ML-HW2

Vincent Jin

2023-02-14

Statistical Machine Learning - Homework 2

Vincent Jin

Person I worked with: Jing Wang

ISL 5.4 - 1 3 5 6 ISL 6.8 - 1 8 9

Resampling Methods

Problem 5.4 - 1

Using basic statistical properties of the variance, as well as single variable calculus, derive (5.6). In other words, prove that α given by (5.6) does indeed minimize $Var(\alpha X + (1-\alpha)Y)$.

Answer By plug in α, we can get:

$$\begin{aligned} Var(\alpha X + (1-\alpha)Y) &= Var(\alpha X) + Var((1-\alpha)Y) + 2Cov(\alpha X, (1-\alpha)Y) \\ &= \alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(X,Y) \\ &= \sigma_X^2\alpha^2 + \sigma_Y^2(1-\alpha)2 + 2\sigma_{XY}(-\alpha^2+\alpha) \end{aligned}$$

Then take a deriviation of the formula, and set to 0 since we want to minimize $Var(\alpha X + (1-\alpha)Y)$:

$$\begin{aligned} 0 &= \frac{d}{d_\alpha}(\sigma_X^2\alpha^2 + \sigma_Y^2(1-\alpha)2 + 2\sigma_{XY}(-\alpha^2+\alpha)) \\ &= 2\sigma_X^2\alpha + 2\sigma_Y^2(1-\alpha)(-1) + 2\sigma_X Y(-2\alpha+1) \\ &= (\sigma_X^2 + \sigma_Y^2 - 2\sigma_X Y)\alpha - \sigma_Y^2 + \sigma_X Y \\ &= \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}} \end{aligned}$$

Problem 5.4-3

We now review k-fold cross-validation. ### (a) Explain how k-fold cross-validation is implemented.

Answer

The k-fold cross-validation simply means that we randomly divide the dataset into k different subsets, and use each one of the k subsets as a validation set while using all other subsets as training sets.

(b) What are the advantages and disadvantages of k-fold crossvalidation relative to:

i. The validation set approach?

Answer

The validation is simply and easy to understand and implemented. However, as compared to k-fold cross-validation, the main drawbacks of validation set approach includes: 1. the test error rate may be highly variable depends on the observations being included in the training and test sets. 2. the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set.

ii. LOOCV?

Answer

As a special case of k-fold cross-validation, instead of splitting the dataset into k groups with length of n/k, in LOOCV approach, the dataset is splitted into k=n groups. This means that each one observation will be used as a validation set while all other observations being used as training set. This approach is most computational extensive and yield lower bias but higher variable results as compared to k-fold cross-validation.

Problem 5.4-5

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.

(a) Fit a logistic regression model that uses income and balance to predict default.

```
library(ISLR)
set.seed(1)
reg1 <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(reg1)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(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.

```
training <- sample(dim(Default)[1], dim(Default)[1]/2)
```

ii. Fit a multiple logistic regression model using only the training observations.

```
reg2 <- glm(default~income+balance, data = Default, family = "binomial", subset = training)
summary(reg2)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default, subset = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5830  -0.1428  -0.0573  -0.0213   3.3395
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.194e+01  6.178e-01 -19.333  < 2e-16 ***
## income       3.262e-05  7.024e-06   4.644 3.41e-06 ***
## balance      5.689e-03  3.158e-04  18.014  < 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: 1523.8  on 4999  degrees of freedom
## Residual deviance:  803.3  on 4997  degrees of freedom
## AIC: 809.3
##
## Number of Fisher Scoring iterations: 8
```

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.

```
reg2.prob <- predict(reg2, newdata = Default[-training, ], type = "response")
reg2.pred <- rep("No", length(reg2.prob))
reg2.pred[reg2.prob > 0.5] = "Yes"
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
mean(reg2.pred != Default[-training, ]$default)
```

```
## [1] 0.0254
```

Answer

The fraction of the observation in the validation set that are missclassified is 2.54%.

(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.

```
reg_result <- list()
per_result <- list()
seed <- c(12, 20, 25)

for (i in seed) {
  set.seed(i)
  training_loop <- sample(dim(Default)[1], dim(Default)[1] / 2)
  reg_temp <- glm(default ~ income + balance, data = Default, family = "binomial", subset = training_loo
  reg_result <- append(reg_result, reg_temp)
  prob_temp <- predict(reg_temp, newdata = Default[-training_loop, ], type = "response")
  pred_temp <- rep("No", length(prob_temp))
  pred_temp[prob_temp > 0.5] <- "Yes"
  mean_temp <- mean(pred_temp != Default[-training_loop, ]$default)
  per_result <- append(per_result, mean_temp * 100)
}
```

Answer

The results from three tries demonstrated that with different observations being included in different training/validation sets, the test error rate may be different.

(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.

```
set.seed(1)
training_dummy <- sample(dim(Default)[1], dim(Default)[1] / 2)
reg3 <- glm(default ~ income+balance+student, data = Default, family = "binomial", subset = training_du
reg3_prob <- predict(reg3, newdata = Default[-training_dummy, ], type = "response")
reg3_pred <- rep("No", length(reg3_prob))
reg3_pred[reg3_prob > 0.5] <- "Yes"
mean(reg3_pred != Default[-training_dummy, ]$default)
```

```
## [1] 0.026
```

Answer

Setting up a dummy variable did not lead to reductions in the test error rate.

Problem 5.4-6

We continue to consider the use of a logistic regression model to predict the probability of default using income and balance on the Default data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.

(a) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.

```
reg4 <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(reg4)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

Answer

Based on the regression results, we can see that the standard error for income as predictor is 4.985×10^{-6} and for balance as predictor is 2.274×10^{-4}.

(b) Write a function, boot.fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

```
boot.fn <- function(data, index) {
  fit.glm <- glm(default ~ income + balance, data = data, family = "binomial", subset = index)
  return(coef(fit.glm))
}
```

(c) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for income and balance.

```
set.seed(1)
library(boot)
boot(Default, boot.fn, 100)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 100)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
## t1* -1.154047e+01  8.556378e-03 4.122015e-01
## t2*  2.080898e-05 -3.993598e-07 4.186088e-06
## t3*  5.647103e-03 -4.116657e-06 2.226242e-04
```

Answer

The results from bootstrap suggesting that the standard error for coefficients is 4.19×10^{-6} for income and 2.23×10^{-4} for balance.

(d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

Answer

Both logistic and bootstrap function provided similar level of precision as indicated by very minor differences among standard error of coefficients. However, the bootstrap tended to provide better estimations with slightly smaller standard errors.

ISL 6.8 - 1, 8, 9

Question 6.8 - 1

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

The models of best subset with k predictors has the smallest training RSS. For the other 2 models, in forward stepwise, the smallest RSS will be the model that augment the predictors in M_{k-1} with one additional predictor, and for backward stepwise it will be M_{k+1}, so it should be the best subset.

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

Answer

Smallest test RSS may be from any of the three models. Although best selection may provide the smaller test RSS because it takes into account more models than the other methods, other models may be able to have smallest test RSS by chance.

(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 back- ward 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 back- ward 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

Question 6.8 - 8

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(1)
x <- rnorm(100)
eps <- rnorm(100)
```

(b) Generate a response vector Y of length n = 100 according to the model Y = b0 +b1X +b2X^2 +b3X^3 + eps, where b0, b1, b2, and b3 are constants of your choice.

```
b0 <- 5
b1 <- -5
b2 <- 10
b3 <- -10
y <- b0 + b1 * x + b2 * x^2 + b3 * x^3 + eps
```

(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 .

