Resampling and Regularization Homework

Tao He

2/6/2022

5.8.We will now perform cross-validation on a simulated data set. (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.

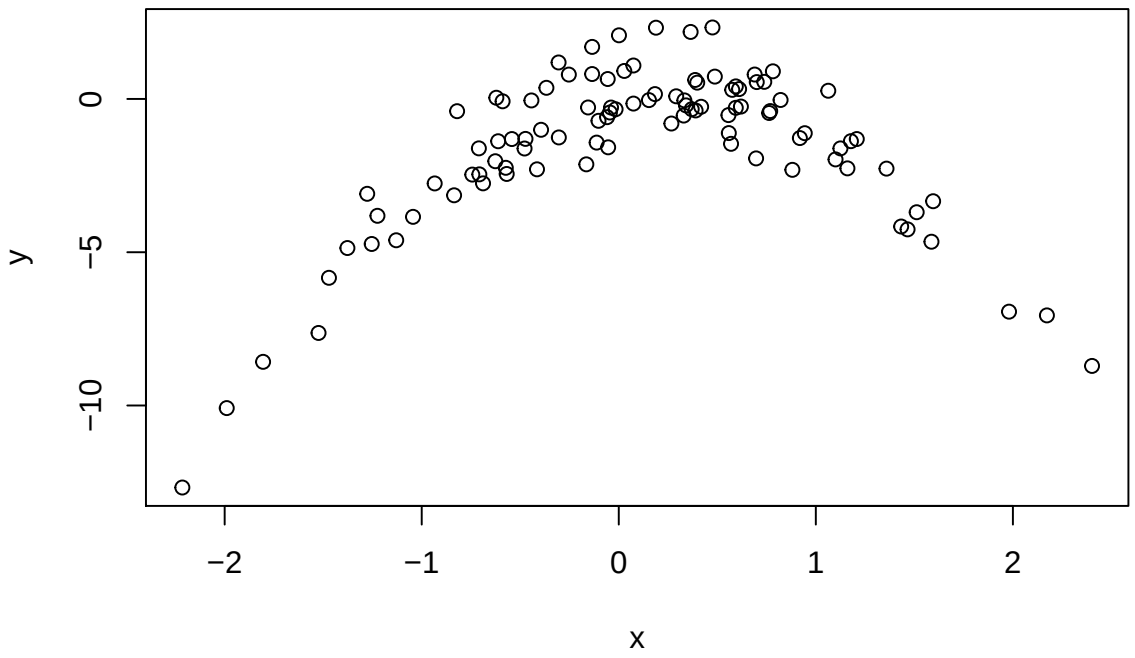
sol.n: In this data set, n is 100 and p is 2. The model is

$$Y = X - 2X^2 + \epsilon$$

.

(b) Create a scatter plot of X against Y. Comment on what you find.

```
plot(x,y)
```



It is obvious that x and y have a curved relationship.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares: Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y.

$$i. Y = \beta_0 + \beta_1 X + \epsilon;$$

```
set.seed(679)
Data <- data.frame(x, y)
fit1 <- glm(y ~ x)
cv.glm(Data, fit1)$delta[1]
```

```
## [1] 7.288162
```

$$ii. Y = \beta_0 + \beta_1 X + \beta_2 X_2 + \epsilon;$$

```
fit2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit2)$delta[1]
```

```
## [1] 0.9374236
```

$$iii. Y = \beta_0 + \beta_1 X + \beta_2 X_2 + \beta_3 X_3 + \epsilon;$$

```
fit3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit3)$delta[1]
```

```
## [1] 0.9566218
```

$$iv. Y = \beta_0 + \beta_1 X + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \epsilon.$$

```
fit4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit4)$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?

```
set.seed(1)
fit1 <- glm(y ~ x)
cv.glm(Data, fit1)$delta[1]
```

```
## [1] 7.288162
```

```
fit2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit2)$delta[1]
```

```
## [1] 0.9374236
```

```
fit3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit3)$delta[1]
```

```
## [1] 0.9566218
```

```
fit4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit4)$delta[1]
```

```
## [1] 0.9539049
```

The results are same as the results in (c). Since LOOCV repeatedly fit the statistical learning method using training sets that contain n − 1 observations, almost as many as are in the entire data set.

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

sol.n: The LOOCV estimate for the test MSE is the average of these n test error estimates and we can see the estimate of the test MAE of fit2 had the smallest LOOVC error. It is not surprising since we saw clearly in (b) that the relation between "x" and "y" is quadratic and curved.

(f) Comment on the statistical significance of the coefficient esti- mates 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?

sol.n:

