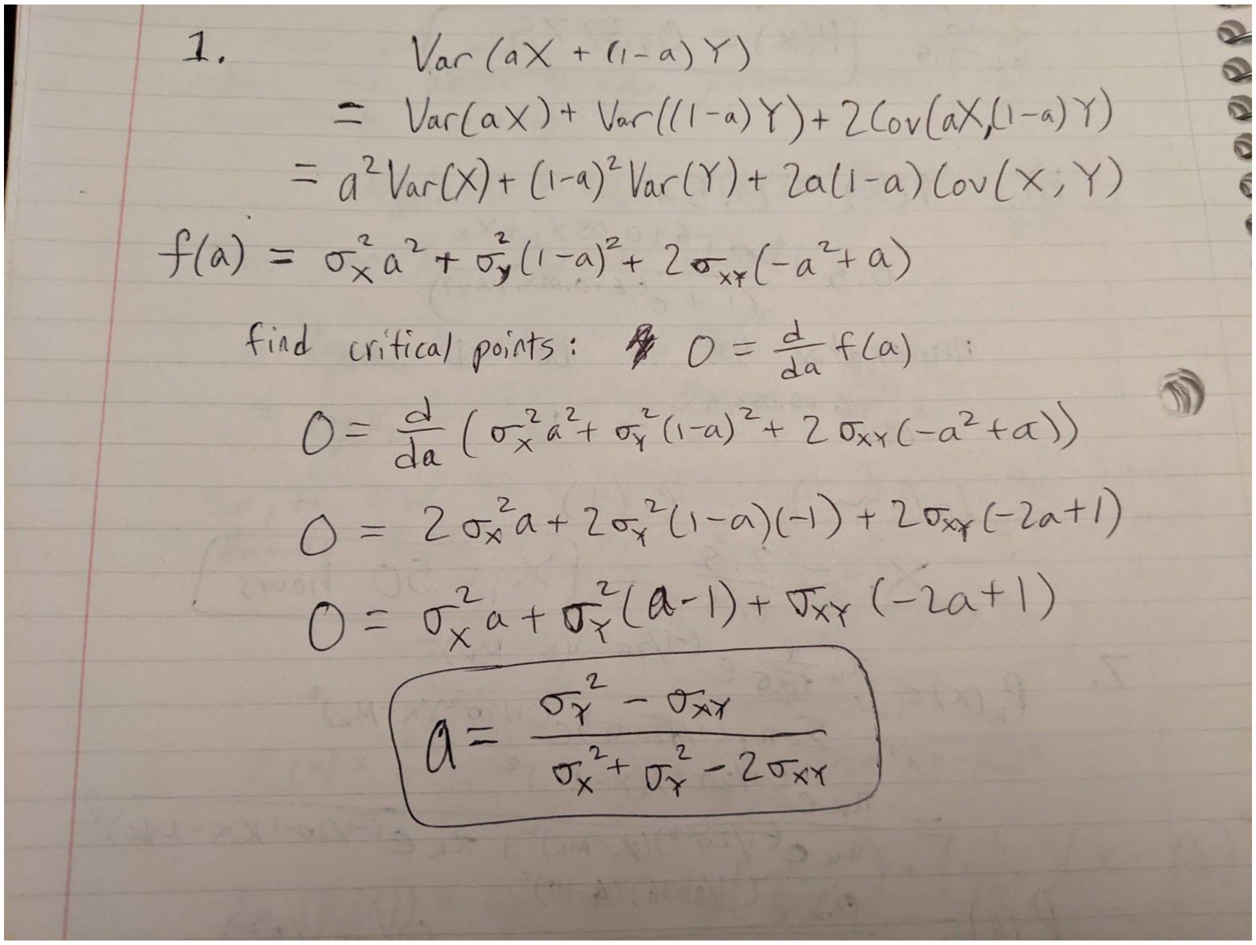
Rakesh Veetekat

Conceptual and theoretical questions

1. Using basic statistical properties of the variance, as well as singlevariable calculus, derive (5.6). In other words, prove that α given by (5.6) does indeed minimize Var(αX + (1 − α)Y). (Ch. 5, Question 1)



1.

$$Var(aX + (1-a)Y)$$

$$= Var(aX) + Var((1-a)Y) + 2Cov(aX,(1-a)Y)$$

$$= a^2 Var(X) + (1-a)^2 Var(Y) + 2a(1-a)\,Cov(X;Y)$$

$$f(a) = \sigma_X^2 a^2 + \sigma_Y^2 (1-a)^2 + 2\sigma_{XY}(-a^2+a)$$

find critical points: ~~#~~ $0 = \frac{d}{da} f(a)$:

$$0 = \frac{d}{da}\left(\sigma_X^2 a^2 + \sigma_Y^2 (1-a)^2 + 2\sigma_{XY}(-a^2+a)\right)$$

$$0 = 2\sigma_X^2 a + 2\sigma_Y^2 (1-a)(-1) + 2\sigma_{XY}(-2a+1)$$

$$0 = \sigma_X^2 a + \sigma_Y^2 (a-1) + \sigma_{XY}(-2a+1)$$

$$\boxed{a = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}}$$

2. We now review k-fold cross-validation. (Ch. 5, Question 3)
   a. Explain how k-fold cross-validation is implemented.

K-fold cross-validation is implemented by first taking the data of n observations and splitting it into “k” folds non-overlapping groups of approximately the same size. The first fold is used as a validation set, while the model is fitted on the remaining folds. After computing the MSE on the observations for the remaining folds, the process is repeated “k” times with different folds being held out each time. After “k” iterations, the test error is estimated by averaging the “k” MSE results.

   b. What are the advantages and disadvantages of k-fold crossvalidation relative to:
      i. The validation set approach?

Compared to the validation set approach, there are 2 advantages to using k-fold cross-validation. First of all, the validation set approach may overestimate the test error rate because it will only use a subset of the data to fit the model, and that means that the approach is trained on fewer observations than k-fold cross-validation. Secondly, the validation set approach could estimate a highly variable test error rate because it depends greatly on the observations used in the training set and observations in the validation set.

ii. LOOCV?

LOOCV is a specialized version of k-fold cross-validation where “k” is the same as the total number of observations in the data, and this means that LOOCV is much more computationally expensive than k-fold cross-validation. Also, LOOCV has a lower bias, but a higher variance than k-fold cross-validation, and k-fold cross-validation with values of 5 or 10 for “k” have been seen to result in test error rates that don’t seem to suffer from high bias or high variance.

3. Suppose that we use some statistical learning method to make a prediction for the response Y for a particular value of the predictor X. Carefully describe how we might estimate the standard deviation of our prediction. (Ch. 5, Question 4)

We could estimate the standard deviation of our prediction by using the bootstrap approach. We can use repeated random samples of the data set we have and the different values for the response, Y, can be used to determine the standard deviation for our prediction.

4. 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: (Ch. 6, Question 1)
   a. Which of the three models with k predictors has the smallest training RSS?

The best subset selection model has the smallest training RSS because the training RSS for the forward stepwise and backward stepwise selection models will depend on the predictors that are selected first for either model.

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

The best subset selection model has the best chance to have the smallest test RSS because it considers more models than the other approaches, but the test RSS could be lower for the other two approaches due to luck.

   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

      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

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

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

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

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

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}|\beta_j| \le s$$

for a particular value of s. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer. (Ch. 6, Question 3)

   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.
      iii. Steadily increase.
      iv. Steadily decrease.
      v. Remain constant.

Steadily decrease.
By increasing s, the model becomes more flexible and this will lead to a decreasing training RSS.

   b. Repeat (a) for test RSS

Decrease initially, and then eventually start increasing in a U shape.
As stated in (a), the model becomes more flexible as s increases, and this means that the test RSS will reduce in the beginning, but will start increasing when overfitting starts.

   c. Repeat (a) for variance

Steadily increase.

The model is becoming more flexible as s increases, and the variance will also steadily increase with the flexibility.

d. Repeat (a) for (squared) bias

Steadily decrease.
Bias decreases when flexibility increases, so the bias will steadily decrease in this model where flexibility is climbing as s increases.

e. Repeat (a) for the irreducible error

Remain constant
Irreducible error is independent of the parameters for the model so it will remain constant as s increases.

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

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

for a particular value of λ. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer. (Ch. 6, Question 4)

a. As we increase λ 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.
   iii. Steadily increase.
   iv. Steadily decrease.
   v. Remain constant.

Steadily increase.
Increasing lambda will decrease the flexibility of the model because beta is becoming more restricted. As the flexibility of the model decreases, the training RSS will steadily increase.

b. Repeat (a) for test RSS

Decrease initially, and then eventually start increasing in a U shape.
As stated in (a), the model is becoming less and less flexible, which means that the test RSS will start out decreasing, but then start increasing when overfitting begins.

c. Repeat (a) for variance

Steadily decrease.
As the flexibility of the model decreases, the variance of the model also steadily decreases.

d. Repeat (a) for (squared) bias

Steadily increase.

