HW5

Anish Mohan

November 1, 2015

Q1

- 1a. Training RSS will start decreasing. β_j's start increasing from 0 to s, hence the value of the training RSS will start decreasing as the β_j's get to their correct values.
- 1b. Test RSS will decrease initially and then increase. Test RSS will decrease as β_j's increase from 0. After a local minima that gives the best value for β_j's the Test RSS will start increasing as the the β_j's are determined from the training set.
- 1c. variance starts increasing β_j's=0 has a constant low variance independent of the data. Variance starts increasing as the s increases from 0.
- 1d. bias starts decreasing β_j's=0 has the highest bias as the model predicts a constant value. As s increases from 0, the bias will start decreasing.
- 1e. Irreducible error remains steady Irreducible by error cannot be determined and continues to stay steady.

Q2

```
+ 2a. Training RSS will start increasing
With $\lambda$=0, the solution is what get from oridinary least squares that
minimize the training MSE. As $\lambda$ starts increasing from 0, the
training error will start increasing as well.
```

- 2b. Test RSS will decrease initially and then increase. Test RSS will initially decrease as λ increases from 0 as the β_j's predicted from training set are able to predict value of the test set with error of margin. However after a certain point that models the best lambda and $beta_j$'s for the test set, the test RSS will start going up.
- 2c. variance starts decreasing λ=0 gives the least squares solution. As λ starts increasing the flexibility of the model starts decreasing and the variance of the model starts decreasing as well.
- 2d. bias starts increasing λ=0 gives the least squares solution. As λ starts increasing the flexibility of the model starts decreasing and the bias of the model starts increasing as well.

- 2e. Irreducible error remains steady Irreducible by error cannot be determined and continues to stay steady.

Q3

- 3a. For k, predictors, the best subset will have the smallest training RSS, because it looks at all k subsets and chooses the subset with lowest RSS.
- 3b. Cannot be reliably predicted and depends on the test data. Best-subset overfits to training data so if it captures the underlying model then the lowest test RSS could be through Best subset. However, forward and backward stepwise selections could also have the least test RSS.
- 3c.
    i. True Forward stepwise is incremental and k+1 the iteration contains all variables of kth iterarion and an additonal variable.
    ii. True Backward stepwise removes one element in each iteration. So kth iteration will have 1 less variable than in k+1 iteration
    iii. False It is not guaranteed to happen.
    iv. False It is not guaranteed to happen.
    v. False K+1 iteration could have elements not in kth iteration.

Q4

- 4a

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

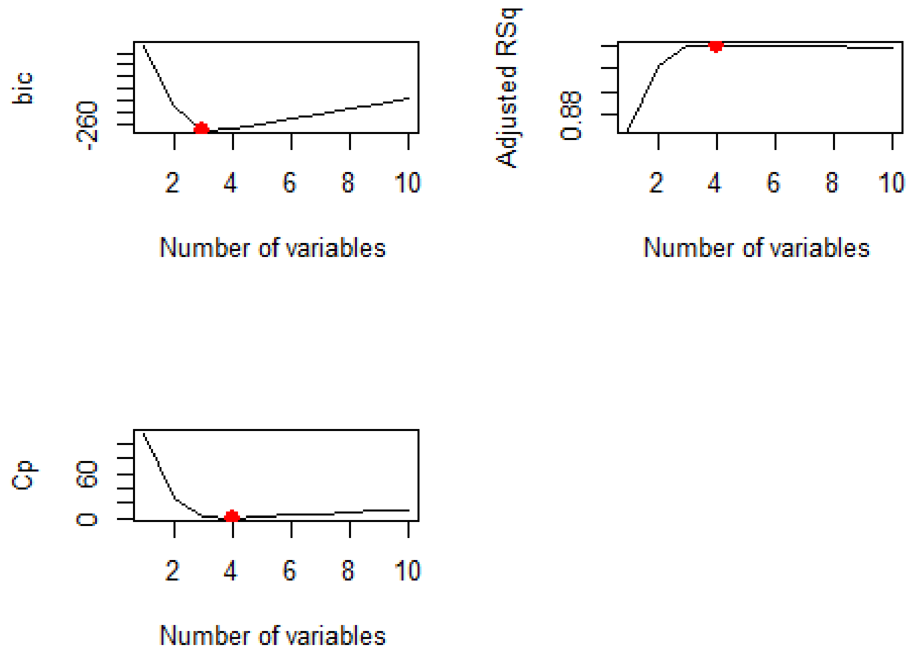
- 4b

```
X2=X^2
X3=X^3

beta0=1
beta1=1
beta2=1
beta3=1
Y=beta0+beta1*X+beta2*X2+beta3*X3+eps
```

