Math 7553 – Spring 2018

HW #2 (Hand In)

Shuddha Chowdhury

Submission Date: 06/02/2018

Chapter 5: Exercise 8

a)

```
 set.seed(1)
 y = rnorm(100)
 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.

Ans:

Here, in this dataset n is 100 and p is 2.

The model used to generate the data in equation form is $Y = X - 2X^2 + \varepsilon$

b)

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

Ans:

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

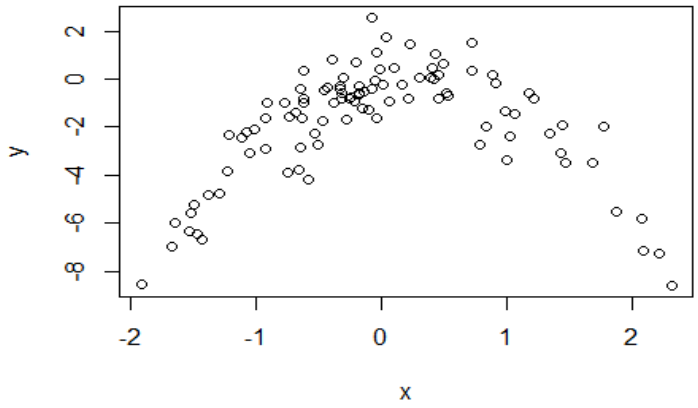


Fig 1: Scatterplot of X against Y

The above plot is a Quadratic Plot. In the X axis, the value of X is from -2 to 2 and in the Y axis the value is from -8 to 2.

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

i. $Y = \beta0 + \beta1X + \varepsilon$

ii. $Y = \beta0 + \beta1X + \beta2X2 + \varepsilon$

iii. $Y = \beta0 + \beta1X + \beta2X2 + \beta3X3 + \varepsilon$

iv. $Y = \beta0 + \beta1X + \beta2X2 + \beta3X3 + \beta4X4 + \varepsilon$

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

Ans:

```
library(boot)
Data = data.frame(x, y)
set.seed(1)
#### solution for equation number 1 ####
glm.fit = glm(y ~ x)
cv.glm(Data, glm.fit)$delta

## [1] 5.890979 5.888812

#### solution for equation number 2 ####
glm.fit = glm(y ~ poly(x, 2))
cv.glm(Data, glm.fit)$delta

## [1] 1.086596 1.086326
```

```
#### solution for equation number 3 ####
glm.fit = glm(y ~ poly(x, 3))
cv.glm(Data, glm.fit)$delta

## [1] 1.102585 1.102227

#### solution for equation number 4 ####
glm.fit = glm(y ~ poly(x, 4))
cv.glm(Data, glm.fit)$delta

## [1] 1.114772 1.114334

# Reference
```


1. GLM in R - Princeton University http://data.princeton.edu/R/glms.html

2. Package 'boot' - CRAN-R https://cran.r-project.org/web/packages/boot/boot.pdf

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

Ans:

```
library(boot)
Data = data.frame(x, y)
set.seed(100)
#### solution for equation number 1 ####
glm.fit = glm(y ~ x)
cv.glm(Data, glm.fit)$delta

## [1] 5.890979 5.888812

#### solution for equation number 3 ####
glm.fit = glm(y ~ poly(x, 2))
cv.glm(Data, glm.fit)$delta

## [1] 1.086596 1.086326

#### solution for equation number 3 ####
glm.fit = glm(y ~ poly(x, 3))
cv.glm(Data, glm.fit)$delta

## [1] 1.102585 1.102227

#### solution for equation number 4 ####
glm.fit = glm(y ~ poly(x, 4))
cv.glm(Data, glm.fit)$delta

## [1] 1.114772 1.114334
```

The results are the exactly same as what I got in (c) because LOOCV will remain same. LOOCV evaluates n folds of a single observation.

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

Ans: Among the models in (c), the quadratic polynomial had the lowest LOOCV (Leave-one-out cross-validation) test error rate. It matched with expectation because the true form of Y is similar with it.

(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?

Ans:

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8914  -0.5244   0.0749   0.5932   2.7796
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.8277     0.1041 -17.549   <2e-16 ***
## poly(x, 4)1   2.3164     1.0415   2.224   0.0285 *
## poly(x, 4)2 -21.0586     1.0415 -20.220   <2e-16 ***
## poly(x, 4)3  -0.3048     1.0415  -0.293   0.7704
## poly(x, 4)4  -0.4926     1.0415  -0.473   0.6373
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.084654)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 103.04  on 95  degrees of freedom
## AIC: 298.78
##
## Number of Fisher Scoring iterations: 2
```

Here we can see that the p-values show statistical significance of linear and quadratic terms. These results agree with the conclusions drawn based on the cross-validation results.

