ANLY512_HW5

Hongyang Zheng

2019/3/18

```
library(leaps)
library(readxl)
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

Problem # 1

a)

Best subset selection has the smallest training RSS, because the models from other two methods depend on the predictors they choose previously as they add/subtract predictors to the kth model.

b)

It depends. Best subset selection may have the smallest test RSS, because it considers more possible models than other two methods and choose the best one from them. However, the other methods might also have the smallest test RSS, if they are lucky enough and pick a model that fits the test data better.

c)

i. TRUE Because the models from forward stepwise selection is nested. The k+1-variable model must contain the variables in the k-variable model.

ii. TRUE Because the models from backward stepwise selection is nested. The k+1-variable model must contain the variables in the k-variable model.

iii. FALSE Usually there is no relationship between the model from forward stepwise selection and the model from backward stepwise selection.

iv. FALSE Usually there is no relationship between the model from forward stepwise selection and the model from backward stepwise selection.

v. FALSE The models from best subset selection are not necessarily nested.

Problem # 8abcd

a)

```r
set.seed(1234)

# Generate X and noise
X = rnorm(100)
eps = rnorm(100)
```

b)

```r
# Generate Y
beta0=2
beta1=1
beta2=-0.5
beta3=0.3

Y=beta0+beta1*X+beta2*(X^2)+beta3*(X^3)+eps
```

c)

```r
# Generate dataframe
data1 = data.frame(y = Y, x = X)

model_10 = regsubsets(y ~ poly(x, 10, raw = T), data = data1, nvmax = 10)
summary.10 = summary(model_10)

# Find the model size for best cp, BIC and adjr2
which.min(summary.10$cp)
```

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

```r
which.min(summary.10$bic)
```

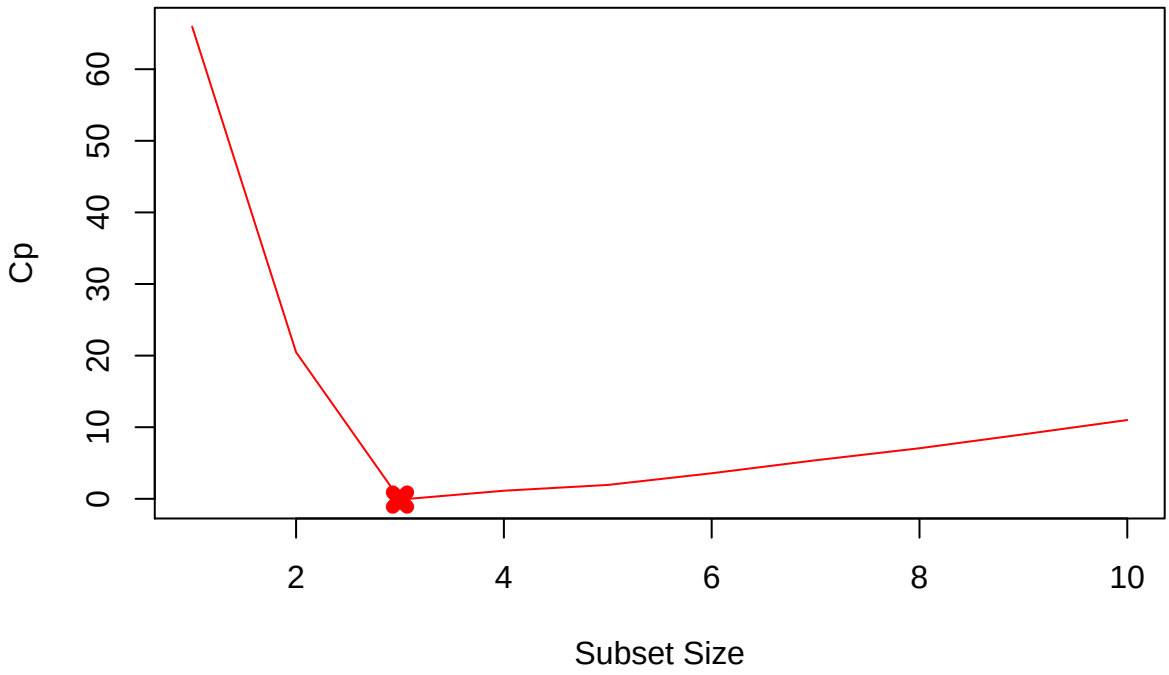
```
## [1] 3
```