```
# install.packages("leaps")

library(leaps)
data <- data.frame(y = y, x = x)
reg <- regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10)
regsum <- summary(reg)
which.min(regsum$cp)
```

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

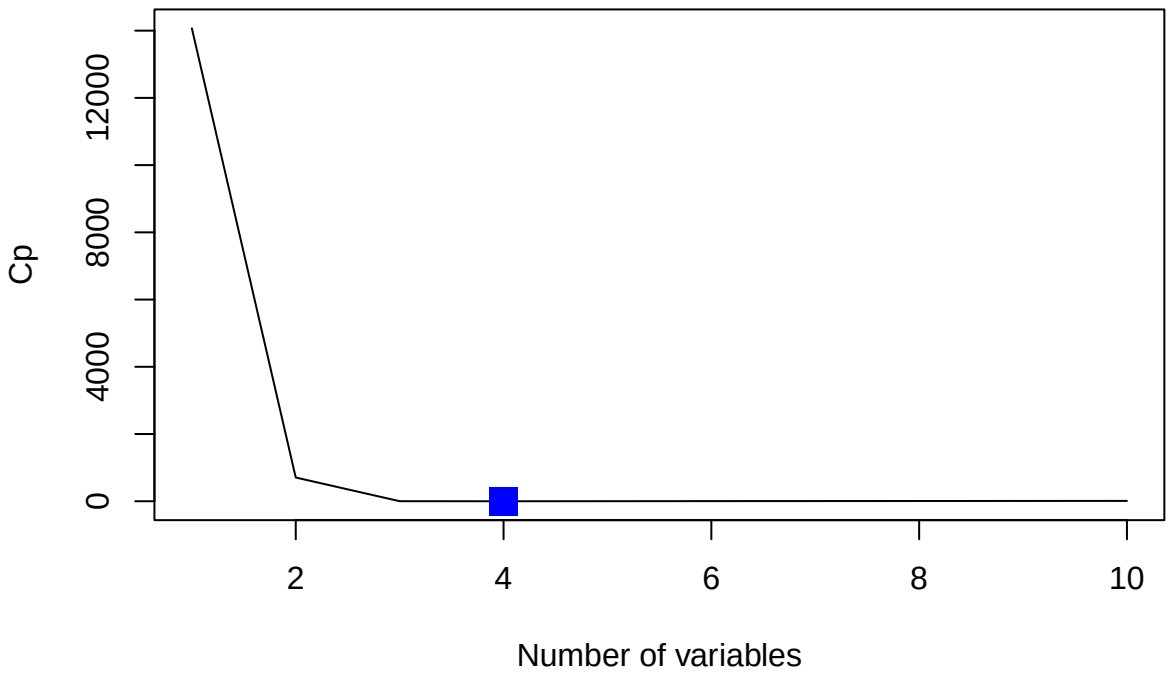
```
which.min(regsum$bic)
```

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

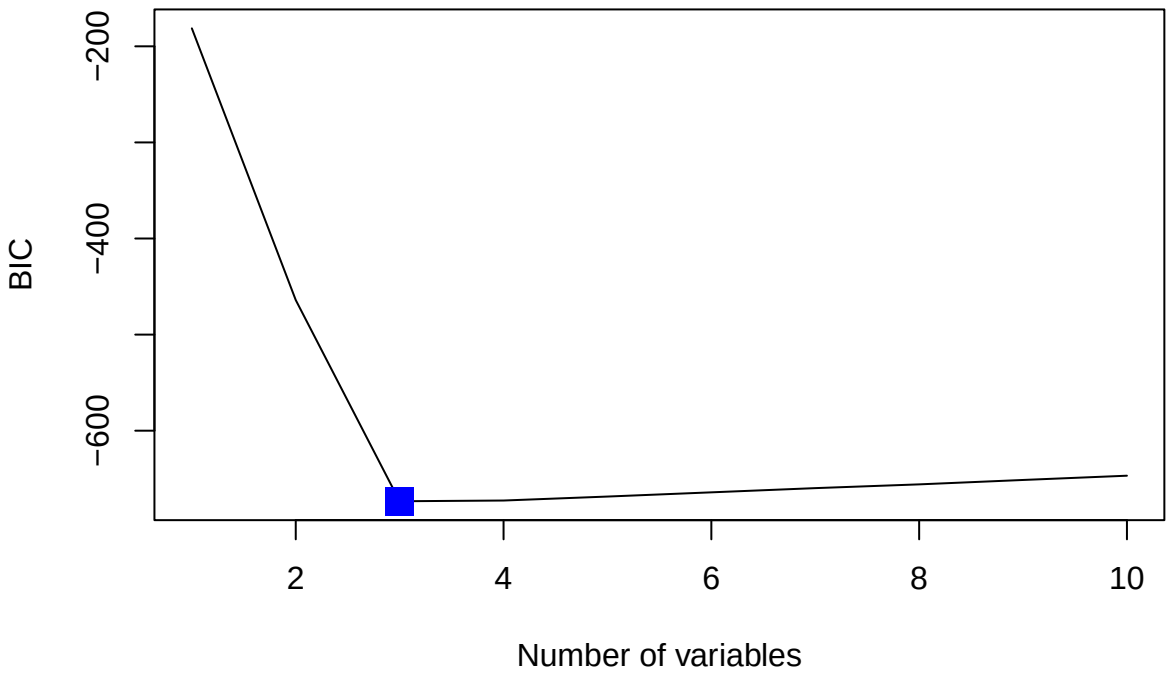
```
which.max(regsum$adjr2)
```

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

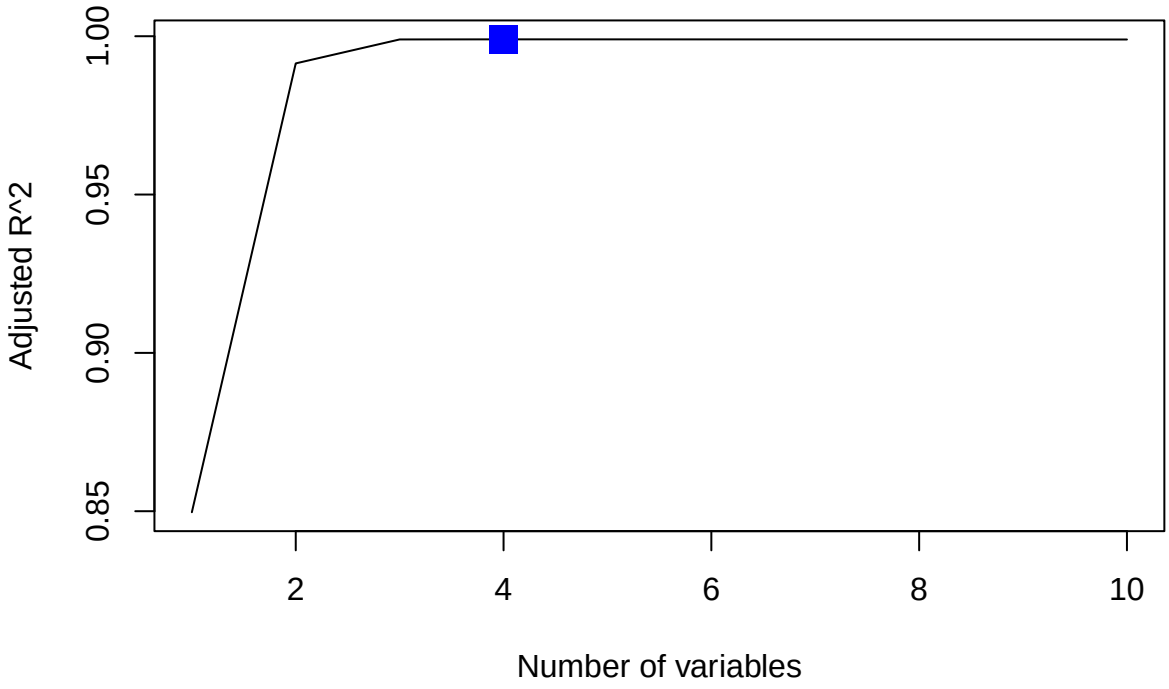
```
plot(regsum$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(4, regsum$cp[which.min(regsum$cp)], col = "blue", cex = 2, pch = 15)
```