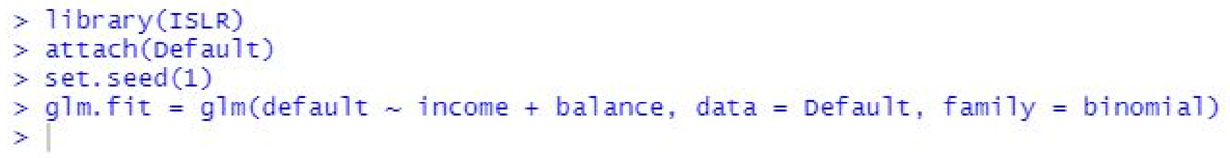
The bias of the model will steadily increase as the flexibility of the model decreases.

e. Repeat (a) for the irreducible error

Remain constant.
Irreducible error is still independent of the parameters for the model, and increasing lambda in the model does not affect the irreducible error at all.

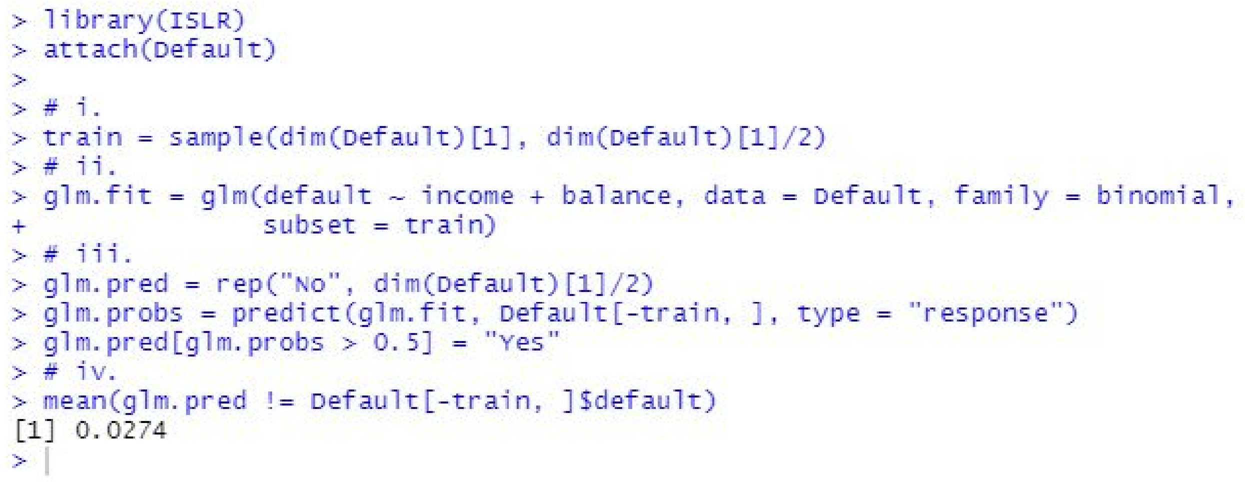
Applied

7. In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis. (Ch. 5, Question 5)
    a. Fit a logistic regression model that uses income and balance to predict default.



```
> library(ISLR)
> attach(Default)
> set.seed(1)
> glm.fit = glm(default ~ income + balance, data = Default, family = binomial)
> 
```

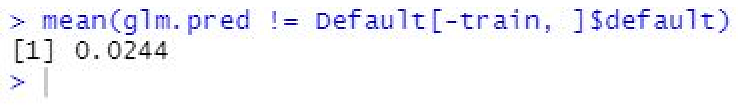
b. Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:
    i. Split the sample set into a training set and a validation set. 5.4 Exercises 199
    ii. Fit a multiple logistic regression model using only the training observations.
    iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.
    iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified



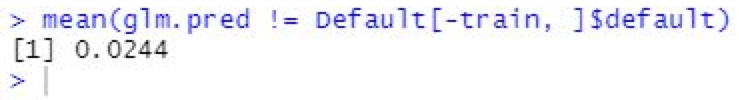
```
> library(ISLR)
> attach(Default)
> 
> # i.
> train = sample(dim(Default)[1], dim(Default)[1]/2)
> # ii.
> glm.fit = glm(default ~ income + balance, data = Default, family = binomial,
+               subset = train)
> # iii.
> glm.pred = rep("No", dim(Default)[1]/2)
> glm.probs = predict(glm.fit, Default[-train, ], type = "response")
> glm.pred[glm.probs > 0.5] = "Yes"
> # iv.
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0274
> 
```

2.74% is the test error for this model.

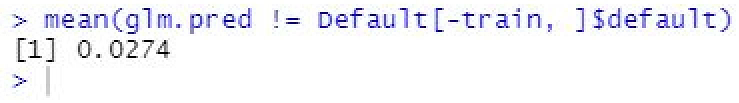
c. Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.



```
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0244
> |
```



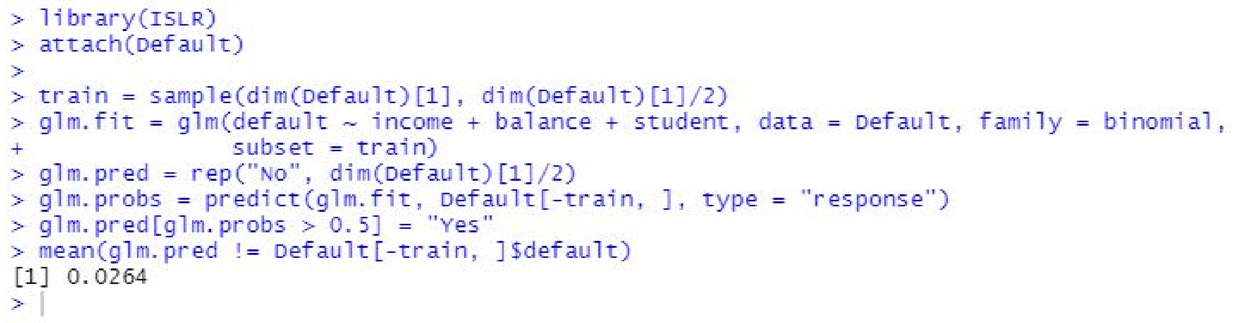
```
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0244
> |
```



```
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0274
> |
```

The varying test error rates of the three different runs averages out to 0.0254, or an average test error rate of 2.54%.

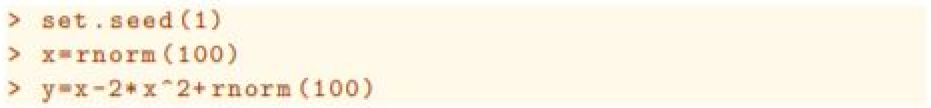
d. Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.



```
> library(ISLR)
> attach(Default)
>
> train = sample(dim(Default)[1], dim(Default)[1]/2)
> glm.fit = glm(default ~ income + balance + student, data = Default, family = binomial,
+               subset = train)
> glm.pred = rep("No", dim(Default)[1]/2)
> glm.probs = predict(glm.fit, Default[-train, ], type = "response")
> glm.pred[glm.probs > 0.5] = "Yes"
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0264
> |
```

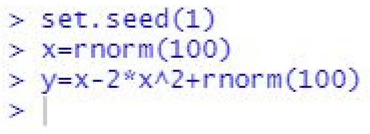
Using a dummy variable for student yields a 2.64% test error rate, so it appears that adding the student dummy variable doesn't lead to a noticeable reduction in test error rate.

8. We will now perform cross-validation on a simulated data set. (Ch. 5, Question 8)
   a. Generate a simulated data set as follows:



```
> set.seed(1)
> x=rnorm(100)
> y=x-2*x^2+rnorm(100)
```