```
summary(fit4)
```

```
##
## 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
```

We can see the coefficients of linear and quadratic terms are significant because of the small p-value. However, the coefficients of cubic and 4th degree terms are not significant. This agree strongly with our cross-validation results which were minimum for the quadratic model.

6.2.For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer. (a) The lasso, relative to least squares, is:

i. More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.
iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
iv. Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

sol.n:

iii. The lasso is less flexible than the least squares method and since it improves prediction accuracy when its increase in bias is less than its decrease in variance.

(b) Repeat (a) for ridge regression relative to least squares.

sol.n:

iii. The ridge regression is less flexible than the least squares method and since it improves prediction accuracy when its increase in bias is less than its decrease in variance.

(c) Repeat (a) for non-linear methods relative to least squares.

sol.n:

ii. The non-linear methods is more flexible than the least square method and since it improves prediction accuracy when its increase in variance is less than its decrease in bias.

6.9.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.

```
attach(College)
x <- model.matrix(Apps~ ., College)[, -1]
y <- College$Apps

grid <- 10^seq(10, -2, length = 100)

set.seed(679)

train <- sample(1:nrow(x), nrow(x)/2)
test <-  (-train)
y.test <-  y[test]
```

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

```
# Linear model using least squares (ridge regression with lambda=0) and test MSE.
fit.linear <- glmnet(x[train,], y[train], alpha = 0, lambda = grid, thresh = 1e-12)
fit.linear.pred <- predict(fit.linear, s = 0, newx = x[test, ], exact = T, x = x[train, ], y = y[train])

err.lm <- mean((fit.linear.pred - y.test)^2)
err.lm
```

```
## [1] 1454265
```

(c) Fit a ridge regression model on the training set, with

$$\lambda$$

chosen by cross-validation. Report the test error obtained.

```
cv.out <- cv.glmnet(x[train,],y[train],alpha=0)
bestlam <- cv.out$lambda.min

ridge.mod <- glmnet(x[train, ], y[train], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test, ])
err.ridge <- mean((ridge.pred - y.test)^2)
err.ridge
```

```
## [1] 2168171
```

(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.

```
cv.out <- cv.glmnet(x[train,],y[train],alpha=1)
bestlam <- cv.out$lambda.min

lasso.mod <- glmnet(x[train,],y[train],alpha=1,lambda=grid)
lasso.pred <- predict(lasso.mod, s=bestlam, newx=x[test,])
err.lasso <- mean((lasso.pred-y.test)^2)
err.lasso
```

```
## [1] 1452932
```

```
lasso.coef = predict(lasso.mod, type="coefficients", s=bestlam)[1:18,]
length(lasso.coef[lasso.coef!= 0])
```

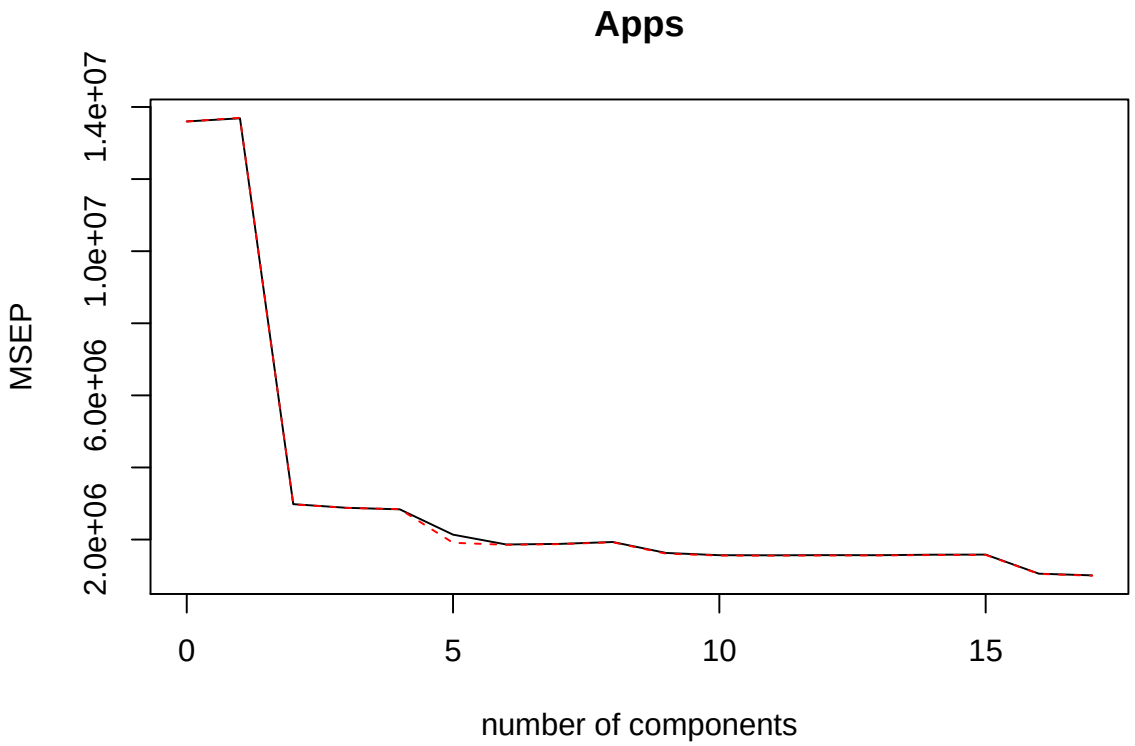
```
## [1] 18
```