Reference

1. GLM in R - Princeton University http://data.princeton.edu/R/glms.html

Chapter 6: Exercise 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.

Ans:

```
set.seed(1)
X = rnorm(100)
eps = rnorm(100)

# Reference
```

1. https://www.rdocumentation.org/packages/compositions/versions/1.40-1/topics/rnorm

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

$$Y = \beta0 + \beta1X + \beta2X2 + \beta3X3 + \varepsilon,$$

where β0, β1, β2, and β3 are constants of your choice.

Ans: Let's select β0=3, β1=2, β2=−3 and β3=0.3

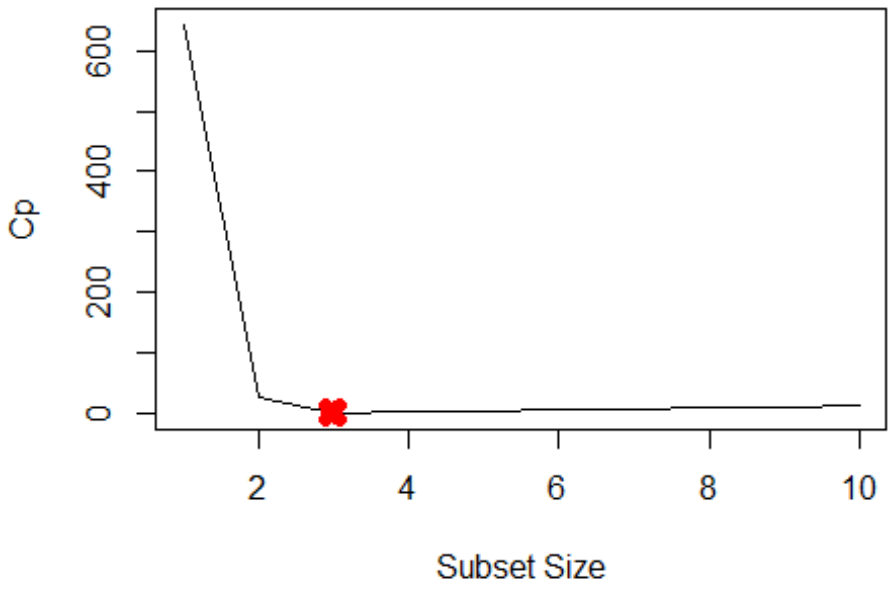
```
beta0 = 3
beta1 = 2
beta2 = -3
beta3 = 0.3
Y = beta0 + beta1 * X + beta2 * X^2 + beta3 * X^3 + eps
```

(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.

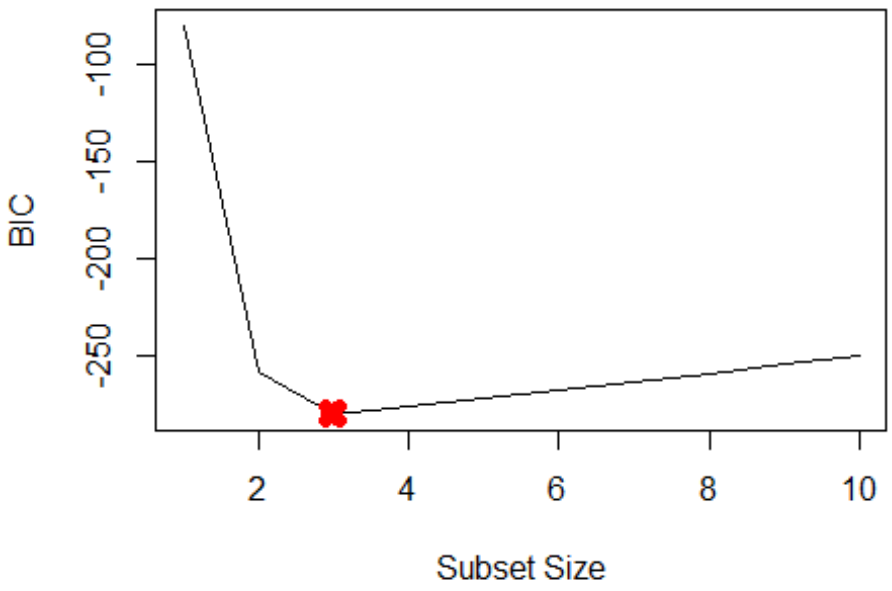
Ans: We use regsubsets to select best model having polynomial of X of degree 10.

```
library(leaps)

## Warning: package 'leaps' was built under R version 3.4.3

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)
```

