Statistical Learning HW 5 - Ch 6

Christian Conroy

March 28, 2019

Book 8a-d (3 points)

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 of length n = 100.

```
set.seed(100)
x <- rnorm(100)
e <- .01*rnorm(100)
```

(b) Generate a response vector Y of length n = 100 according to the model Y = ??0 + ??1X + ??2X2 + ??3X3 + , where ??0, ??1, ??2, and ??3 are constants of your choice.

```
y <- -2 + 2*x + 3* I(x^2) + 4*I(x^3) + e
```

(c) Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors X, X2,. . . ,X10. What is the best model obtained according to Cp, BIC, and adjusted R2? 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 containing both X and Y.

```
# Calculate

df <- data.frame(x, y)

full_subsets <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9

full_subsets_summary = summary(full_subsets, matrix.logical = TRUE)

full_subsets_summary$cp
```

```
## [1] 2.82e+07 1.76e+06 3.82e+00 3.85e+00 5.31e+00 6.01e+00 6.52e+00 7.15e+00
```

```
full_subsets_summary$bic
```

```
## [1]  -280  -553 -1530 -1527 -1523 -1520 -1517 -1514
```

```
full_subsets_summary$adjr2
```

```
## [1] 0.944 0.996 1.000 1.000 1.000 1.000 1.000 1.000
```

```
summary(lm(y ~ x + I(x^2) + I(x^3)), data = df)
```

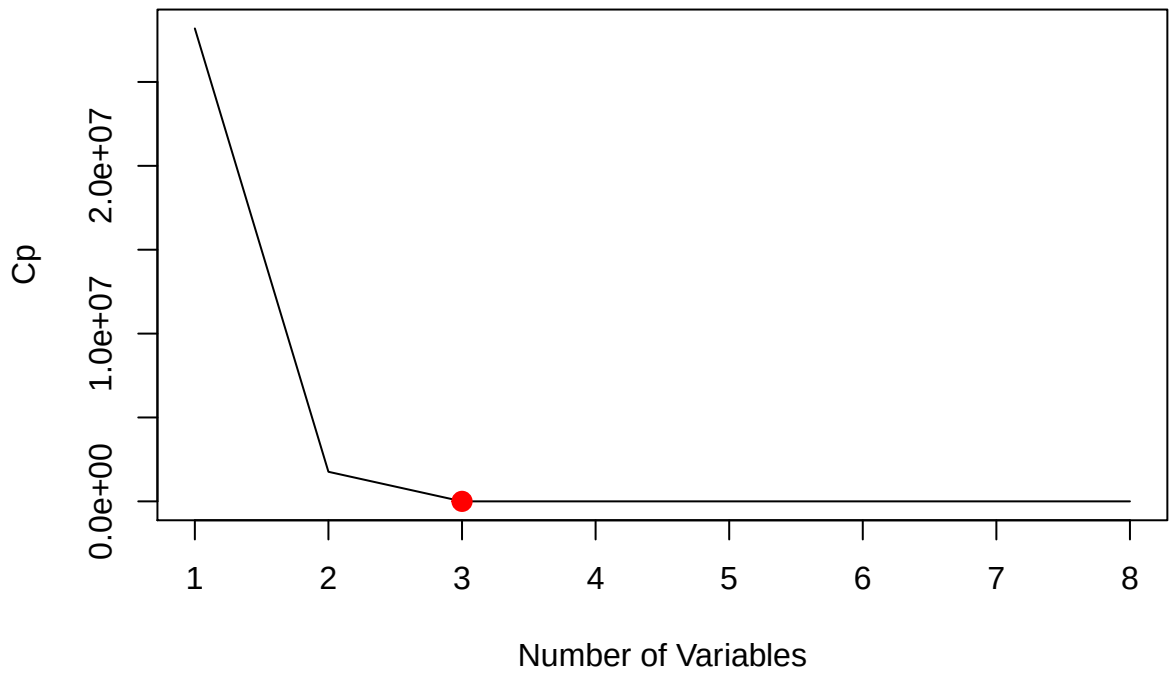
```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.02003 -0.00453 -0.00119  0.00510  0.01838
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.000604   0.000977   -2047   <2e-16 ***
## x            1.998067   0.001504    1329   <2e-16 ***
## I(x^2)       3.000655   0.000568    5284   <2e-16 ***
## I(x^3)       4.000249   0.000426    9400   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00792 on 96 degrees of freedom
## Multiple R-squared:     1,   Adjusted R-squared:     1
## F-statistic: 1.7e+08 on 3 and 96 DF,  p-value: <2e-16
```

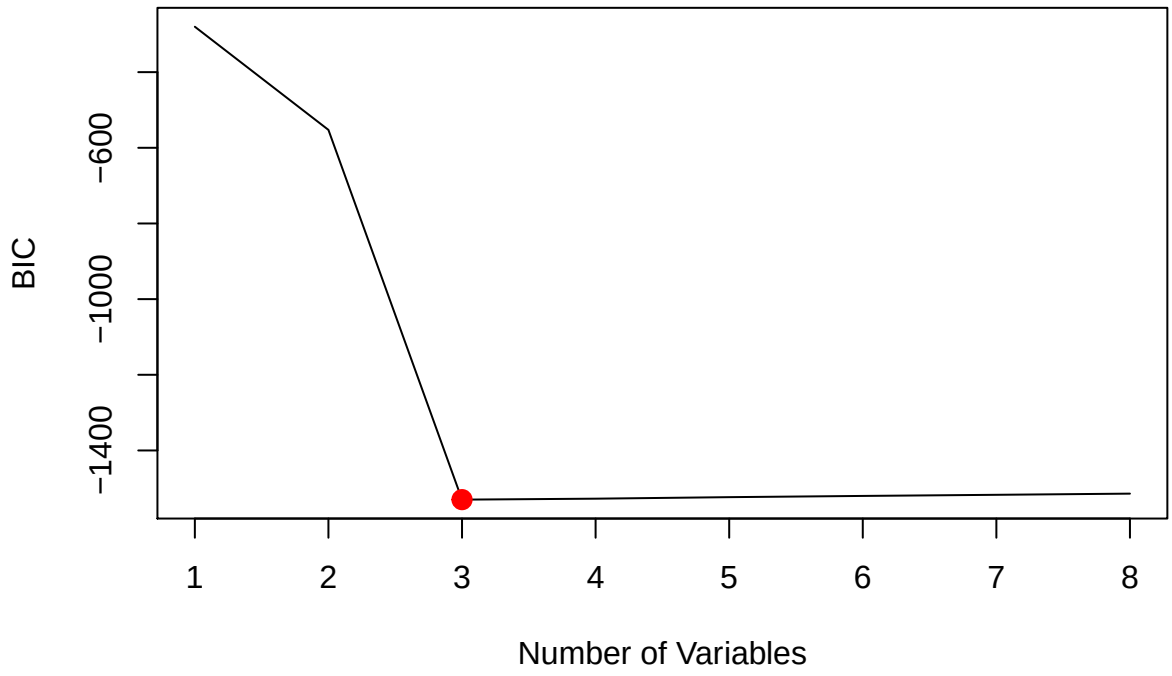
According to Cp, the best model is the third polynomial. According to BIC, the best model is the third polynomial. According to AdjR2, the best model is the third polynomial

The regression coefficient on the third polynomial is 4.000249.