```
lasso.coef
```

```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -476.48313070 -470.92627629    1.37345659   -0.86044195   46.31239757
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
##  -12.71711651    0.15671246    0.03984880   -0.02220130    0.14182006
##         Books      Personal           PhD      Terminal     S.F.Ratio
##    0.68694877   -0.06843998  -11.75019935   -1.94726605    4.34068796
##   perc.alumni        Expend     Grad.Rate
##   -9.43638315    0.04144067    8.91595273
```

(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.

```
pcr.fit <- pcr(Apps~ ., data = College, subset = train, scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```



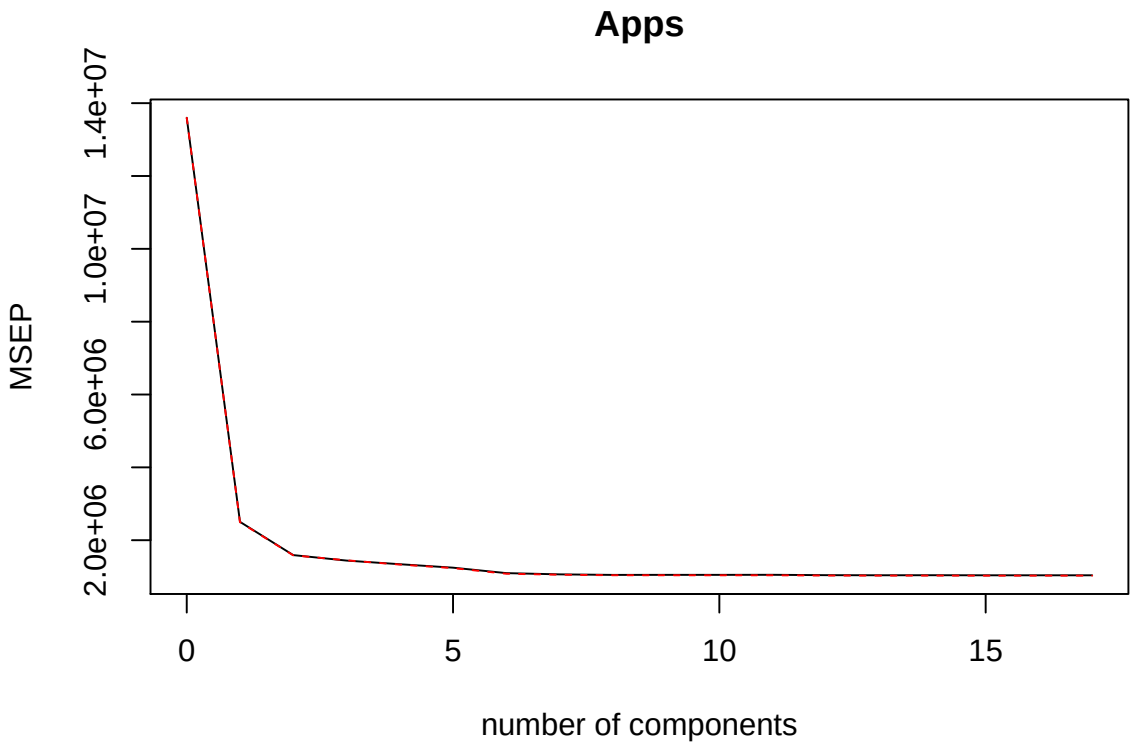
Now we find that the lowest cross-validation error occurs when M = 16 components are used. We compute the test MSE as follows.

```
pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 16)
err.pcr <- mean((pcr.pred - y.test)^2)
err.pcr
```

```
## [1] 1476735
```