```r
which.max(summary.10$adjr2)
```

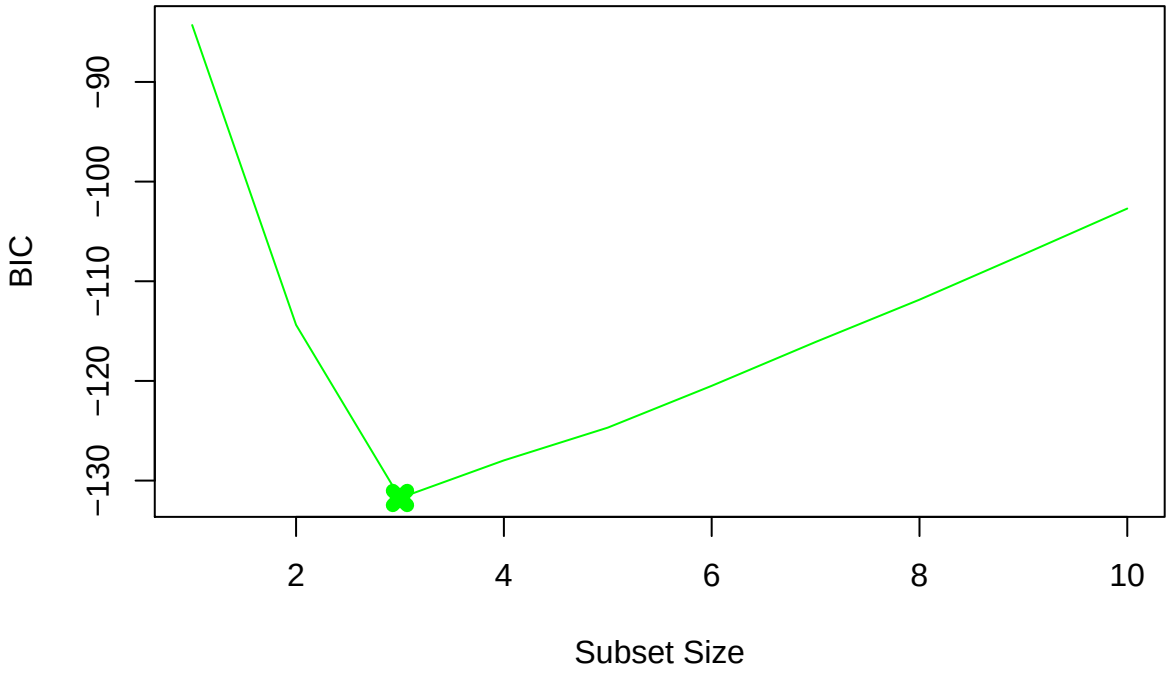
```
## [1] 5
```

The best model indicated by Cp and BIC contains 3 predictors , while the best model indicated by adjusted R-square contains 3 predictors.

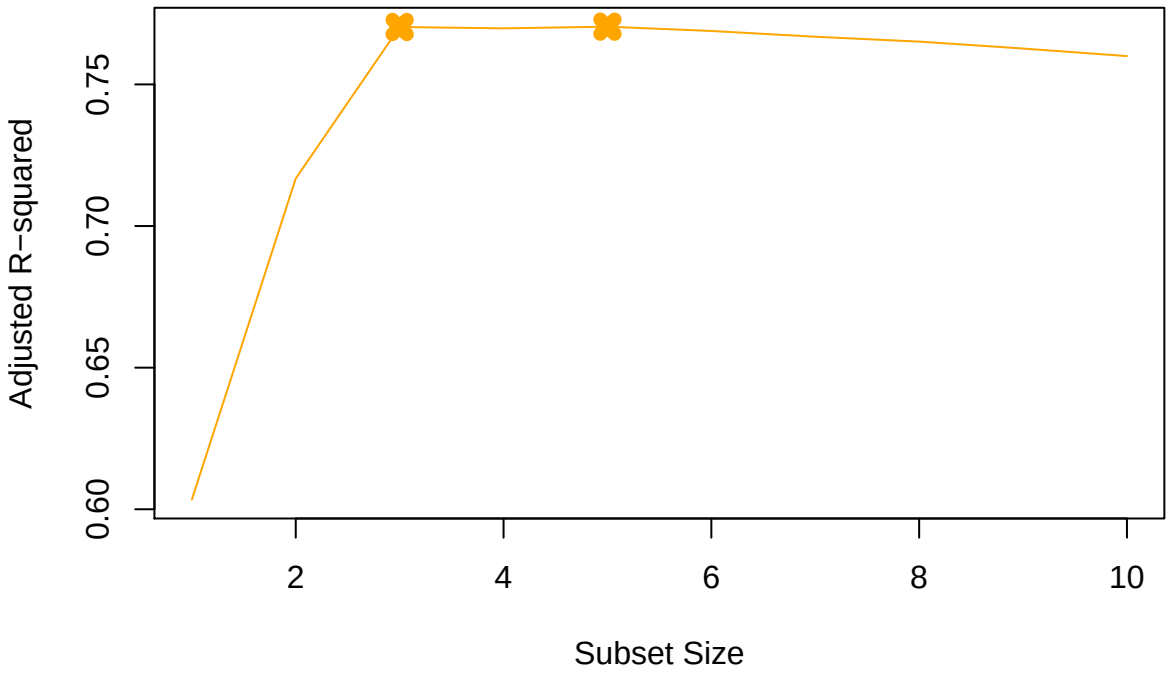
```r
# Draw plot cp
plot(summary.10$cp, xlab = "Subset Size", ylab = 'Cp', type = "l", col='red')
points(3, summary.10$cp[3], pch = 4, col = "red", lwd = 7)
```




```
# Draw plot BIC
plot(summary.10$bic, xlab = "Subset Size", ylab = 'BIC', type = "l", col='green')
points(3, summary.10$bic[3], pch = 4, col = "green", lwd = 7)
```




```
# Draw plot adjr2
plot(summary.10$adjr2, xlab = "Subset Size", ylab = 'Adjusted R-squared', type = "l", col='orange')
points(5, summary.10$adjr2[5], pch = 4, col = "orange", lwd = 7)
points(3, summary.10$adjr2[3], pch = 4, col = "orange", lwd = 7)
```



From the above graph, we can see that Cp and BIC reach the minimum when subset size = 3, and adjusted R-square reaches the maximum when subset size = 5. However, we can see from the adjr2 plot, the difference between 3 and 5 is very small. So I think the model containing 3 variables is the best model. The coefficient of the model is showed below:

```
# Coefficient
coefficients(model_10, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             2.1324697             0.9125864            -0.6063732
## poly(x, 10, raw = T)3
##             0.3323054
```

The best model is:

$$Y1 = 2.13 + 0.91X - 0.61X^2 + 0.33X^3$$

The coefficient is close to my choice of coefficient.

d)

```
# Forward stepwise selection
model_10.fwd = regsubsets(y ~ poly(x, 10, raw = T), data = data1, nvmax = 10,  method = "forward")
summary.10.fwd = summary(model_10.fwd)

