# RSM8512 Assignment - Model Selection

Yanbing Chen 1009958752 2023/11/16

## Question 1 [20 marks]

We perform best subset, forward stepwise, and backward stepwise selection on a single data set. For each approach, we obtain $p + 1$ models, containing $0, 1, 2, ..., p$ predictors. Explain your answers.

(a) Which of the three models with $k$ predictors has the smallest training RSS?

**Answer:** best subset selection has the smallest training RSS. For k-variable model, the best subset selection considers all $C(p, k)$ models while the other two methods determine only the models derived by a certain path.

(b) Which of the three models with $k$ predictors has the smallest test RSS?

**Answer:** Best subset selection has the smallest test RSS because the best subset selection considers more models than the others.

(c) True or False:

i. The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the $(k + 1)$-variable model identified by forward stepwise selection.

**Answer:** True. Because in forward stepwise selection, the k+1-variable model uses k variables in the former iteration and 1 new variable which can result in the smallest RSS.

(ii) The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the $(k + 1)$ -variable model identified by backward stepwise selection.

**Answer:** True. Because in backward stepwise selection, the k-variable model uses k+1 variables in the former iteration and drop 1 variable among those variables which can result in the smallest RSS.

(iii) The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the $(k+1)$ -variable model identified by forward stepwise selection.

**Answer:** False. Because forward stepwise selection and backward steps selection starts from two directions and may lead two different paths.

(iv) The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the $(k+1)$ -variable model identified by forward stepwise selection.

**Answer:** False. Because forward stepwise selection and backward steps selection starts from two directions and may lead two different paths.

(v)The predictors in the k-variable model identified by best subset are a subset of the predictors in the $(k+1)$ variable model identified by best subset selection.

**Answer:** False. Because best subset selection chooses p predictors among all k predictors whose model has the smallest RSS. But forward stepwise selection chooses predictions depends on former iterations.

## Question 3 [10 marks]

Suppose we estimate the regression coefficients in a linear regression model by minimizing

$$\Sigma_{i=1}^{n}(y_i - \beta_0 - \Sigma_{j=1}^{p}\beta_j x_{ij})^2$$

subject to

$$\Sigma_{j=1}^{p}|\beta_j| <= s$$

For a particular value of s. For parts (a) through (e), indicate which of i.through v. is correct. Justify your answer.

(a) As we increase s from 0, the training RSS will:

i. Increase initially, and then eventually start decreasing in an inverted U shape. ii Decrease initially, and then eventually start increasing in a U shape.
ii. Steadily increase.
iii. Steadily decrease.
iv. Remain constant.

**Answer:** (iv) Steadily decreases. As we loosen the constraint against $\beta$, all $\beta$ increase from 0 to their least square estimate values. The training RSS starts from the maximum and it steadily decreases to the Ordinary Least Square RSS

(b) Repeat (a) for test RSS.

**Answer:** (ii) Decrease initially, and then eventually start increasing in a U shape: When $s = 0$, all $\beta$ $s$ are 0, the model is extremely simple and has a high test RSS. As we increase s, beta s assume non-zero values and model starts fitting well on test data and so test RSS decreases. Eventually, as beta s approach their full blown OLS values, they start overfitting to the training data, increasing test RSS.

(c) Repeat (a) for variance.

**Answer:** (iii) Steadily increase: When $s = 0$, the model effectively predicts a constant and has almost no variance. As we increase s, the models includes more $\beta$ s and their values start increasing. At this point, the values of $\beta$ $s$ become highly dependent on training data, thus increasing the variance.

(d) Repeat (a) for (squared) bias.

**Answer:** (iv) Steadily decrease: When s=0 , the model effectively predicts a constant and hence the prediction is far from actual value. Thus bias is high. As $s$ increases, more $\beta$ $s$ become non-zero and thus the model continues to fit training data better. And thus, bias decreases.

(e) Repeat (a) for the irreducible error.

**Answer:** (v) Remains constant: By definition, irreducible error is model independent and hence irrespective of the choice of $s$, remains constant.

