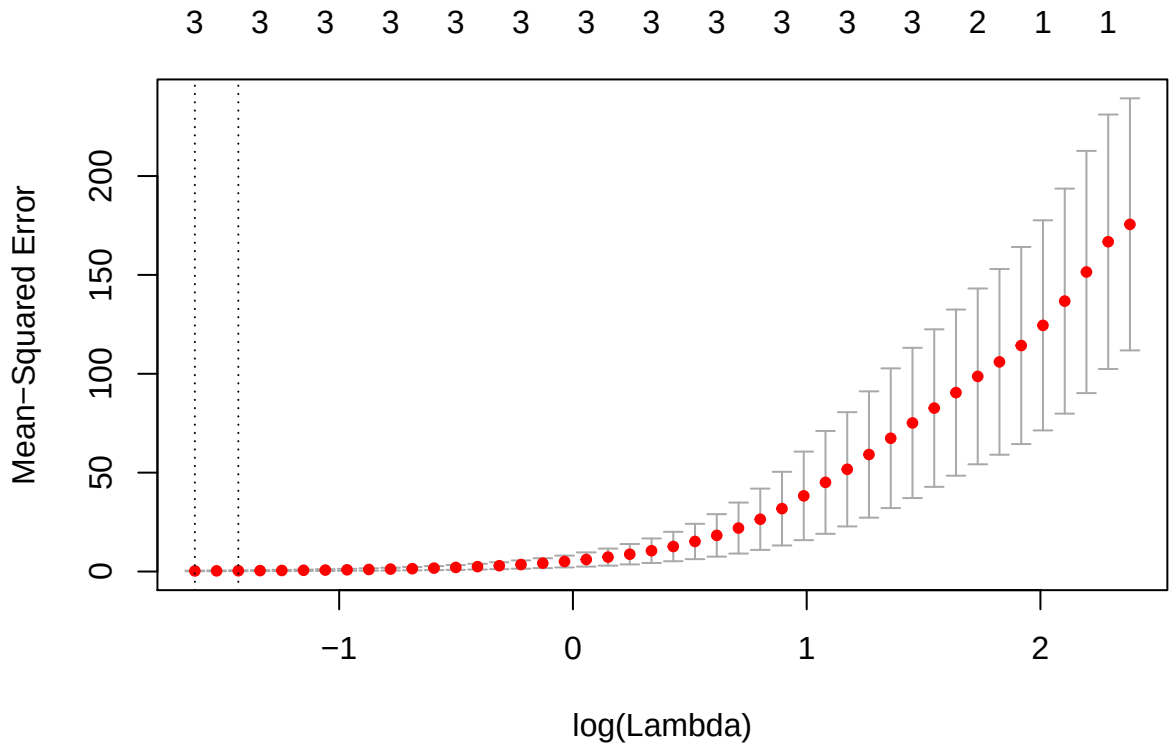
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

```
mat = model.matrix(y ~ poly(x, 10), data = df)[,-1]
lasso = cv.glmnet(mat, y, alpha = 1)
lambda_fit = lasso$lambda.min
lambda_fit
```

```
## [1] 0.1983984
```

```
plot(lasso)
```



```
model_fit = glmnet(mat, y)
predict(model_fit, s = lambda_fit, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)     4.482519
## poly(x, 10)1  106.386721
## poly(x, 10)2   44.067876
## poly(x, 10)3   57.974797
## poly(x, 10)4    .
## poly(x, 10)5    .
## poly(x, 10)6    .
## poly(x, 10)7    .
## poly(x, 10)8    .
## poly(x, 10)9    .
## poly(x, 10)10   .
```

Lasso also picks X^1, X^2, X^3.

(f) Set $\beta_7 = 7$

```
b7 = 7
y = b0 + b7*x^7 + eps
data = data.frame(y, x)
fit = regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10)
fit_sum = summary(fit)
which.min(fit_sum$cp)
```

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

```
which.min(fit_sum$bic)
```

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

```
which.max(fit_sum$adjr2)
```

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

```
coefficients(fit, id = 2)
```

```
##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##            1.01762259          -0.03542711            7.00038880
```

```
coefficients(fit, id = 1)
```

```
##           (Intercept) poly(x, 10, raw = T)7
##             0.9897351             7.0001926
```

```
coefficients(fit, id = 4)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            1.01906311           0.07285040          -0.04044178
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
##           -0.06316317           7.00228344
```

BIC picks only 1 variable while Cp and adjustedR2 pick more variables.