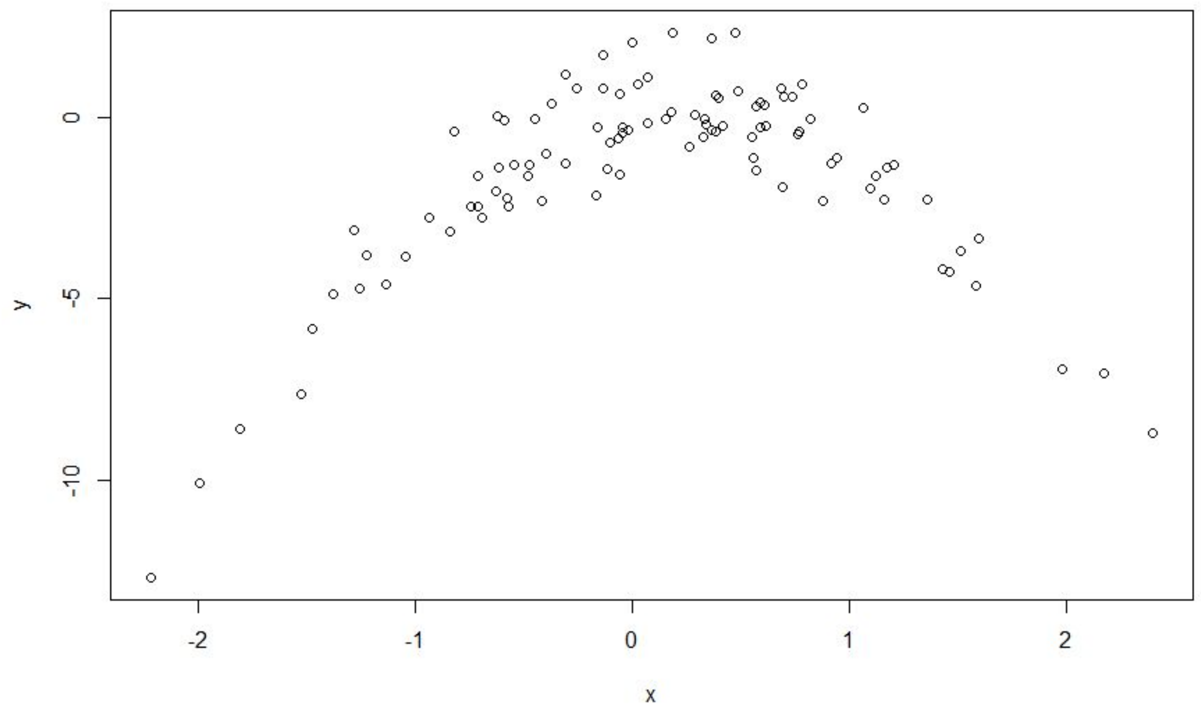
In this data set, what is n and what is p? Write out the model used to generate the data in equation form.



```
> set.seed(1)
> x=rnorm(100)
> y=x-2*x^2+rnorm(100)
> |
```

In this dataset, n is 100 and p is 2.
The model is y = x - 2x^2 + ϵ

b. Create a scatterplot of X against Y . Comment on what you find.

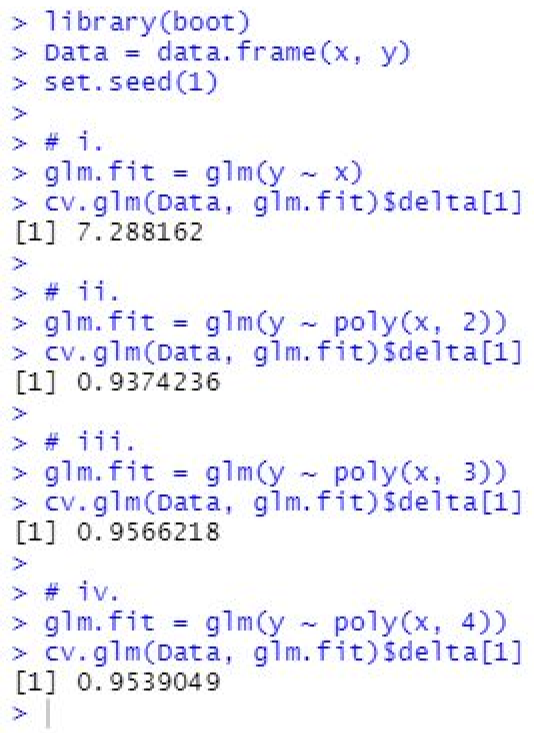


The plot appears to show a quadratic relationship with an inverted U shape.

c. Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

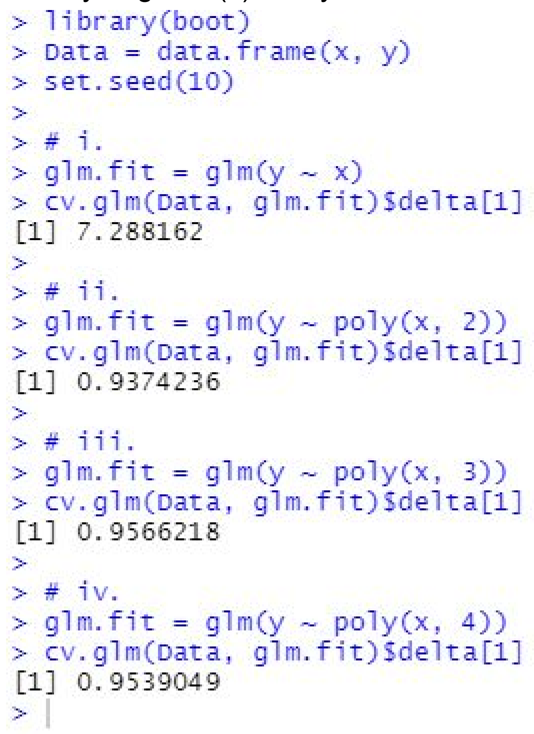
i. $Y = \beta_0 + \beta_1 X + \epsilon$
ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon.$

Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y



```
> library(boot)
> Data = data.frame(x, y)
> set.seed(1)
>
> # i.
> glm.fit = glm(y ~ x)
> cv.glm(Data, glm.fit)$delta[1]
[1] 7.288162
>
> # ii.
> glm.fit = glm(y ~ poly(x, 2))
> cv.glm(Data, glm.fit)$delta[1]
[1] 0.9374236
>
> # iii.
> glm.fit = glm(y ~ poly(x, 3))
> cv.glm(Data, glm.fit)$delta[1]
[1] 0.9566218
>
> # iv.
> glm.fit = glm(y ~ poly(x, 4))
> cv.glm(Data, glm.fit)$delta[1]
[1] 0.9539049
> |
```

d. Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?



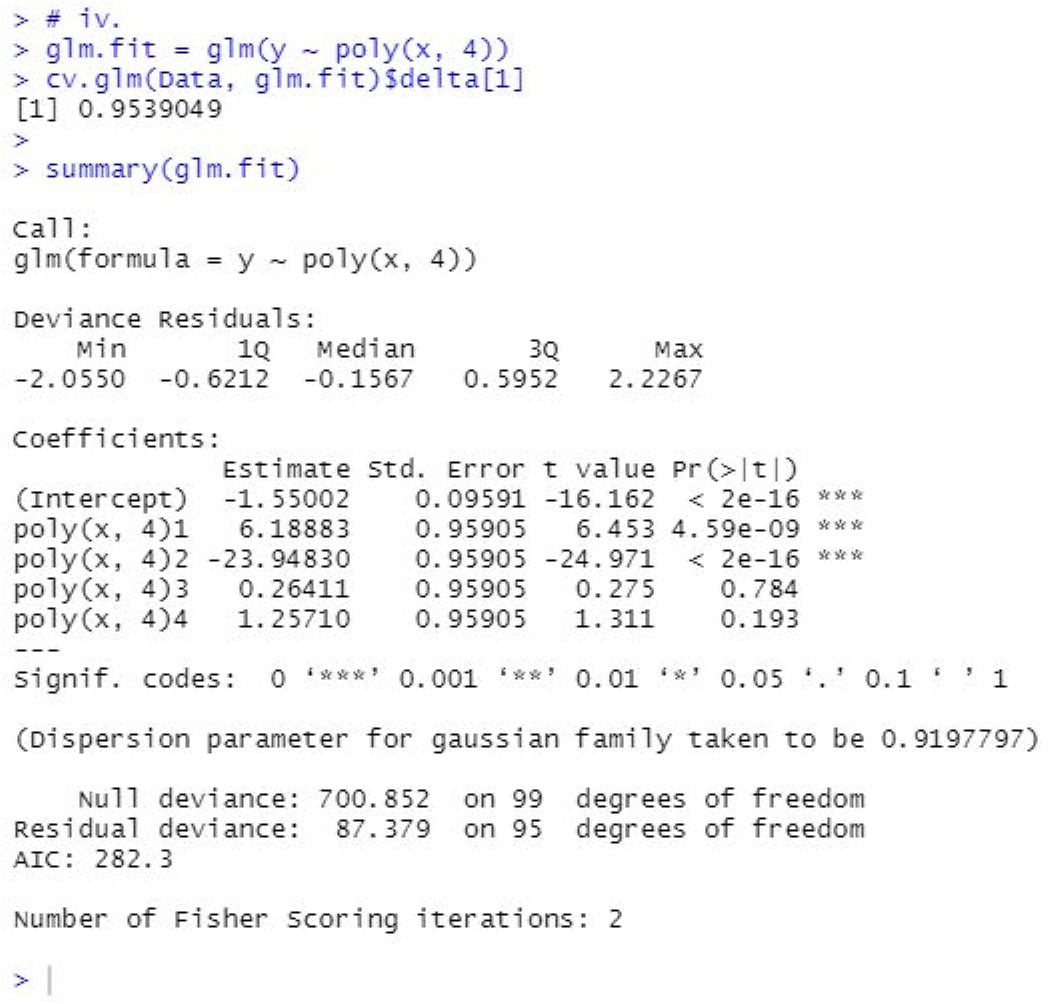
```
> library(boot)
> Data = data.frame(x, y)
> set.seed(10)
>
> # i.
> glm.fit = glm(y ~ x)
> cv.glm(Data, glm.fit)$delta[1]
[1] 7.288162
>
> # ii.
> glm.fit = glm(y ~ poly(x, 2))
> cv.glm(Data, glm.fit)$delta[1]
[1] 0.9374236
>
> # iii.
> glm.fit = glm(y ~ poly(x, 3))
> cv.glm(Data, glm.fit)$delta[1]
[1] 0.9566218
>
> # iv.
> glm.fit = glm(y ~ poly(x, 4))
> cv.glm(Data, glm.fit)$delta[1]
[1] 0.9539049
> |
```

The results are the same as (c) because LOOCV calculates n folds for one observation, regardless of the seed parameter.

e. Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The quadratic model performs the best because it has the smallest LOOCV error at 0.9374236. This is within expectations because a quadratic function was used to generate the data.

