ch5,6exercises

Lauren Temple

2/3/2022

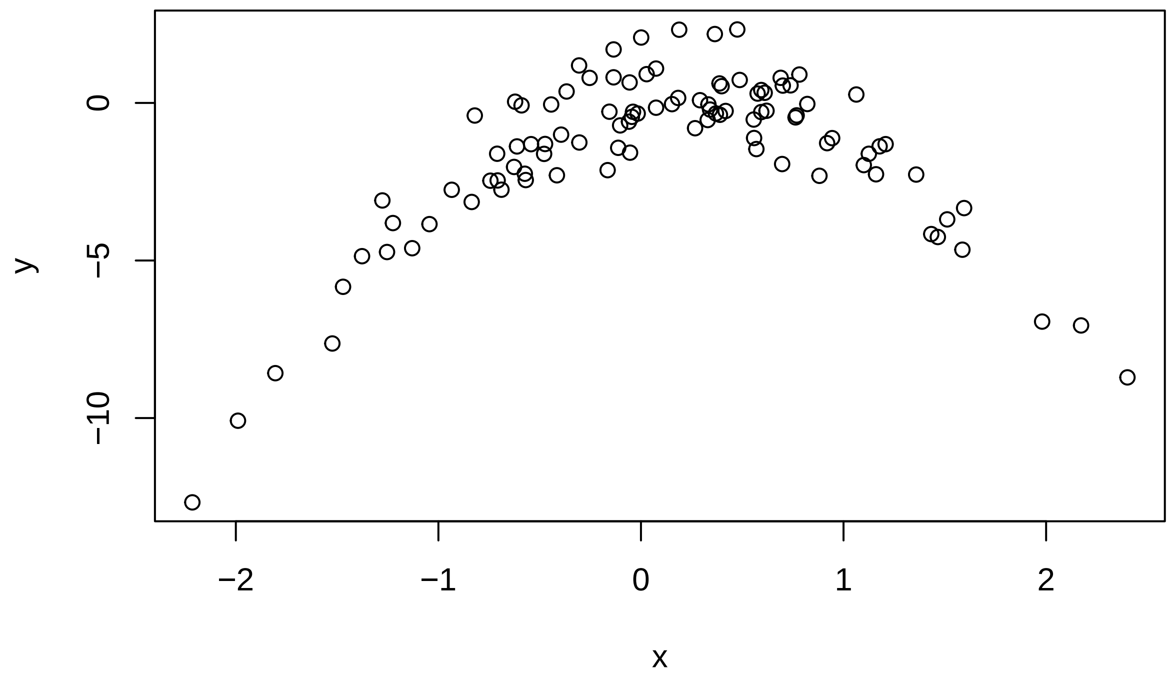
5.8 a.

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

n= 100 p=2 Y= X - 2X^2 + e

b.

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



This plot appears to be quadratic and ranges from -2 to 2 on the x axis and -10 to 2 on the y axis.

c.

```
set.seed(42)
data5.8 = data.frame(x, y)
```

```
i <- glm(y ~ x)
cv.glm(data5.8, i)$delta
```

```
## [1] 7.288162 7.284744
```

```
ii <- glm(y ~ poly(x, 2))
cv.glm(data5.8, ii)$delta
```

```
## [1] 0.9374236 0.9371789
```

```
iii <- glm(y ~ poly(x, 3))
cv.glm(data5.8, iii)$delta
```

```
## [1] 0.9566218 0.9562538
```

```
iv <- glm(y ~ poly(x, 4))
cv.glm(data5.8, iv)$delta
```

```
## [1] 0.9539049 0.9534453
```

d.

```
set.seed(123)
data5.8 = data.frame(x, y)
```

```
i <- glm(y ~ x)
cv.glm(data5.8, i)$delta
```

```
## [1] 7.288162 7.284744
```

```
ii <- glm(y ~ poly(x, 2))
cv.glm(data5.8, ii)$delta
```

```
## [1] 0.9374236 0.9371789
```

```
iii <- glm(y ~ poly(x, 3))
cv.glm(data5.8, iii)$delta
```

```
## [1] 0.9566218 0.9562538
```

```
iv <- glm(y ~ poly(x, 4))
cv.glm(data5.8, iv)$delta
```

```
## [1] 0.9539049 0.9534453
```

The results are the same with a different seed because LOOCV evaluates n folds of a single observation.

e. The second model, the quadratic one had the lowest LOOCV test error rate due to the fact that it most closely matches the true form of Y we saw in the scatter plot.