# Find the model size for best cp, BIC and adjr2
which.min(summary.10.fwd$cp)
```

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

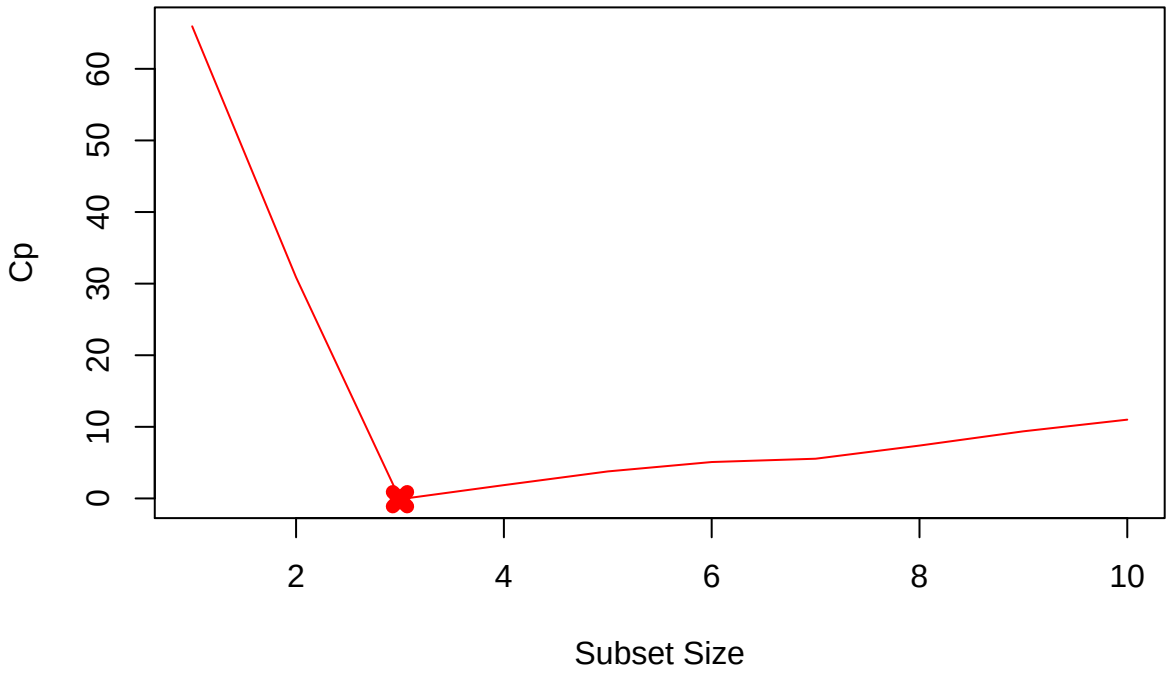
```
which.min(summary.10.fwd$bic)
```

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

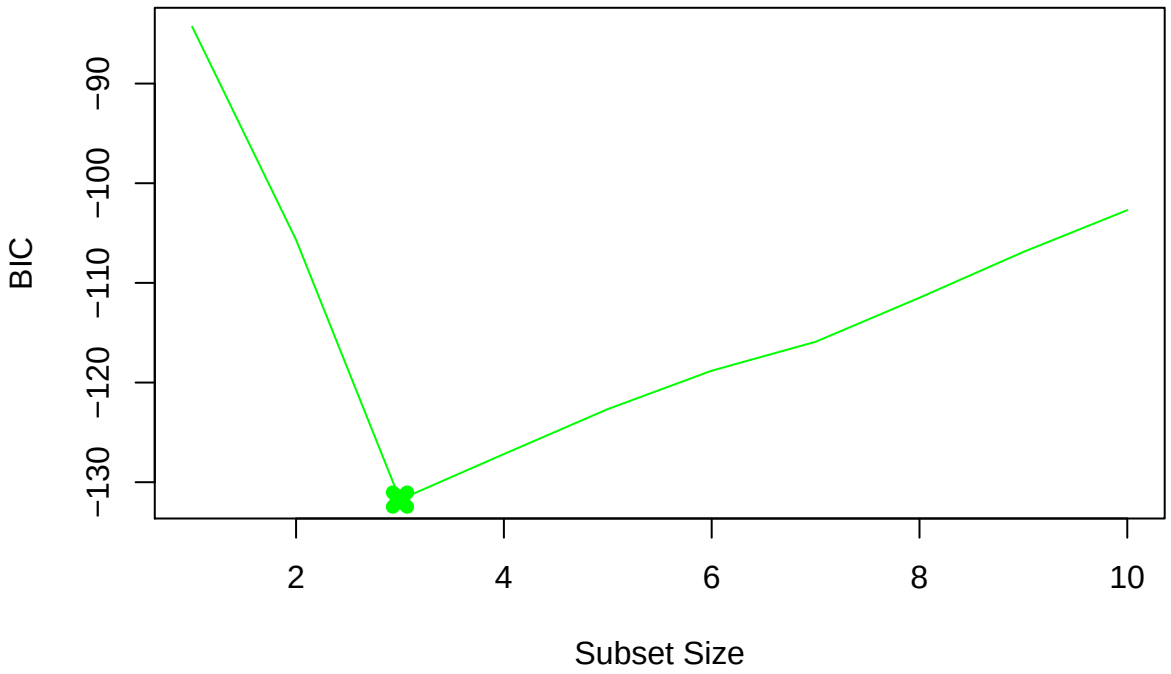
```
which.max(summary.10.fwd$adjr2)
```

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

```
# Draw plot cp
plot(summary.10.fwd$cp, xlab = "Subset Size", ylab = 'Cp', type = "l", col='red')
points(3, summary.10.fwd$cp[3], pch = 4, col = "red", lwd = 7)
```

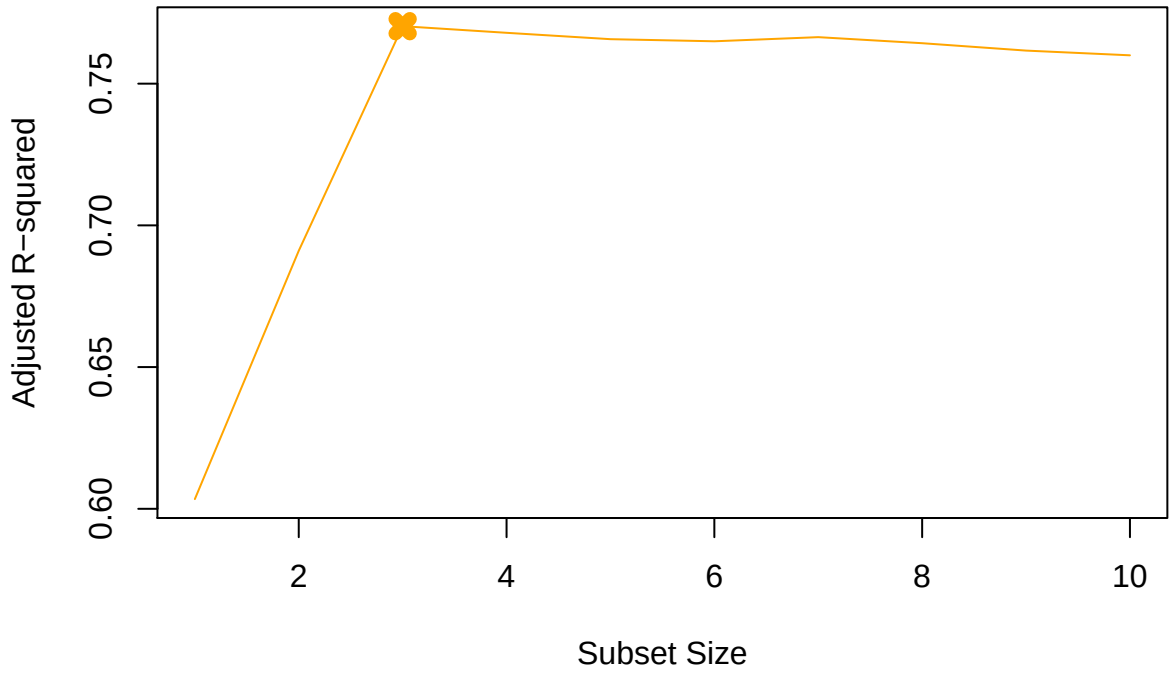