(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 = College, subset=train, scale = TRUE, validation = "CV")
validationplot(pls.fit, val.type="MSEP")
```



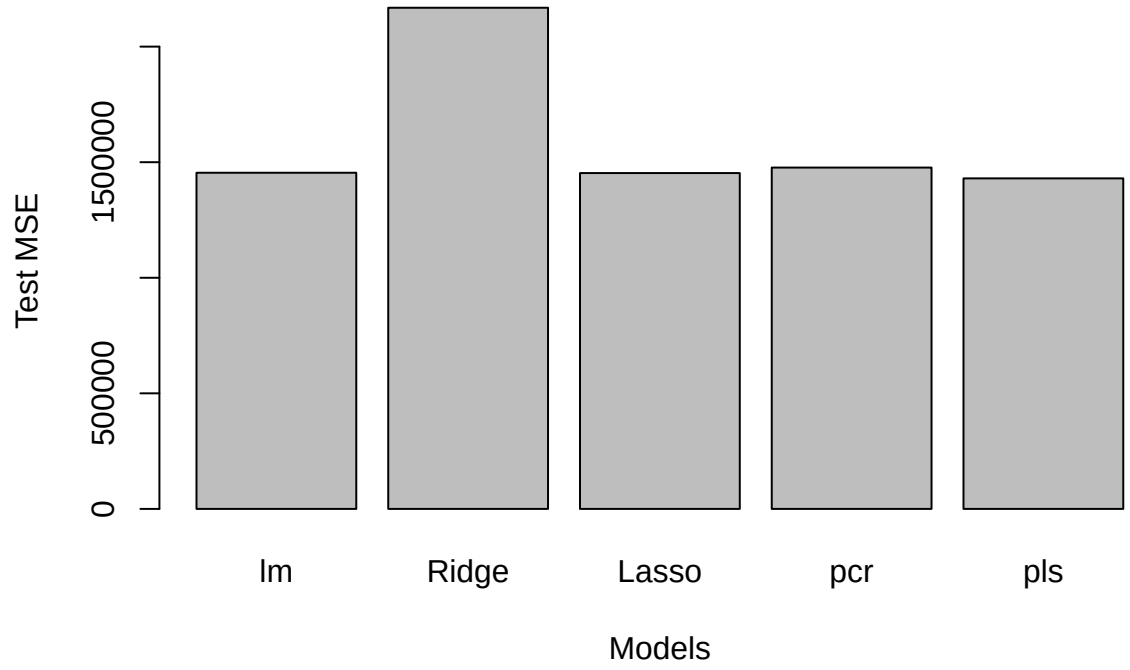
Now we find that the lowest cross-validation error occurs when M = 8 components are used. We compute the test MSE as follows.

```
pls.pred <- predict(pls.fit, x[test, ], ncomp = 8)
err.pls <- mean((pls.pred - y.test)^2)
err.pls
```

```
## [1] 1429903
```

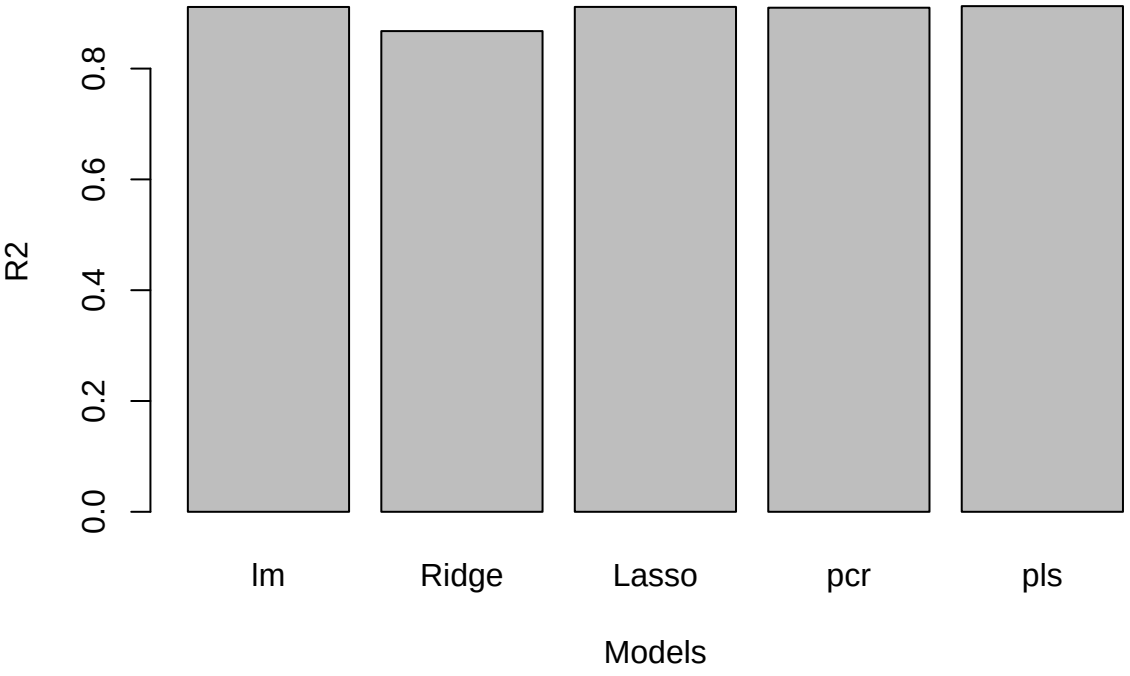
(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