```
plot(regsum$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(3, regsum$bic[which.min(regsum$bic)], col = "blue", cex = 2, pch = 15)
```




```
plot(regsum$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(4, regsum$adjr2[which.max(regsum$adjr2)], col = "blue", cex = 2, pch = 15)
```




```
coef(reg,which.min(regsum$cp))
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            5.07200775           -4.61254404            9.84575641
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)5
##          -10.44202574            0.08072292
```

```
coef(reg, which.min(regsum$bic))
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##              5.061507             -5.024720              9.876209
## poly(x, 10, raw = T)3
##             -9.982361
```

```
coef(reg, which.max(regsum$adjr2))
```

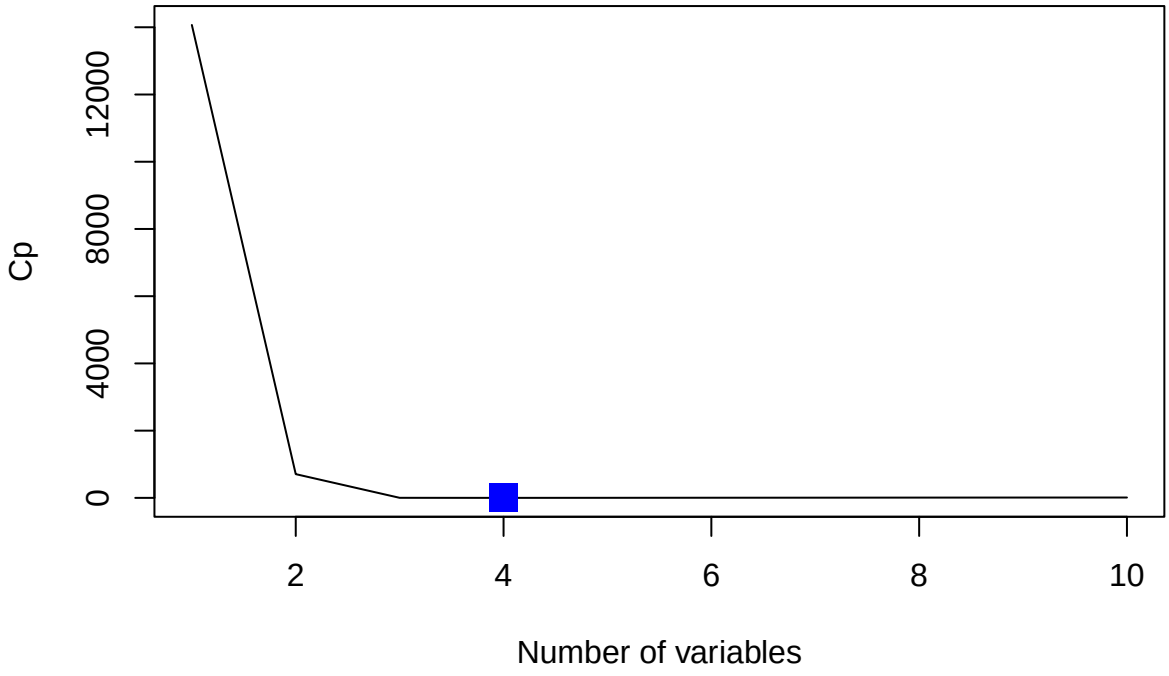
```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            5.07200775           -4.61254404            9.84575641
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)5
##          -10.44202574            0.08072292
```

Answer

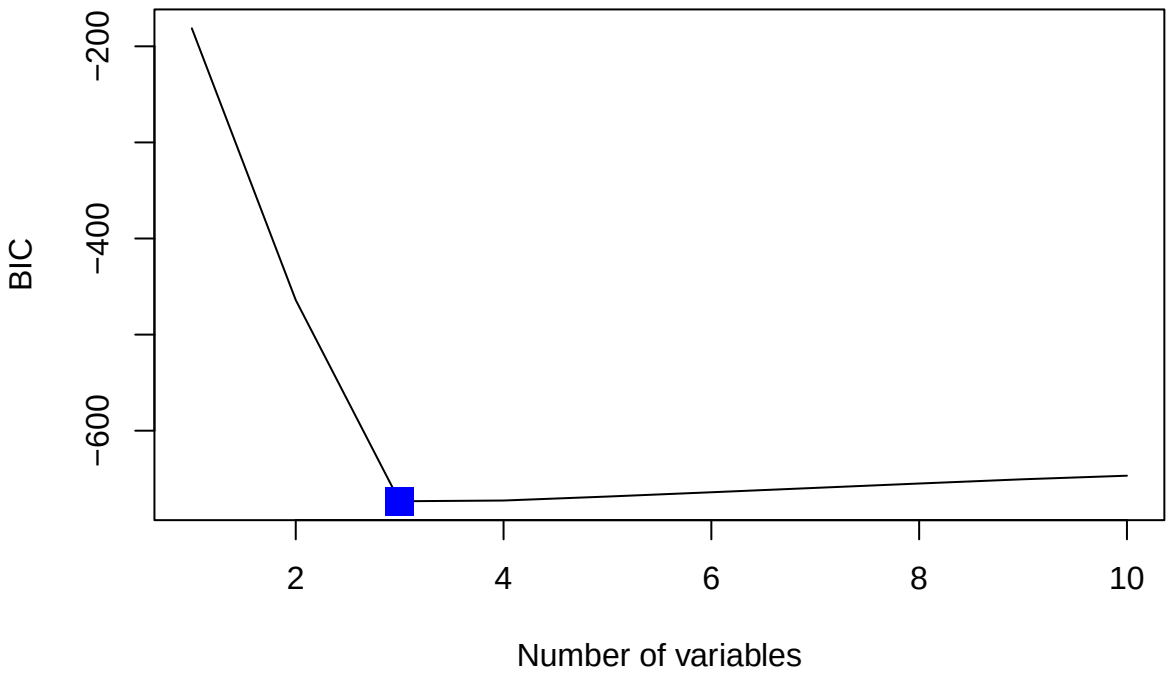
The results have suggested that the 4-variables model was picked in Cp, coefs are: 5.07, -4.61, 9.85, -10.44; the 3-variables model was pciked with BIC, coefficients are: 5.06, -5.02, 9.88, -8.98; the 4-variables model with adjusted R^2, coefficients are: 5.07, -4.61, 9.85, -10.44.

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

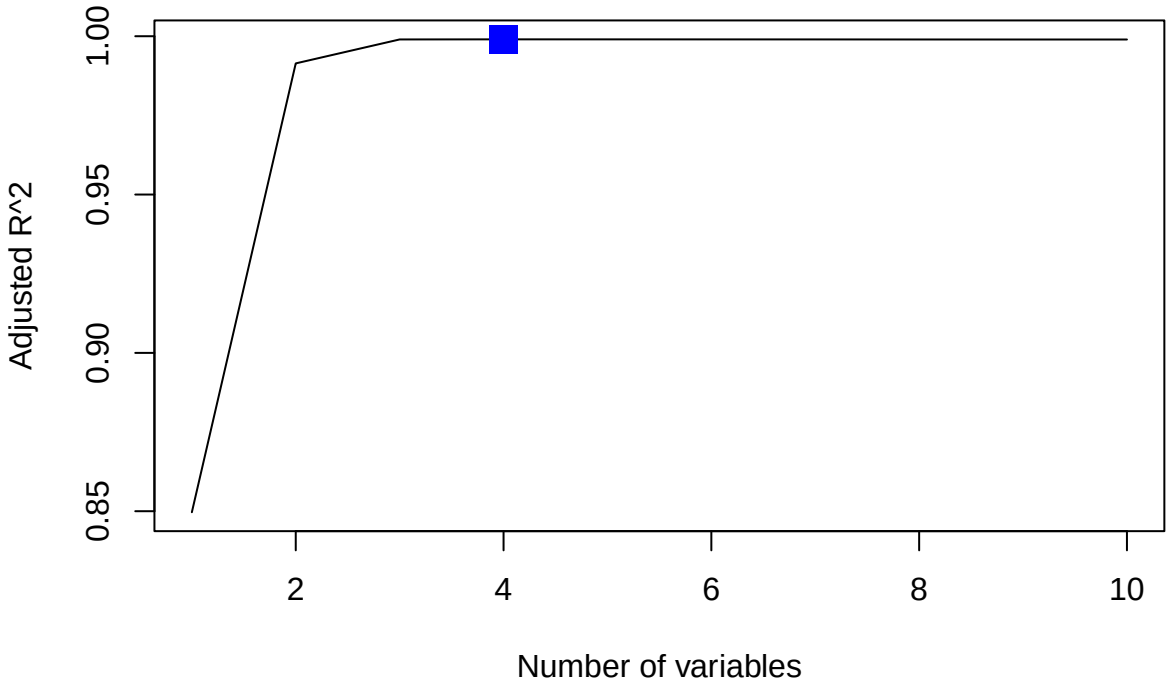
```
fwd.data <- regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10, method = "forward")
sumfwd <- summary(fwd.data)