```
# Draw plot BIC
plot(summary.10.fwd$bic, xlab = "Subset Size", ylab = 'BIC', type = "l", col='green')
points(3, summary.10.fwd$bic[3], pch = 4, col = "green", lwd = 7)
```




```
# Draw plot adjr2
plot(summary.10.fwd$adjr2, xlab = "Subset Size", ylab = 'Adjusted R-squared', type = "l", col='orange')
```

```
points(3, summary.10.fwd$adjr2[3], pch = 4, col = "orange", lwd = 7)
```



From the above results and graph, we can see that Cp and BIC reach the minimum when subset size = 3, and adjusted R-square reaches the maximum when subset size = 3 too. The best model contains 3 variables and the coefficient is showed below.

```
coefficients(model_10.fwd, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             2.1324697             0.9125864            -0.6063732
## poly(x, 10, raw = T)3
##             0.3323054
```

The best model is:

$$Y2 = 2.13 + 0.91X - 0.61X^2 + 0.33X^3$$

The model is same as the model from best subset selection.

```
# Backward stepwise selection
model_10.bwd = regsubsets(y ~ poly(x, 10, raw = T), data = data1, nvmax = 10,  method = "backward")
summary.10.bwd = summary(model_10.bwd)

# Find the model size for best cp, BIC and adjr2
which.min(summary.10.bwd$cp)
```

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

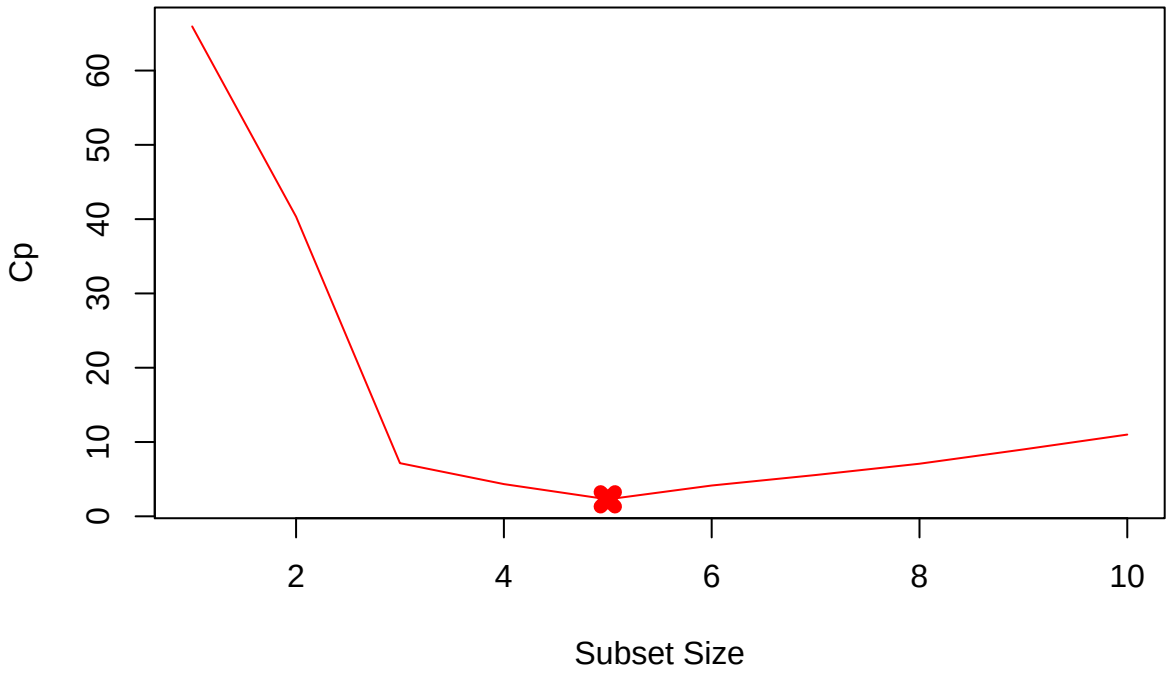
```
which.min(summary.10.bwd$bic)
```

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

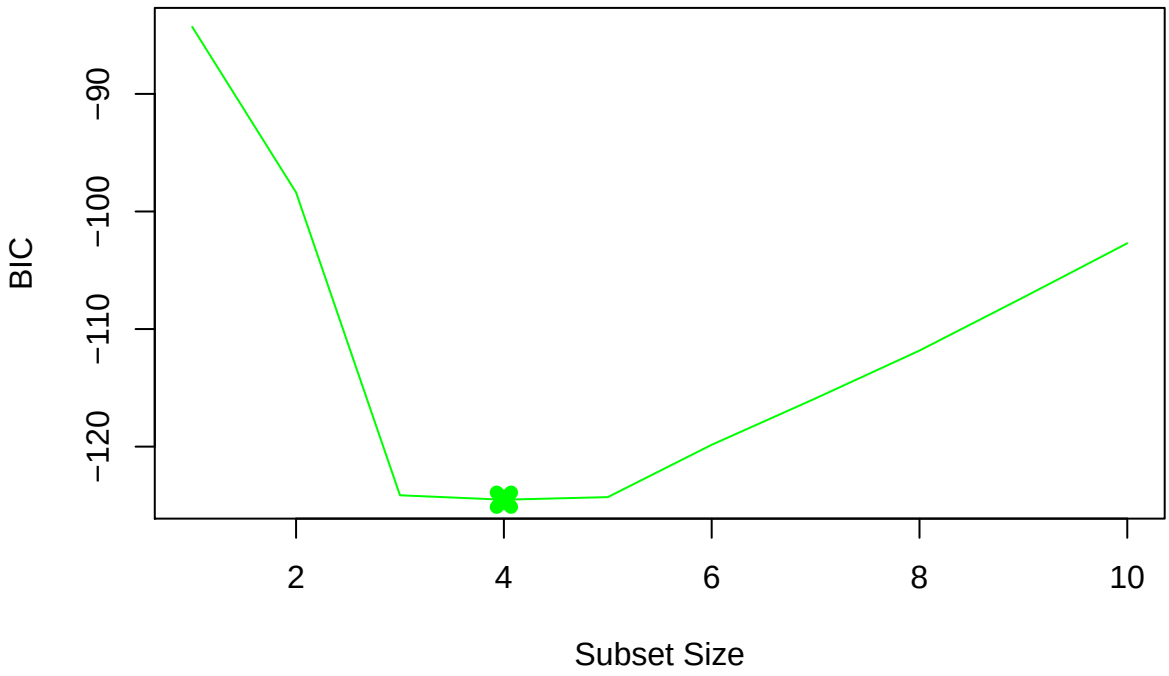
```
which.max(summary.10.bwd$adjr2)
```

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

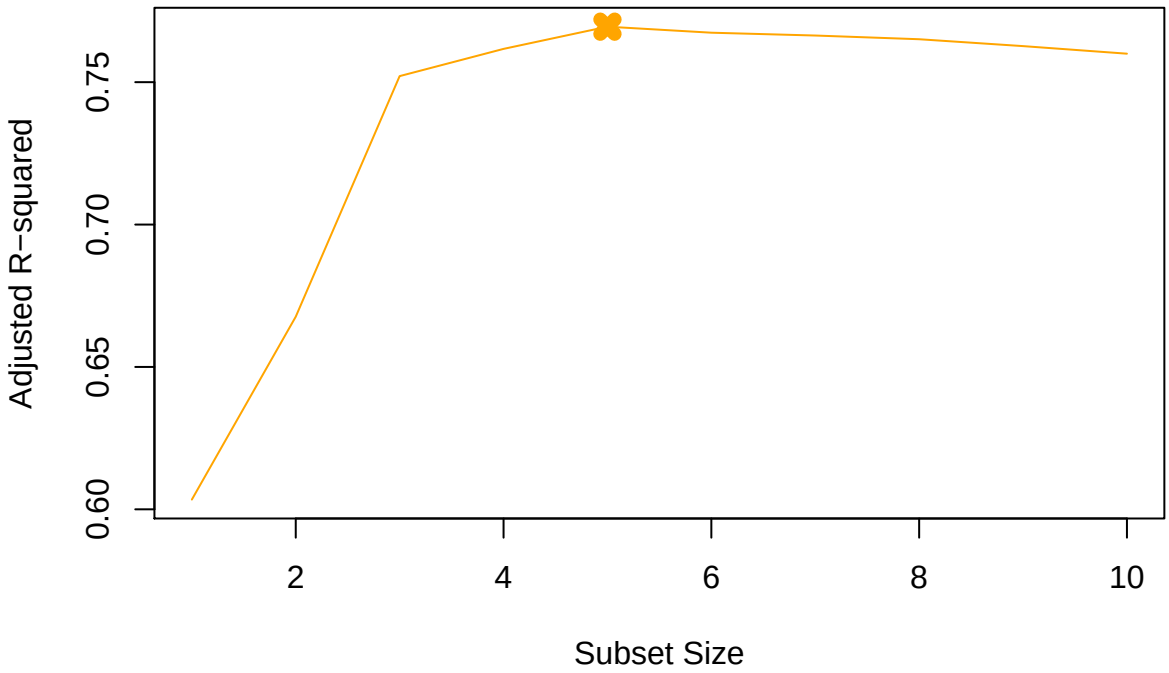
```r
# Draw plot cp
plot(summary.10.bwd$cp, xlab = "Subset Size", ylab = 'Cp', type = "l", col='red')
points(5, summary.10.bwd$cp[5], pch = 4, col = "red", lwd = 7)
```