```
err.all <- c(err.lm, err.ridge, err.lasso, err.pcr, err.pls)
barplot(err.all, xlab="Models", ylab="Test MSE", names=c("lm", "Ridge", "Lasso", "pcr", "pls"))
```



All the test errors resulting from these five approaches are nearly the same, except the Ridge models. Lasso and pls models have the lower test MSE.

```
# compute the R^2 to test the accuracy
test.avg <- mean(y.test)
lm.r2    <- 1 - mean((fit.linear.pred - y.test)^2) / mean((test.avg - y.test)^2)
ridge.r2 <- 1 - mean((ridge.pred - y.test)^2) / mean((test.avg - y.test)^2)
lasso.r2 <- 1 - mean((lasso.pred - y.test)^2) / mean((test.avg - y.test)^2)
pcr.r2   <- 1 - mean((pcr.pred - y.test)^2) / mean((test.avg - y.test)^2)
pls.r2   <- 1 - mean((pls.pred - y.test)^2) / mean((test.avg - y.test)^2)
barplot(c(lm.r2, ridge.r2, lasso.r2, pcr.r2, pls.r2), xlab="Models", ylab="R2",
        names=c("lm", "Ridge", "Lasso", "pcr", "pls"))
```



All the models have R^2 with over 0.8.We can predict the number of college applications received confidently.

6.10. 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

$$Y = X\beta + \epsilon$$

, where

$$\beta$$

has some elements that are exactly equal to zero.

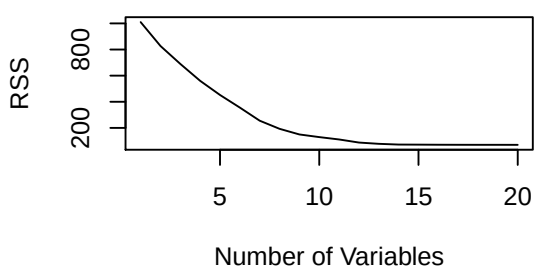
```
set.seed(2021)
# Set the parameters
p <- 20
n <- 1000
# Create a matrix of n \times p
X <- matrix(rnorm(n*p), n, p)
# Create a vector of beta and randomly assign 0 to some betas
beta <- rnorm(p)
beta[1] = beta[3] = beta[5] = beta[7] = beta[9] = 0
# Generating the error terms
error <- rnorm(n)
# Multiplying X and beta to get y
Y <- X %*% beta + error
```

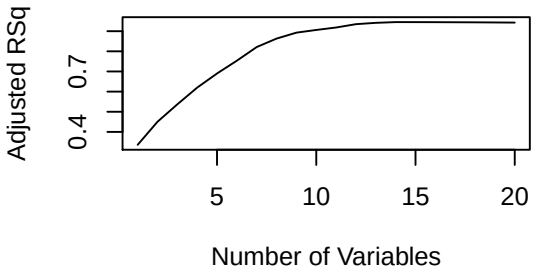
(b) Split your data set into a training set containing 100 observations and a test set containing 900 observations.

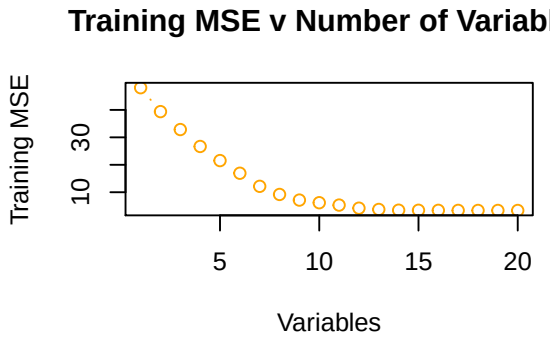
```
# Create a vector train of length 100 and assign them random
# index number chosen from 1 to n.
train <- sample(seq(n), 100, replace=FALSE)
# Create a vector test which contains rest of the indices
test <- (-train)
# Subset X matrix for the given sequence in train and test
x.train = X[train,]
x.test = X[test,]
y.train = Y[train]
y.test = Y[test]
train <- data.frame(Y = y.train, X = x.train)
test <- data.frame(Y = y.test, X = x.test)
```

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