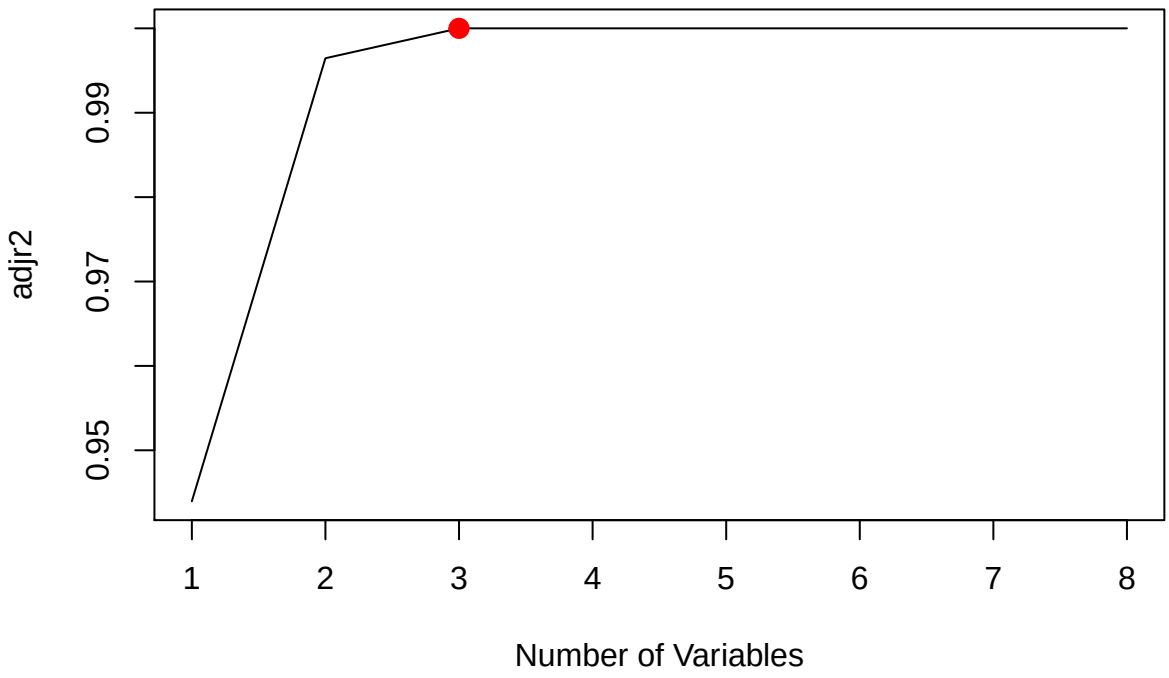
```
### Plot

# CP
#  Plot the CP value
plot(full_subsets_summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary$cp)
points(c,full_subsets_summary$cp[c],col="red",cex=2,pch=20)
```



```r
# BIC
#  Plot the BIC value
plot(full_subsets_summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary$bic)
points(c,full_subsets_summary$bic[c],col="red",cex=2,pch=20)
```




```r
# AdjR2
#  Plot the adjr2 value
plot(full_subsets_summary$adjr2,xlab="Number of Variables",ylab="adjr2",type='l')
# Find and mark the min value
c=which.max(full_subsets_summary$adjr2)
points(3,full_subsets_summary$adjr2[3],col="red",cex=2,pch=20)
```




```
# Have to manually adjust last plot because there looks to be slight difference to where 8 is higher, b
```

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

```
# Forward

full_subsets_forward <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8)

full_subsets_summary_forward = summary(full_subsets_forward, matrix.logical = TRUE)

full_subsets_summary_forward$cp
```

```
## [1] 2.82e+07 1.76e+06 3.82e+00 3.85e+00 5.31e+00 6.56e+00 8.41e+00 1.03e+01
```

```
full_subsets_summary_forward$bic
```

```
## [1]  -280  -553 -1530 -1527 -1523 -1520 -1515 -1511
```

```
full_subsets_summary_forward$adjr2
```

```
## [1] 0.944 0.996 1.000 1.000 1.000 1.000 1.000 1.000
```

```
# Backward

full_subsets_backward <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8

full_subsets_summary_backward = summary(full_subsets_backward, matrix.logical = TRUE)
```

```
full_subsets_summary_backward$cp
```

```
## [1] 2.82e+07 1.76e+06 3.82e+00 3.89e+00 5.50e+00 6.52e+00 6.52e+00 7.25e+00
```

```
full_subsets_summary_backward$bic
```

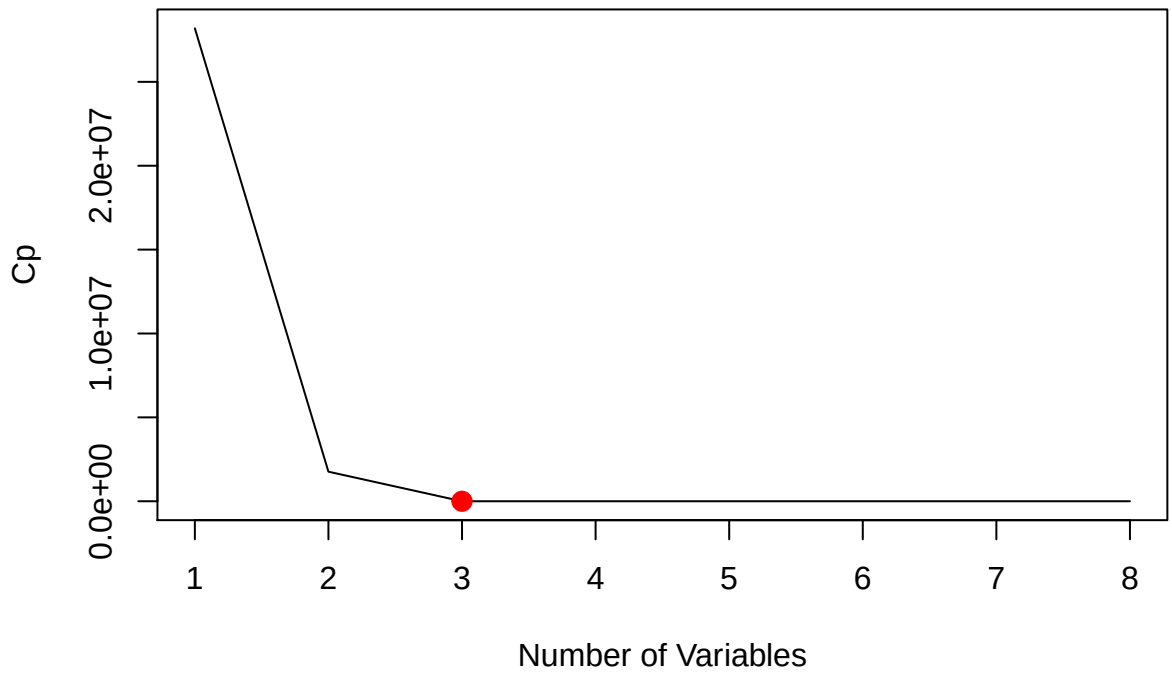
```
## [1]  -280  -553 -1530 -1527 -1523 -1520 -1517 -1514
```

```
full_subsets_summary_backward$adjr2
```

```
## [1] 0.944 0.996 1.000 1.000 1.000 1.000 1.000 1.000
```