## Question 8 [30 marks]

In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

(a) Use the `rnorm()` function to generate a predictor $X$ of length $n = 100$, as well as a noise vector $\varepsilon$ of length $n = 100$.

```
set.seed(1234)
X = rnorm(100)
epsilon = rnorm(100)
```

(b) Generate a response vector $Y$ of length $n = 100$ according to the model $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$ where $\beta_0, \beta_1, \beta_2, \beta_3$ are constants of your choice.

```
beta_0 = 3
beta_1 = 2
beta_2 = -3
beta_3 = 0.3
Y = beta_0 + beta_1*X + beta_2*X^2 + beta_3*X^3 + epsilon
```

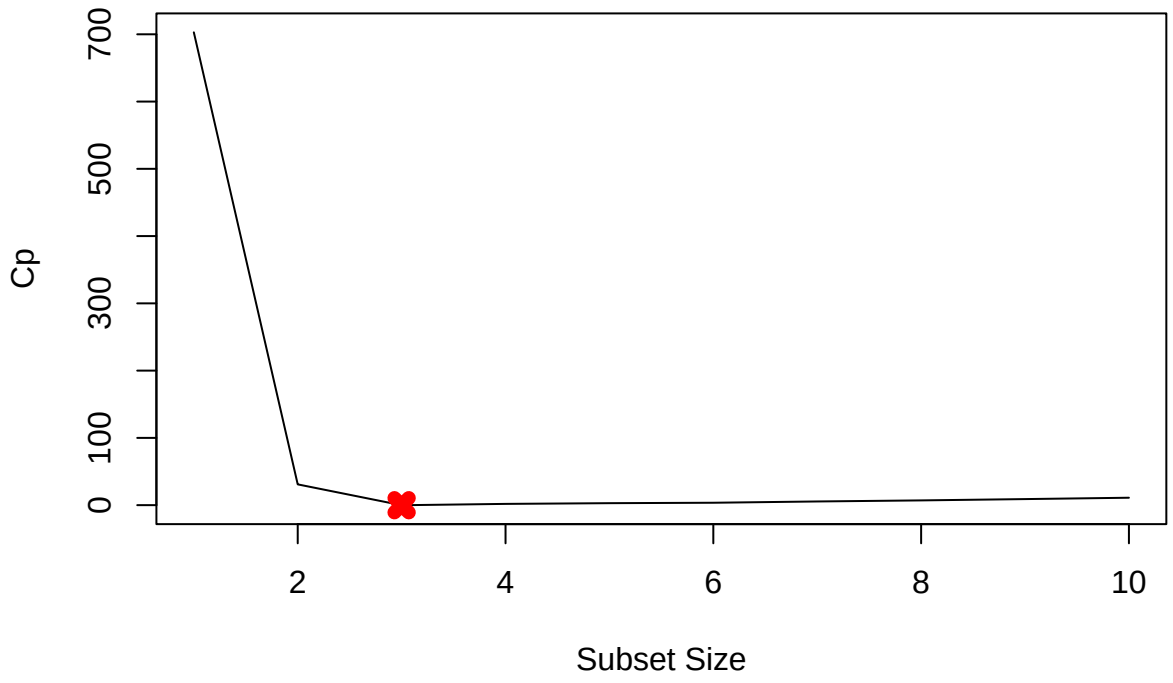
(c) Use the `regsubsets()` function to perform best subset selection in order to choose the best model containing the predictors $X, X^2, ..., X^{10}$. What is the beset model obtained according to $C_p$, BIC, and adjusted $R^2$? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the `data.frame()` function to create a single data set constraining both $X$ and $Y$.

```
df_full = data.frame(y = Y, x = X)
model_full = regsubsets(y~poly(x,10, raw = T), data = df_full, nvmax = 10)
model_full_summary = summary(model_full)
```

```
# Model size for best cp
which.min(model_full_summary$cp)
```