f.

```
summary(i)
```

```
##
## Call:
## glm(formula = y ~ x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.5161  -0.6800   0.6812   1.5491   3.8183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.6254     0.2619  -6.205 1.31e-08 ***
## x             0.6925     0.2909   2.380   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.760719)
##
##     Null deviance: 700.85  on 99  degrees of freedom
## Residual deviance: 662.55  on 98  degrees of freedom
## AIC: 478.88
##
## Number of Fisher Scoring iterations: 2
```

```
summary(ii)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9650  -0.6254  -0.1288   0.5803   2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5500     0.0958  -16.18  < 2e-16 ***
## poly(x, 2)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, 2)2 -23.9483     0.9580  -25.00  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9178258)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  89.029  on 97  degrees of freedom
## AIC: 280.17
##
## Number of Fisher Scoring iterations: 2
```

```
summary(iii)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 3))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9765  -0.6302  -0.1227   0.5545   2.2843
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09626 -16.102  < 2e-16 ***
## poly(x, 3)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, 3)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, 3)3   0.26411    0.96263   0.274    0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9266599)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  88.959  on 96  degrees of freedom
## AIC: 282.09
##
## Number of Fisher Scoring iterations: 2
```

```
summary(iv)
```

```
##
## 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 show statistical significance for the linear and quadratic terms but not the terms of higher degree polynomials.

6.2 a. iii. Lasso relative to least squares is less flexible and will give improved prediction accuracy when its increase in bias is less than its decrease in variance. As lambda increases, flexibility of fit decreases, and the estimated coefficients decrease with some being zero. This leads to a substantial decrease in the variance of the predictions for a small increase in bias.

b. iii. Ridge regression relative to least squares is less flexible and will give improved prediction accuracy when its increase in bias is less than its decrease in variance. As lambda increases, flexibility of fit decreases, and the estimated coefficients decrease with none being zero. This leads to a substantial decrease in the variance of the predictions for a small increase in bias.

c. ii. Non-linear models relative to least squares will be more flexible, and give improved prediction accuracy when its increase in variance is less than its decrease in bias. Predictions will improve if the variance rises less than a decrease in the bias which is the bias-variance trade off.

6.9

```
pacman::p_load(ISLR2, glmnet)
data(College)
data69 <- College
```

a.

```
set.seed(42)
x <- model.matrix(Apps ~ ., College)[,-1]
y <- College$Apps
grid <- 10^seq(10, -2, length= 100)
```

```
train <- sample(1:nrow(x), nrow(x)/1.3)
test <- (-train)
y.test <- y[test]
```

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

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

err.linear
```

```
## [1] 1704492
```

```r
train.df <- data.frame(College[train,])
test.df <- data.frame(College[test,])
```

```r
lm.fit <- lm(Apps ~ ., data= train.df)
lm.pred <- predict(lm.fit, test.df, type= c("response"))
err.lm <- mean((lm.pred - test.df$Apps)^2)
err.lm
```

```
## [1] 1704464
```

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

```r
set.seed(42)
cv.out <- cv.glmnet(x[train,], y[train], alpha= 0)
bestlam.ridge <- cv.out$lambda.min

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

```
## [1] 2359439
```

```r
bestlam.ridge
```

```
## [1] 338.6721
```

The best value of lambda for ridge regression, chosen by cv is about 338.67. The test error obtained is 2359439.

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

```r
set.seed(42)
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda= grid)

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

lasso.pred <- predict(lasso.mod, s= bestlam.lasso, newx = x[test,])
err.lasso <- mean((lasso.pred - y.test)^2)

bestlam.lasso
```

```
## [1] 8.788624
```

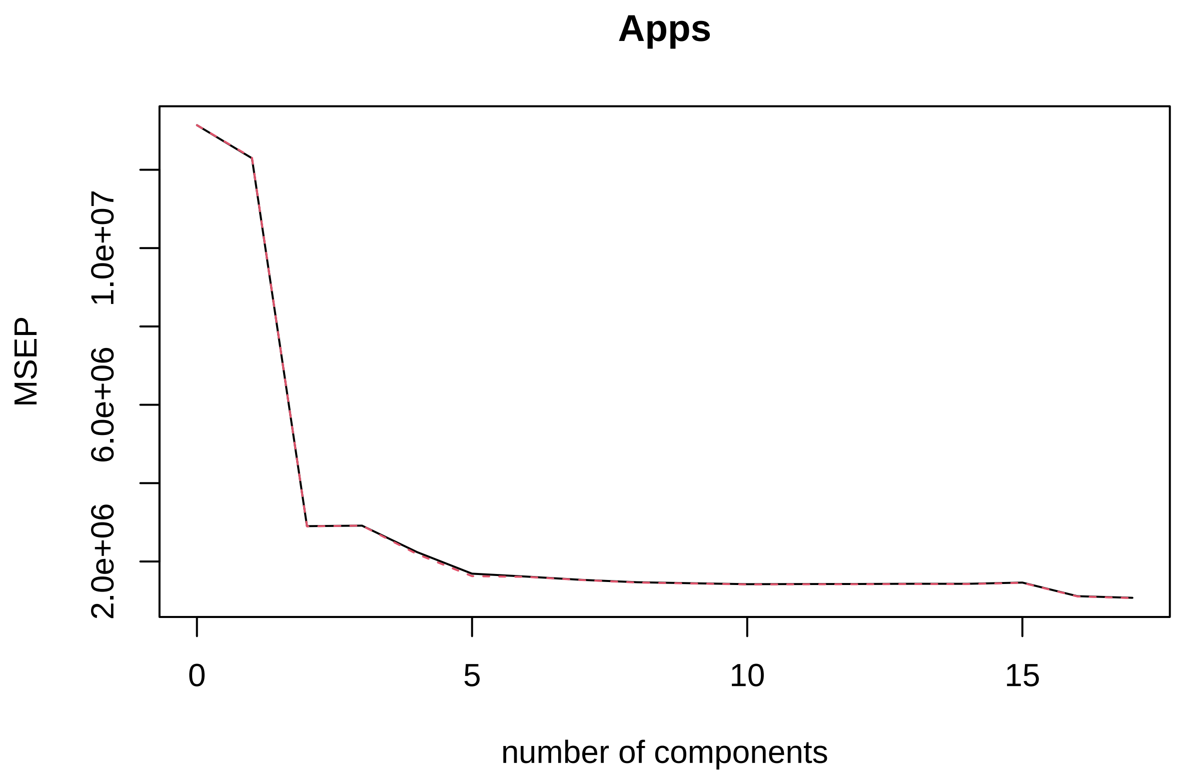
```
err.lasso
```