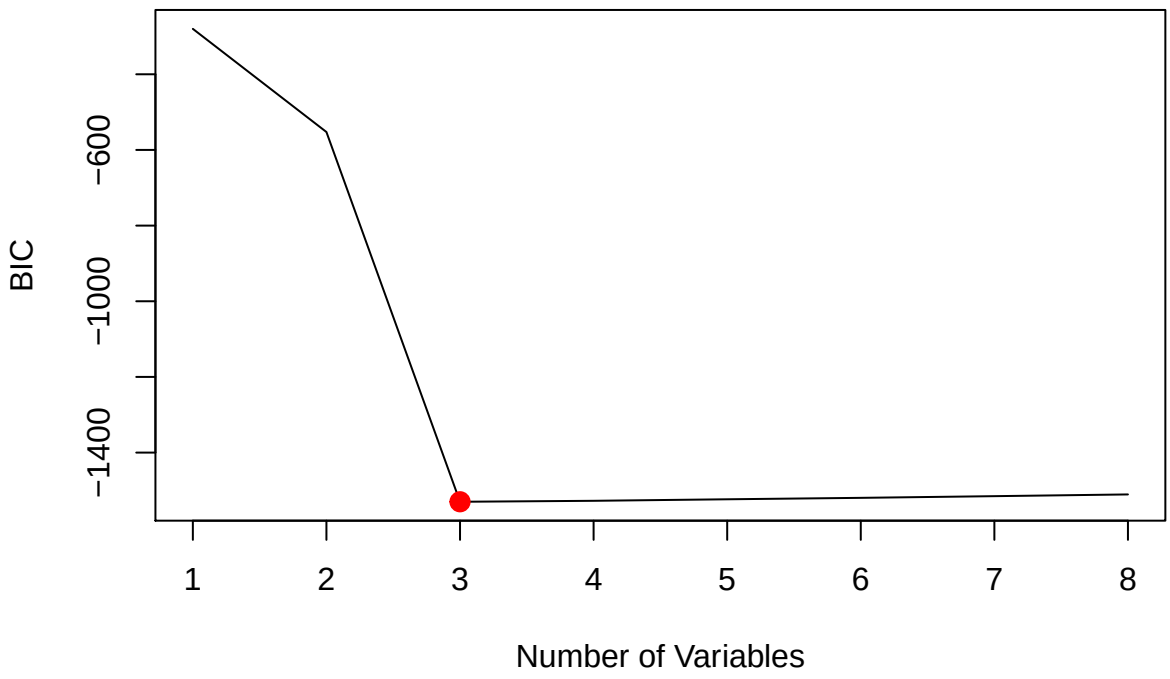
When using both forward and backwards stepwise regression, Cp, BIC, and AdjR2 all also show the cubic model to be the best.

```
### Plot - Forward

# CP
#  Plot the CP value
plot(full_subsets_summary_forward$cp,xlab="Number of Variables",ylab="Cp",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary_forward$cp)
points(c,full_subsets_summary_forward$cp[c],col="red",cex=2,pch=20)
```

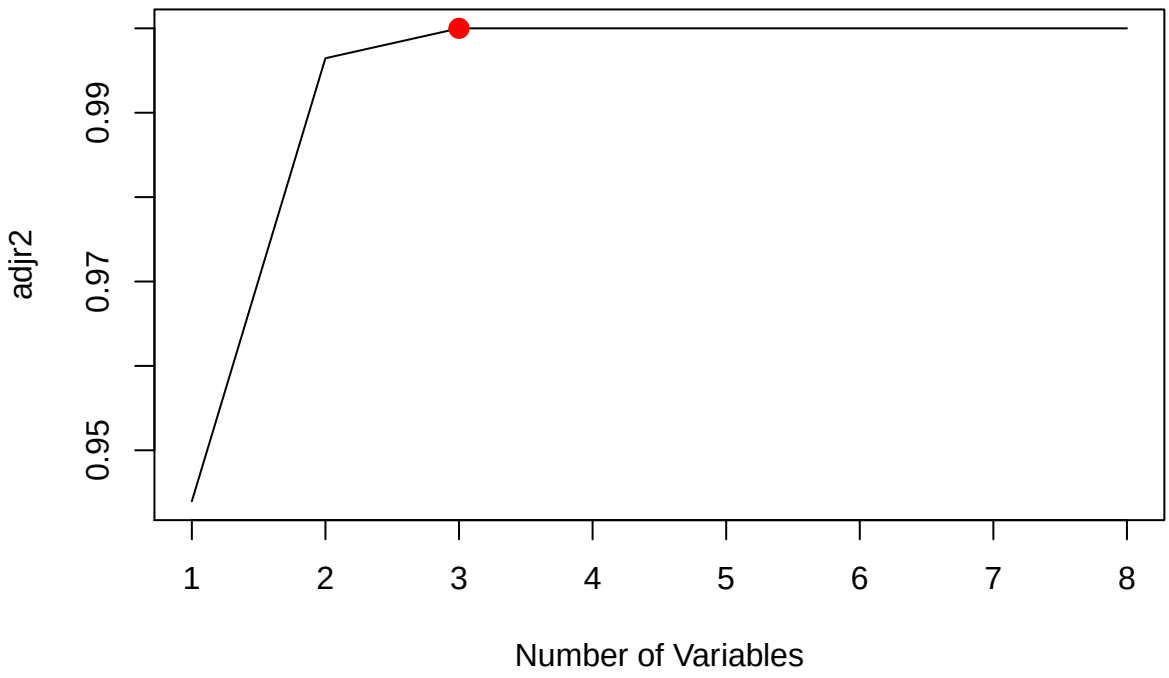



```
# BIC
#  Plot the BIC value
plot(full_subsets_summary_forward$bic,xlab="Number of Variables",ylab="BIC",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary_forward$bic)
```