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

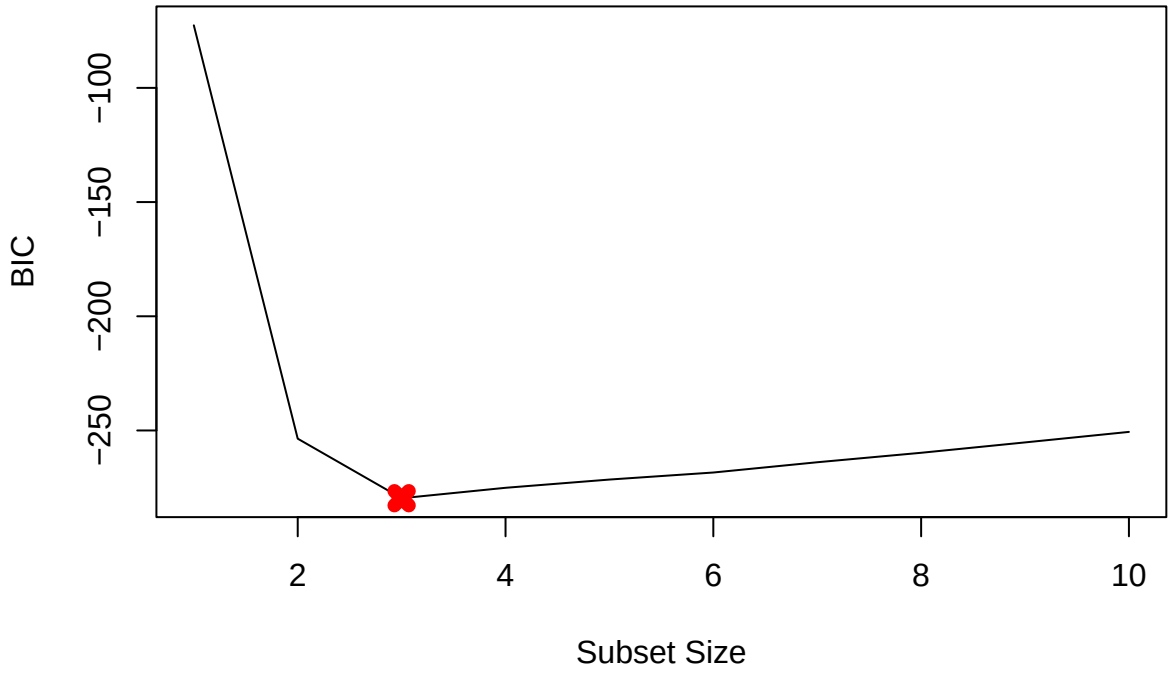
```
plot(model_full_summary$cp, xlab = "Subset Size", ylab = "Cp", pch = 20, type = "l")
points(3, model_full_summary$cp[3], pch = 4, col = "red", lwd = 7)
```




```
# Model size for best BIC
which.min(model_full_summary$bic)
```

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

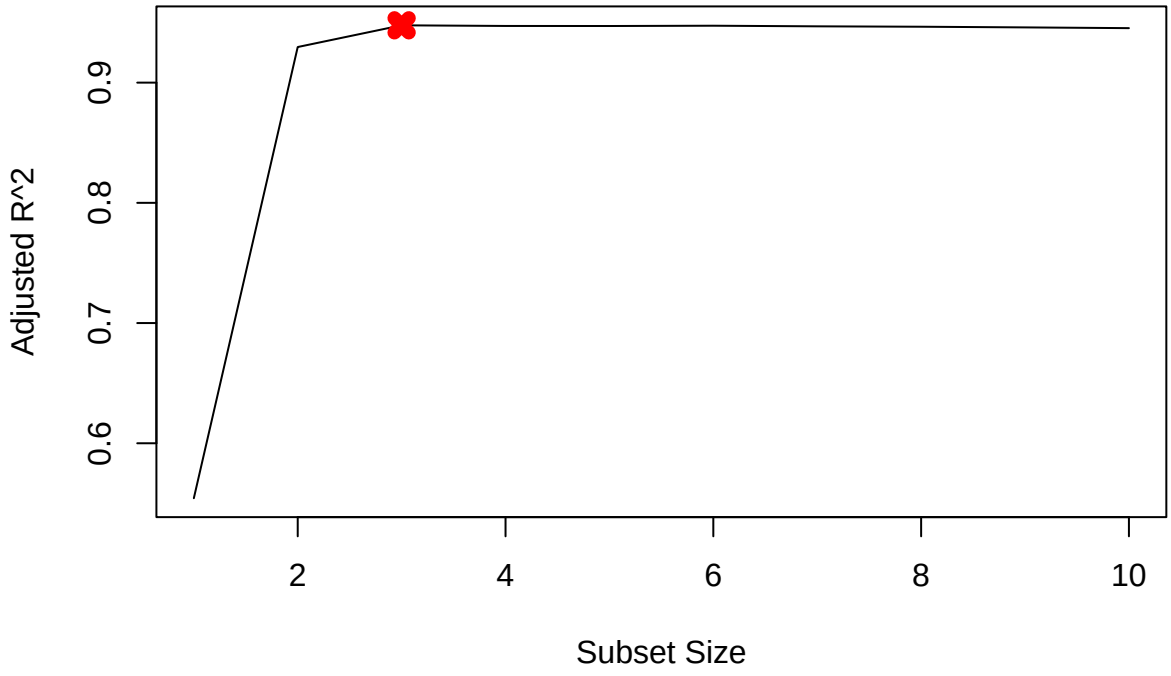
```
plot(model_full_summary$bic, xlab = "Subset Size", ylab = "BIC", pch = 20, type = "l")
points(3, model_full_summary$bic[3], pch = 4, col = "red", lwd = 7)
```




```
# Model size for best adjusted R^2
which.max(model_full_summary$adjr2)
```