```
## [1] 1793449
```

The best value of lambda for lasso regression, chosen by cv is about 8.79. The test error obtained is 1793449.

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

```
pacman::p_load(pls)
set.seed(42)
pcr.fit <- pcr(Apps ~ ., data= College, subset= train, scale= T, validation= "CV")
validationplot(pcr.fit, val.type="MSEP")
```



```
set.seed(42)
pcr.pred <- predict(pcr.fit, x[test,], ncomp= 5)
err.pcr <- mean((pcr.pred - y.test)^2)
err.pcr
```

```
## [1] 5023811
```

The test set MSE using M=5 is 5023811.

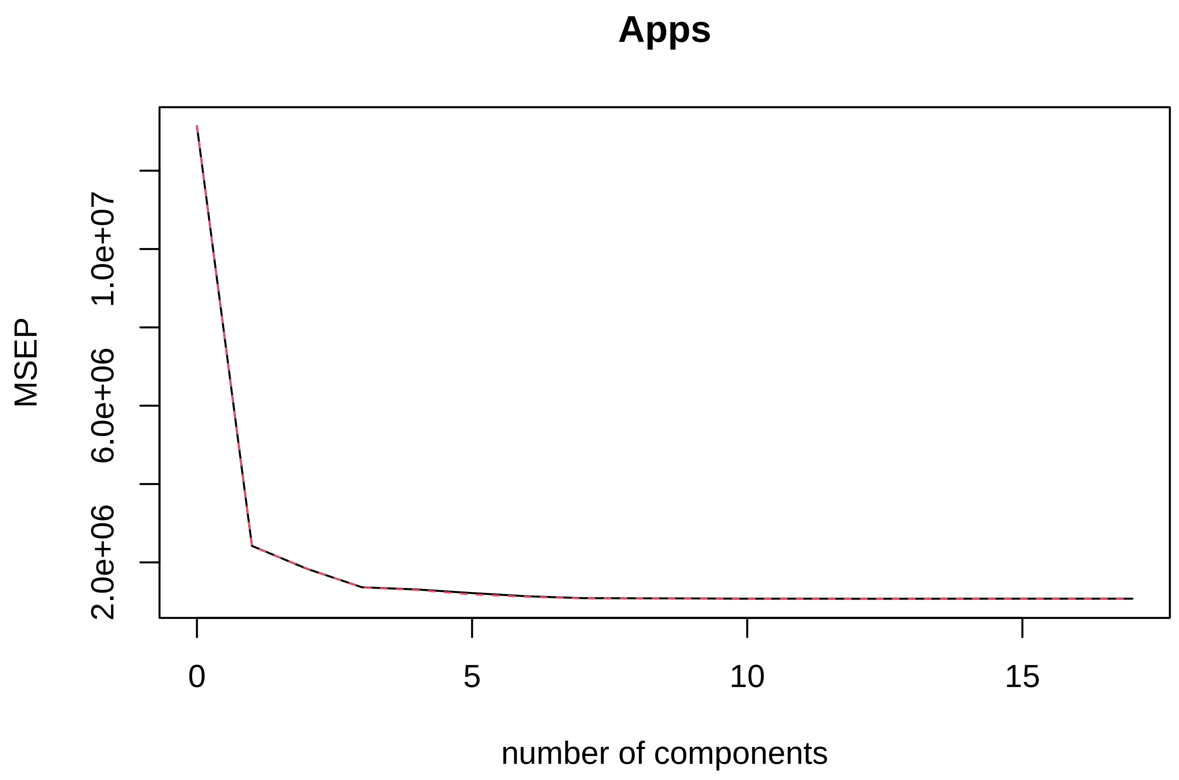
```
pcr.pred <- predict(pcr.fit, x[test,], ncomp= 16)
err.pcr <- mean((pcr.pred - y.test)^2)
err.pcr
```