plot(sumfwd$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(which.min(sumfwd$cp), sumfwd$cp[which.min(sumfwd$cp)], col = "blue", cex = 2, pch = 15)
```



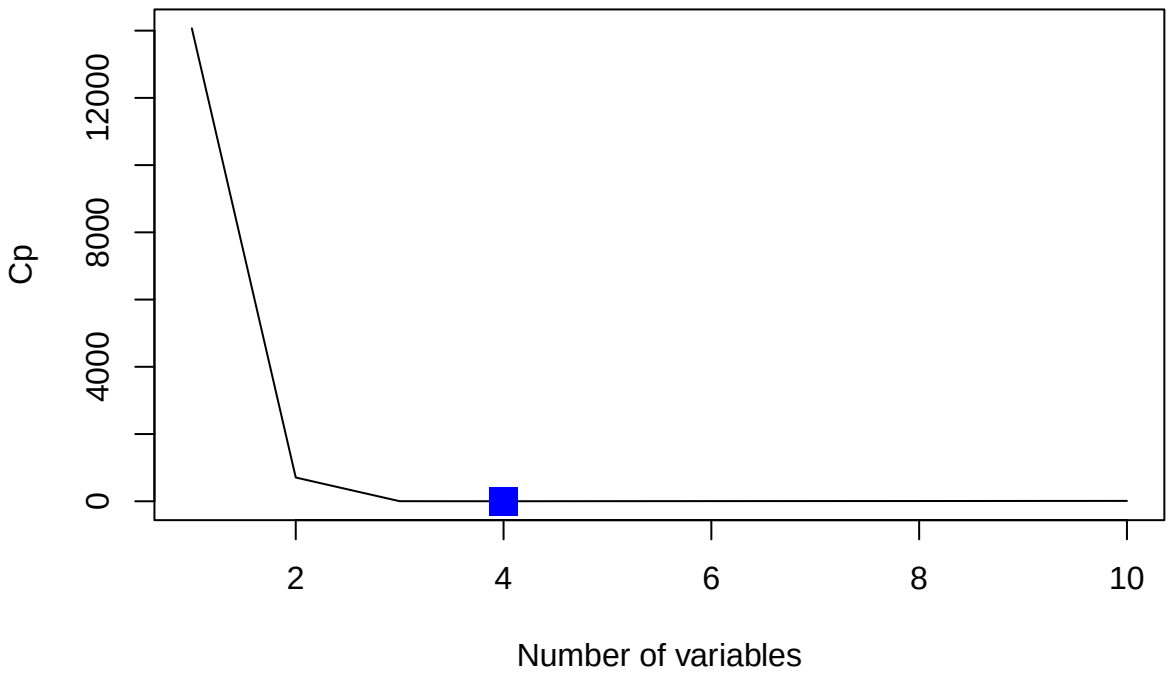

```
plot(sumfwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(sumfwd$bic), sumfwd$bic[which.min(sumfwd$bic)], col = "blue", cex = 2, pch = 15)
```



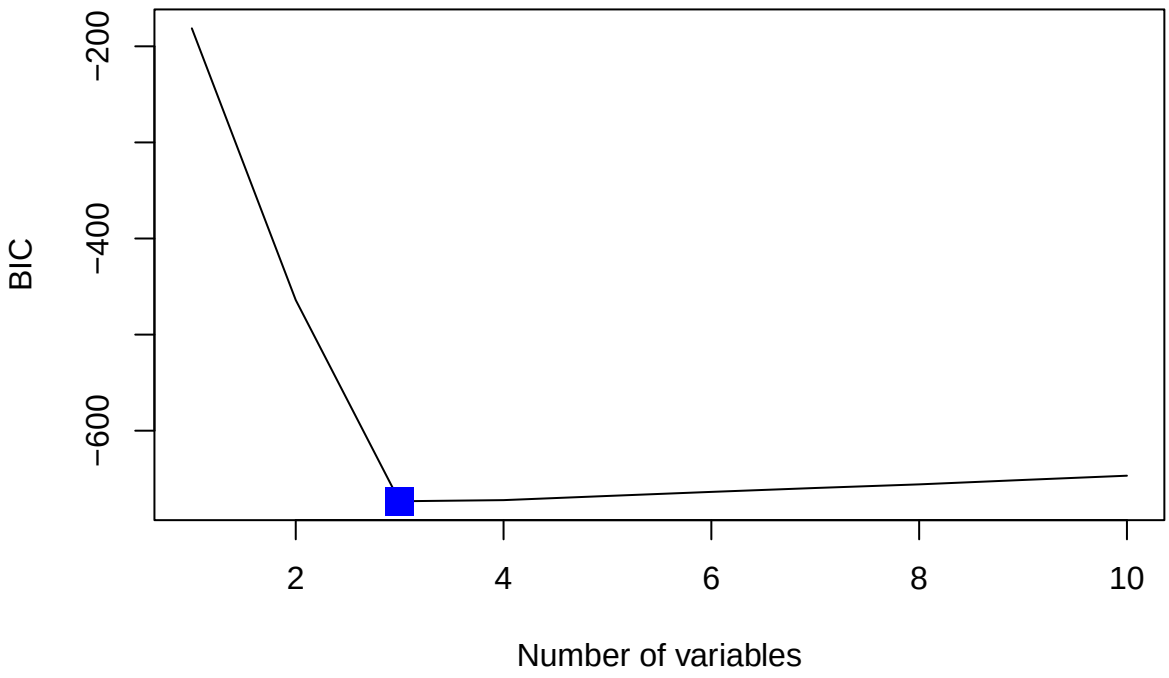