```r
# Draw plot BIC
plot(summary.10.bwd$bic, xlab = "Subset Size", ylab = 'BIC', type = "l", col='green')
points(4, summary.10.bwd$bic[4], pch = 4, col = "green", lwd = 7)
```




```r
# Draw plot adjr2
plot(summary.10.bwd$adjr2, xlab = "Subset Size", ylab = 'Adjusted R-squared', type = "l", col='orange')
points(5, summary.10.bwd$adjr2[5], pch = 4, col = "orange", lwd = 7)
```



From the above graph, we can see that Cp reaches minimum when subset size = 5, and BIC reach the minimum when subset size = 4, and adjusted R-square reaches the maximum when subset size = 5. However, we can see from the BIC plot, the difference between 4 and 5 is very small. So I think the model containing 5 variables is the best model. The coefficient of the model is showed below:

```
coefficients(model_10.bwd, id = 5)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)4
##            2.00759276           1.09964771          -0.38204601
## poly(x, 10, raw = T)5 poly(x, 10, raw = T)6 poly(x, 10, raw = T)7
##            0.16791149           0.05198532          -0.02137972
```

The best model is:

$$Y3 = 2 + 1.1X - 0.38X^4 + 0.17X^5 + 0.05X^6 - 0.02X^7$$

The model is very different from the model from best subset selection. The new model contains more high-degree term.

Problem # 10

a)

```
# Generate data set, noise and beta
set.seed(134)
x = matrix(rnorm(1000 * 20), 1000, 20)
B = rnorm(20)
eps = rnorm(20)

# Let some beta = 0
B[5] = 0
B[9] = 0
B[10] = 0
B[19] = 0
```

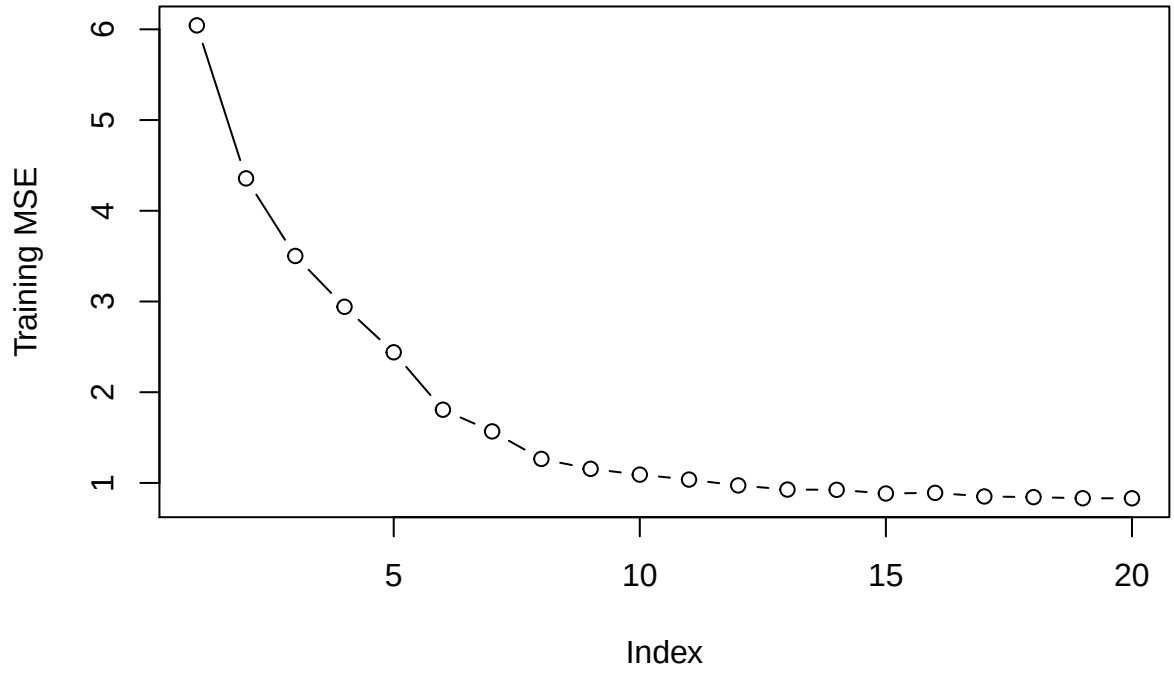
```r
# Generate y
y = x %*% B + eps
```

b)