P5

```
library(ISLR)
data('College')
typeof(College)
```

```
## [1] "list"
```

```
dim(College)
```

```
## [1] 777  18
```

(a) train test split: portion: 0.7 train, 0.3 test

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------ tidyverse 1.2.1 --
```

```
## v tibble  2.1.3     v purrr   0.3.2
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v tibble  2.1.3     v forcats 0.4.0
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x purrr::accumulate() masks foreach::accumulate()
## x tidyr::expand()     masks Matrix::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x purrr::lift()       masks caret::lift()
## x tidyr::pack()       masks Matrix::pack()
## x tidyr::unpack()     masks Matrix::unpack()
## x purrr::when()       masks foreach::when()
```

```
library(caTools)

set.seed(1)
sample = sample.split(College,SplitRatio = 0.7)
training = subset(College,sample ==TRUE)
testing = subset(College, sample==FALSE)

preprocessor = preProcess(training, method = c('center', 'scale'))
training = predict(preprocessor, training)
testing = predict(preprocessor, testing)

encoder = dummyVars(Apps ~ ., data = training)
x_train = predict(encoder, training)
x_test = predict(encoder, testing)

y_train = training$Apps
y_test = testing$Apps
```

(b) linear

```
lm <- lm(Apps ~ ., data = training)

pred <- predict(lm, testing)
summary(pred)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.85863 -0.56372 -0.26918  0.04561  0.32488  4.34931
```

```
postResample(pred, testing$Apps)
```

```
##      RMSE  Rsquared       MAE
## 0.2668491 0.9133310 0.1577525
```

(c) ridge

```
ridge = cv.glmnet(x_train, y_train, alpha = 0)
ridge_fit = glmnet(x_train, y_train, alpha = 0)
lambda_fit = ridge$lambda.min
lambda_fit
```

```
## [1] 0.09487939
```

```
ridge_pred = predict(ridge_fit, s = lambda_fit, newx = x_test)
mean((ridge_pred - y_test)^2)
```

```
## [1] 0.06936452
```

(d) lasso

```r
lasso = cv.glmnet(x_train, y_train, alpha = 1)
lasso_fit = glmnet(x_train, y_train, alpha = 1)
lambda_fit = lasso$lambda.min
lambda_fit
```

```
## [1] 0.0005063428
```

```r
lasso_pred = predict(lasso_fit, s = lambda_fit, newx = x_test)
mean((lasso_pred - y_test)^2)
```

```
## [1] 0.07072009
```

```r
predict(lasso_fit, s = lambda_fit, newx = x_test, type = "coefficient")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -2.968360e-02
## Private.No   1.084915e-01
## Private.Yes -4.459203e-13
## Accept       1.027663e+00
## Enroll      -2.389106e-01
## Top10perc    2.204527e-01
## Top25perc   -6.211397e-02
## F.Undergrad  7.919157e-02
## P.Undergrad  3.221691e-02
## Outstate    -7.979188e-02
## Room.Board   3.150316e-02
## Books        1.308923e-03
## Personal    -4.235766e-05
## PhD         -3.613806e-02
## Terminal    -1.414909e-02
## S.F.Ratio    1.295488e-02
## perc.alumni  7.988022e-03
## Expend       7.384076e-02
## Grad.Rate    3.414070e-02
```

P6 (a)

```r
set.seed(1)
p = 20
n = 1000
x = matrix(rnorm(n * p), n, p)
beta = rnorm(p)
eps = rnorm(p)

beta[3] = 0
beta[8] = 0
beta[10] = 0
beta[15] = 0
y = x %*% beta + eps
dim(y)
```

```
## [1] 1000    1
```

(b) train test split

```
spliter = sample(seq(1000), 100, replace = FALSE)
x_train = x[spliter, ]
x_test = x[-spliter, ]