```
plot(sumfwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(sumfwd$adjr2), sumfwd$adjr2[which.max(sumfwd$adjr2)], col = "blue", cex = 2, pch = 15)
```



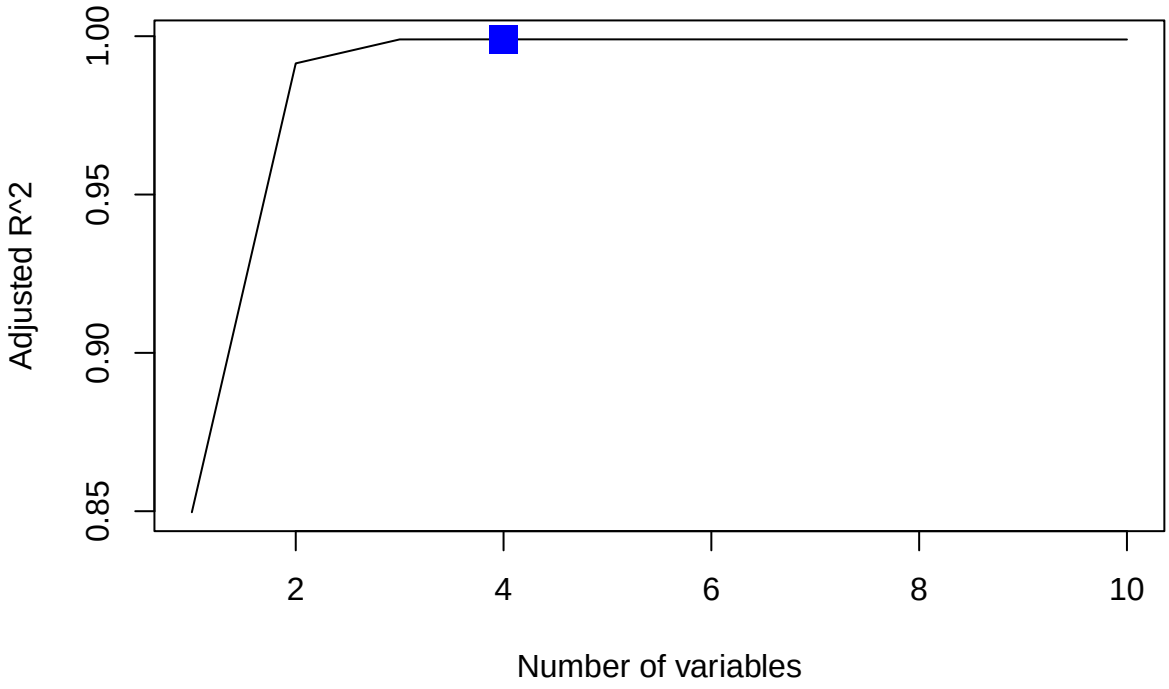

```
bwd.data <- regsubsets(y ~ poly(x, 10, raw = T), data = data, nvmax = 10, method = "backward")
sumbwd <- summary(bwd.data)

plot(sumbwd$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(which.min(sumbwd$cp), sumbwd$cp[which.min(sumbwd$cp)], col = "blue", cex = 2, pch = 15)
```




```
plot(sumbwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(sumbwd$bic), sumbwd$bic[which.min(sumbwd$bic)], col = "blue", cex = 2, pch = 15)
```




```
plot(sumbwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(sumbwd$adjr2), sumbwd$adjr2[which.max(sumbwd$adjr2)], col = "blue", cex = 2, pch = 15)
```



Answer

Forward and Backward select both suggested the same variable numbers as compared to results in (a).

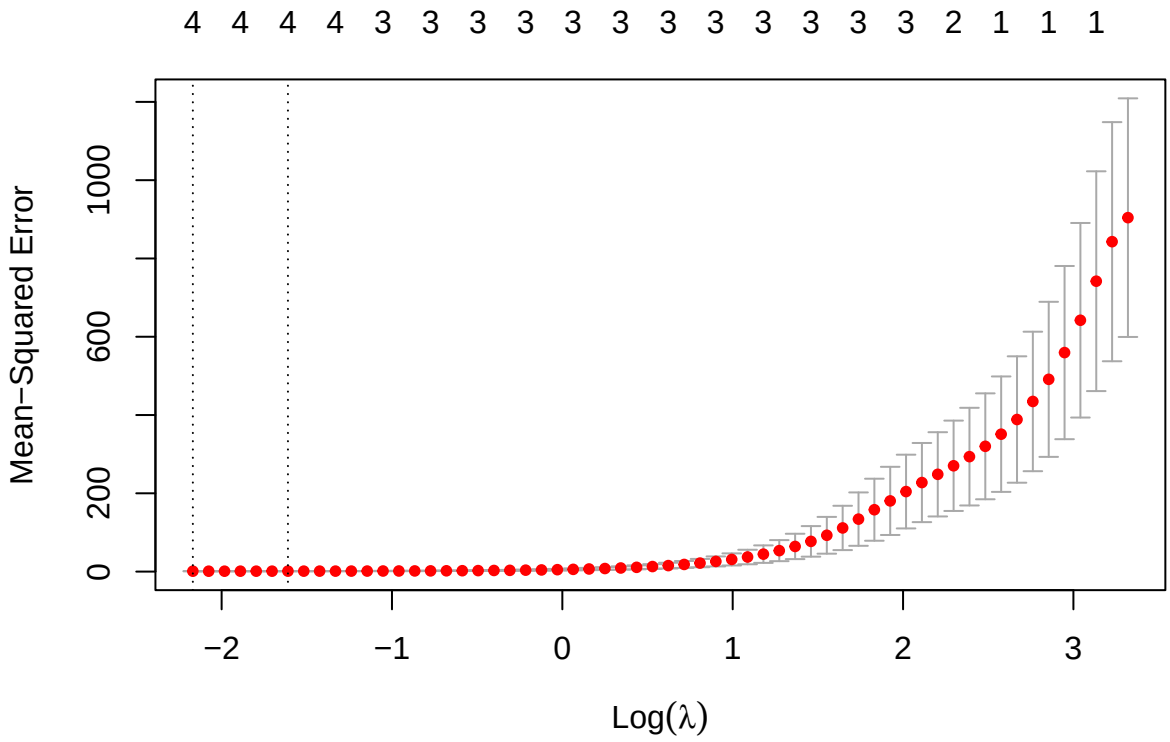
(e) Now fit a lasso model to the simulated data, again using X,X2, . . . , X 10 as predictors. Use cross-validation to select the optimal value of lambda. Create plots of the cross-validation error as a function of lambda. Report the resulting coefficient estimates, and discuss the results obtained.

```
# install.packages("glmnet")

library(glmnet)
```

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

```
## Loaded glmnet 4.1-6
```

```
mat <- model.matrix(y ~ poly(x, 10, raw = T), data = data)[, -1]
lasso.cv <- cv.glmnet(mat, y, alpha = 1)
plot(lasso.cv)
```




```
best <- lasso.cv$lambda.min
best
```

```
## [1] 0.1143051
```

```
fit <- glmnet(mat, y, alpha = 1)
predict(fit, s = best, type = "coefficients")[1:11,]
```