```
## [1] 1992462
```

The MSE using M=16 is 1992464.

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

```
set.seed(42)
pls.fit <- plsr(Apps ~ ., data= College, subset= train, scale= T, validation= "CV")
validationplot(pls.fit, val.type= "MSEP")
```



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

```
## [1] 1752400
```

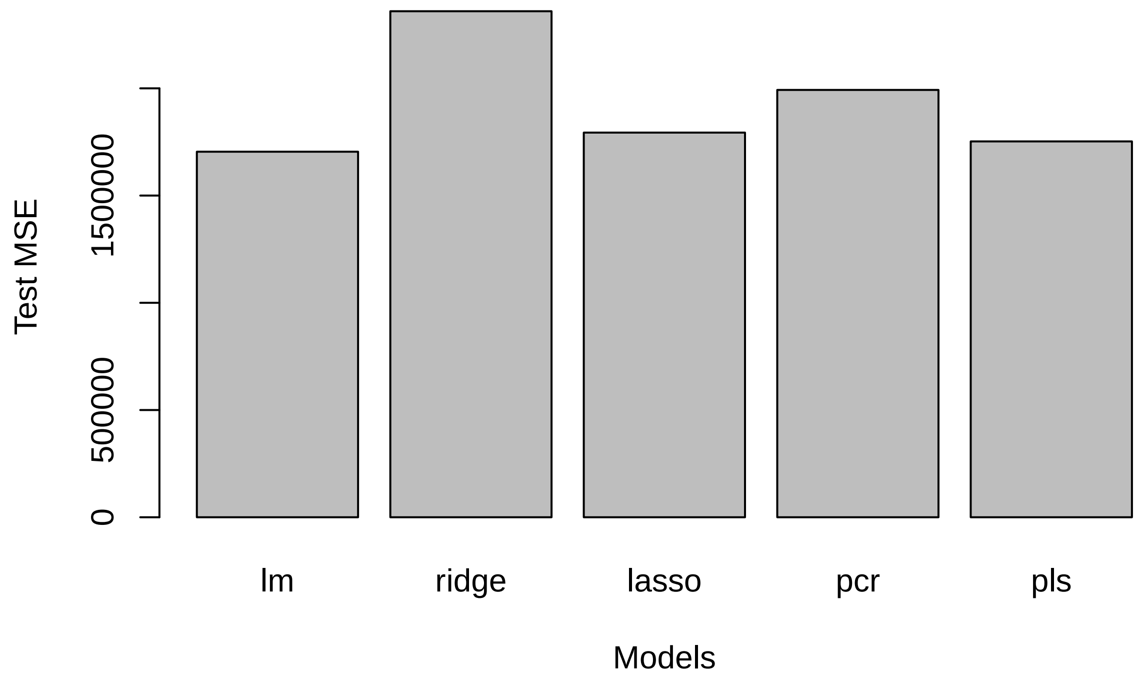
The MSE is 1752400 when M=8.

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)
table(err.all, names= c("lm", "ridge", "lasso", "pcr", "pls"))
```

```
##                    names
## err.all              lasso lm pcr pls ridge
##   1704463.84724487      0  1   0   0     0
##   1752399.68812991      0  0   0   1     0
##   1793449.20813037      1  0   0   0     0
##   1992461.54049088      0  0   1   0     0
##   2359438.94097798      0  0   0   0     1
```

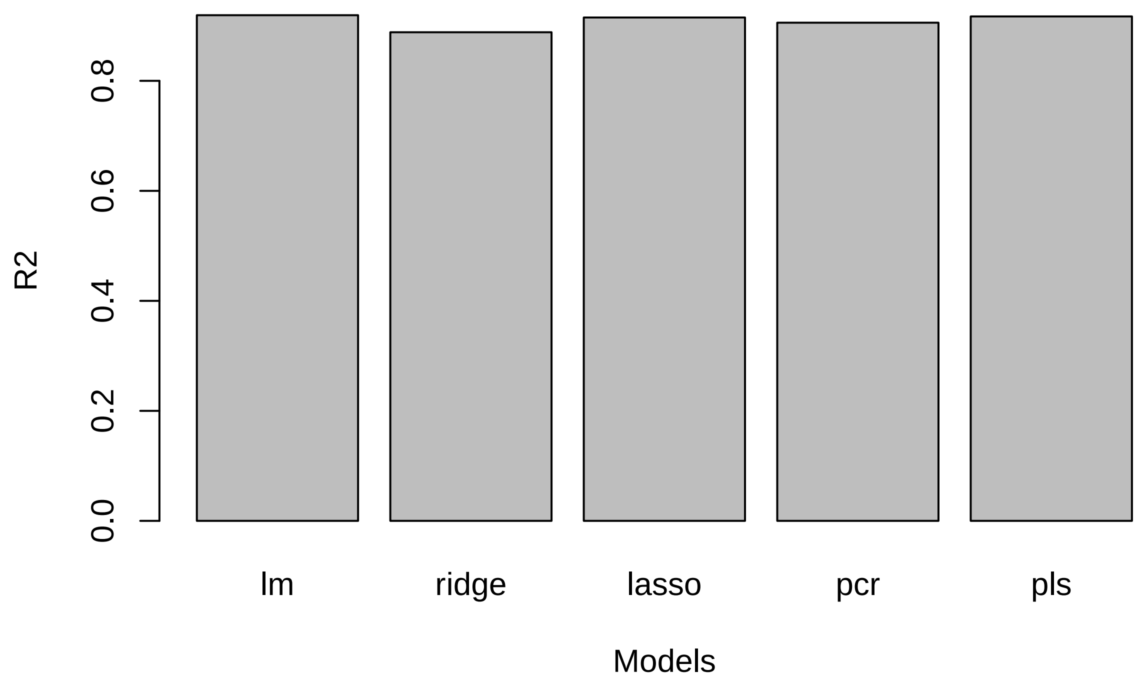
```
barplot(err.all, xlab="Models", ylab="Test MSE", names=c("lm", "ridge", "lasso", "pcr", "pls"))
```