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

```
plot(model_full_summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R^2", pch = 20, type = "l
points(3, model_full_summary$adjr2[3], pch = 4, col = "red", lwd = 7)
```




```
coef(model_full, 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             3.1324697             1.9125864            -3.1063732
## poly(x, 10, raw = T)3
##             0.3323054
```

(d) Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)?

```
# Forward stepwise
model_fwd = regsubsets(y~poly(x, 10, raw = T), data = df_full, nvmax = 10, method = "forward")
model_fwd_summary = summary(model_fwd)

which.min(model_fwd_summary$cp)
```

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

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

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

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

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

```r
# Backward stepwise
model_bwd = regsubsets(y~poly(x, 10, raw = T), data = df_full, nvmax = 10, method = "backward")
model_bwd_summary = summary(model_bwd)

which.min(model_bwd_summary$cp)
```

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

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

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

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

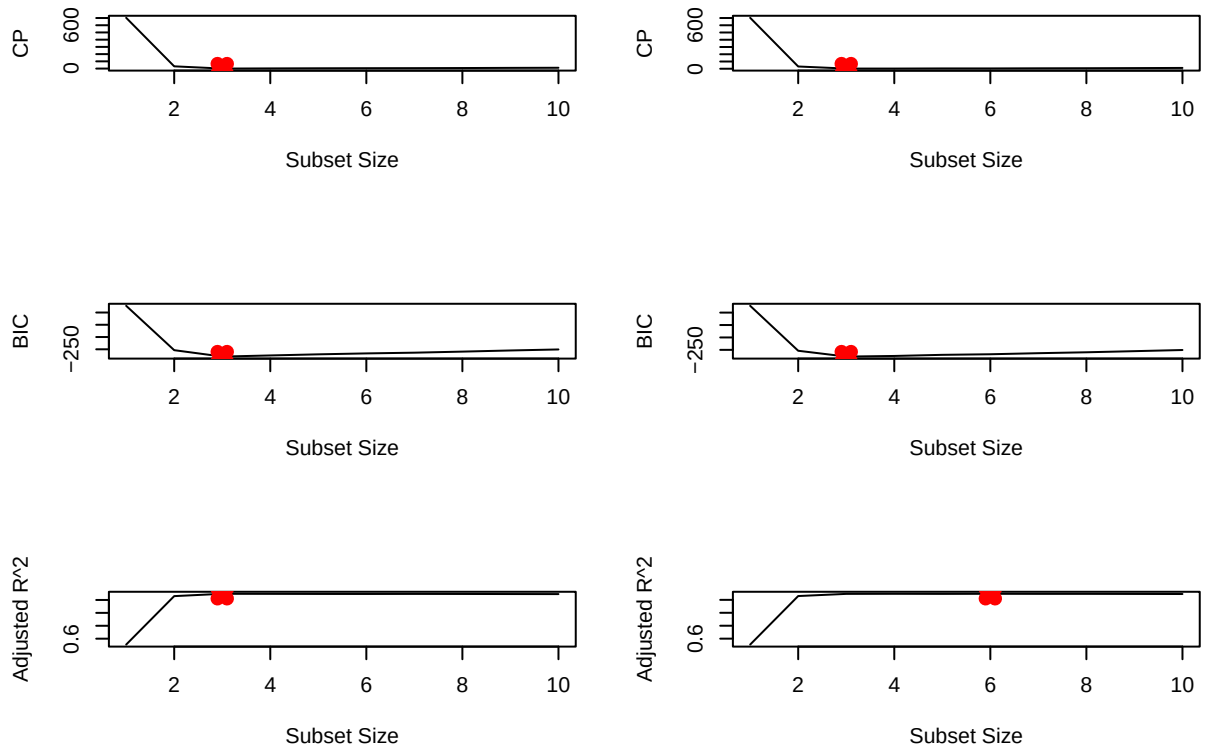
```
## [1] 6
```

```r
# Plot the statistics
par(mfrow = c(3, 2))
plot(model_fwd_summary$cp, xlab = "Subset Size", ylab = "CP", pch = 20, type = "l")
points(3, model_fwd_summary$cp[3], pch = 4, col = "red", lwd = 7)
plot(model_bwd_summary$cp, xlab = "Subset Size", ylab = "CP", pch = 20, type = "l")
points(3, model_bwd_summary$cp[3], pch = 4, col = "red", lwd = 7)

plot(model_fwd_summary$bic, xlab = "Subset Size", ylab = "BIC", pch = 20, type = "l")
points(3, model_fwd_summary$bic[3], pch = 4, col = "red", lwd = 7)
plot(model_bwd_summary$bic, xlab = "Subset Size", ylab = "BIC", pch = 20, type = "l")
points(3, model_bwd_summary$bic[3], pch = 4, col = "red", lwd = 7)

plot(model_fwd_summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R^2", pch = 20, type = "l"
```

```
points(3, model_fwd_summary$adjr2[3], pch = 4, col = "red", lwd = 7)
plot(model_bwd_summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R^2", pch = 20, type = "l"
points(6, model_bwd_summary$adjr2[6], pch = 4, col = "red", lwd = 7)
```