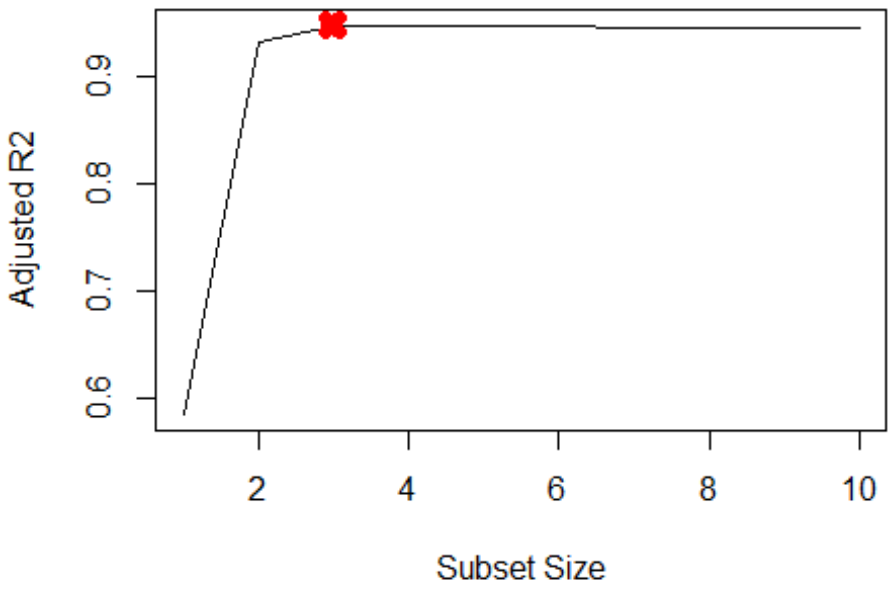
```r
# Find the model size for best cp, BIC and adjr2
which.min(mod.summary$cp)

## [1] 3

which.min(mod.summary$bic)

## [1] 3

which.max(mod.summary$adjr2)

## [1] 3

# Plot of cp, BIC and adjr2
plot(mod.summary$cp, xlab = "Subset Size", ylab = "Cp", pch = 20, type = "l")
points(3, mod.summary$cp[3], pch = 4, col = "red", lwd = 7)
```