All of the models gave similar results, ridge regression has the highest MSE, and lm, lasso, pls are all between 1700000-1800000.

```
test.avg = mean(y.test)
lm.r2 = 1 - mean((lm.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", "
```



```
table(c(lm.r2, ridge.r2, lasso.r2, pcr.r2, pls.r2), names=c("lm", "ridge", "lasso", "pcr", "pls") )
```

```
##                     names
##                      lasso lm pcr pls ridge
##   0.888325545967592     0  0   0   0     1
##   0.905694929904539     0  0   1   0     0
##   0.915114369914656     1  0   0   0     0
##   0.917057282127694     0  0   0   1     0
##   0.919326130355316     0  1   0   0     0
```

All of the R2 values are around 0.88 or above which means we can be confident in the accuracy of the model predictions.

6.10 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(42)
n <- 1000
p <- 20
X <- matrix(rnorm(n*p), n, p)
B <- sample(-10:10, 20)
B
```

```
##  [1]   4   5 -10   2   7  -4   3  -6   8  -1   1   0  -8  10  -3  -2  -7   6  -9
## [20]  -5
```

```
e <- rnorm(1000, mean= 0, sd=0.1)
Y <- X%*%B + e
df <- data.frame(X, Y)
```

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

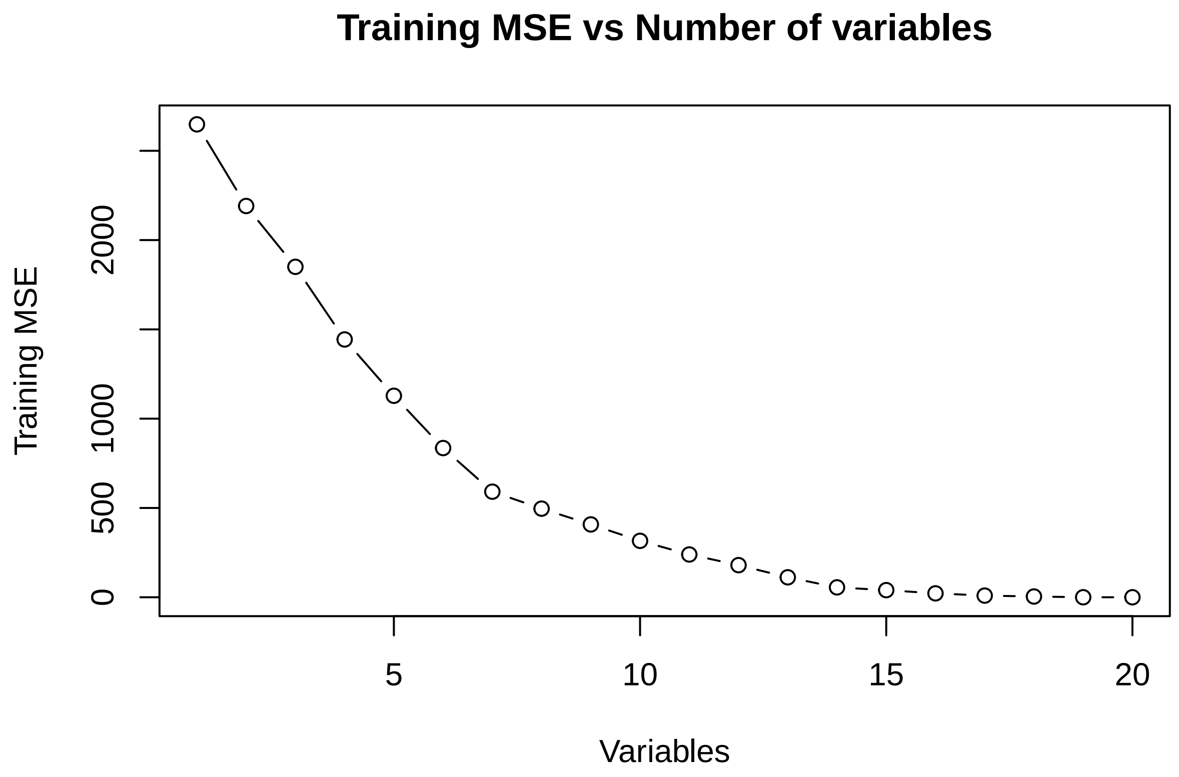
```
library(caTools)
sample <- sample.split(df$Y, 0.1)
train <- subset(df, sample == T)
test <- subset(df, sample == F)
```

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