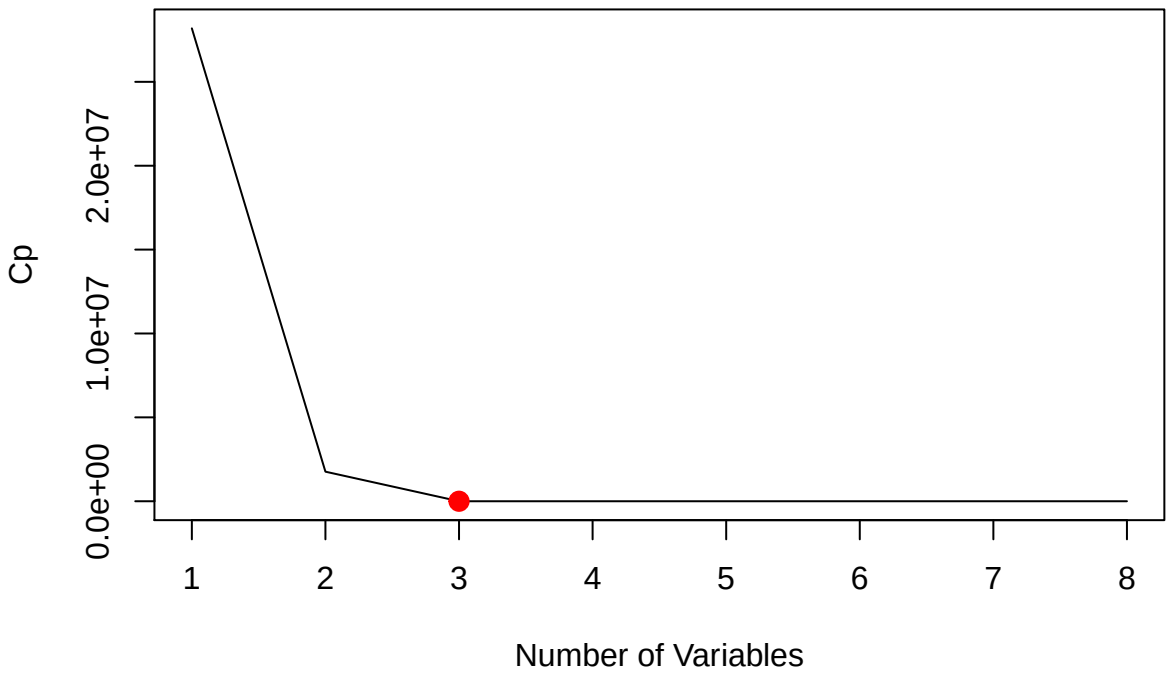
```
points(c,full_subsets_summary_forward$bic[c],col="red",cex=2,pch=20)
```



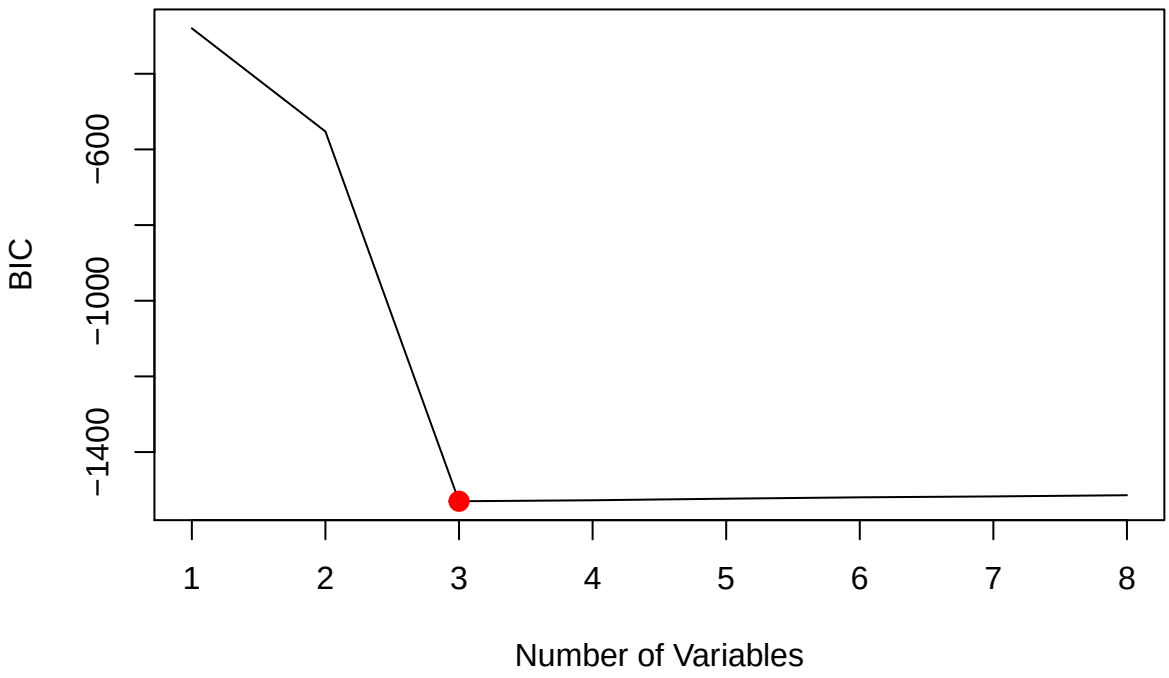

```
# AdjR2
#  Plot the adjr2 value
plot(full_subsets_summary_forward$adjr2,xlab="Number of Variables",ylab="adjr2",type='l')
# Find and mark the min value
c=which.max(full_subsets_summary_forward$adjr2)
points(3,full_subsets_summary_forward$adjr2[3],col="red",cex=2,pch=20)
```



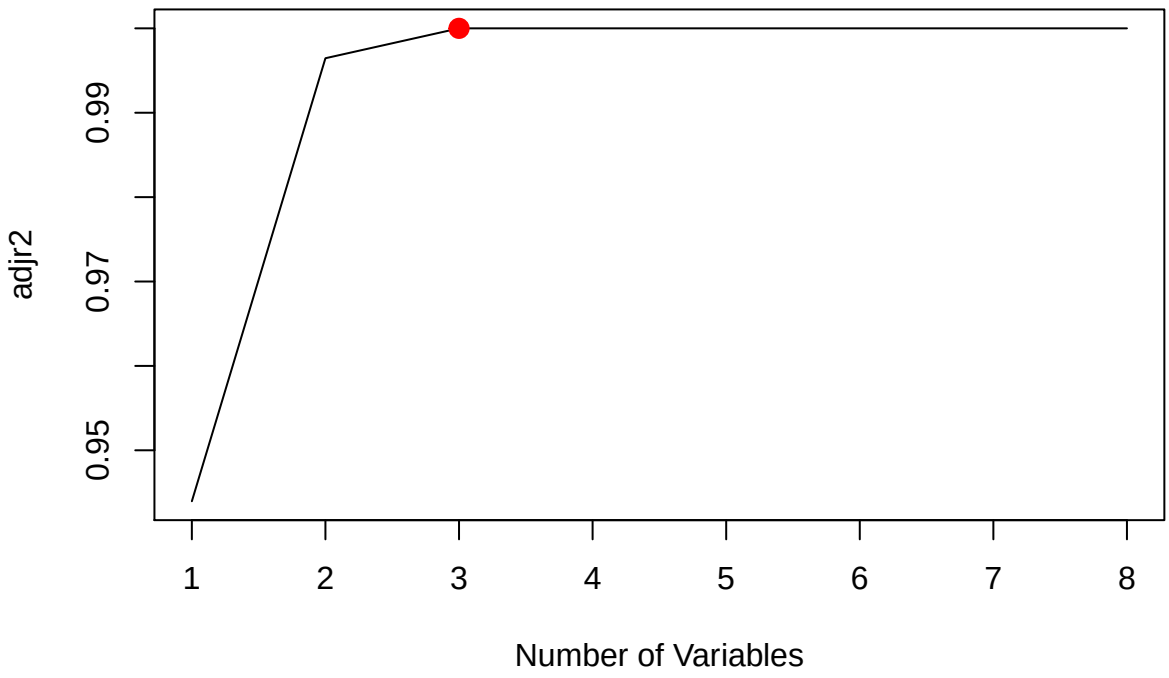

```
# Have to manually adjust last plot because there looks to be slight difference to where 8 is higher, b

### Plot - Backward

# CP
#  Plot the CP value
plot(full_subsets_summary_backward$cp,xlab="Number of Variables",ylab="Cp",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary_backward$cp)
points(c,full_subsets_summary_backward$cp[c],col="red",cex=2,pch=20)
```




```
# BIC
#  Plot the BIC value
plot(full_subsets_summary_backward$bic,xlab="Number of Variables",ylab="BIC",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary_backward$bic)
points(c,full_subsets_summary_backward$bic[c],col="red",cex=2,pch=20)
```




```
# AdjR2
#  Plot the adjr2 value
plot(full_subsets_summary_backward$adjr2,xlab="Number of Variables",ylab="adjr2",type='l')
# Find and mark the min value
c=which.max(full_subsets_summary_backward$adjr2)
points(3,full_subsets_summary_backward$adjr2[3],col="red",cex=2,pch=20)
```




```
# Have to manually adjust last plot because there looks to be slight difference to where 8 is higher, b
```