```
regfit.full <- regsubsets(Y~ ., data = train,  nvmax = p)
reg.summary <- summary(regfit.full)

par(mfrow = c(2, 2))
plot(reg.summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")

train.mse <- (reg.summary$rss)/length(train)
plot(train.mse, xlab = "Variables", ylab = "Training MSE",
     main = "Training MSE v Number of Variables", pch = 1, type = "b", col="orange")
```

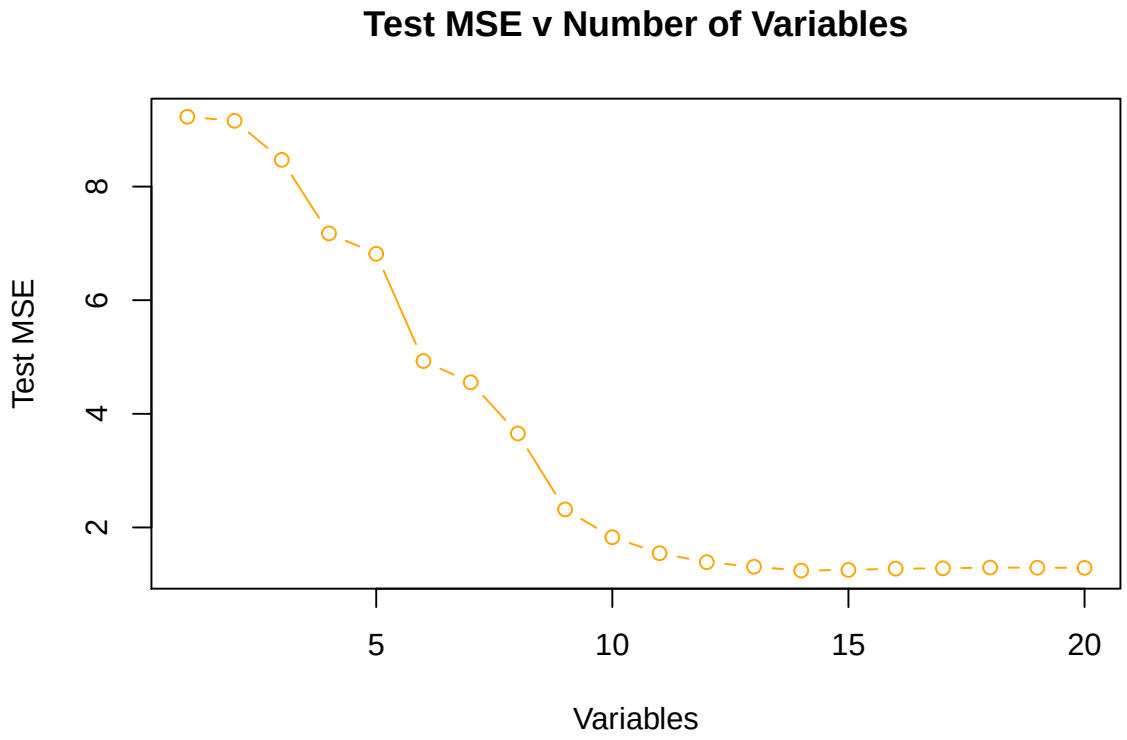






(d) Plot the test set MSE associated with the best model of each size.

```
test.mse <- rep(NA,20)
for(i in 1:20){
    model <- lm.regsubsets(regfit.full, i)
    model.pred <- predict(model, newdata=test, type=c("response"))
    test.mse[i] <-  mean((test$Y-model.pred)^2)
  }
# Plot
plot(1:20, test.mse, xlab = "Variables",ylab = "Test MSE",main = "Test MSE v Number of Variables",
     pch = 1, type = "b", col = "orange")
```



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

```
which.min(test.mse)
```

```
## [1] 14
```

Minimum Test MSE occurs at a model with 14 variables.

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

```
coef(regfit.full, 14)
```

```
##  (Intercept)          X.2          X.4          X.6          X.8         X.10
##  0.007858244 -0.966883088  0.488157768 -0.895973419 -1.224552344 -1.128006294
##         X.12         X.13         X.14         X.15         X.16         X.17
##  0.727396529 -1.020616370 -2.492471074 -0.493006806 -0.852328431  0.352594406
##         X.18         X.19         X.20
##  0.259739532  1.009661083 -0.613034766
```