```r
# Split the data
train = sample(1:1000, 100, replace = FALSE)
y.train = y[train, ]
y.test = y[-train, ]
x.train = x[train, ]
x.test = x[-train, ]

data.train=data.frame(x = x.train, y = y.train)
data.test=data.frame(x = x.test, y = y.test)
```

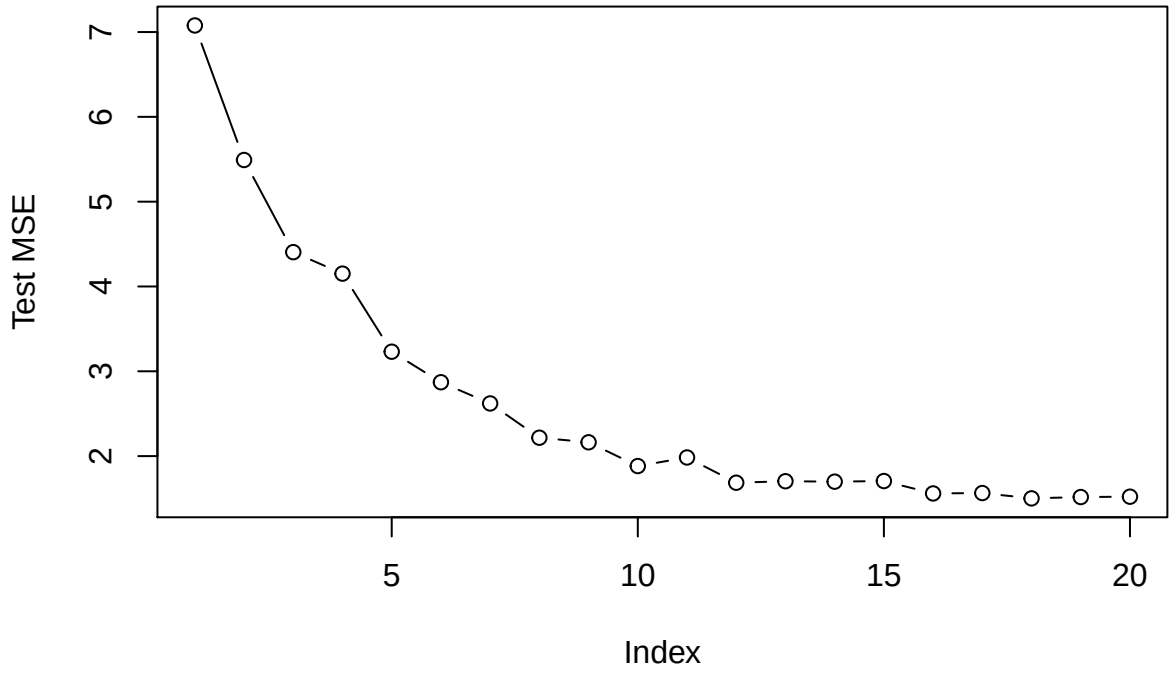
c)

```r
# Best subset selection
model.full = regsubsets(y ~ ., data = data.train, nvmax = 20)

# Training error for each model
errors1 = rep(NA, 20)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
for (i in 1:20) {
    coefi = coef(model.full, id = i)
    pred = as.matrix(x.train[, x_cols %in% names(coefi)]) %*% coefi[names(coefi) %in% x_cols]
    errors1[i] = mean((y.train - pred)^2)
}

# Plot training error
plot(errors1, ylab = "Training MSE", type = "b")
```



d)

```
# Test error for each model
errors2 = rep(NA, 20)
for (i in 1:20) {
    coefi = coef(model.full, id = i)
    pred = as.matrix(x.test[, x_cols %in% names(coefi)]) %*% coefi[names(coefi) %in% x_cols]
    errors2[i] = mean((y.test - pred)^2)
}

# Plot test error
plot(errors2, ylab = "Test MSE", type = "b")
```



e)

```
# Find minimum test error
which.min(errors2)
```

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

When the model contains 18 variables, it has the minimum test error.

f)