The plots look identical to what we had with best subset.

Extra 43 (3 points)

Suppose you are given a single data set with p = 3 predictors X1, . . . , X3. In order to add flexibility to a linear model, you have decided to include also all powers X2i and all interaction terms XiXj as variables.

a) How many variables are there now? How many different models are there with exactly 3 variables? You don't have to list them.

Assuming we are only considering models where the main effects are included with the polynomial effects (i.e. following the hierarchy principle), there are 10 models with exactly 3 variables. If not following the hierchy principle (Sample of 3 from 9 choices without replacement and where order does not matter), there are 84 models with exactly 3 variables. We still technically only have three variables, but there are nine possible covriates if we are including quadratics and interactions.

b) List all models with exactly 3 variables that satisfy the hierarchy principle.

Y ~ X1 X2 X3 Y ~ X1 X12 X2 Y ~ X1 X12 X3 Y ~ X2 X22 X1 Y ~ X2 X22 X3 Y ~ X3 X32 X1 Y ~ X3 X32 X2

Y ~ X1 X2 X2X1 Y ~ X1 X3 X3X1 Y ~ X2 X3 X2X3

c) Propose a modification of the forward selection method such that all selected models satisfy the hierarchy principle. Explain it in words. You don't have to write down a formal algorithm.

The greedy search would have to be adjusted so that quadratics are only added when the main effects are already included and that interactions are only added when both of the main effects are already included. From an algorithm stand point, this would consist of a simple ifthen statement that says if, for example, X1 is already in the model, add the quadratic term X1^2, or is X1 and X3 are already in the model, add the interaction term X1X3.

Extra 44 (3 points)

Consider the concrete strength data from problem 37. There are eight predictors and one response. Use forward selection and backward selection to identify good models with k = 1, . . . , 8 variables. Plot the BIC values for both sequences of models in the same graph. Do the sequences of models agree?

```
concrete <- read_excel("Concrete_Data.xls")
colnames(concrete) <- c("cementkg", "blustfur", "flyash", "superplas", "courseagg", "fineagg", "age", "
```

```
# Forward

full_subsets_forward <- regsubsets(CCS ~ cementkg + blustfur + flyash + superplas + courseagg + fineagg

full_subsets_summary_forward = summary(full_subsets_forward, matrix.logical = TRUE)

full_subsets_summary_forward$cp
```

```
## [1] 28.88 18.29 15.87 16.30 13.99  7.17  8.00
```

```
full_subsets_summary_forward$bic
```

```
## [1] -68.7 -74.2 -71.6 -66.2 -63.6 -65.5 -59.7
```

```
full_subsets_summary_forward$adjr2
```

```
## [1] 0.0762 0.0863 0.0893 0.0898 0.0927 0.0996 0.0997
```

```
# Backward

full_subsets_backward <- regsubsets(CCS ~ cementkg + blustfur + flyash + superplas + courseagg + fineag

full_subsets_summary_backward = summary(full_subsets_backward, matrix.logical = TRUE)

full_subsets_summary_backward$cp
```

```
## [1] 89.11 67.91 46.17 35.43  7.45  7.17  8.00
```

```
full_subsets_summary_backward$bic
```

```
## [1] -11.5 -26.3 -42.0 -47.4 -70.1 -65.5 -59.7
```

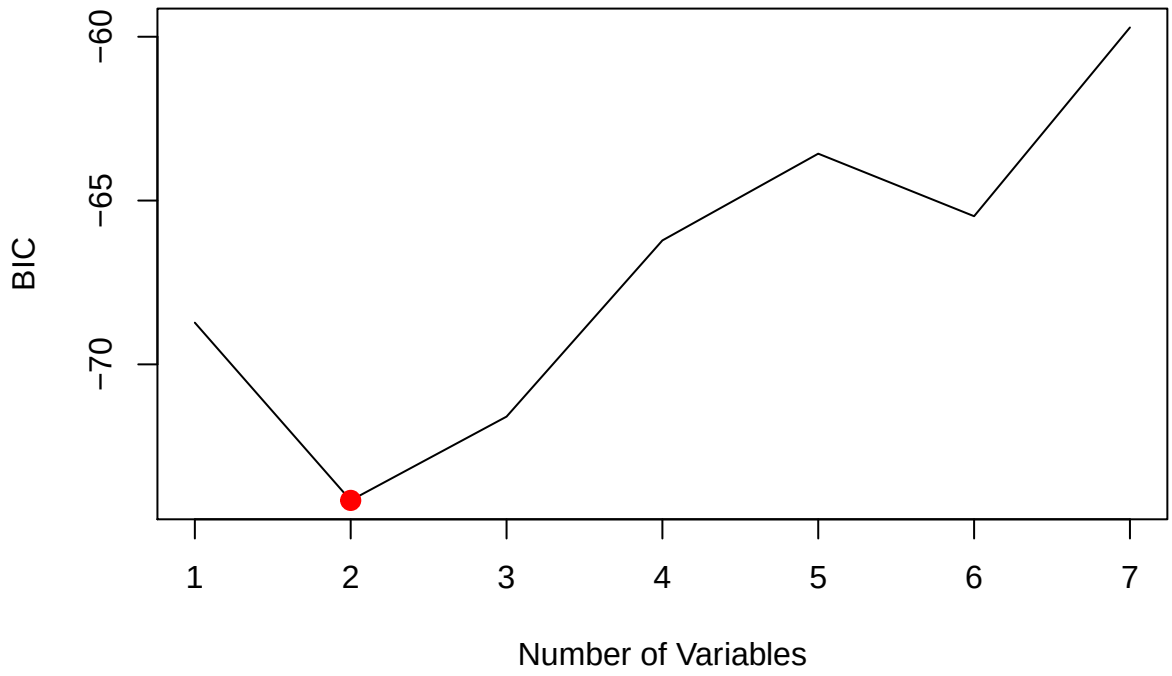
```
full_subsets_summary_backward$adjr2
```

```
## [1] 0.0234 0.0428 0.0627 0.0730 0.0984 0.0996 0.0997
```