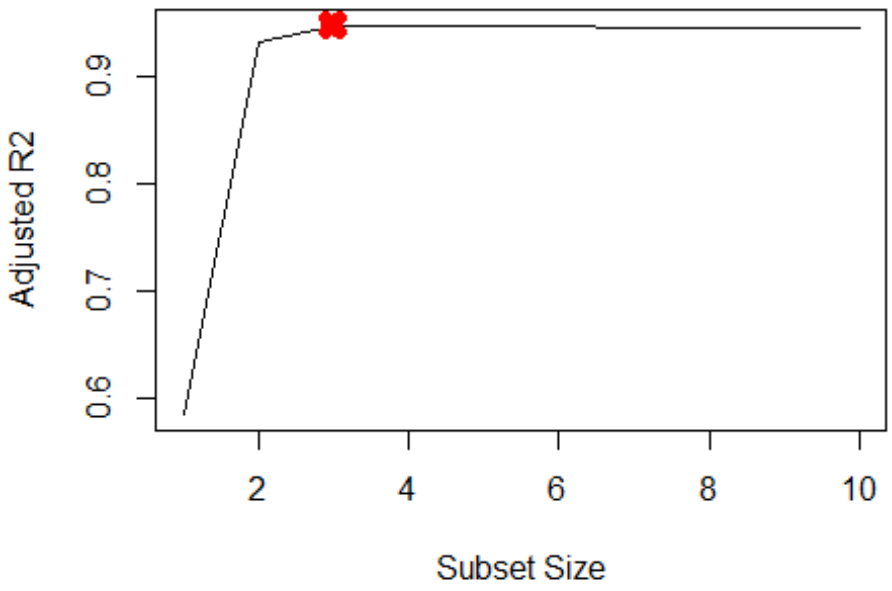

```r
# Plot cp, BIC and adjr2
plot(mod.summary$bic, xlab = "Subset Size", ylab = "BIC", pch = 20, type = "l
")
points(3, mod.summary$bic[3], pch = 4, col = "red", lwd = 7)
```




```
# Plot cp, BIC and adjr2
plot(mod.summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R2", pch = 20,
    type = "l")
points(3, mod.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
```




```
# Plot cp, BIC and adjr2
plot(mod.summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R2", pch = 20,
    type = "l")
points(3, mod.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
```



We find that with Cp, BIC and the adjusted R2 criteria, 3, 3 and 3 variable models are respectively picked.

```
coefficients(mod.full, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            3.07627412            2.35623596           -3.16514887
## poly(x, 10, raw = T)7
##            0.01046843
```

Here all statistics pic X^7 over X^3. The remaining coefficients are quite close to β s.

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

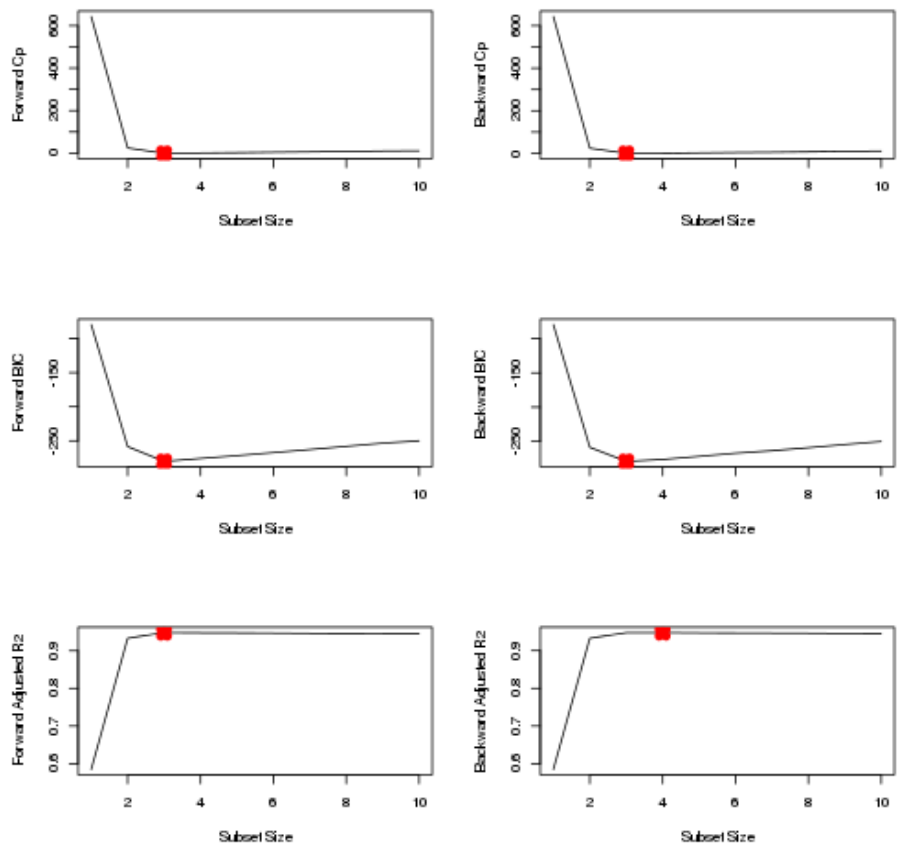
Ans: We need to fit forward and backward stepwise models to the data.