```
# Find coefficient
coef(model.full, id = 18)
```

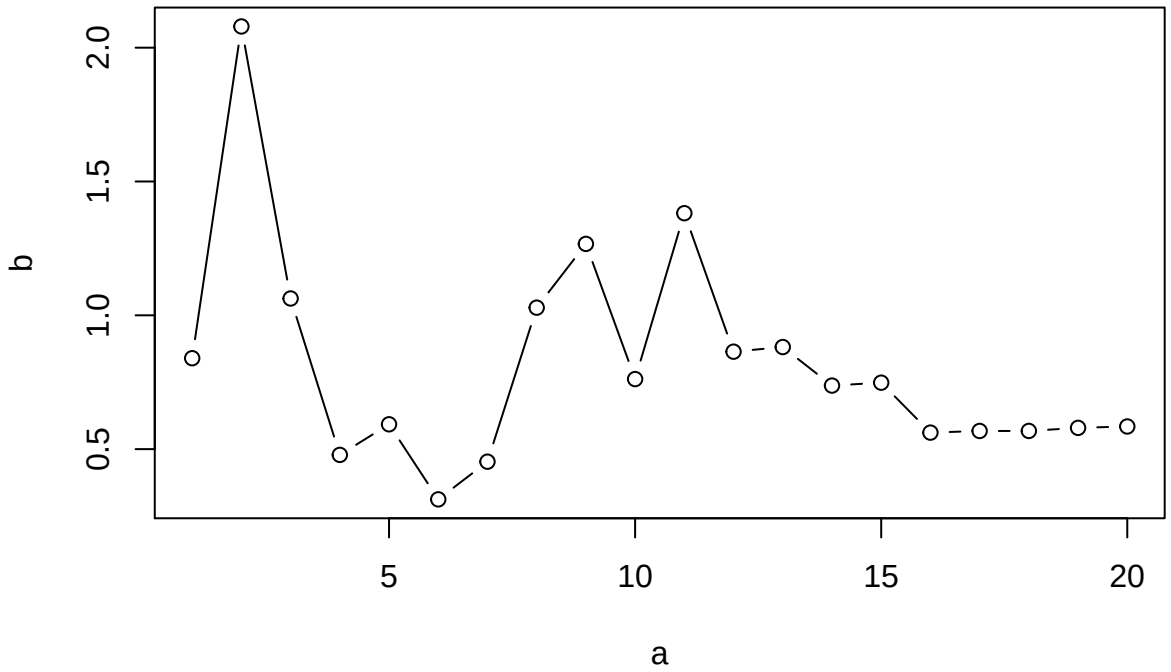
```
## (Intercept)         x.1         x.2         x.3         x.4         x.5
##  -0.2465488   0.2328804  -0.4564635  -0.6889561  -0.3459996   0.1540830
##         x.7         x.8         x.9        x.10        x.11        x.12
##   0.1828332   1.2011950   0.1941792  -0.1824485  -0.4892839   0.2938043
##        x.13        x.14        x.15        x.16        x.17        x.18
##   0.1473360  -0.2485434   0.7545060   0.3940041   1.7948894  -0.9464592
##        x.20
```

```
##  -0.9072423
```

For the true model, the beta for x.5, x.9, x.10, x.19 are 0. For this best model, it does not include x.19, but include x.5, x.9 and x.10 which should be 0. The best model also excludes another predictor x.6 which should not be 0.

g)

```
a = rep(NA, 20)
b = rep(NA, 20)

for (i in 1:20) {
    coefi = coef(model.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
}

# Make a plot
plot(x = a, y = b, type = "b")
```




```
# Find minimum
which.min(b)
```

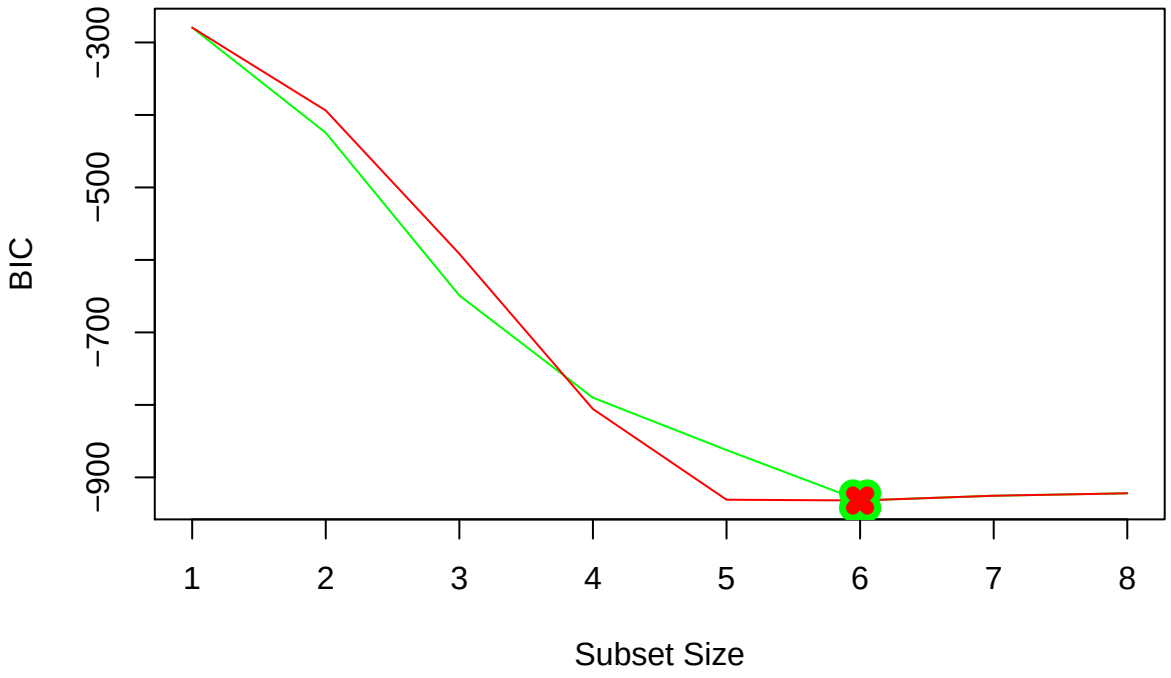
```
## [1] 6
```

Model with 5 predictors minimizes the error between the estimated and true coefficients. Test error is minimized with 18 parameter model. A better fit of true coefficients doesn't mean the model will have a lower test MSE.

Problem Xtra # 44

```
# Load the data and change the names
Concrete <- read_excel("~/Desktop/HW/Concrete_Data.xls")
names(Concrete)=c('Cement', 'Slag', 'Fly', 'Water', 'Super', 'Coarse', 'Fine', 'Age', 'Strength')
```

```
# Forward stepwise selection
model.fwd = regsubsets(Strength ~ ., data = Concrete, nvmax = 8,  method = "forward")
summary.fwd = summary(model.fwd)

# Backward stepwise selection
model.bwd = regsubsets(Strength ~ ., data = Concrete, nvmax = 8,  method = "backward")
summary.bwd = summary(model.bwd)