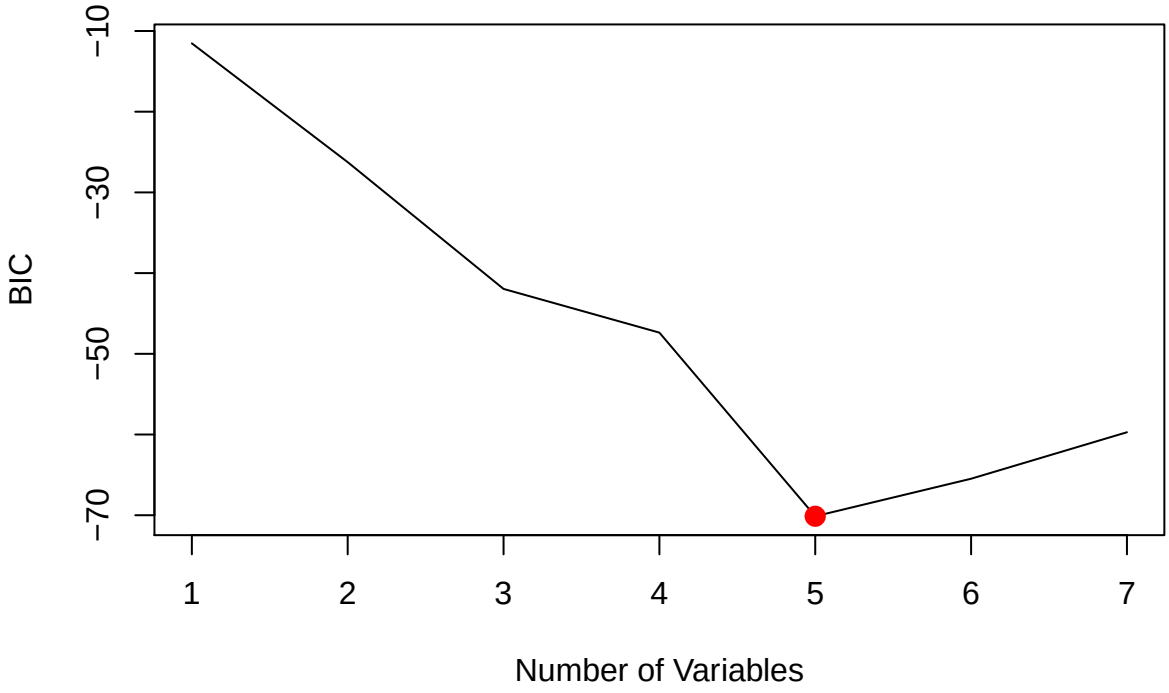
In looking at the Cp, BIC, and Adj R2, forward subset selection indicates that 6,2, and 7 respectively are best. Cp and BIC though are most useful in this case as just adding covariates may boost adjusted R2 despite the penalty imposed based on the addition of covariates.

In looking at the Cp, BIC, and Adj R2, forward subset selection indicates that 6, 5, and 7 respectively are best. Cp and BIC though are most useful in this case as just adding covariates may boost adjusted R2 despite the penalty imposed based on the addition of covariates.

```
### Plot - Forward

# BIC
#  Plot the BIC value
plot(full_subsets_summary_forward$bic,xlab="Number of Variables",ylab="BIC",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary_forward$bic)
points(c,full_subsets_summary_forward$bic[c],col="red",cex=2,pch=20)
```




```
### Plot - Backward

# BIC
#  Plot the BIC value
plot(full_subsets_summary_backward$bic,xlab="Number of Variables",ylab="BIC",type='l')
# Find and mark the min value
c=which.min(full_subsets_summary_backward$bic)
points(c,full_subsets_summary_backward$bic[c],col="red",cex=2,pch=20)
```



The plot shows that forward and backward subset selection can produce different results as here forward favors the model with 2 variables and backward the model with 5.

(Ask about this. This kind of makes sense but I'm not sure)

Book 1 (5 Points)

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:

That is, we fit all p models selection that contain exactly one predictor, all (p 2) = p(p???1)/2 models that contain exactly two predictors, and so forth.

Here best is defined as having the smallest RSS, or equivalently largest R2.

This task must be performed with care, because the RSS of these p + 1 models decreases monotonically, and the R2 increases monotonically, as the number of features included in the models increases. Therefore, if we use these statistics to select the best model, then we will always end up with a model involving all of the variables. The problem is that a low RSS or a high R2 indicates a model with a low training error, whereas we wish to choose a model that has a low test error.

While best subset selection is a simple and conceptually appealing approach, it suffers from computational limitations. The number of possible models that must be considered grows rapidly as p increases. In general, there are 2p models that involve subsets of p predictors. So if p = 10, then there are approximately 1,000 possible models to be considered, and if 6.1 Subset Selection 207 p = 20, then there are over one million possibilities!

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

The model selected through best subset selection will have the smallest training RSS because it will select a model involving all of the variables and the RSS of these p + 1 models decreases monotonically as the number of the features increases. In the case of best subset selection, the larger the search space, the higher the chance of finding models that look good on the training data.

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

A low training error by no means guarantees a low test error. As mentioned above, it will not necessarily be the model chosen through best subset selection even though the train RSS may be small. Any of the models could ultimately have the smallest test RSS.

(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.

TRUE. Because we are moving forward in adding predictors, the k+1 variable model will include all k features chosen plus additional features that decrease error..

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.

TRUE. Because we are moving forward in adding predictors, the k+1 variable model will include all k features chosen minus features that do not decrease the error.

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.

FALSE. Forward and Backward stepwise do not always give us the same conclusions in terms of the best model.

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 backward stepwise selection.

FALSE. Forward and Backward stepwise do not always give us the same conclusions in terms of the best model.

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.

FALSE. Because of the different combinatorial possibilities in the k-variable model versus the (k+1) variable model, they are not guaranteed to select the same best model.

•

Book 10 (5 points)

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 = 1,000 observations, and an associated quantitative response vector generated according to the model Y = X?? + , where ?? has some elements that are exactly equal to zero

```
set.seed(1)
data <- matrix(rnorm(1000 *20), nrow = 1000, ncol = 20)
beta=rnorm(20,sd=10)
beta[c(1:5)]=0
e=rnorm(1000)
Y=as.vector(data%*%beta+e)
```