y_train = y[spliter, ]
y_test = y[-spliter, ]
dim(x_train)
```

```
## [1] 100  20
```

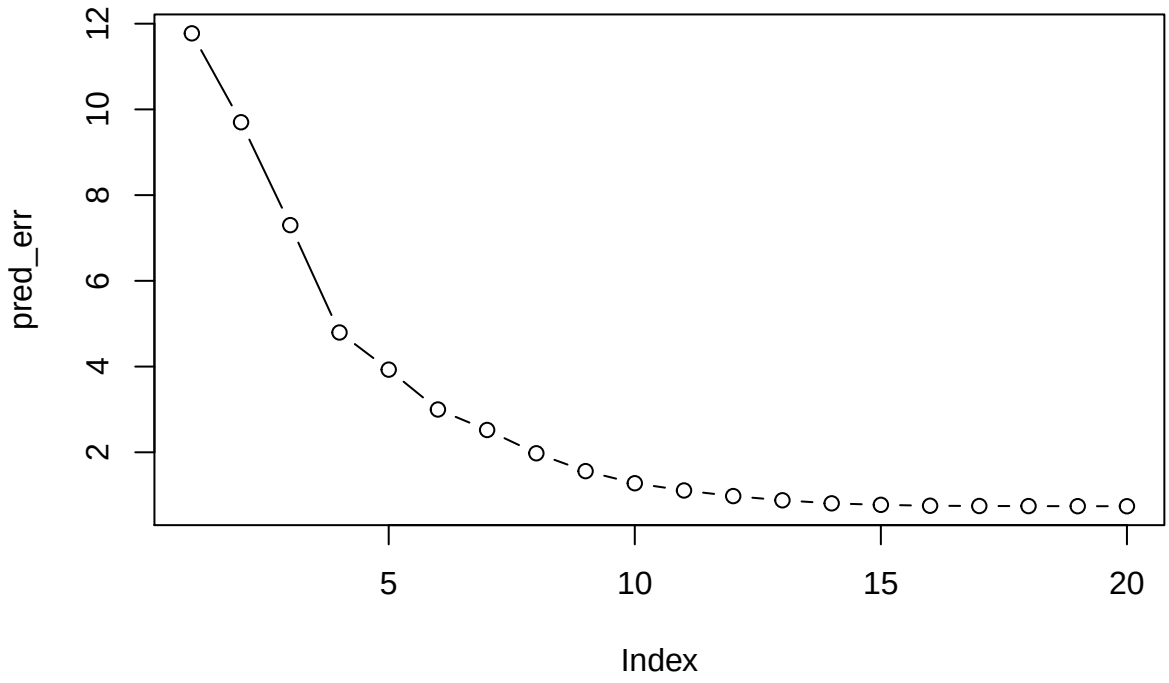
```
dim(x_test)
```

```
## [1] 900  20
```

(c) subset selection

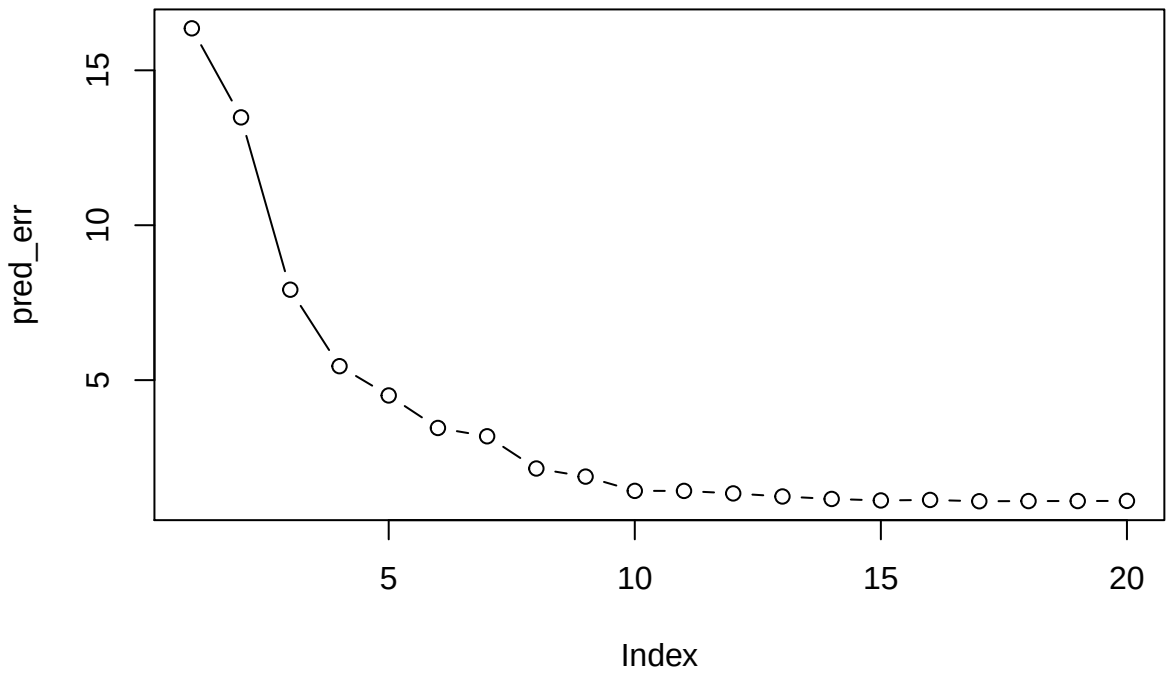
```
df = data.frame(x = x_train, y = y_train)
fit= regsubsets(y ~ ., data = df,nvmax = p)

pred_err = seq(1,p)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
# compute pred err for each p
for (i in 1:p) {
  coefi = coef(fit, id = i)
  pred =as.matrix(x_train[, x_cols %in% names(coefi)]) %*% coefi[names(coefi) %in%x_cols]
  pred_err[i] = mean((y_train - pred)^2)
}
plot(pred_err, type = "b")
```



(d) test err