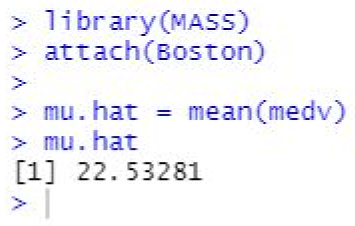
f. Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?



```
> # iv.
> glm.fit = glm(y ~ poly(x, 4))
> cv.glm(Data, glm.fit)$delta[1]
[1] 0.9539049
>
> summary(glm.fit)

Call:
glm(formula = y ~ poly(x, 4))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0550  -0.6212  -0.1567   0.5952   2.2267

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
poly(x, 4)3   0.26411    0.95905   0.275    0.784
poly(x, 4)4   1.25710    0.95905   1.311    0.193
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.9197797)

    Null deviance: 700.852  on 99  degrees of freedom
Residual deviance:  87.379  on 95  degrees of freedom
AIC: 282.3

Number of Fisher Scoring iterations: 2

> |
```

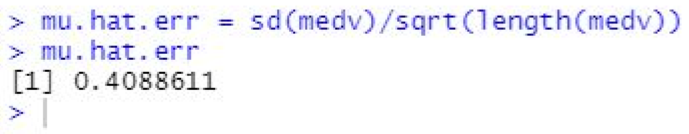
The p-values of the linear and quadratic terms are the only ones that are significant, which is expected because the quadratic model did the best, as seen in (d) and (e).

9. We will now consider the Boston housing data set, from the MASS library. (Ch.5, Question 9)
   a. Based on this data set, provide an estimate for the population mean of medv. Call this estimate ˆμ.



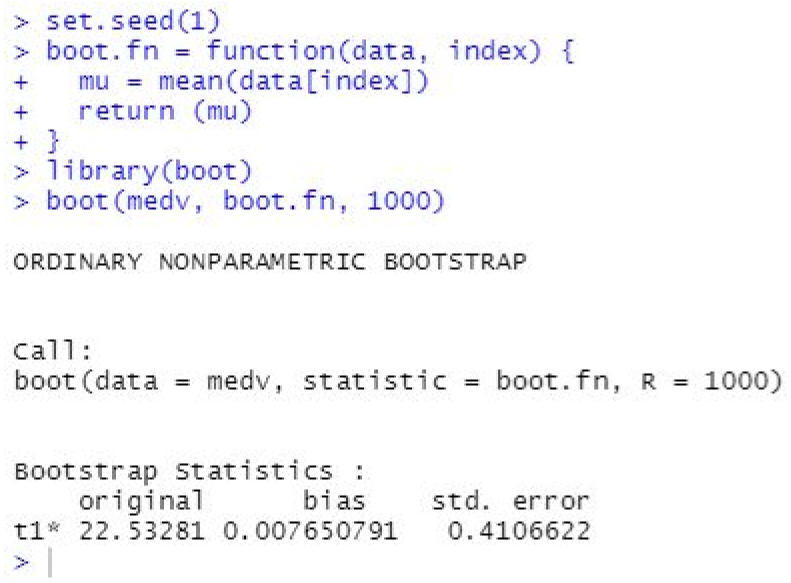
```
> library(MASS)
> attach(Boston)
>
> mu.hat = mean(medv)
> mu.hat
[1] 22.53281
> |
```

b. Provide an estimate of the standard error of ˆμ. Interpret this result. Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.



```
> mu.hat.err = sd(medv)/sqrt(length(medv))
> mu.hat.err
[1] 0.4088611
> |
```

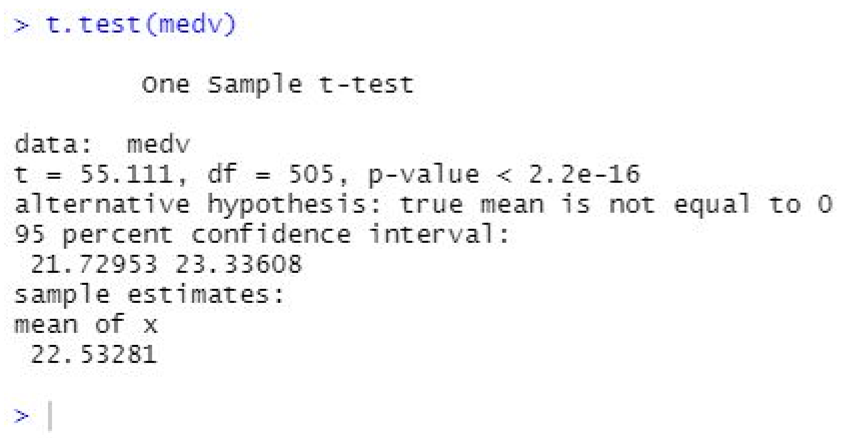
c. Now estimate the standard error of ˆμ using the bootstrap. How does this compare to your answer from (b)?



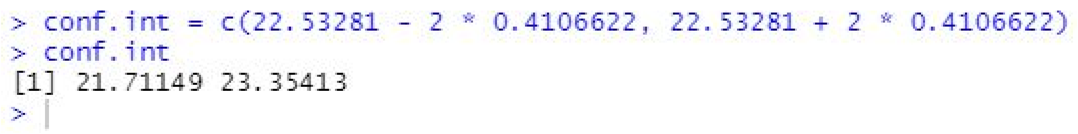
```
> set.seed(1)
> boot.fn = function(data, index) {
+   mu = mean(data[index])
+   return (mu)
+ }
> library(boot)
> boot(medv, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original      bias    std. error
t1* 22.53281 0.007650791   0.4106622
> |
```

The result from (b) is very similar to the estimate here using bootstrap. The two values are 0.4088611 versus 0.4106622.

d. Based on your bootstrap estimate from (c), provide a 95 % confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv). Hint: You can approximate a 95 % confidence interval using the formula [ˆμ − 2SE(ˆμ), μˆ + 2SE(ˆμ)].



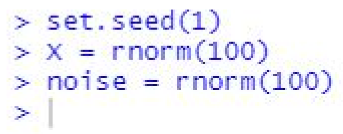
```
> t.test(medv)

        One Sample t-test

data:  medv
t = 55.111, df = 505, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.72953 23.33608
sample estimates:
mean of x 
 22.53281 

> 
```



```
> conf.int = c(22.53281 - 2 * 0.4106622, 22.53281 + 2 * 0.4106622)
> conf.int
[1] 21.71149 23.35413
> 
```

The confidence interval calculated using bootstrap is very close to the one obtained through t.test().

10. In this exercise, we will generate simulated data, and will then use this data to perform best subset selection. (Ch. 6, Question 8)
    a. Use the rnorm() function to generate a predictor X of length n = 100, as well as a noise vector of length n = 100.

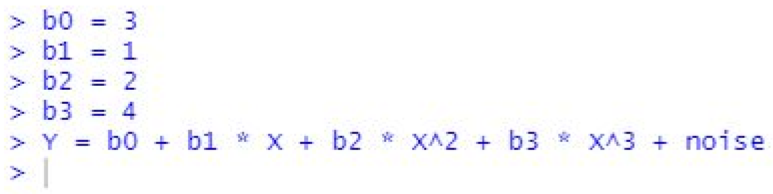


```
> set.seed(1)
> X = rnorm(100)
> noise = rnorm(100)
> 
```