```
beta
```

```
##  [1]  0.00000000 -0.93037070  0.00000000  0.40029582  0.00000000 -0.98970461
##  [7]  0.00000000 -1.14771580  0.00000000 -1.19977808  0.04756078  0.57822388
## [13] -0.83583341 -2.56123942 -0.44683306 -0.87682071  0.25668015  0.27719406
## [19]  1.03237577 -0.58525249
```

(g) Create a plot displaying

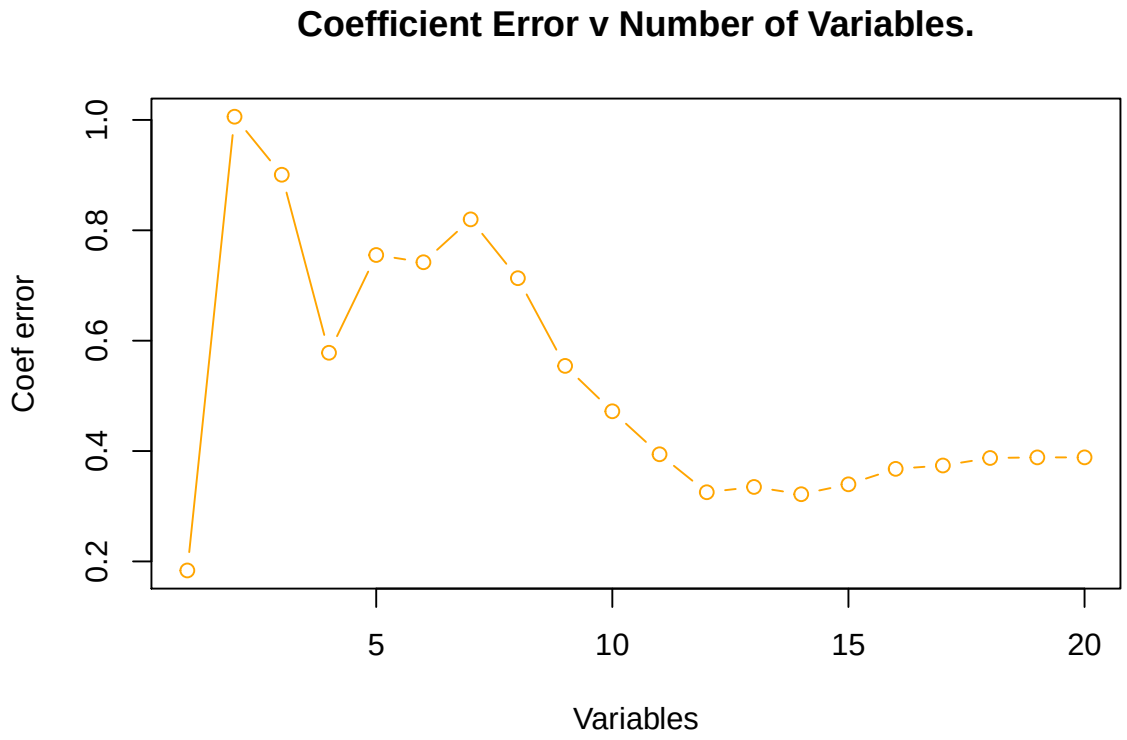
$$\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$$

for a range of values of r, where

$$\hat{\beta}_j^r$$

is the jth 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)?

```
# Original coefficient values transposed and columns renamed.
beta <- as.data.frame(t(beta))
names(beta) <- paste0('X.', 1:(ncol(beta)))

# Loop to calculate root squared errors of the true and estimated coefficients.
coef.err <- rep(NA,20)
for (i in 1:20){
  a <- coef(regfit.full, i)
  coef.err[i] <- sqrt(sum(((a[-1] - beta[names(a)[-1]])^2)))
}
plot(1:20, coef.err, xlab = "Variables", ylab = "Coef error",
     main="Coefficient Error v Number of Variables.", pch = 1, type = "b", col = "orange")
```




```
which.min(coef.err)
```

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

6.11. We will now try to predict per capita crime rate in the Boston data set. (a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

```
# Using Lasso regresson method.
set.seed(679)

x <- model.matrix(crim~., Boston)[,-1]
y <- Boston$crim
```

```
grid <- 10^seq(10, -2, length=100)
train <- sample(1:nrow(x), nrow(x)/1.3)
test <- (-train)
y.test <- y[test]

set.seed(679)
cv.out <- cv.glmnet(x[train,], y[train], alpha=1)
bestlam <- cv.out$lambda.min

lasso.mod <- glmnet(x[train,],y[train],alpha=1,lambda=grid)
lasso.pred <- predict(lasso.mod, s=bestlam, newx=x[test,])
mean((lasso.pred-y.test)^2)
```

```
## [1] 16.20087
```

```
lasso.coef <- predict(lasso.mod, type="coefficients", s=bestlam)[1:13,]
lasso.coef
```