```
##            (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##             5.19422218            -4.95402817             9.61122433
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)4  poly(x, 10, raw = T)5
##            -9.95258205             0.03486649             0.00000000
##  poly(x, 10, raw = T)6  poly(x, 10, raw = T)7  poly(x, 10, raw = T)8
##             0.00000000             0.00000000             0.00000000
##  poly(x, 10, raw = T)9 poly(x, 10, raw = T)10
##             0.00000000             0.00000000
```

Answer

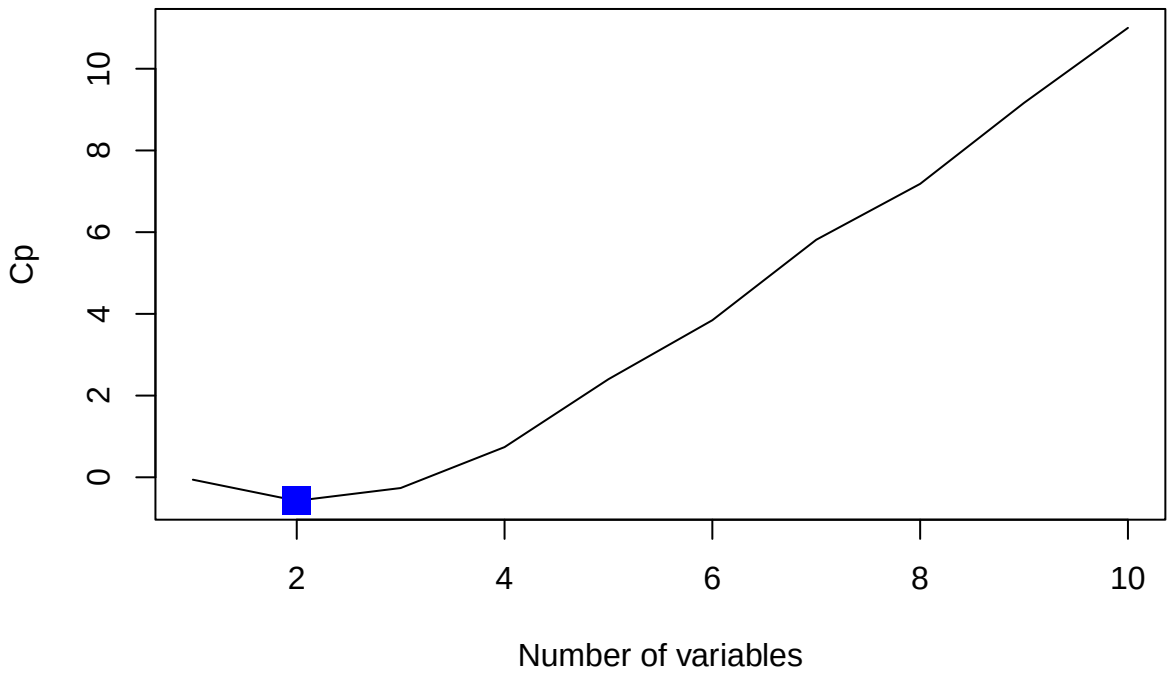
The lasso method picked X, X^2, X^3, X^4 as variable for the model.

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

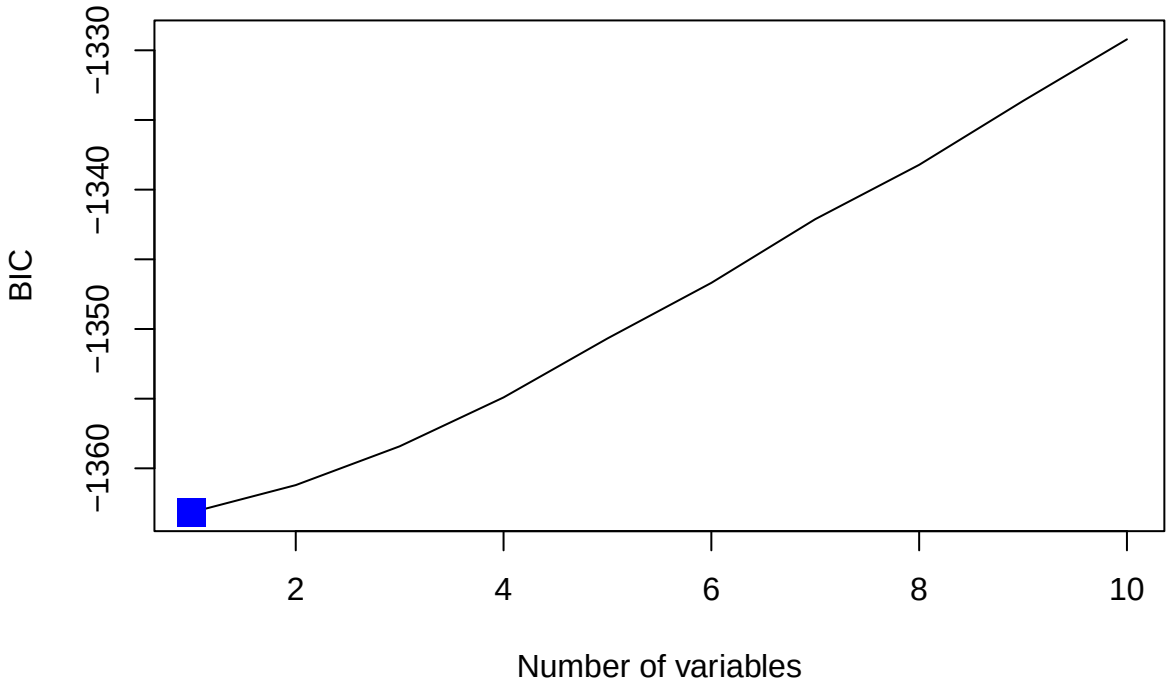
$$Y = \beta_0 + \beta_7 X^7 + \epsilon$$

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

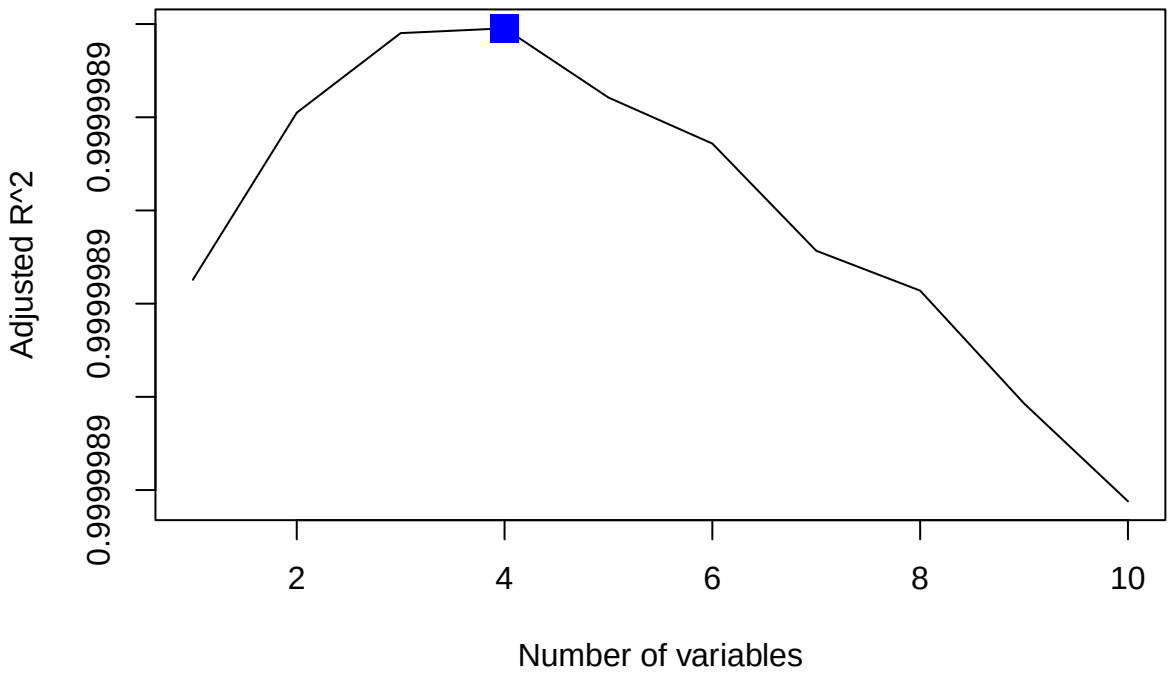
```
b7 <- 15
y1 <- b0 + b7 * x^7 + eps
data_7 <- data.frame (y=y1, x=x)
reg5 <- regsubsets(y1 ~ poly(x, 10, raw = T), data = data_7, nvmax = 10)
reg5_sum <- summary(reg5)

plot(reg5_sum$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(which.min(reg5_sum$cp), reg5_sum$cp[which.min(reg5_sum$cp)], col = "blue", cex = 2, pch = 15)
```