# Draw plot BIC
plot(summary.fwd$bic, xlab = "Subset Size", ylab = 'BIC', type = "l", col='green')
lines(summary.bwd$bic, xlab = "Subset Size", ylab = 'BIC', type = "l", col='red')
a=which.min(summary.fwd$bic)
b=which.min(summary.bwd$bic)
points(a, summary.fwd$bic[a], pch = 4, col = "green", lwd = 14)
points(b, summary.bwd$bic[b], pch = 4, col = "red", lwd = 7)
```



Both reach minimum bic when subset size = 6. The change tendency is similar for forward stepwise selection and backward stepwise selection, except for backward, the bic decreases rapidily from 4 to 5 and then almost no change, while for forward, the bic decreases gradually from 4 to 6.

Problem Xtra # 43

a)

There are $X_1, X_2, X_3, X_1^2, X_2^2, X_3^2, X_1X_2, X_2X_3, X_1X_3$, total 9 variables. There are $\frac{9!}{3!6!} = \frac{9*8*7}{3*2*1} = 84$

b)

1. $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_7 X_1 X_2$
2. $Y = \beta_0 + \beta_2 X_2 + \beta_3 X_3 + \beta_8 X_2 X_3$
3. $Y = \beta_0 + \beta_1 X_1 + \beta_3 X_3 + \beta_9 X_1 X_3$
4. $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_4 X_1^2$
5. $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_5 X_2^2$
6. $Y = \beta_0 + \beta_1 X_1 + \beta_3 X_3 + \beta_4 X_1^2$
7. $Y = \beta_0 + \beta_1 X_1 + \beta_3 X_3 + \beta_6 X_3^2$
8. $Y = \beta_0 + \beta_2 X_2 + \beta_3 X_3 + \beta_5 X_2^2$
9. $Y = \beta_0 + \beta_2 X_2 + \beta_3 X_3 + \beta_6 X_3^2$

c)

When we done the selection for 1-variable model, we exclude the interactive term that does not contain the variable included in the 1-variable model; we also exclude the high-degree terms that are 2 or more degree higher than the variable included in the 1-variable model. Then we perform the forward stepwise selection for the 2-variable model. We add back all the variables, and then exclude the interactive term that does not contain the variable included in the 2-variable model; we also exclude the high-degree terms that are 2 or more degree higher than the variable included in the 2-variable model. Then we perform the forward stepwise selection for the 3-variable model and so on until we get the n-variable model. And the next step is the same as forward stepwise selection.

Problem Xtra # 46

```
set.seed(123)
# Load data
mnist=load('~/Desktop/other/data/mnist_all.RData')
```

```
# Generate dataframe
# Extract data for digit1 and digit3
y.nist = train$y
index <- (y.nist == 1 | y.nist == 3)
x.nist <- train$x[index,]
x.train <- as.data.frame(x.nist)

y.nist1 = test$y
index1 <- (y.nist1 == 1 | y.nist1 == 3)
x.nist1 <- test$x[index1,]
x.test <- as.data.frame(x.nist1)

# y=1 if it is digit1, y=0 if it is digit3
x.train$y <- 0
x.train$y[y.nist[index] == 1] <- 1
x.test$y <- 0
x.test$y[y.nist1[index1] == 1] <- 1
```

```r
# Calculate variance for each pixel for train data
variance.train=rep(0,784)
for (i in 1:784)
{
  variance.train[i]=var(x.train[,i])
}
variance.train=as.data.frame(variance.train)
train=x.train[,variance.train != 0]
train$y=x.train$y

# Calculate variance for each pixel for test data
variance.test=rep(0,784)
for (i in 1:784)
{
  variance.test[i]=var(x.test[,i])
}
variance.test=as.data.frame(variance.test)
test=x.test[,variance.test != 0]
test$y=x.test$y
```

a)

```r
# Calculate auc score for each pixel
auc.train=rep(0,624)
for (i in 1:624)
{
  model=glm(train$y~train[,i], data=train, family = 'binomial')
  pred.model=predict(model, data=train, type = 'response')
  auc.train[i]=auc(roc(train$y, pred.model), col=4)
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# Find the maximum
a=which.max(auc.train)
colnames(train[a])
```

```
## [1] "V490"
```

```r
# The best model is
model.best1=glm(y~V490, data=train, family = 'binomial')
summary(model.best1)
```

```
##
## Call:
## glm(formula = y ~ V490, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6575  -0.4663   0.2437   0.2519   2.1321
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.1642252  0.0397429  -54.46   <2e-16 ***
## V490         0.0222187  0.0003285   67.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 17816.8  on 12872  degrees of freedom
## Residual deviance:  7490.3  on 12871  degrees of freedom
## AIC: 7494.3
##
## Number of Fisher Scoring iterations: 5
```

```r
# The AUC score is
auc.train[a]
```

```
## [1] 0.9438168
```

b)

```r
# Calculate auc score for one more pixel
train2=train[,-a]
auc.train2=rep(0,623)
for (i in 1:623)
{
  model=glm(train2$y~train$V490+train2[,i], data=train2, family = 'binomial')
  pred.model=predict(model, type = 'response')
  auc.train2[i]=auc(roc(train2$y, pred.model), col=4)
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# Find the maximum
b=which.max(auc.train2)
colnames(train2[b])
```

```
## [1] "V495"
```

```r
# The best model is
model.best2=glm(y~V490+V495, data=train, family = 'binomial')
summary(model.best2)
```

```
##
## Call:
## glm(formula = y ~ V490 + V495, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8621  -0.0588   0.1832   0.1888   4.2552
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.0391967  0.0465356  -22.33   <2e-16 ***
## V490         0.0200710  0.0004085   49.13   <2e-16 ***
## V495        -0.0316761  0.0013822  -22.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 17816.8  on 12872  degrees of freedom
## Residual deviance:  5044.6  on 12870  degrees of freedom
```

```
## AIC: 5050.6
##
## Number of Fisher Scoring iterations: 8
```

```
# The AUC score is
auc.train2[b]
```

```
## [1] 0.974748
```

c)

```
# Test the first model
pred.model.test1=predict(model.best1, newdata=test, family = 'r')
auc.test1=auc(roc(test$y, pred.model.test1), col=4)
auc.test1
```

```
## Area under the curve: 0.9575
```

```
# Test the second model
pred.model.test2=predict(model.best2, newdata=test, family = 'r')
auc.test2=auc(roc(test$y, pred.model.test2), col=4)
auc.test2
```

```
## Area under the curve: 0.9824
```

From the result, we can see that the auc score for the first model is 0.96, which is less than the auc score from the second model 0.98. Therefore, the second model is really better than the first model.

d)

part a, we examine 624 models, and part b we examine 623 models. Therefore we examine in total 1247 models.

If I want to continue this process and make the best logistic model with 10 pixels, then I need to examine 624+623+622+621+620+619+618+617+616+615=6195 models in total.