The results of CP and BIC from forward step and backward step are the same as the results in (c) except backward stepwise with adjusted R2. The coefficients are below.

```
coef(model_fwd, 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             3.1324697             1.9125864            -3.1063732
## poly(x, 10, raw = T)3
##             0.3323054
```

```
coef(model_bwd, 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            3.18525533            2.30024572           -3.12687927
```

```
## poly(x, 10, raw = T)5
##            0.04732764
```

```
coef(model_bwd, 6)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            3.00757607            2.09964860           -2.49994084
## poly(x, 10, raw = T)4 poly(x, 10, raw = T)5 poly(x, 10, raw = T)6
##           -0.38207642            0.16791365            0.05198902
## poly(x, 10, raw = T)7
##           -0.02138017
```

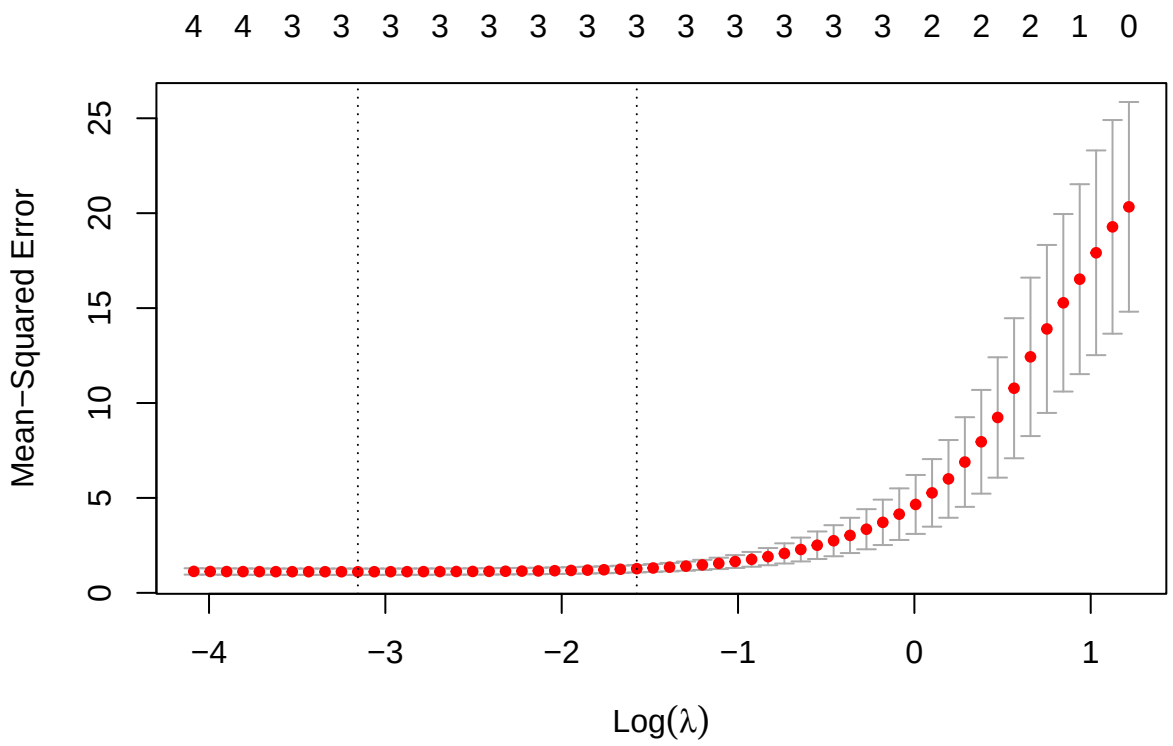
The forward stepwise picks $X^3$. Backward stepwise with 3 variables picks $X^5$ which with 6 variables picks $X^7$ and $X^6$. All coefficients above are close to $\beta$ value.