    b. Generate a response vector Y of length n = 100 according to the model

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon,$$

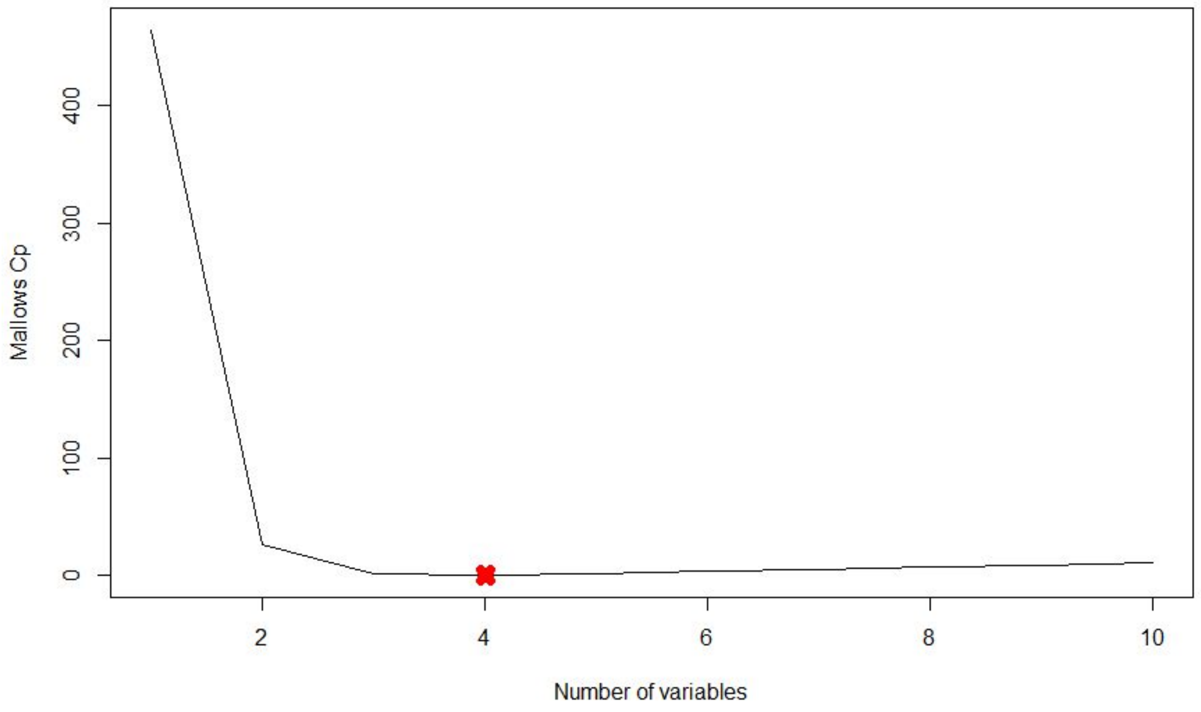
where β_0, β_1, β_2, and β_3 are constants of your choice.

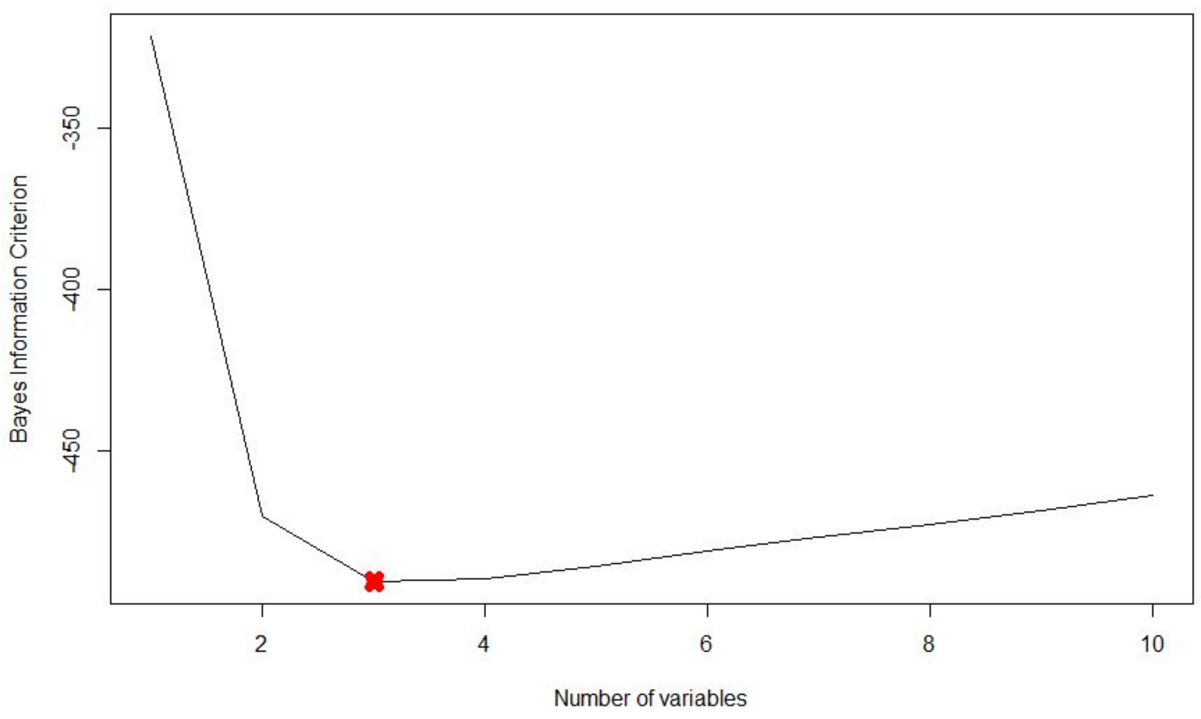


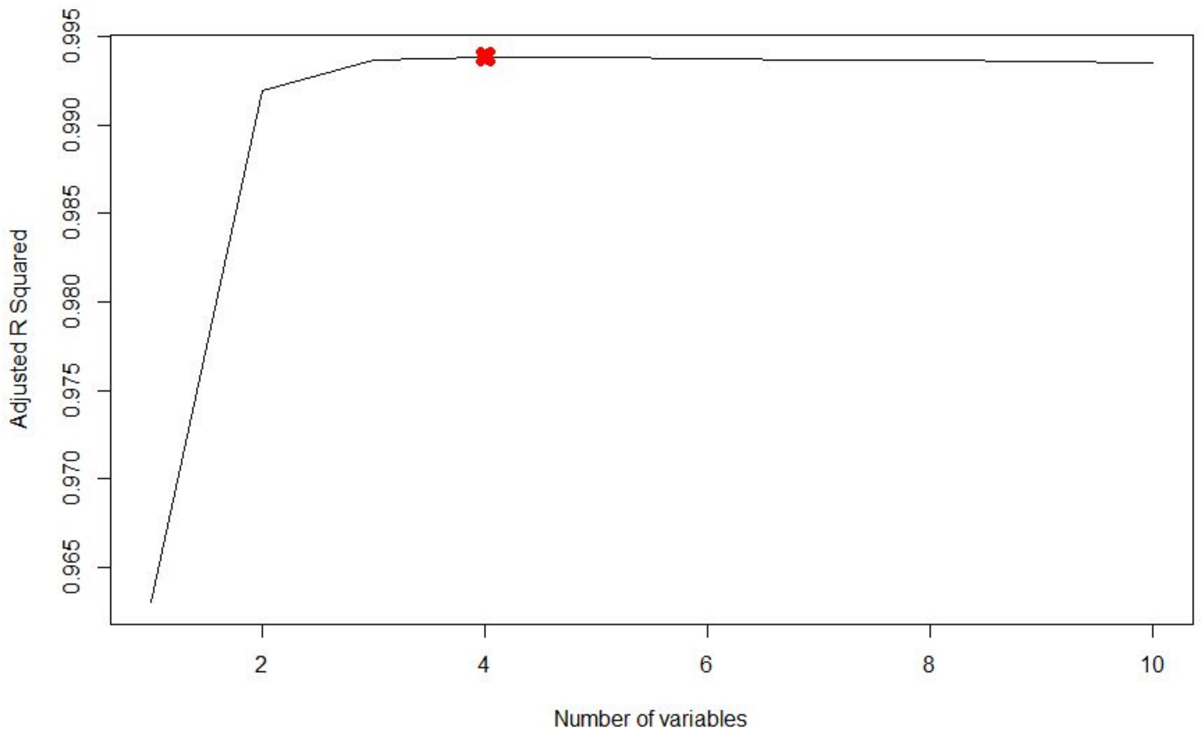
```
> b0 = 3
> b1 = 1
> b2 = 2
> b3 = 4
> Y = b0 + b1 * X + b2 * X^2 + b3 * X^3 + noise
> 
```

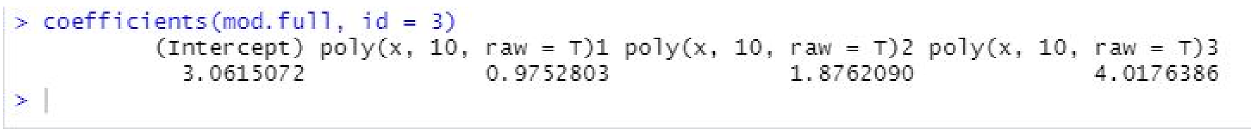
    c. Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors X, X2,...,X10. What is the best model obtained according to Cp, BIC, and adjusted R2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained.