```
for (i in 1:p) {
  coefi = coef(fit, id = i)
  pred =as.matrix(x_test[, x_cols %in% names(coefi)]) %*% coefi[names(coefi) %in%x_cols]
  pred_err[i] = mean((y_test - pred)^2)
}
plot(pred_err, type = "b")
```



(e)

```
which.min(pred_err)
```

```
## [1] 17
```

For model size 17, the test set MSE takes its min value.

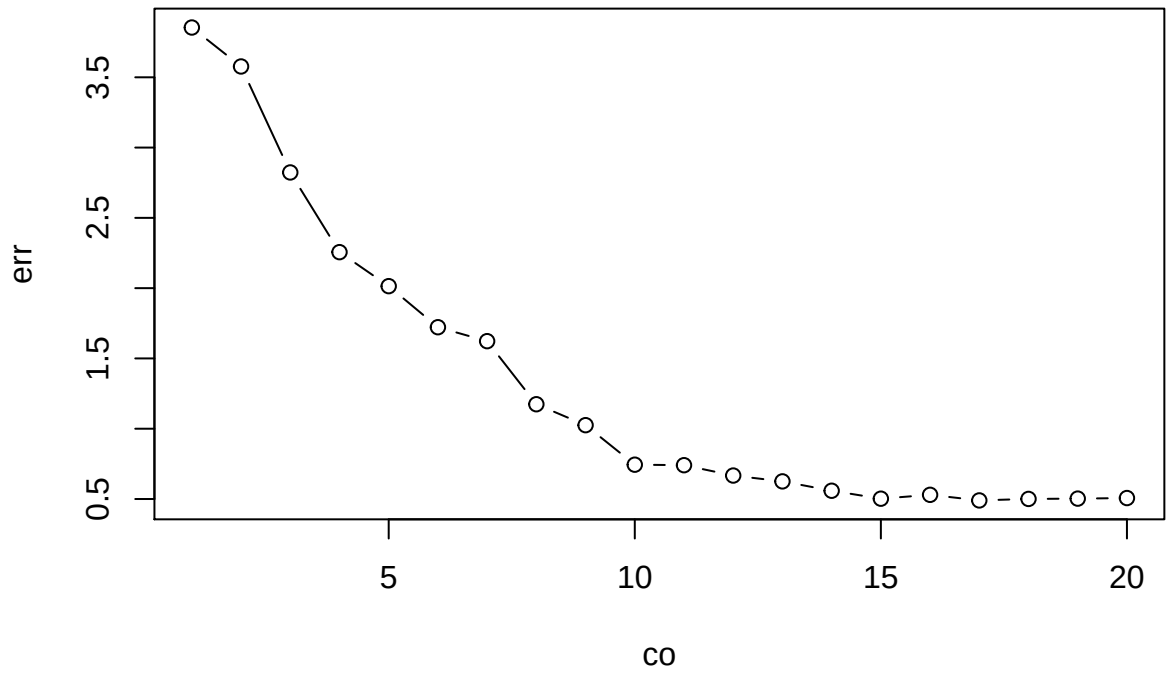
(f)

```
coef(fit, id = 17)
```

```
##   (Intercept)           x.1           x.2           x.3           x.4
##  0.0009424275  0.4177374536  0.0843139824 -0.1307606026 -1.8089460311
##           x.5           x.6           x.7           x.9          x.11
##  0.9254258079 -0.1921288988 -1.4945850480  1.8574851995  0.8611485099
##          x.12          x.13          x.14          x.16          x.17
##  0.7613779752 -0.2881103138 -0.6326761053 -0.3795376558  0.3362069098
##          x.18          x.19          x.20
##  1.5531615952  0.9145988856 -0.8806963623
```

```
pred_err = seq(1,p)
co = seq(1,p)
err = seq(1,p)
```

```r
for (i in 1:p) {
  coefi = coef(fit, id = i)
  co[i] = length(coefi) - 1
  err[i] = sqrt(sum((beta[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_cols])^2) + sum(beta[!(
}
plot(x = co, y = err, type = "b")
```



The trend of two lines are very similar, but the test MST plot decreases more drastically at the beginning.