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

Ans:

```
mod.fwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,
    method = "forward")
```

```
mod.bwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,
    method = "backward")
fwd.summary = summary(mod.fwd)
bwd.summary = summary(mod.bwd)
which.min(fwd.summary$cp)

## [1] 3

which.min(bwd.summary$cp)

## [1] 3

which.min(fwd.summary$bic)

## [1] 3

which.min(bwd.summary$bic)

## [1] 3

which.max(fwd.summary$adjr2)

## [1] 3

which.max(bwd.summary$adjr2)

## [1] 3

# Plotting of the statistical data
par(mfrow = c(3, 2))
plot(fwd.summary$cp, xlab = "Subset Size", ylab = "Forward Cp", pch = 20,
type = "l")
points(3, fwd.summary$cp[3], pch = 4, col = "red", lwd = 7)
plot(bwd.summary$cp, xlab = "Subset Size", ylab = "Backward Cp", pch = 20,
type = "l")
points(3, bwd.summary$cp[3], pch = 4, col = "red", lwd = 7)
plot(fwd.summary$bic, xlab = "Subset Size", ylab = "Forward BIC", pch = 20,
    type = "l")
points(3, fwd.summary$bic[3], pch = 4, col = "red", lwd = 7)
plot(bwd.summary$bic, xlab = "Subset Size", ylab = "Backward BIC", pch = 20,
    type = "l")
points(3, bwd.summary$bic[3], pch = 4, col = "red", lwd = 7)
plot(fwd.summary$adjr2, xlab = "Subset Size", ylab = "Forward Adjusted R2",
    pch = 20, type = "l")
points(3, fwd.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
plot(bwd.summary$adjr2, xlab = "Subset Size", ylab = "Backward Adjusted R2",
    pch = 20, type = "l")
points(4, bwd.summary$adjr2[4], pch = 4, col = "red", lwd = 7)
```



From the above figure,We see that all statistics pick 3 variable models except stepwise with adjusted R2. The coefficients are given below.

```
coefficients(mod.fwd, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##               3.07627               2.35624              -3.16515
## poly(x, 10, raw = T)7
##               0.01047
```

```
coefficients(mod.bwd, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##               3.07888               2.41982              -3.17724
## poly(x, 10, raw = T)9
##               0.00187
```

```
coefficients(mod.fwd, id = 4)
```

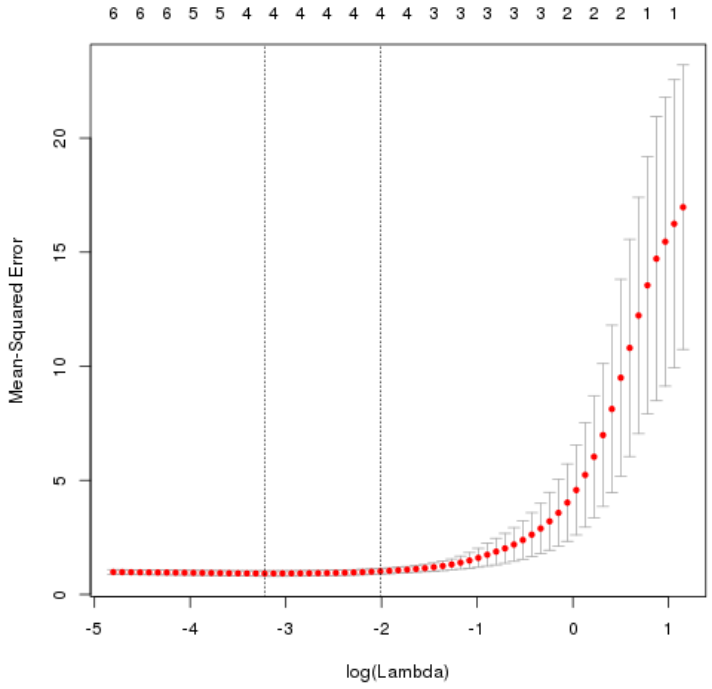
```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##              3.112359              2.369859             -3.275727
## poly(x, 10, raw = T)4 poly(x, 10, raw = T)7
##              0.027674              0.009997
```

Here we notice that the forward stepwise function picks X^7 over X^3. Backward stepwise function with 3 variables picks X^9 but backward stepwise with 4 variables picks X^4 and X^7. All the other coefficients are close to β s.