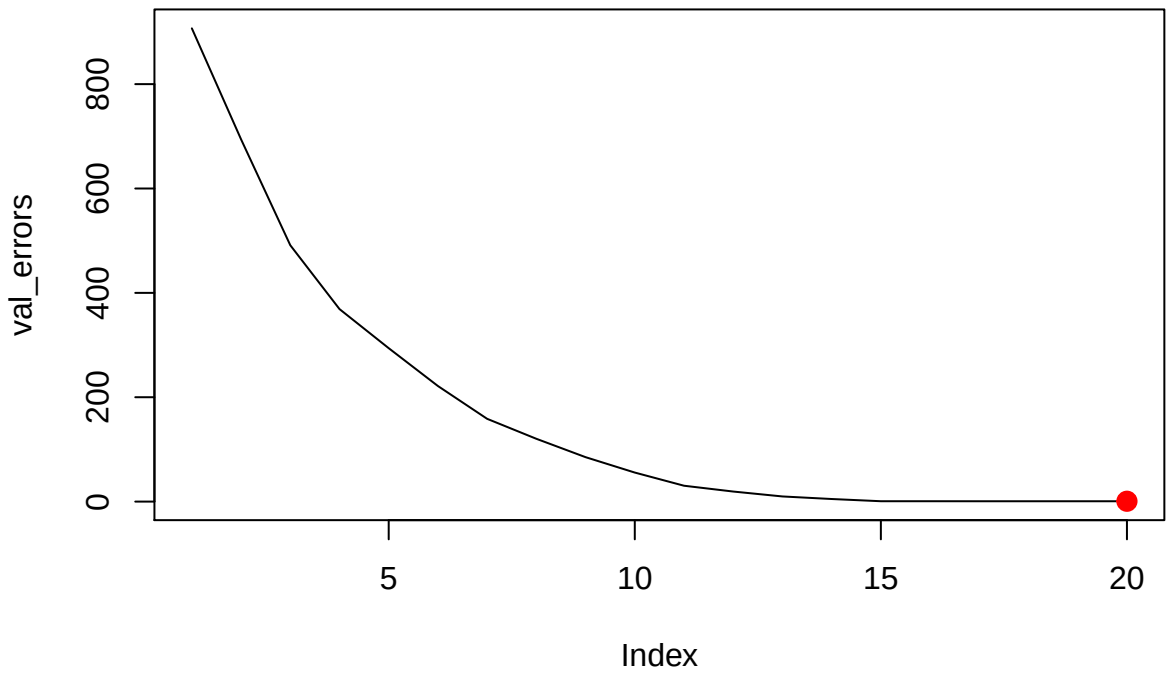
```
data <- cbind(data.frame(Y=Y),data)
```

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

```
train <- sample(1000, 100)
data_train <- data[train,]
data_test <- data[-train,]
```

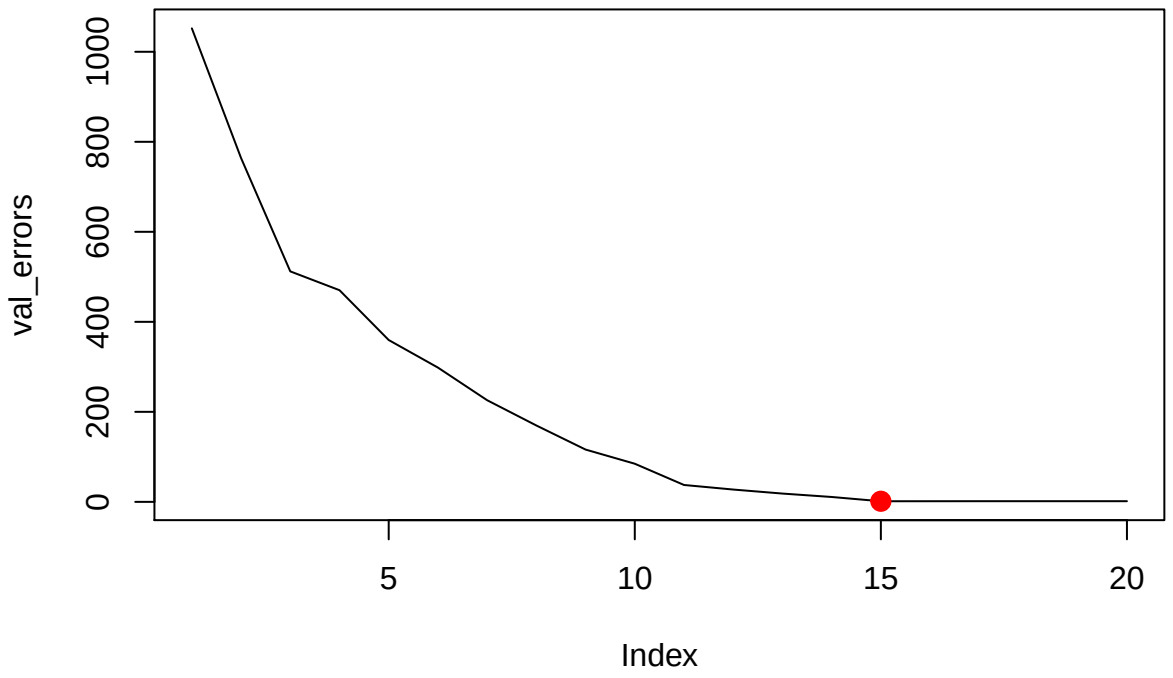
(c) Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

```
# Calculate
regfit_best_train = regsubsets(Y~., data = data_train, nvmax = 20)

train_mat = model.matrix (Y ~., data = data_train)

val_errors = rep(NA,20)

# Iterates over each size i
for(i in 1:20){

    # Extract the vector of predictors in the best fit model on i predictors
    coefi = coef(regfit_best_train, id = i)

    # Make predictions using matrix multiplication of the train matirx and the coefficients vector
    pred = train_mat[,names(coefi)]%*%coefi

    # Calculate the MSE
    val_errors[i] = mean((data_train$Y-pred)^2)
}

# Plot MSE
min = which.min(val_errors)
plot(val_errors, type = 'l')
points(min, val_errors[min][1], col = "red", cex = 2, pch = 20)
```



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

```
# Calculate

test_mat = model.matrix (Y ~., data = data_test)

val_errors = rep(NA,20)

# Iterates over each size i
for(i in 1:20){

    # Extract the vector of predictors in the best fit model on i predictors
    coefi = coef(regfit_best_train, id = i)

    # Make predictions using matrix multiplication of the test matirx and the coefficients vector
    pred = test_mat[,names(coefi)]%*%coefi

    # Calculate the MSE
    val_errors[i] = mean((data_test$Y-pred)^2)
}

# Plot MSE
min = which.min(val_errors)
plot(val_errors, type = 'l')
points(min, val_errors[min][1], col = "red", cex = 2, pch = 20)
```



(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.

The test set MSE takes on its minimum value at a model size of 15.