Note you will need to use the data.frame() function to create a single data set containing both X and Y . 6.8 Exercises 263




Bayes Information Criterion
-350
-400
-450
2
4
6
8
10
Number of variables

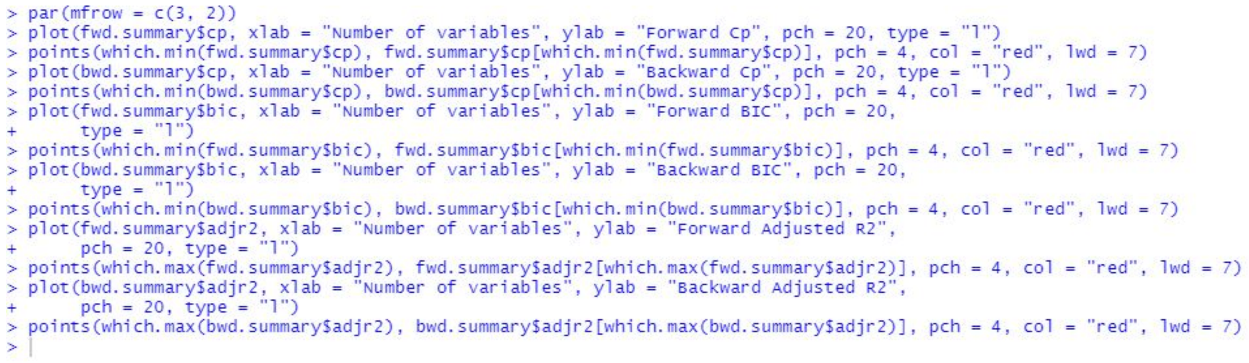

Adjusted R Squared
0.995
0.990
0.985
0.980
0.975
0.970
0.965
2
4
6
8
10
Number of variables



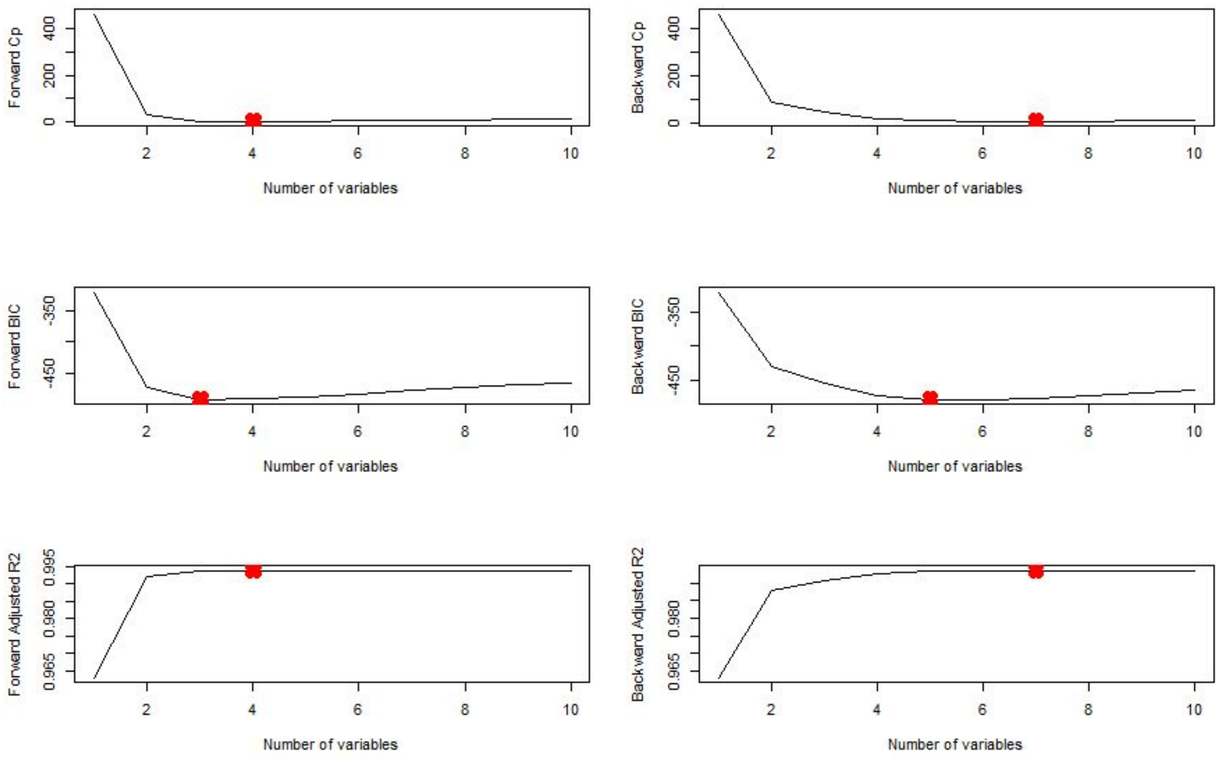
```
> coefficients(mod.full, id = 3)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)3
          3.0615072             0.9752803             1.8762090             4.0176386
> |
```

According to Cp, BIC, and adjusted R2, the best model is X3.

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

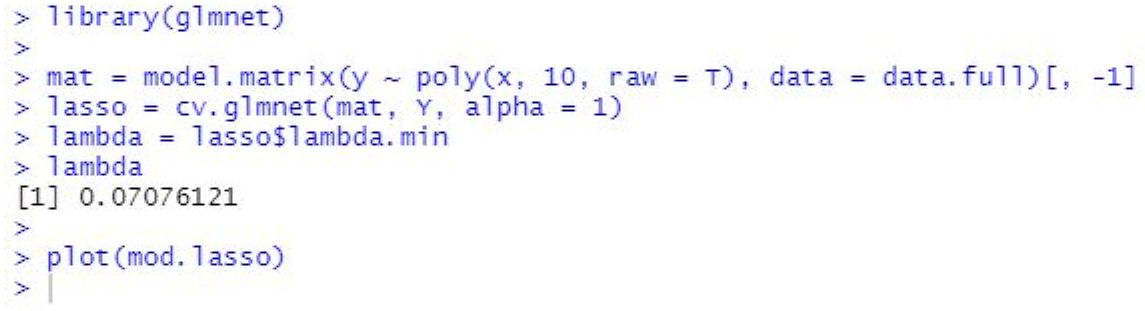


```
> par(mfrow = c(3, 2))
> plot(fwd.summary$cp, xlab = "Number of variables", ylab = "Forward Cp", pch = 20, type = "l")
> points(which.min(fwd.summary$cp), fwd.summary$cp[which.min(fwd.summary$cp)], pch = 4, col = "red", lwd = 7)
> plot(bwd.summary$cp, xlab = "Number of variables", ylab = "Backward Cp", pch = 20, type = "l")
> points(which.min(bwd.summary$cp), bwd.summary$cp[which.min(bwd.summary$cp)], pch = 4, col = "red", lwd = 7)
> plot(fwd.summary$bic, xlab = "Number of variables", ylab = "Forward BIC", pch = 20,
+      type = "l")
> points(which.min(fwd.summary$bic), fwd.summary$bic[which.min(fwd.summary$bic)], pch = 4, col = "red", lwd = 7)
> plot(bwd.summary$bic, xlab = "Number of variables", ylab = "Backward BIC", pch = 20,
+      type = "l")
> points(which.min(bwd.summary$bic), bwd.summary$bic[which.min(bwd.summary$bic)], pch = 4, col = "red", lwd = 7)
> plot(fwd.summary$adjr2, xlab = "Number of variables", ylab = "Forward Adjusted R2",
+      pch = 20, type = "l")
> points(which.max(fwd.summary$adjr2), fwd.summary$adjr2[which.max(fwd.summary$adjr2)], pch = 4, col = "red", lwd = 7)
> plot(bwd.summary$adjr2, xlab = "Number of variables", ylab = "Backward Adjusted R2",
+      pch = 20, type = "l")
> points(which.max(bwd.summary$adjr2), bwd.summary$adjr2[which.max(bwd.summary$adjr2)], pch = 4, col = "red", lwd = 7)
> |
```