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.

Ans:

```
# We need to train (least absolute shrinkage and selection operator)lasso on the data.

library(glmnet)

xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]
mod.lasso = cv.glmnet(xmat, Y, alpha = 1)
best.lambda = mod.lasso$lambda.min
best.lambda

## [1] 0.03991

plot(mod.lasso)
```




```
# Next we fit the model on entire data set using best lambda value
```

```
best.model = glmnet(xmat, Y, alpha = 1)
predict(best.model, s = best.lambda, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)             3.0398151
## poly(x, 10, raw = T)1   2.2303371
## poly(x, 10, raw = T)2  -3.1033193
## poly(x, 10, raw = T)3   .
## poly(x, 10, raw = T)4   .
## poly(x, 10, raw = T)5   0.0498411
## poly(x, 10, raw = T)6   .
## poly(x, 10, raw = T)7   0.0008068
## poly(x, 10, raw = T)8   .
## poly(x, 10, raw = T)9   .
## poly(x, 10, raw = T)10  .
```

Lasso also picks X^5 over X^3. It also picks X^7 with negligible coefficient.

Reference:

1. Package 'glmnet' - https://cran.r-project.org/web/packages/glmnet/glmnet.pdf
2. Ridge Regression and the Lasso - https://www.r-bloggers.com/ridge-regression-and-the-lasso/

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

$$Y = \beta_0 + \beta_7 X7 + \varepsilon,$$

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

Ans: At first, we can create new Y with different $\beta_7 = 7$.

```
beta7 = 7
Y = beta0 + beta7 * X^7 + eps
# Make prediction 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)

# Finding the model size for best cp, BIC and adjr2
which.min(mod.summary$cp)
```

```
## [1] 2
```

```
which.min(mod.summary$bic)
```

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

```
which.max(mod.summary$adjr2)
```

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

```
coefficients(mod.full, id = 1)
```

```
##           (Intercept) poly(x, 10, raw = T)7
##                 2.959                 7.001
```

```
coefficients(mod.full, id = 2)
```

```
##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##                3.0705               -0.1417                7.0016
```

```
coefficients(mod.full, id = 4)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##                3.0763                0.2914               -0.1618
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
##               -0.2527                7.0091
```

We can observe that the BIC picks the most accurate and correct 1 variable model with matching coefficients. We also notice that other criteria also pick additional variables.

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

```
## [1] 12.37
```

```
best.model = glmnet(xmat, Y, alpha = 1)
predict(best.model, s = best.lambda, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)            3.820
## 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  6.797
## poly(x, 10, raw = T)8  .
## poly(x, 10, raw = T)9  .
## poly(x, 10, raw = T)10 .
```

Here Lasso picks the best 1 variable model but intercept is quite off. We notice here the value is 3.8 vs 3.

Chapter 6: Exercise 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.

Ans:

```
library(ISLR)
set.seed(11)
sum(is.na(College))
```

```
## [1] 0
```

```
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.

Ans:

Number of applications is the Apps variable.

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

```
## [1] 1538442
```

Here the Test RSS value is 1538442

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

Ans:

We can pick λ using College.train and then report error on College.test

```
library(glmnet)
```

```
train.mat = model.matrix(Apps~., data=College.train)
test.mat = model.matrix(Apps~., data=College.test)
grid = 10 ^ seq(4, -2, length=100)
```

```
mod.ridge = cv.glmnet(train.mat, College.train[, "Apps"], alpha=0,
lambda=grid, thresh=1e-12)
lambda.best = mod.ridge$lambda.min
lambda.best
```

```
## [1] 18.74
```

```
ridge.pred = predict(mod.ridge, newx=test.mat, s=lambda.best)
mean((College.test[, "Apps"] - ridge.pred)^2)
```