- 4c

```
library(leaps)

## Warning: package 'leaps' was built under R version 3.2.2
```

```r
  df=data.frame(y=Y,x=X)
  regfit.X=regsubsets(y~poly(x,10,raw=T), data=df, nvmax=10)
  regfitx.summary=summary(regfit.X)

  par(mfrow=c(2,2))

  plot(regfitx.summary$bic, xlab="Number of variables", ylab="bic",type =
"l")
  k=which.min(regfitx.summary$bic)
  points(k,regfitx.summary$bic[k],col="red",cex=2,pch=20)

  plot(regfitx.summary$adjr2, xlab="Number of variables", ylab="Adjusted
RSq",type = "l")
  k=which.max(regfitx.summary$adjr2)
  points(k,regfitx.summary$adjr2[k],col="red",cex=2,pch=20)

  plot(regfitx.summary$cp, xlab="Number of variables",
ylab="Cp",type="l")
  k=which.min(regfitx.summary$cp)
  points(k,regfitx.summary$cp[k],col="red",cex=2,pch=20)

  coefficients(regfit.X,3)

##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             1.0615072             0.9752803             0.8762090
## poly(x, 10, raw = T)3
##             1.0176386

  coefficients(regfit.X,4)

##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            1.07200775            1.38745596            0.84575641
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)5
##            0.55797426            0.08072292
```



3 variable model picks X, X^2 and X^3 4 variable model picks X, X^2, X^3 and X^5

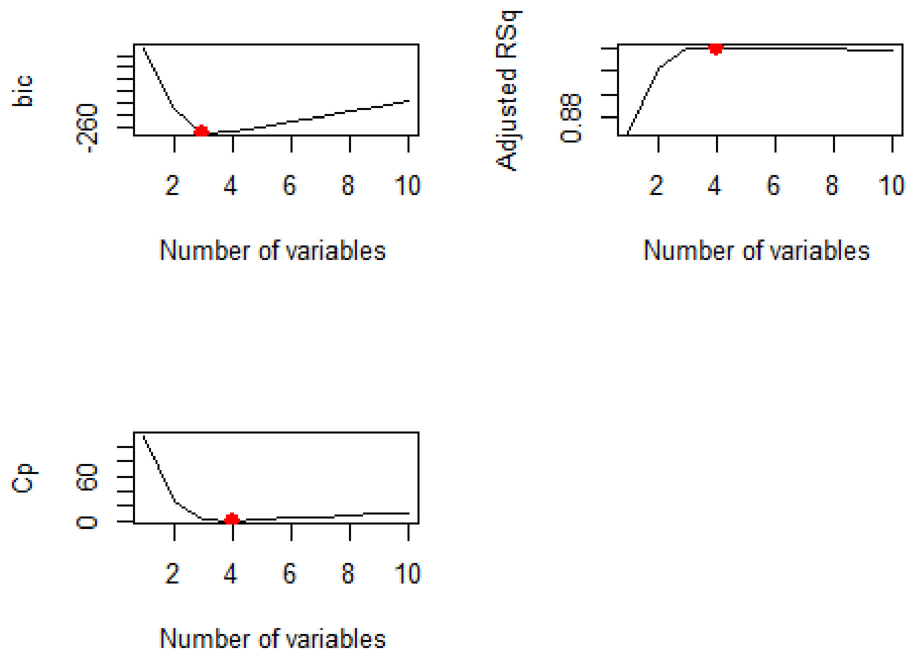
- 4d.

```
  regfit.fwd=regsubsets(y~poly(x,10,raw=T), data=df, nvmax=10,
method="forward")
  regfitfwd.summary=summary(regfit.X)

  par(mfrow=c(2,2))

  plot(regfitfwd.summary$bic, xlab="Number of variables", ylab="bic",type
= "l")
  k=which.min(regfitfwd.summary$bic)
  points(k,regfitfwd.summary$bic[k],col="red",cex=2,pch=20)

  plot(regfitfwd.summary$adjr2, xlab="Number of variables",
ylab="Adjusted RSq",type = "l")
  k=which.max(regfitfwd.summary$adjr2)
  points(k,regfitfwd.summary$adjr2[k],col="red",cex=2,pch=20)

  plot(regfitfwd.summary$cp, xlab="Number of variables",
ylab="Cp",type="l")
  k=which.min(regfitfwd.summary$cp)
  points(k,regfitfwd.summary$cp[k],col="red",cex=2,pch=20)

  coefficients(regfit.fwd,3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             1.0615072             0.9752803             0.8762090
## poly(x, 10, raw = T)3
##             1.0176386
```