```
library(leaps)
regfit.full <- regsubsets(Y ~ ., data= train, nvmax= 20)
reg.summary <- summary(regfit.full)

train.mse <- (reg.summary$rss)/length(train)
```

```
plot(1:20, train.mse, xlab= "Variables", ylab= "Training MSE", main= "Training MSE vs Number of variabl
```



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

```
library(HH)
```

```
## Warning: package 'HH' was built under R version 4.1.2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
## Loading required package: grid
```

```
## Loading required package: latticeExtra
```

```
##
## Attaching package: 'latticeExtra'
```

```
## The following object is masked from 'package:ggplot2':
##
##     layer
```

```
## Loading required package: multcomp
```

```
## Warning: package 'multcomp' was built under R version 4.1.2
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:boot':
##
##     aml
```

```
## Loading required package: TH.data
```

```
## Warning: package 'TH.data' was built under R version 4.1.2
```

```
##
## Attaching package: 'TH.data'
## The following object is masked from 'package:MASS':
##
##     geyser
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
##
## Attaching package: 'HH'
```

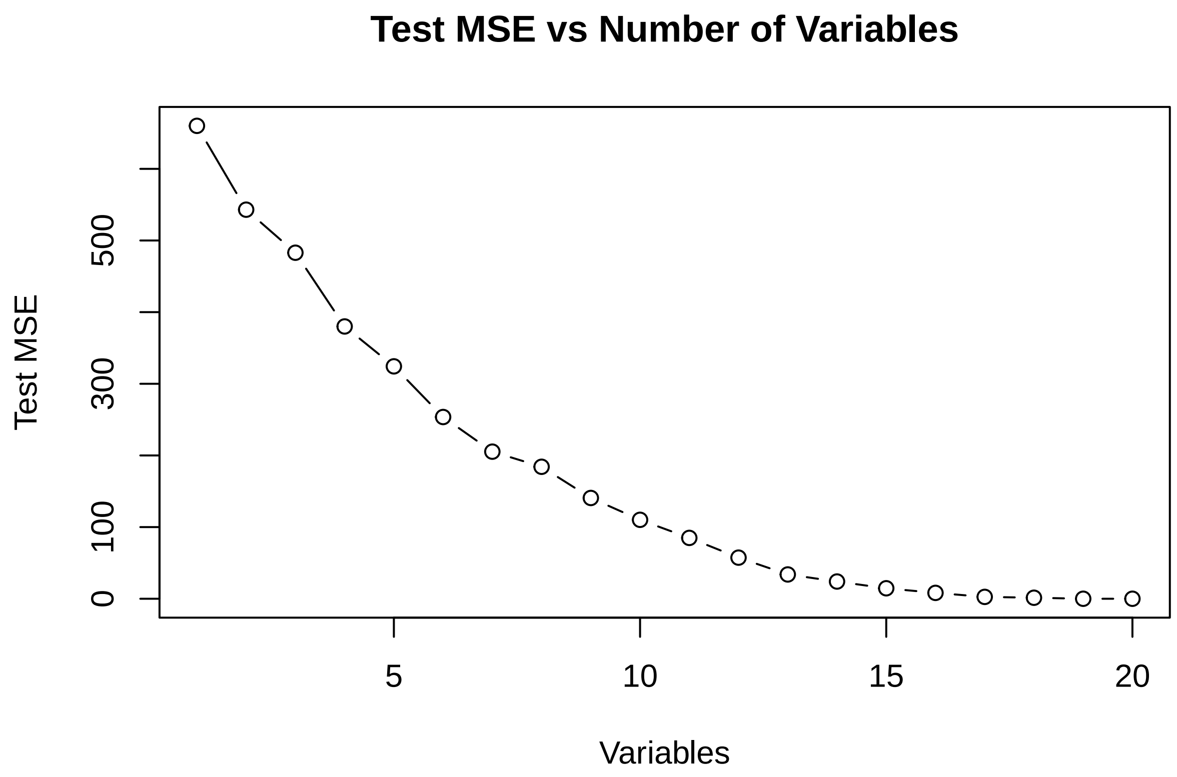
```
## The following object is masked from 'package:boot':
##
##     logit
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
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(1:20, test.mse, xlab= "Variables", ylab= "Test MSE", main= "Test MSE vs Number of Variables", pch=
```



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] 19
```

The minimum test mse occurs at a model size of 19 variables. As the flexibility of this model increases it is a better fit to the data set.