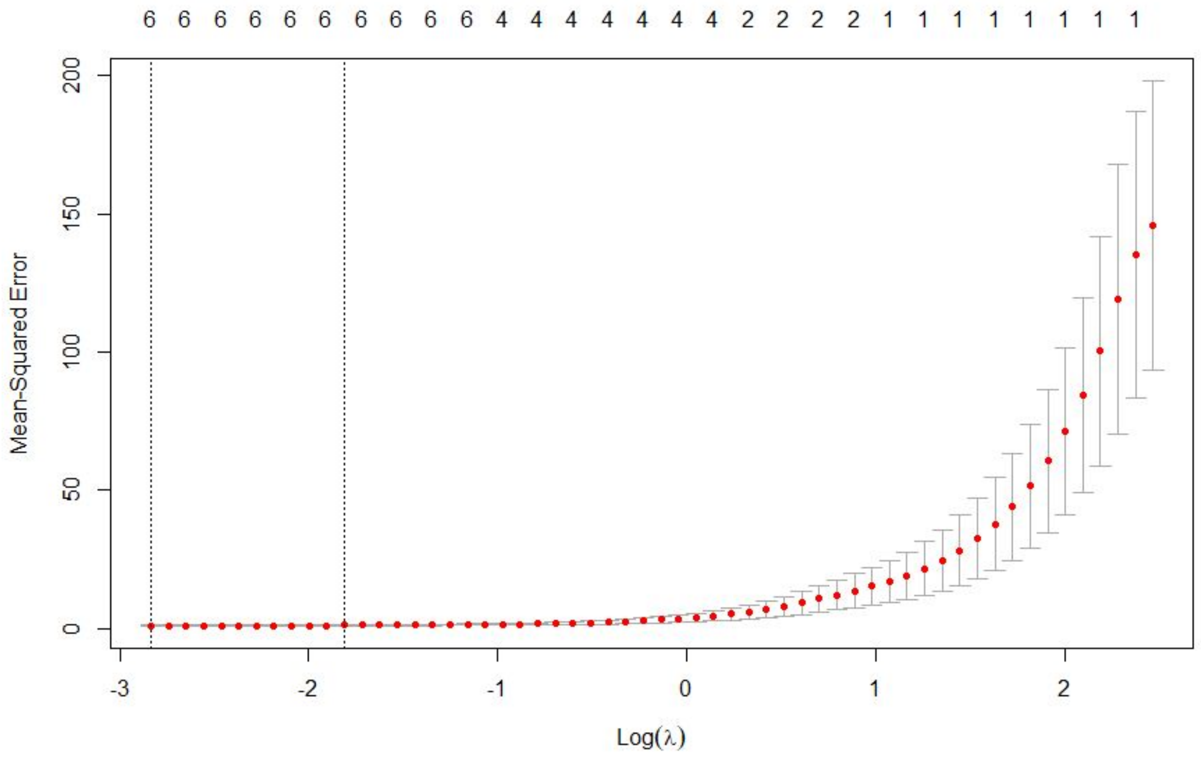


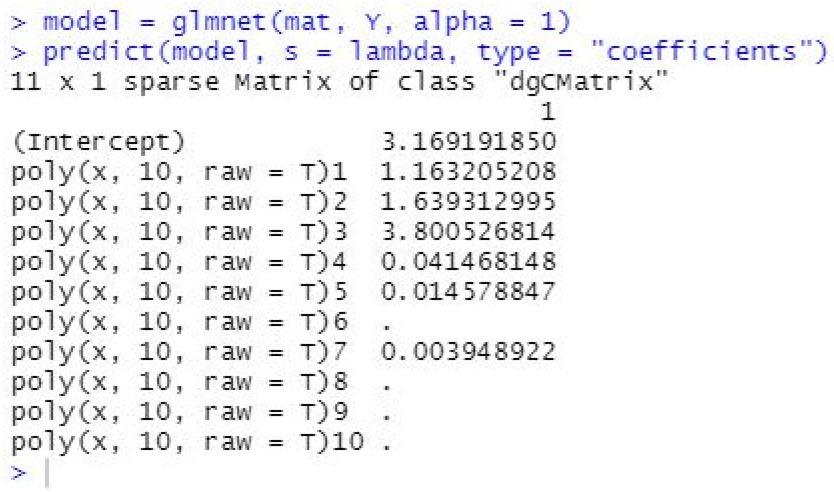
Compared to the results in (c), both the forward stepwise and backward stepwise selection results are very similar for Cp, BIC, and adjusted R2.

e. Now fit a lasso model to the simulated data, again using X, X2, ...,X10 as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.



```
> library(glmnet)
>
> mat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]
> lasso = cv.glmnet(mat, Y, alpha = 1)
> lambda = lasso$lambda.min
> lambda
[1] 0.07076121
>
> plot(mod.lasso)
>
```





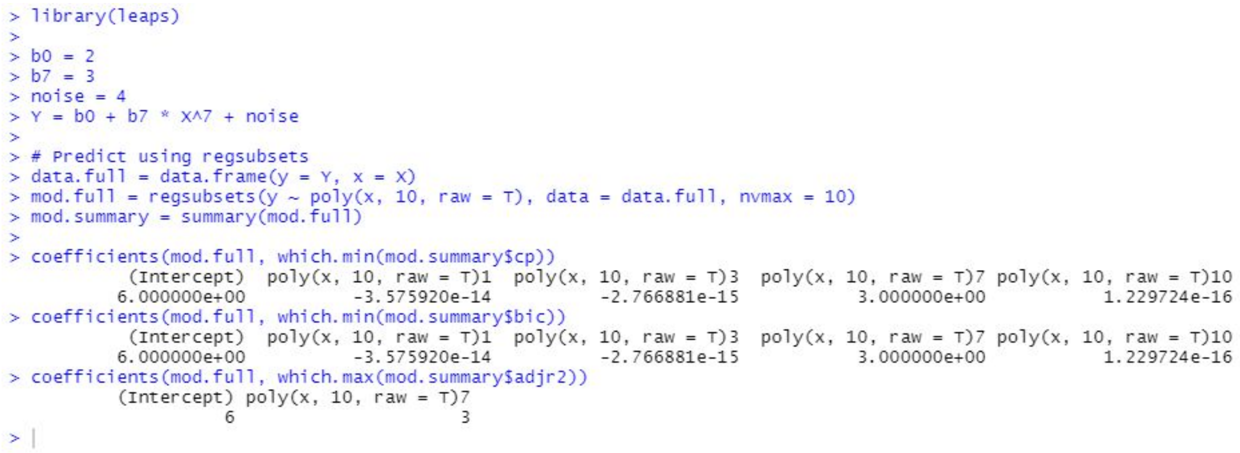
```
> model = glmnet(mat, Y, alpha = 1)
> predict(model, s = lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                                 1
(Intercept)           3.169191850
poly(x, 10, raw = T)1  1.163205208
poly(x, 10, raw = T)2  1.639312995
poly(x, 10, raw = T)3  3.800526814
poly(x, 10, raw = T)4  0.041468148
poly(x, 10, raw = T)5  0.014578847
poly(x, 10, raw = T)6  .
poly(x, 10, raw = T)7  0.003948922
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  .
poly(x, 10, raw = T)10 .
> |
```

The results show that the lasso model would select X5 and X7.

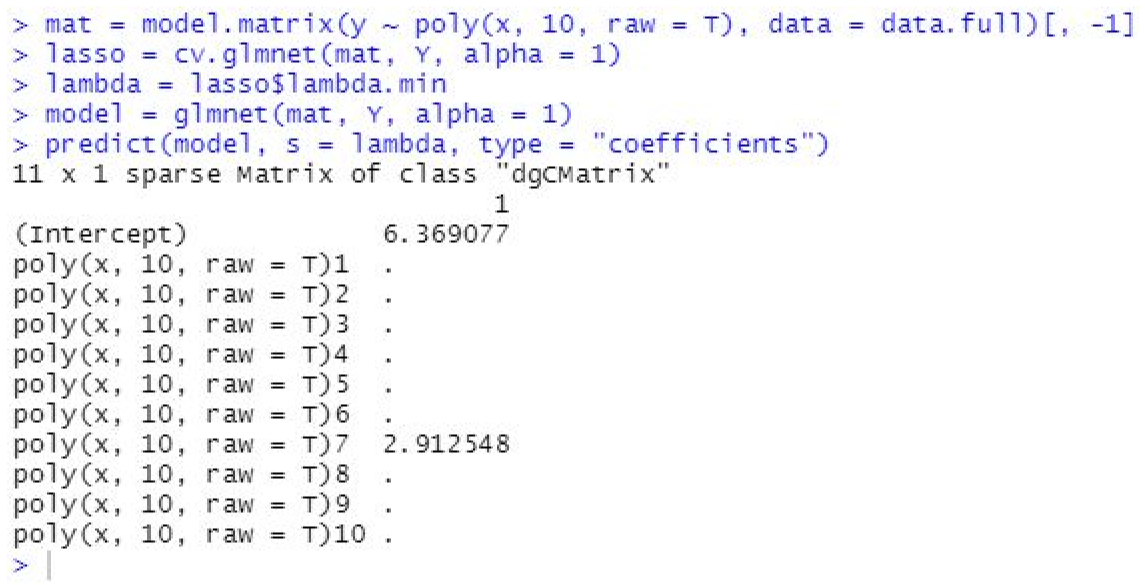
f. Now generate a response vector Y according to the model