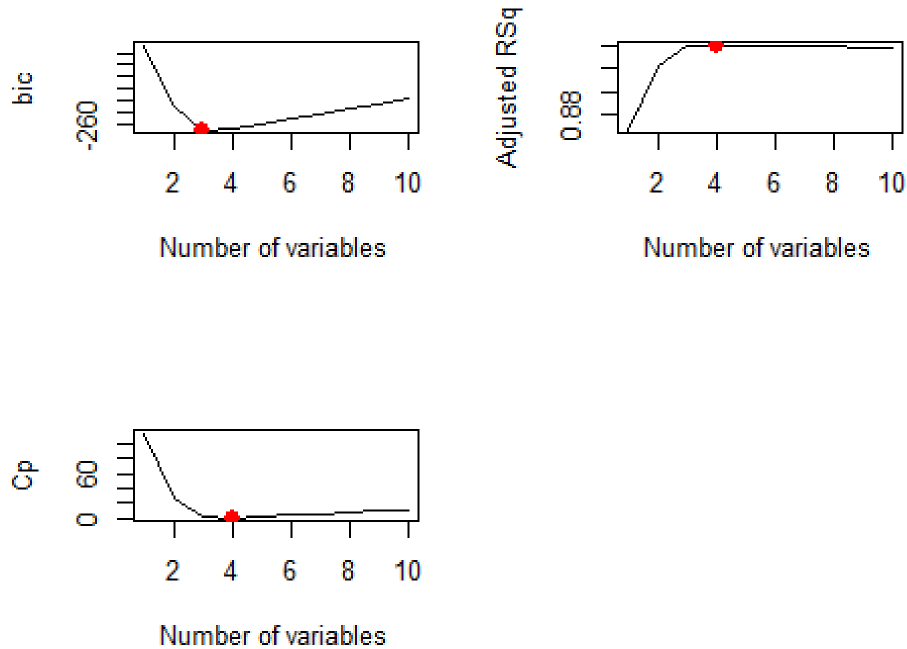
```
  coefficients(regfit.fwd,4)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            1.07200775            1.38745596            0.84575641
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)5
##            0.55797426            0.08072292
```




```
  #Backward
  regfit.bwd=regsubsets(y~poly(x,10,raw=T), data=df, nvmax=10,
method="backward")
  regfitbwd.summary=summary(regfit.X)

  par(mfrow=c(2,2))
  plot(regfitbwd.summary$bic, xlab="Number of variables", ylab="bic",type
= "l")
  k=which.min(regfitbwd.summary$bic)
  k

## [1] 3

  points(k,regfitbwd.summary$bic[k],col="red",cex=2,pch=20)
```

```r
  plot(regfitbwd.summary$adjr2, xlab="Number of variables",
ylab="Adjusted RSq",type = "l")
  k=which.max(regfitbwd.summary$adjr2)
  k

## [1] 4

  points(k,regfitbwd.summary$adjr2[k],col="red",cex=2,pch=20)

  plot(regfitbwd.summary$cp, xlab="Number of variables",
ylab="Cp",type="l")
  k=which.min(regfitbwd.summary$cp)
  k

## [1] 4

  points(k,regfitbwd.summary$cp[k],col="red",cex=2,pch=20)

  coefficients(regfit.bwd,3)

##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             1.0615072             0.9752803             0.8762090
## poly(x, 10, raw = T)3
##             1.0176386

  coefficients(regfit.bwd,4)

##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##            1.15670295            1.03082564            0.59010182
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)4
##            0.99086710            0.06978542
```