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, 19)
```

```
##  (Intercept)           X1           X2           X3           X4           X5
## -0.007424176  3.983432628  5.000439613 -9.993778569  2.014972835  7.005325110
##           X6           X7           X8           X9          X10          X11
## -3.997272378  3.025057084 -5.998249207  7.991177053 -1.010247909  0.987336642
##          X13          X14          X15          X16          X17          X18
## -8.008843753 10.011012725 -3.010372106 -1.992976504 -6.977821808  6.018409190
##          X19          X20
## -8.999705047 -5.001709474
```

```
B
```

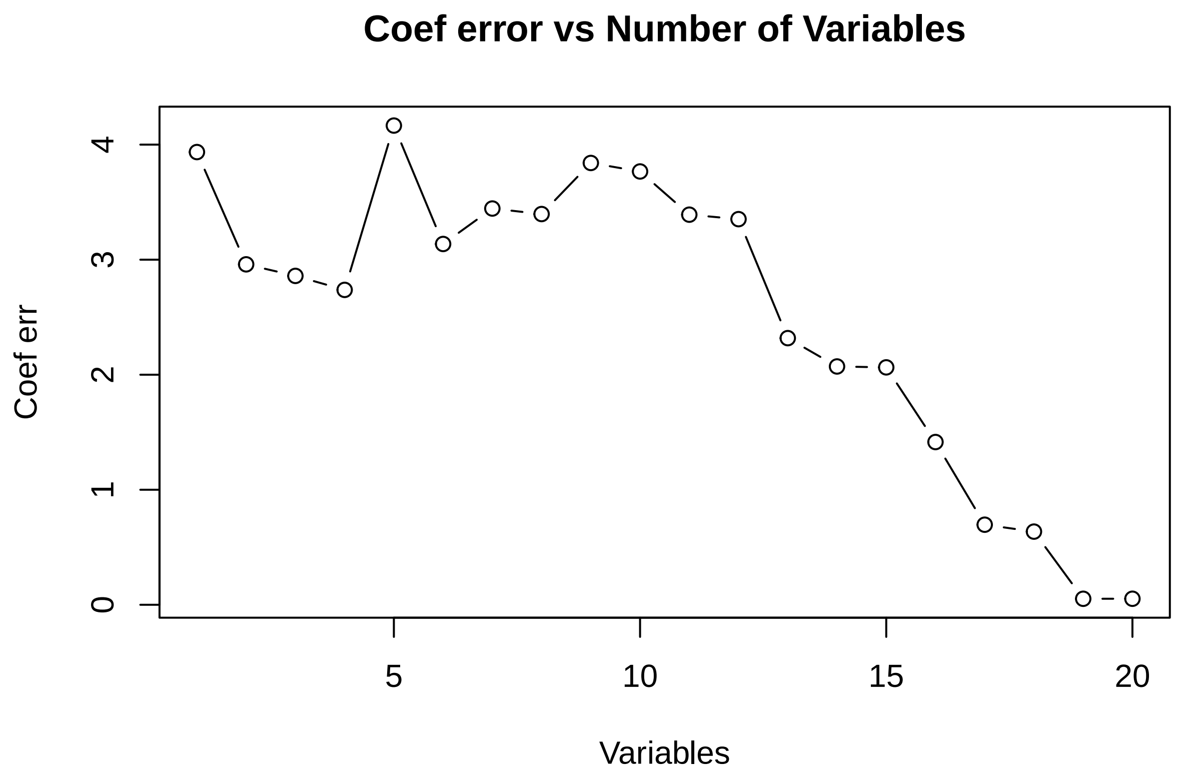
```
##  [1]   4   5 -10   2   7  -4   3  -6   8  -1   1   0  -8  10  -3  -2  -7   6  -9
## [20]  -5
```

g. Create a plot displaying G)p j=1(j − ˆr j)2 for a range of values of r, where ˆr j 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)?

```
B <- as.data.frame(t(B))
names(B) <- paste0('X', 1:(ncol(B)))
```

```
coef.err <- rep(NA, 20)
for(i in 1:20){
  a <- coef(regfit.full, i)
  coef.err[i] <- sqrt(sum(((a[-1] - B[names(a)[-1]])^2)))
}

plot(1:20, coef.err, xlab= "Variables", ylab= "Coef err", main= "Coef error vs Number of Variables", pc
```



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

```
## [1] 19
```

6.11 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.