```
plot(reg5_sum$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(reg5_sum$bic), reg5_sum$bic[which.min(reg5_sum$bic)], col = "blue", cex = 2, pch = 15)
```




```
plot(reg5_sum$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(reg5_sum$adjr2), reg5_sum$adjr2[which.max(reg5_sum$adjr2)], col = "blue", cex = 2, pch
```




```
coef(reg5, 2)
```

```
##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##             5.0704904            -0.1417084            15.0015552
```

```
coef(reg5, 1)
```

```
##           (Intercept) poly(x, 10, raw = T)7
##               4.95894              15.00077
```

```
coef(reg5, 4)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             5.0762524             0.2914016            -0.1617671
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
##            -0.2526527            15.0091338
```

```
mat_7 <- model.matrix(y1 ~ poly(x, 10, raw = T), data = data_7)[, -1]
lasso.cv_7 <- cv.glmnet(mat_7, y1, alpha = 1)
best_7 <- lasso.cv_7$lambda.min
best_7
```

```
## [1] 26.5031
```

```
fit_7 <- glmnet(mat_7, y, alpha = 1)
predict(fit_7, s = best_7, type = "coefficients")[1:11, ]
```

```
##            (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##             10.5092472             0.0000000             0.0000000
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)4  poly(x, 10, raw = T)5
##             -0.4383893             0.0000000             0.0000000
##  poly(x, 10, raw = T)6  poly(x, 10, raw = T)7  poly(x, 10, raw = T)8
##              0.0000000             0.0000000             0.0000000
##  poly(x, 10, raw = T)9 poly(x, 10, raw = T)10
##              0.0000000             0.0000000
```

Answer

Cp, BIC, adjusted R^2 picked 2, 1, 4 variable model respectively while the lasso method picked a single variable model of X^3.

Question 6.8 - 9

In this exercise, we will predict the number of applications received using the other variables in the College data set.

(a) Split the data set into a training set and a test set.

```
library(ISLR)
data(College)
set.seed(1)
training_col <- sample(1:dim(College)[1], dim(College)[1] / 2)
test_col <- -training_col
col_tr <- College[training_col, ]
col_te <- College[test_col, ]
```

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

```
reg6 <- lm(Apps~., data = col_tr)
reg6_pred <- predict(reg6, col_te)
mean((reg6_pred - col_te$Apps)^2)
```

```
## [1] 1135758
```

Answer

The MSE is 1.14×10^6.

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

```
mat_tr <- model.matrix(Apps ~ ., data = col_tr)
mat_te <- model.matrix(Apps ~ ., data = col_te)
grid <- 10 ^ seq(4, -2, length = 100)
ridge <- glmnet(mat_tr, col_tr$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
ridge_cv <- cv.glmnet(mat_tr, col_tr$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
best_ridge <- ridge_cv$lambda.min
best_ridge
```

```
## [1] 0.01
```

```
ridge_pred <- predict(ridge, s = best_ridge, newx = mat_te)
mean((ridge_pred - col_te$Apps)^2)
```

```
## [1] 1135714
```

Answer

The ridge regression provided slightly smaller MSE, but overall similar.

(d) Fit a lasso model on the training set, with lambda chosen by cross- validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
lasso_cv <- cv.glmnet(mat_tr, col_tr$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
best_lasso <- lasso_cv$lambda.min
best_lasso
```

```
## [1] 0.01
```

```
lasso_pred <- predict(lasso_cv, newx=mat_te, s=best_lasso)
mean((col_te[, "Apps"] - lasso_pred)^2)
```

```
## [1] 1135660
```

```
lasso_fit <- glmnet(mat_tr, col_tr$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
predict(lasso_fit, s = best_lasso, type = "coefficients")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -7.900363e+02
## (Intercept)  .
## PrivateYes  -3.070103e+02
## Accept       1.779328e+00
## Enroll      -1.469508e+00
## Top10perc    6.672214e+01
## Top25perc   -2.230442e+01
## F.Undergrad  9.258974e-02
## P.Undergrad  9.408838e-03
## Outstate    -1.083495e-01
## Room.Board   2.115147e-01
## Books        2.912105e-01
```

```
## Personal     6.120406e-03
## PhD         -1.547200e+01
## Terminal     6.409503e+00
## S.F.Ratio    2.282638e+01
## perc.alumni  1.130498e+00
## Expend       4.856697e-02
## Grad.Rate    7.488081e+00
```

Answer

The lasso method provided even slightly smaller MSE, but overall similar. All of the predictors had non-zero coefficient estimates.

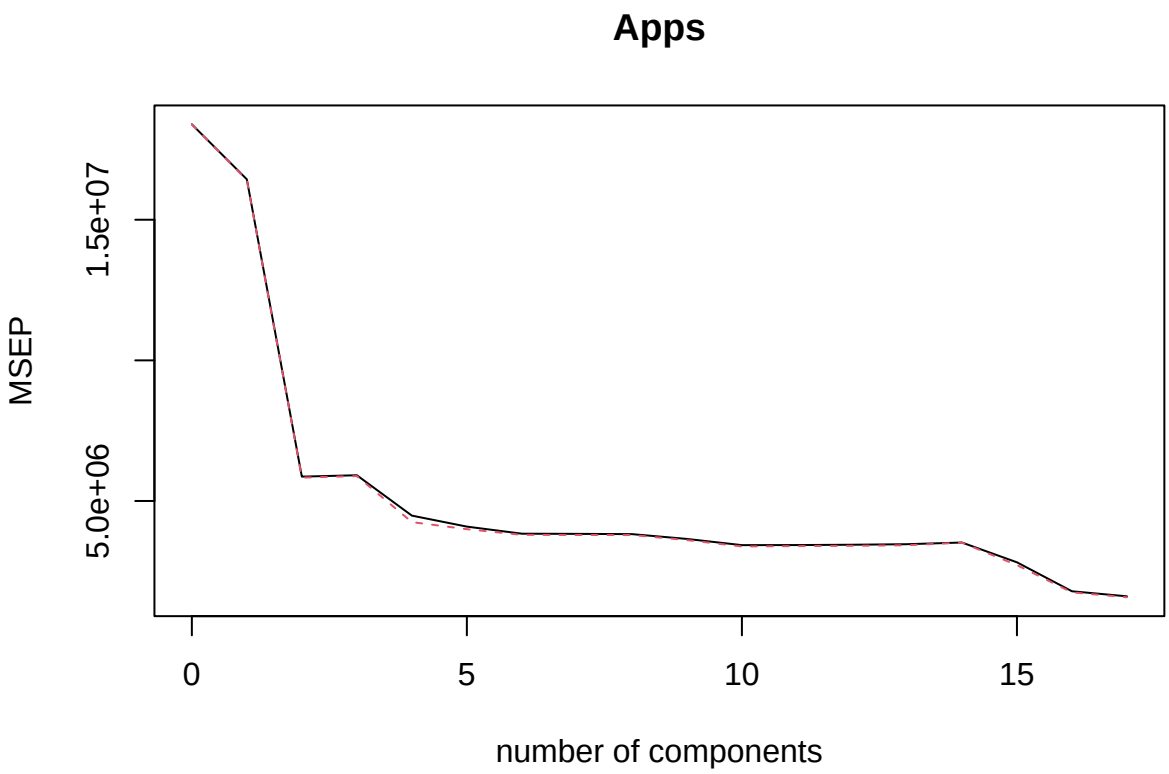
(e) Fit a PCR model on the training set, with M chosen by cross- validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
# install.packages("pls")

library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
pcr_fit <- pcr(Apps ~ ., data = col_tr, scale = TRUE, validation = "CV")
validationplot(pcr_fit, val.type = "MSEP")
```