```
##  (Intercept)           zn        indus         chas          nox           rm
## 14.671036211  0.037361741 -0.064657322 -0.434722774 -9.139546156  0.307855206
##          age          dis          rad          tax      ptratio        black
##  0.000000000 -0.834324360  0.533547236  0.000000000 -0.229980971 -0.009023112
##        lstat
##  0.155996648
```

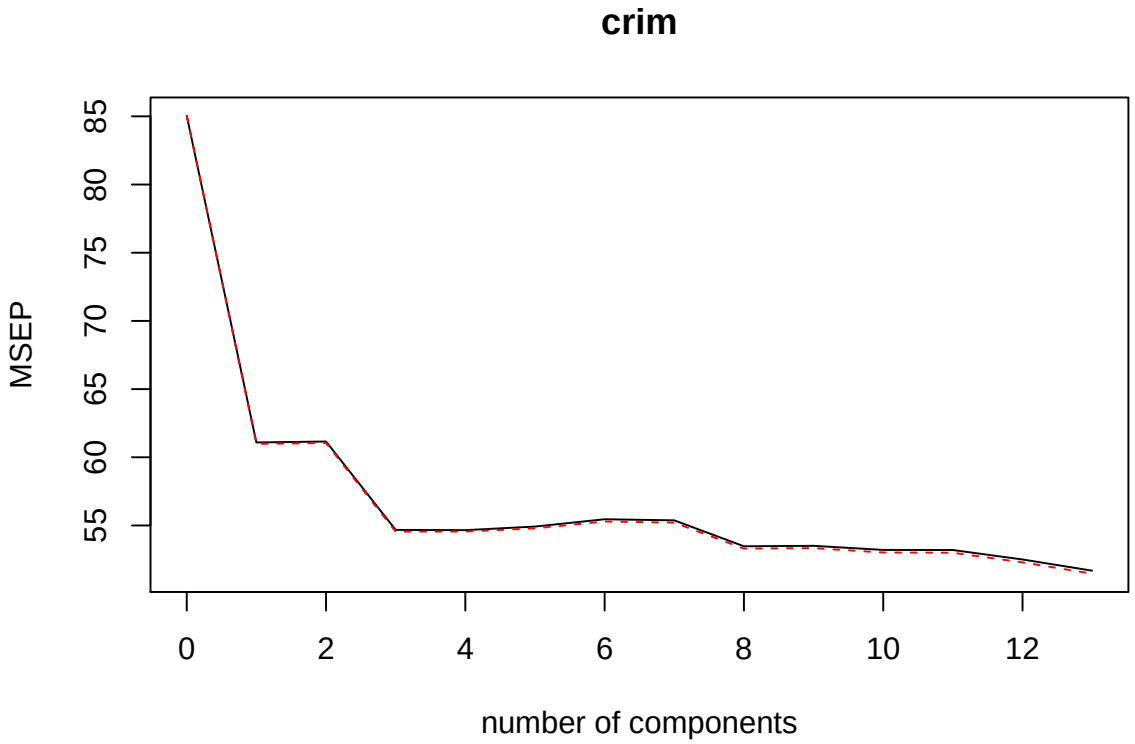
```
# Using Ridge regression
cv.out <- cv.glmnet(x[train,], y[train], alpha=0)
bestlam <- cv.out$lambda.min

glm.mod <- glmnet(x[train,],y[train], alpha=0, lambda=grid, thresh=1e-12)
glm.pred <- predict(glm.mod, s=bestlam, newx=x[test,])
mean((glm.pred-y.test)^2)
```

```
## [1] 16.22523
```

```
glm.coef <- predict(glm.mod, type="coefficients", s=bestlam)[1:13,]
glm.coef
```

```
##  (Intercept)           zn        indus         chas          nox           rm
##  8.503498260  0.030664536 -0.073717210 -0.581490399 -5.174004272  0.396112393
##          age          dis          rad          tax      ptratio        black
## -0.001492583 -0.673865392  0.388822155  0.005191953 -0.128822660 -0.010201622
##        lstat
##  0.171684394
```

```
# Using PCR regression
pcr.fit <- pcr(crim~., data=Boston, subset=train, scale=T, validation="CV")
validationplot(pcr.fit, val.type="MSEP")
```




```
pcr.pred <- predict(pcr.fit, x[test,], ncomp = 8)
mean((pcr.pred-y.test)^2)
```

```
## [1] 17.35708
```

(b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross- validation, or some other reasonable alternative, as opposed to using training error.

sol.n: The lasso model performs well on this data set since it has lowest test MSE and it is more interpretable.

(c) Does your chosen model involve all of the features in the data set? Why or why not?

sol.n:

My chosen model involves 11 variables in the data set.