```
## [1] 1608859
```

We notice here that the Test RSS is slightly higher that OLS, 1608859.

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

Ans:

We can pick λ using College.train and report error on College.test

```
mod.lasso = cv.glmnet(train.mat, College.train[, "Apps"], alpha=1,
lambda=grid, thresh=1e-12)
lambda.best = mod.lasso$lambda.min
lambda.best
```

```
## [1] 21.54
```

```
lasso.pred = predict(mod.lasso, newx=test.mat, s=lambda.best)
mean((College.test[, "Apps"] - lasso.pred)^2)
```

```
## [1] 1635280
```

Again, Test RSS value is slightly higher that OLS, 1635280.

The coefficients is like this:

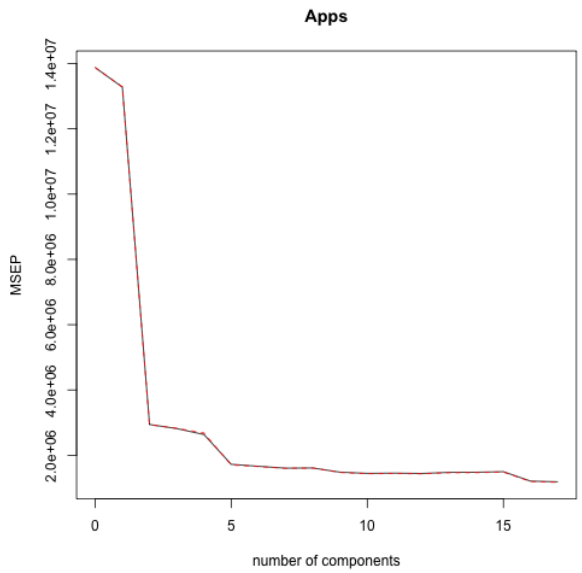
```
mod.lasso = glmnet(model.matrix(Apps~., data=College), College[, "Apps"],
alpha=1)
predict(mod.lasso, s=lambda.best, type="coefficients")
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) -6.038e+02
## (Intercept)  .
## PrivateYes  -4.235e+02
## Accept       1.455e+00
## Enroll      -2.004e-01
## Top10perc    3.368e+01
```

```
## Top25perc   -2.403e+00
## F.Undergrad  .
## P.Undergrad  2.086e-02
## Outstate    -5.782e-02
## Room.Board   1.246e-01
## Books        .
## Personal     1.833e-05
## PhD         -5.601e+00
## Terminal    -3.314e+00
## S.F.Ratio    4.479e+00
## perc.alumni -9.797e-01
## Expend       6.968e-02
## Grad.Rate    5.160e+00
```

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.

Ans :

```
# r pls library
library(pls)

##
## Attaching package: 'pls'
##
## The following object(s) are masked from 'package:stats':
##
##     loadings

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



```
pcr.pred = predict(pcr.fit, College.test, ncomp=10)
mean((College.test[, "Apps"] - data.frame(pcr.pred))^2)
```

```
## [1] 3014496
```

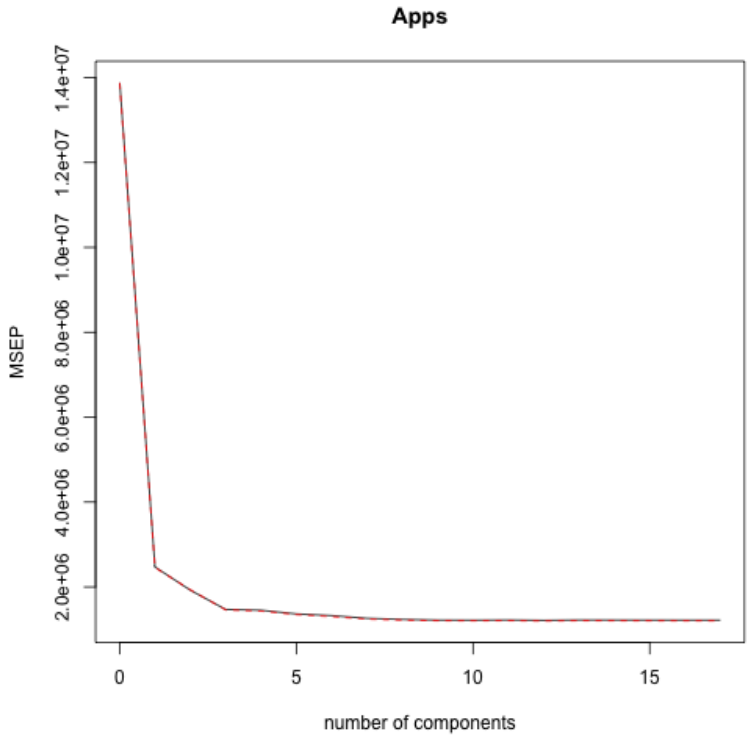
We notice that Test RSS for PCR is about 3014496.