```
pcr_pred <- predict(pcr_fit, col_te, ncomp = 10)
mean((pcr_pred - col_te$Apps)^2)
```

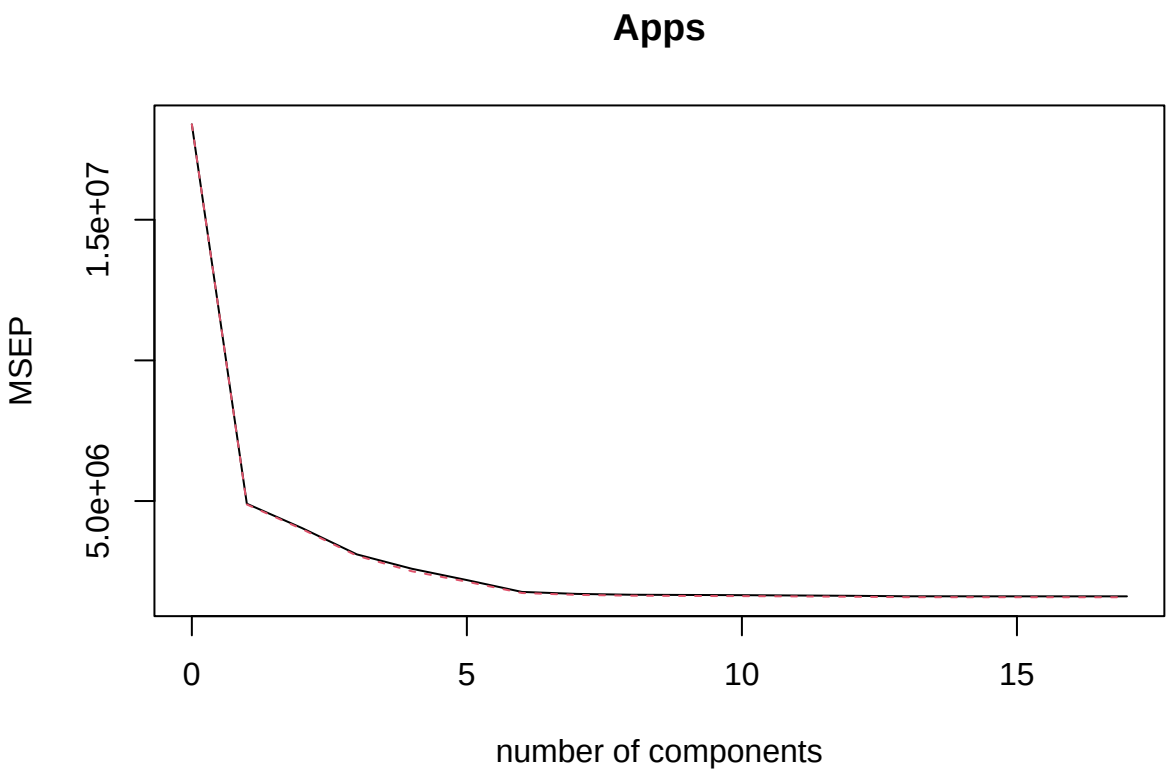
```
## [1] 1723100
```

Answer

The PCR method provided higher MSE as compared to least square method.

(f) Fit a PLS model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
pls_fit <- plsr(Apps ~ ., data = col_tr, scale = TRUE, validation = "CV")
validationplot(pls_fit, val.type = "MSEP")
```




```
pls_pred <- predict(pls_fit, col_te, ncomp = 10)
mean((pls_pred - col_te$Apps)^2)
```

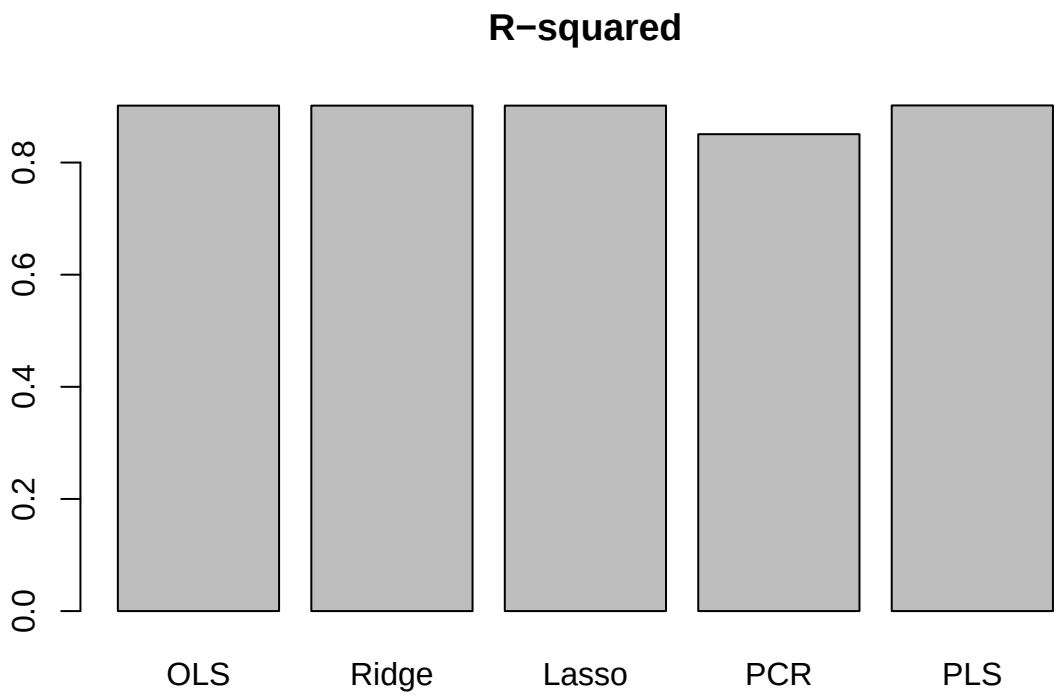
```
## [1] 1131661
```

Answer

The pls method provided lower MSE as comapred to least square method.

(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

```
testavg <- mean(col_te[, "Apps"])
lmr2 <- 1 - mean((reg6_pred - col_te$Apps)^2) / mean((testavg - col_te$Apps)^2)
ridger2 <- 1 - mean((ridge_pred - col_te$Apps)^2) / mean((testavg - col_te$Apps)^2)
lassor2 <- 1 - mean((lasso_pred - col_te$Apps)^2) / mean((testavg - col_te$Apps)^2)
pcrr2 <- 1 - mean((pcr_pred - col_te$Apps)^2) / mean((testavg - col_te$Apps)^2)
plsr2 <- 1 - mean((pls_pred - col_te$Apps)^2) / mean((testavg - col_te$Apps)^2)
barplot(c(lmr2, ridger2, lassor2, pcrr2, plsr2), col="grey", names.arg=c("OLS", "Ridge", "Lasso", "PCR"
```



Answer

As the plot depicted, all models except PCR method had similar R-square value, which means every method except PCR method predicted college application with high accuracy.