$$Y = \beta_0 + \beta_7 X^7 + \epsilon,$$

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



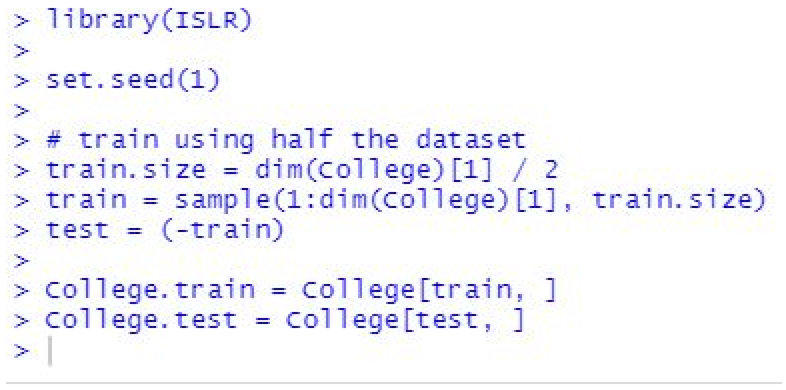
```
> library(leaps)
>
> b0 = 2
> b7 = 3
> noise = 4
> Y = b0 + b7 * X^7 + noise
>
> # Predict using regsubsets
> data.full = data.frame(y = Y, x = X)
> mod.full = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10)
> mod.summary = summary(mod.full)
>
> coefficients(mod.full, which.min(mod.summary$cp))
          (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)3  poly(x, 10, raw = T)7 poly(x, 10, raw = T)10
         6.000000e+00          -3.575920e-14          -2.766881e-15           3.000000e+00           1.229724e-16
> coefficients(mod.full, which.min(mod.summary$bic))
          (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)3  poly(x, 10, raw = T)7 poly(x, 10, raw = T)10
         6.000000e+00          -3.575920e-14          -2.766881e-15           3.000000e+00           1.229724e-16
> coefficients(mod.full, which.max(mod.summary$adjr2))
         (Intercept) poly(x, 10, raw = T)7
                   6                     3
> |
```



```
> mat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]
> lasso = cv.glmnet(mat, Y, alpha = 1)
> lambda = lasso$lambda.min
> model = glmnet(mat, Y, alpha = 1)
> predict(model, s = lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                               1
(Intercept)            6.369077
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  2.912548
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  .
poly(x, 10, raw = T)10 .
> |
```

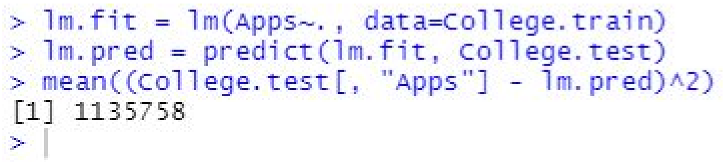
The first snippet shows an implementation best subset selection, while the second shows an implementation of lasso. The results show that the best subset selection had adjusted R squared select the most accurate model with one variable for X7, and lasso also returned the similar results with a model for X7. The intercept is a little bit different where best subset selection returned 6 and lasso returned 6.37.

11. In this exercise, we will predict the number of applications received using the other variables in the College data set. (Ch. 6, Question 9)
    a. Split the data set into a training set and a test set.



```
> library(ISLR)
>
> set.seed(1)
>
> # train using half the dataset
> train.size = dim(College)[1] / 2
> train = sample(1:dim(College)[1], train.size)
> test = (-train)
>
> College.train = College[train, ]
> College.test = College[test, ]
> |
```

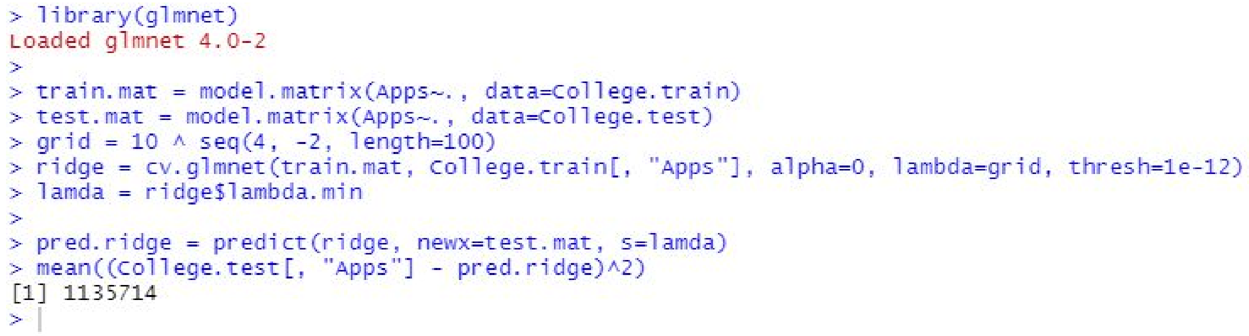
    b. Fit a linear model using least squares on the training set, and report the test error obtained.



```
> lm.fit = lm(Apps~., data=College.train)
> lm.pred = predict(lm.fit, College.test)
> mean((College.test[, "Apps"] - lm.pred)^2)
[1] 1135758
> |
```

The mean squared error for the linear model is 1.135758 * 10^6.

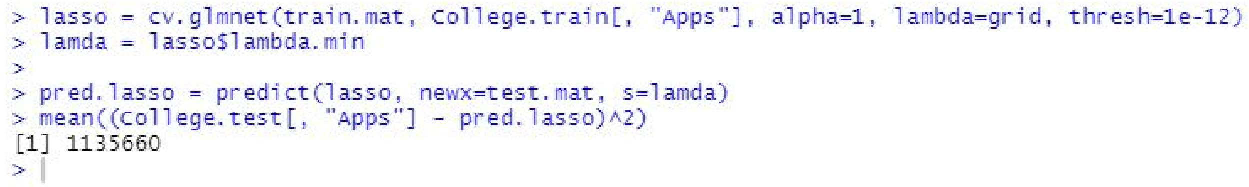
c. Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.



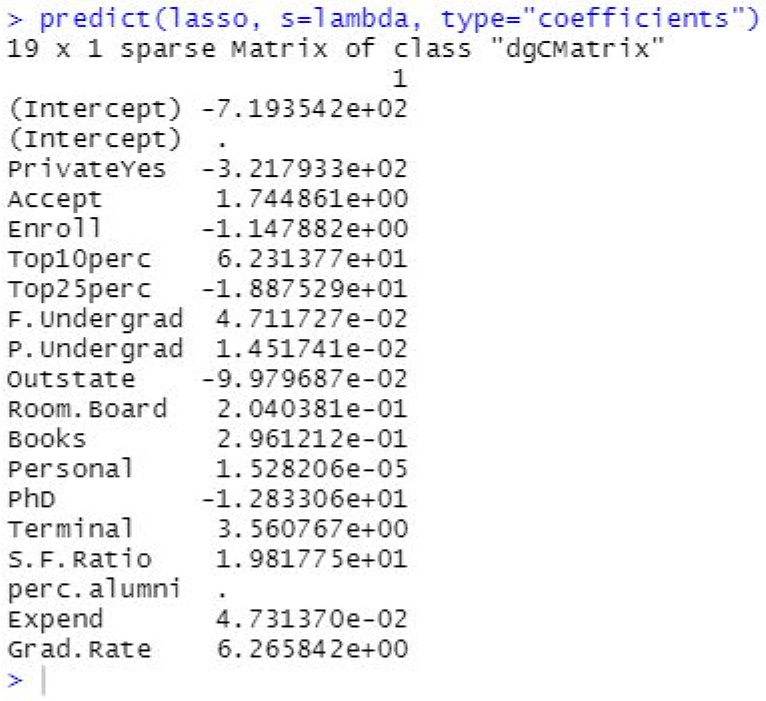
```
> library(glmnet)
Loaded glmnet 4.0-2
>
> train.mat = model.matrix(Apps~., data=College.train)
> test.mat = model.matrix(Apps~., data=College.test)
> grid = 10 ^ seq(4, -2, length=100)
> ridge = cv.glmnet(train.mat, College.train[, "Apps"], alpha=0, lambda=grid, thresh=1e-12)
> lamda = ridge$lambda.min
>
> pred.ridge = predict(ridge, newx=test.mat, s=lamda)
> mean((College.test[, "Apps"] - pred.ridge)^2)
[1] 1135714
> |
```

The test error for the ridge regression model is 1.135714 * 10^6, which is very close to the error recorded in (b).

d. Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.



```
> lasso = cv.glmnet(train.mat, College.train[, "Apps"], alpha=1, lambda=grid, thresh=1e-12)
> lamda = lasso$lambda.min
>
> pred.lasso = predict(lasso, newx=test.mat, s=lamda)
> mean((College.test[, "Apps"] - pred.lasso)^2)
[1] 1135660
> |
```



```
> predict(lasso, s=lambda, type="coefficients")
19 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept) -7.193542e+02
(Intercept)  .
PrivateYes  -3.217933e+02
Accept       1.744861e+00
Enroll      -1.147882e+00
Top10perc    6.231377e+01
Top25perc   -1.887529e+01
F.Undergrad  4.711727e-02
P.Undergrad  1.451741e-02
Outstate    -9.979687e-02
Room.Board   2.040381e-01
Books        2.961212e-01
Personal     1.528206e-05
PhD         -1.283306e+01
Terminal     3.560767e+00
S.F.Ratio    1.981775e+01
perc.alumni  .
Expend       4.731370e-02
Grad.Rate    6.265842e+00
> |
```

The test error of the lasso model is 1.135660 * 10^6, and the non-zero coefficient estimates are shown above.