(e) Now fit a lasso model to the simulated data, again using $X, X^2, ..., X^{10}$ as predictors. Use cross-validation to select the optimal value of $\lambda$. Create plots of the cross-validation error as a function of $lambda$. Report the resulting coefficient estimates, and discuss the results obtained.

```
#library(glmnet)
xmat = model.matrix(y~poly(x,10, raw = T), data = df_full)[,-1]
lasso.mod = cv.glmnet(xmat, Y, alpha = 1)
best.lambda = lasso.mod$lambda.min
best.lambda
```

```
## [1] 0.04260095
```

```
plot(lasso.mod)
```




```
# Fit the model on the whole dataset by using the best lambda
best.model = glmnet(xmat, Y, alpha = 1)
predict(best.model, s = best.lambda, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)             3.0853710
## poly(x, 10, raw = T)1   1.8980459
## poly(x, 10, raw = T)2  -3.0608806
## poly(x, 10, raw = T)3   0.3180291
## 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   .
```

Lasso only picks $X^1$, $X^2$ and $X^3$ and passed $X^6$ and $X^7$.

(f) Now generate a response vector $Y$ according to the model

$$Y = \beta_0 + \beta_7 X^7 + \varepsilon$$

and perform best subset selection and the lasso. Discuss the results obtained.

```
beta_7 = 7
Y_new = beta_0 + beta_7*X^7 + epsilon
```

```
df_new = data.frame(y = Y_new, x = X)
mod_full = regsubsets(y~poly(x,10, raw = T), data = df_new, nvmax = 10)
mod.summary = summary(mod_full)

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

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

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

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

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

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

```
coef(mod_full, 1)
```

```
##           (Intercept) poly(x, 10, raw = T)7
##              3.042105              6.999908
```

```
coef(mod_full, 2)
```

```
##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##             3.1471879            -0.1074417             7.0004274
```

From the results, CP and BIC picks the most accurate 1 variable with matching coefficients. And adjusted R^2 picks 2 variables, adding an additional variable $X^2$.

```
xmat = model.matrix(y~poly(x, 10 , raw = T), data = df_new)[,-1]
mod.lasso = cv.glmnet(xmat, Y_new, alpha = 1)
best.lambda = mod.lasso$lambda.min
best.lambda
```

```
## [1] 20.34881
```

```
best.model = glmnet(xmat, Y_new, alpha = 1)
predict(best.model, s = best.lambda, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)           4.952989
## poly(x, 10, raw = T)1  .
## poly(x, 10, raw = T)2  .
## 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  6.795857
## poly(x, 10, raw = T)8  .
## poly(x, 10, raw = T)9  .
## poly(x, 10, raw = T)10 .
```

Lasso also picks the best 1 variable model but the intercept is larger. (4.95 > 3.04)

## Question 10 [20 marks]

We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

(a) Generate a data set with $p = 20$ features, $n = 1000$ observations, and an associated quantitative response vector generated according to the model

$$Y = X\beta + \varepsilon$$

where $\beta$ has some elements that are exactly equal to zero.

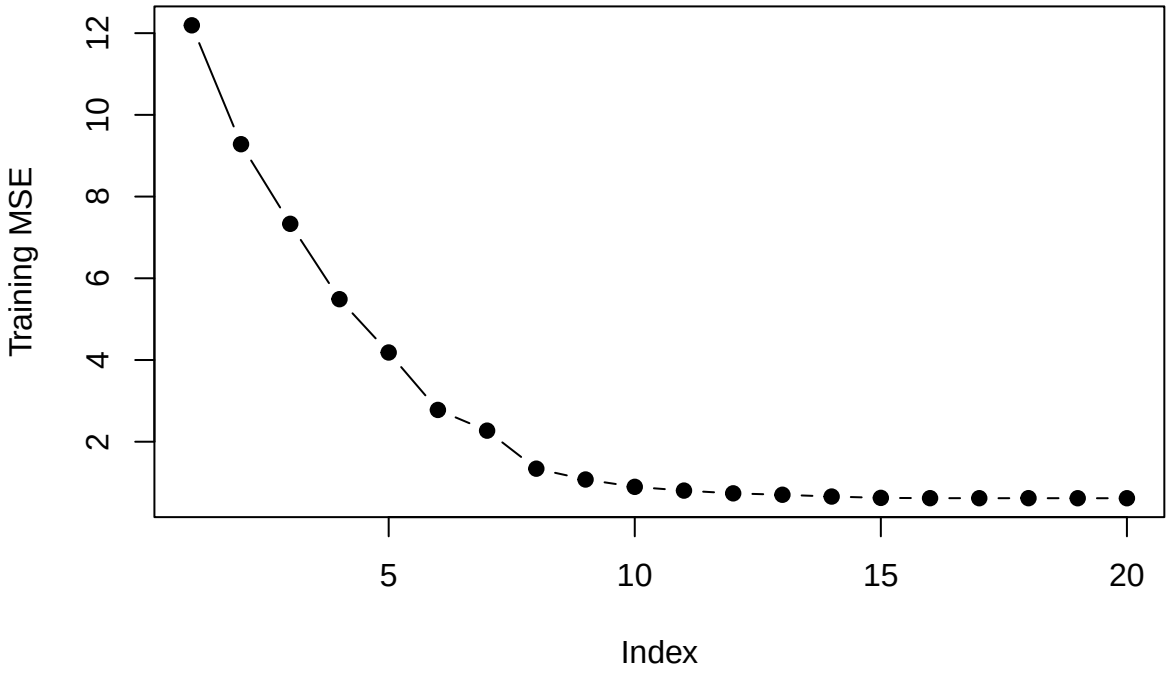
```
p = 20
n = 1000
x = matrix(rnorm(n*p),n,p)
B = rnorm(p)
B[2] = 0
B[4] = 0
B[10] = 0
B[12] = 0
B[18] = 0
epsilon = rnorm(p)
y = x %*% B +epsilon
```

(b) Split your data set into a training set containing 100 observations and a test set containing 900 observations.