(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_best_train, 15)
```

```
## (Intercept)         `6`         `7`         `8`         `9`        `10`
##       0.168      -2.603     -13.969       7.354      20.233       7.386
##        `11`        `12`        `13`        `14`        `15`        `16`
##       8.597       5.528      -2.930      -6.905      -7.035      -2.594
##        `17`        `18`        `19`        `20`
##       3.106      16.548       8.950      -9.936
```

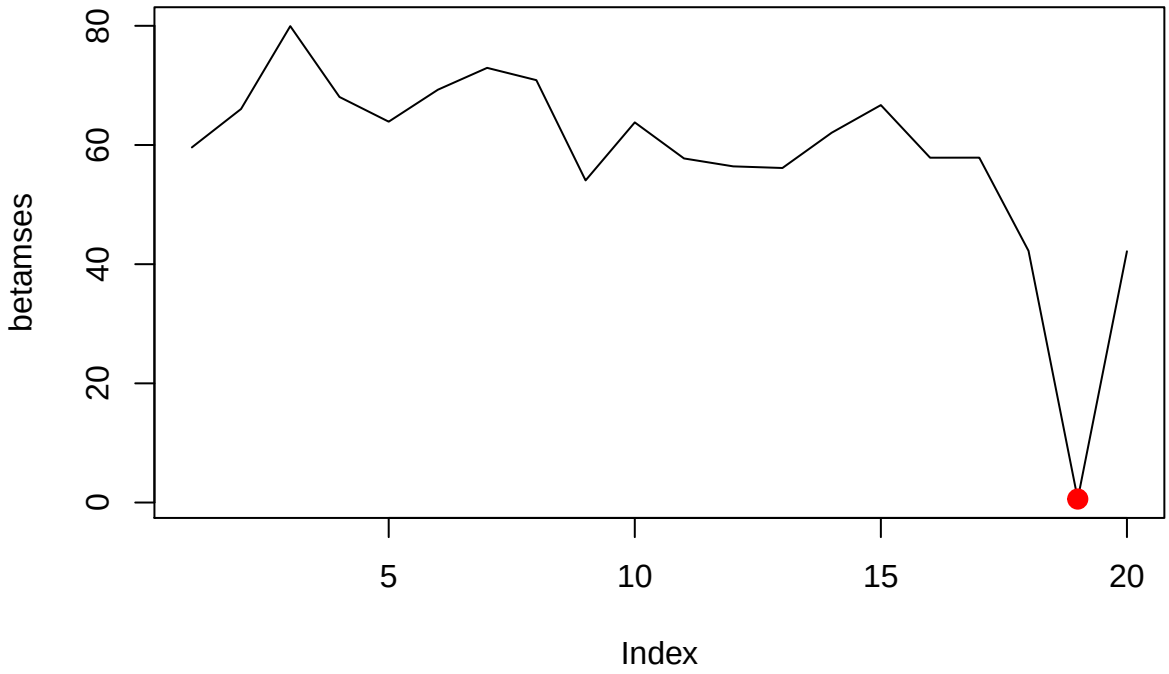
```
beta
```

```
##  [1]   0.00   0.00   0.00   0.00   0.00  -2.84 -14.10   7.23  20.31   7.30
## [11]   8.79   5.55  -2.85  -6.75  -7.15  -2.71   3.13  16.70   8.92 -10.15
```

The coefficient for the 15th element of the true model used to generate the data (-7.15) is very similar to the X15 coefficient (-7.035) from the best model as determined by best subbset selection.

(g) Create a plot displaying 'pj=1(??j ??? ??ˆrj)2 for a range of values 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)?

```r
# Using 1:20 as values of r
betamses <- sapply(1:20,function(r){
  # Get the coefficients for models of 1:20 parameters
  beta.est <-coef(regfit_best_train,id=r)
  # Get MSE for each model (What the problem requests. Bj from original true model generation)
  return(sqrt(sum((beta-beta.est)^2)))
})

min = which.min(betamses)
plot(betamses, type = 'l')
points(min, betamses[min][1], col = "red", cex = 2, pch = 20)
```



We do not have a consistent decline here because we are subtracting the coefficient from the betas we created, which included 0 elements.

Extra 46 (5 points)

We'll use the MNIST image classification data again, available as mnist_all.RData that were used in class during the last two weeks. We want to distinguish between 1 and 3. Extract the relevant training data and place them in a data frame. Remove all variables (pixels) that have zero variance, i.e. pixels that have the same value for both digits. Repeat this for the test data. In this problem, you will do two forward selection steps for finding good logistic models.

```r
mnist <-load('mnist_all.RData')
```

```
mnist_train <- data.frame(train$n, train$x, train$y)
mnist_train <- mnist_train[mnist_train$train.y == 1 | mnist_train$train.y == 3,]
mnist_train <- mnist_train[ - as.numeric(which(apply(mnist_train, 2, var) == 0))]
mnist_test <- data.frame(test$n, test$x, test$y)
mnist_test <- mnist_test[mnist_test$test.y == 1 | mnist_test$test.y == 3,]
mnist_test <- mnist_test[ - as.numeric(which(apply(mnist_test, 2, var) == 0))]
```

a) Find the pixel that gives the best logistic model for the training data, using the area under the ROC curve as a criterion. Do this with a complete search. Do not show the output of all logistic models!

```
logreg <- glm(factor(train.y) ~ X43, data = mnist_train, family = binomial)
pred <- predict(logreg, type = 'response')
auc(mnist_train$train.y, pred)
```

```
## Area under the curve: 0.5
```

```
predictors <- names(mnist_train)[1:624]

logreg <- function(x) {
  logreg <- glm(paste("factor(train.y) ~", x), data = mnist_train, family = binomial)
  pred <- predict(logreg, type = 'response')
  AUC <- auc(mnist_train$train.y, pred)
  full <- cbind(x, AUC)
  return(full)
}

onevar <- lapply(predictors, logreg)
onevar<- do.call(rbind.data.frame, onevar)

onevar$AUC <- as.numeric(as.character(onevar$AUC))
onevar$x[which.max(onevar$AUC)]
```

```
##    x
## X490
## 624 Levels: X43 X44 X45 X67 X68 X69 X70 X71 X72 X73 X74 X75 X76 X77 ... X751
```

The variable for which the one-variable model shows the highest AUC is pixel 490.

b) Now find one more pixel such that the resulting logistic model using the pixel from a) together the new one has the best area under the ROC curve. Do this with a complete search. Minimize the output.

```
predictors2 <- names(mnist_train)[c(1:383,385:624)]

logreg2 <- function(x) {
  logreg <- glm(paste("factor(train.y) ~ X490 + ", x), data = mnist_train, family = binomial)
  pred <- predict(logreg, type = 'response')
  AUC <- auc(mnist_train$train.y, pred)
  full <- cbind(x, AUC)
  return(full)
}

twovar <- lapply(predictors2, logreg2)
twovar<- do.call(rbind.data.frame, twovar)

twovar$AUC <- as.numeric(as.character(twovar$AUC))
twovar$x[which.max(twovar$AUC)]
```

```
##    x
## X495
## 623 Levels: X43 X44 X45 X67 X68 X69 X70 X71 X72 X73 X74 X75 X76 X77 ... X751
```

The variable for which the two-variable model shows the highest AUC is pixel 495.

c) Use the test data to decide whether the second model is really better than the first one.

```
# Model One
ModOne <- glm(factor(train.y) ~ X490, data = mnist_train, family = binomial)
pred <- predict(ModOne, newdata = mnist_test, type = 'response')
auc(mnist_test$test.y, pred)
```

```
## Area under the curve: 0.958
```

```
# Model Two
ModTwo <- glm(factor(test.y) ~ X490 + X495, data = mnist_test, family = binomial)
pred <- predict(ModTwo, newdata = mnist_test, type = 'response')
auc(mnist_test$test.y, pred)
```

```
## Area under the curve: 0.982
```

According to the respective AUCs for the test, the second model is indeed the better model.

d) How many logistic models altogether have you examined? How many will you have to examine if you want to continue this process and make the best logistic model with 10 pixels?

Overall, I have examined $624 + 623 = 1247$ models. If I wanted to continue the proicess and make the best logistic model with 10 pixels, I would examine 6195 models.