```
library(ISLR2)
data(Boston)
Boston <- Boston
```

lasso

```
set.seed(42)
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]
```

```
library(glmnet)
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda= grid)

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

lasso.pred <- predict(lasso.mod, s= bestlam.lasso, newx = x[test,])
err.lasso <- mean((lasso.pred - y.test)^2)
lasso.coef <- predict(lasso.mod, type= "coefficients", s=bestlam.lasso)[1:13,]

lasso.coef
```

```
##   (Intercept)            zn         indus          chas           nox
##  18.236375396   0.051219129  -0.068135334  -0.987808007 -12.742082428
##            rm           age           dis           rad           tax
##   0.627984047   0.000000000  -1.204129314   0.654960616  -0.003800747
##       ptratio         lstat          medv
##  -0.374073785   0.122970665  -0.260274865
```

```
err.lasso
```

```
## [1] 16.46346
```

```
bestlam.lasso
```

```
## [1] 0.02803198
```

ridge

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

glm.mod <- glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
```

```r
glm.pred <- predict(glm.mod, s=bestlam.ridge, newx=x[test,])
err.ridge <- mean((glm.pred-y.test)^2)
glm.coef <- predict(glm.mod, type="coefficients", s=bestlam.ridge)[1:13,]

glm.coef
```

```
##   (Intercept)            zn         indus          chas           nox
##  9.3034165249  0.0390253535 -0.0888987268 -1.0154687190 -7.6559946425
##            rm           age           dis           rad           tax
##  0.6405802753 -0.0001648025 -0.9190104794  0.4990693145  0.0026979889
##       ptratio         lstat          medv
## -0.2376591066  0.1530155531 -0.2042215910
```

```r
err.ridge
```

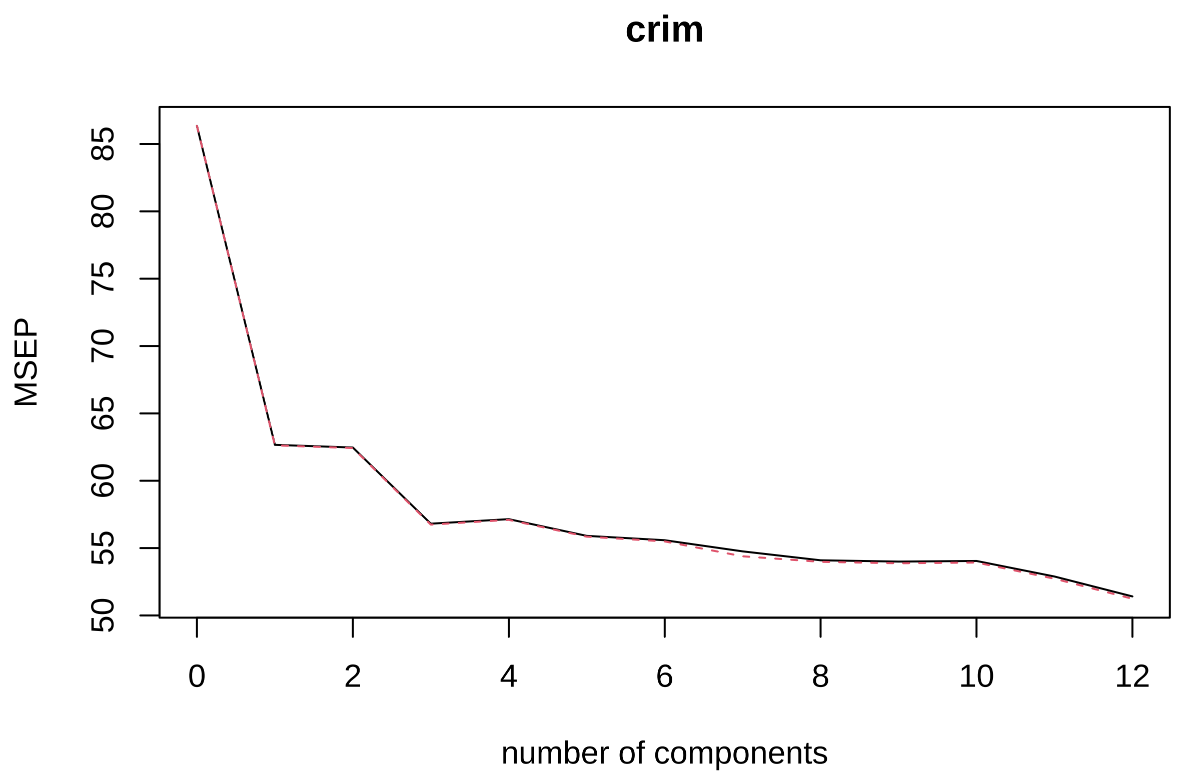
```
## [1] 14.55418
```

```r
bestlam.ridge
```

```
## [1] 0.5632277
```

pcr

```r
library(pls)
pcr.fit <- pcr(crim~., data=Boston, subset=train, scale=T, validation="CV")
validationplot(pcr.fit, val.type="MSEP")
```



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

err.pcr
```

```
## [1] 14.43794
```

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, crossvalidation, or some other reasonable alternative, as opposed to using training error.

```r
err.all <- c(err.ridge, err.lasso, err.pcr)
table(err.all, names= c("ridge", "lasso", "pcr"))
```

```
##                    names
## err.all             lasso pcr ridge
##   14.4379443733024     0   1     0
##   14.5541800987381     0   0     1
##   16.4634557052613     1   0     0
```

I would use either pcr or lasso since they have the smallest test MSE.

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

The lasso model involves 11 variables, one of them resulted in a zero coefficient.