```
train = sample(seq(1000), 100, replace = FALSE)
y.train = y[train, ]
y.test = y[-train, ]
x.train = x[train, ]
x.test = x[-train, ]
```

(c) Perform best subset selection on the training set, and plot the traning set MSE associated with the best model of each size.

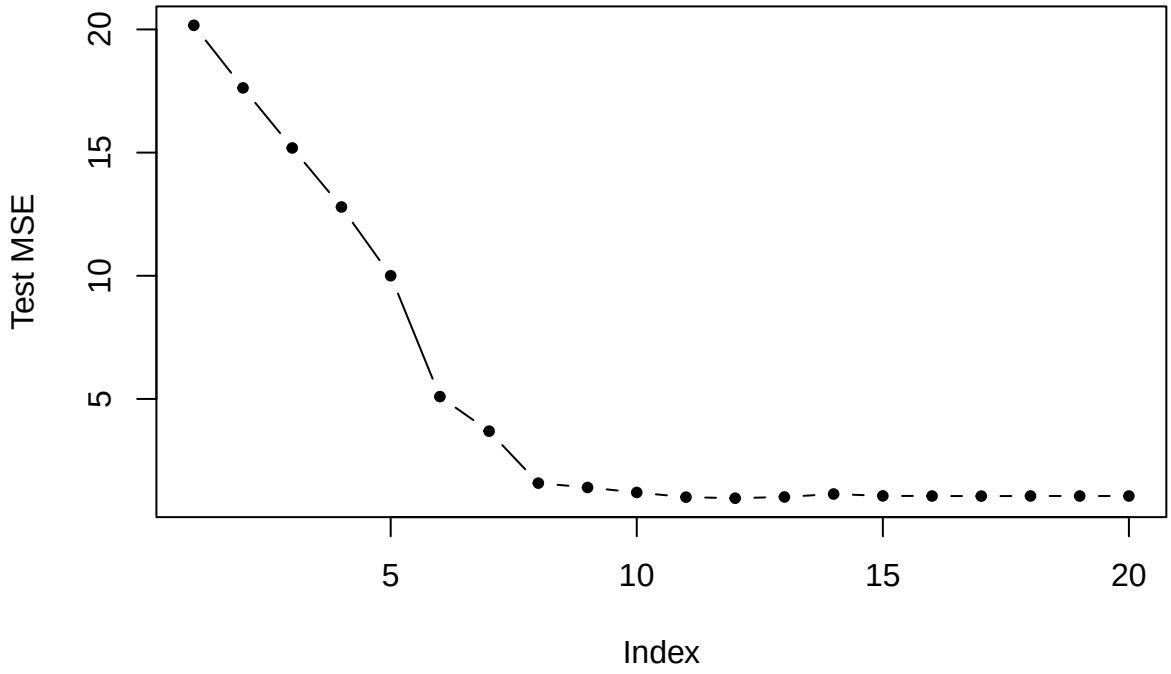
```
df2 = data.frame(x = x.train, y = y.train)
regfit.full = regsubsets(y~., data = df2, nvmax = p)
val.errors = rep(NA, p)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
for (i in 1:p){
    coefi = coef(regfit.full, id = i)
    pred = as.matrix(x.train[, x_cols %in% names(coefi)]) %*% coefi[names(coefi) %in%
        x_cols]
    val.errors[i] = mean((y.train - pred)^2)
}
plot(val.errors, ylab = "Training MSE", pch = 19, type = "b")
```




(d) Plot the test set MSE associated with the best model of each size.