Reference:
1. pls: Partial Least Squares and Principal Component Regression - https://cran.r-project.org/web/packages/pls/index.html

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.

Ans : We can use validation to fit pls.

```
pls.fit = plsr(Apps~., data=College.train, scale=T, validation="CV")
validationplot(pls.fit, val.type="MSEP")
```




```
pls.pred = predict(pls.fit, College.test, ncomp=10)
mean((College.test[, "Apps"] - data.frame(pls.pred))^2)
```

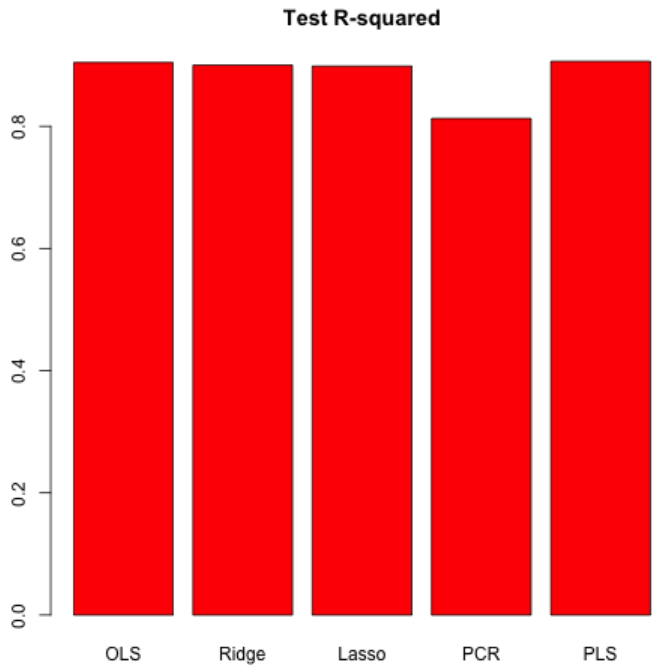
```
## [1] 1508987
```

We notice Test RSS for PLS is about 1508987.

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?

Ans: Here, Results for OLS, Lasso, Ridge are comparable. Lasso reduces the F.Undergrad and Books variables to zero and shrinks coefficients of other variables. Here are the test R2 for all models.

```
test.avg = mean(College.test[, "Apps"])
lm.test.r2 = 1 - mean((College.test[, "Apps"] - lm.pred)^2)
/mean((College.test[, "Apps"] - test.avg)^2)
ridge.test.r2 = 1 - mean((College.test[, "Apps"] - ridge.pred)^2)
/mean((College.test[, "Apps"] - test.avg)^2)
lasso.test.r2 = 1 - mean((College.test[, "Apps"] - lasso.pred)^2)
/mean((College.test[, "Apps"] - test.avg)^2)
pcr.test.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pcr.pred))^2)
/mean((College.test[, "Apps"] - test.avg)^2)
pls.test.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pls.pred))^2)
/mean((College.test[, "Apps"] - test.avg)^2)
barplot(c(lm.test.r2, ridge.test.r2, lasso.test.r2, pcr.test.r2,
pls.test.r2), col="red", names.arg=c("OLS", "Ridge", "Lasso", "PCR", "PLS"),
main="Test R-squared")
```



The above plot shows that test R2 for all models except PCR are around 0.9, with PLS having slightly higher test R2 than others. PCR has a smaller test R2 of less than 0.8. All models except PCR predict college applications with high accuracy.