```
val.errors = rep(NA, p)

for(i in 1:p) {
  coefi = coef(regfit.full, id = i)
  pred = as.matrix(x.test[, x_cols %in% names(coefi)]) %*% coefi[names(coefi) %in%
    x_cols]
  val.errors[i] = mean((y.test - pred)^2)
}
plot(val.errors, ylab = "Test MSE", pch = 20, type = "b")
```




(e) For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.

```
which.min(val.errors)
```

```
## [1] 12
```

Model with 13 parameter has the smallest MSE.

(f) How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values.

```
coef(regfit.full, 13)
```

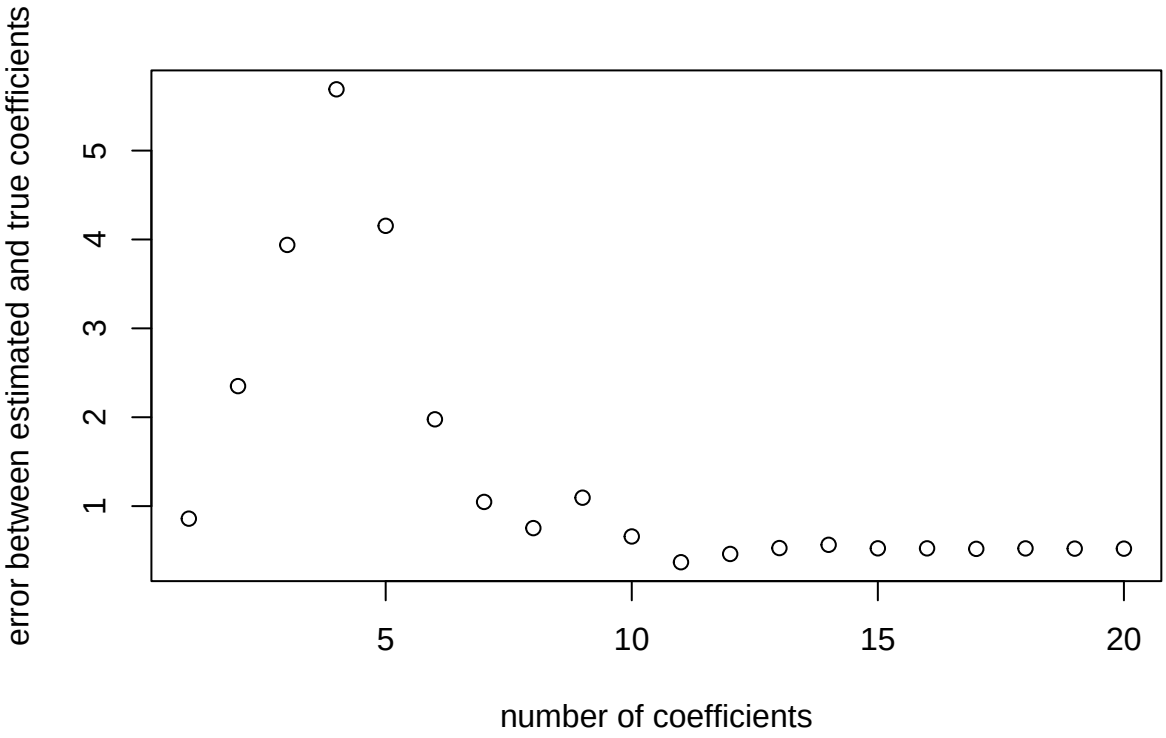
```
## (Intercept)         x.1         x.5         x.7         x.8         x.9
##   0.2178320  -1.6338588   2.7387282   1.1179478   1.0441334   0.4395431
```

```
##        x.12        x.13        x.14        x.15        x.16        x.17
##   0.2584232   0.2949516  -0.3705702  -1.5982596  -1.8066350   2.2826727
##        x.19        x.20
##  -0.4650055   1.4741548
```

This best model caught all coefficients.

(g) Create a plot displaying $\sqrt{\Sigma_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ for a range of values of $r$, where $\hat{\beta}_j^r$ is the $j$th coefficient estimate for the best model containing $r$ coefficients. Comment on what you observe. How does this compare to the test MSE plot from (d)?

```
val.errors = rep(NA, p)
a = rep(NA, p)
b = rep(NA, p)
for (i in 1:p) {
    coefi = coef(regfit.full, id = i)
    a[i] = length(coefi) - 1
    b[i] = sqrt(sum((B[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_cols])^2) +
        sum(B[!(x_cols %in% names(coefi))])^2)
}
plot(x = a, y = b, xlab = "number of coefficients", ylab = "error between estimated and true co
```


error between estimated and true coefficients
1
2
3
4
5
5
10
15
20
number of coefficients