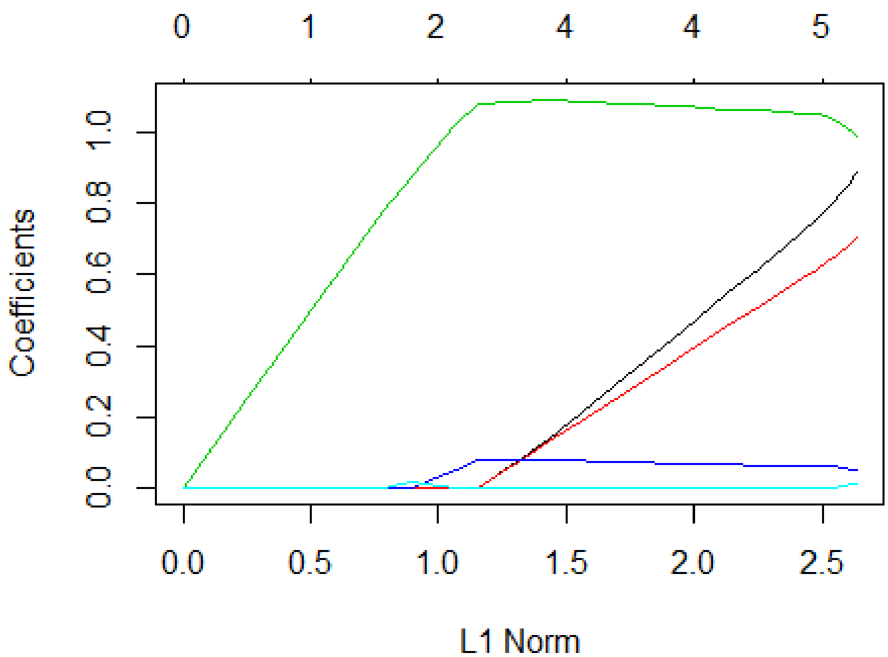


Statistics from Forward and Backward models show 3 and 4 variable models are optimal. Additionally,3 variable model picks X, X^2 and X^3 and 4 variable model picks X, X^2, X^3 and X^5. These results are similar to results in 4c.

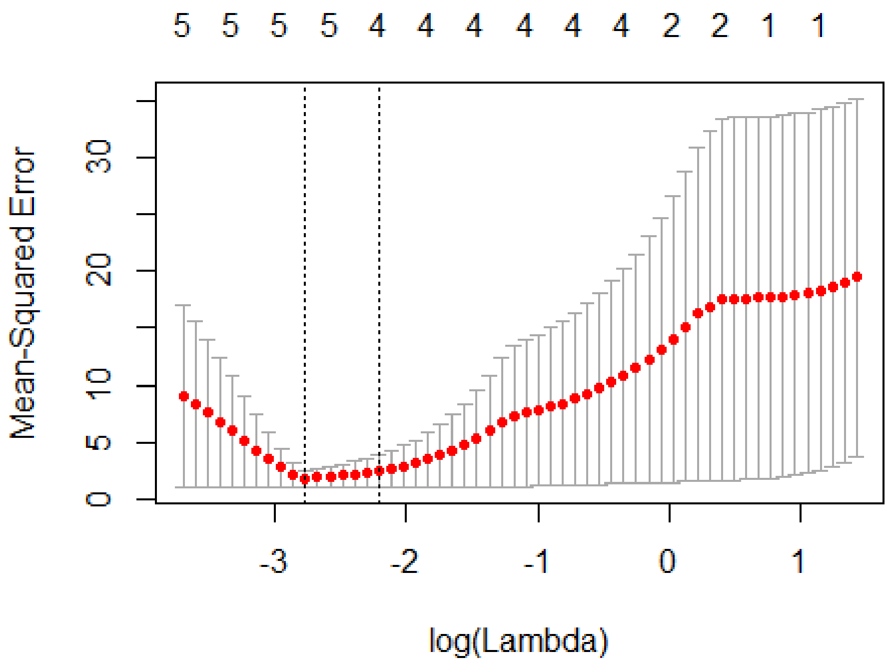
- 4e

```
par(mfrow=c(1,1))
library(glmnet)

## Warning: package 'glmnet' was built under R version 3.2.2

## Loading required package: Matrix
## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.2.2

## Loaded glmnet 2.0-2

xnew=model.matrix(y~poly(x,10,raw=T),data=df)[,-1]
grid=10^seq(10,-2,length=100)

set.seed(1)
train=sample(1:nrow(xnew),nrow(xnew)/2)
test=(-train)
Y.test=Y[test]
```

```r
lasso.mod=glmnet(xnew[train,],Y[train], alpha=1,lambda=grid)
plot(lasso.mod)
```




```r
cv.out=cv.glmnet(xnew[train,], Y[train], alpha=1)
plot(cv.out)
```




```
  bestlam=cv.out$lambda.min

  lasso.pred=predict(lasso.mod,s=bestlam,newx=xnew[test,],
type="coefficients")
  lasso.pred

## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)            1.047519228
## poly(x, 10, raw = T)1  0.791739971
## poly(x, 10, raw = T)2  0.640112595
## poly(x, 10, raw = T)3  1.041070112
## poly(x, 10, raw = T)4  0.060629481
## poly(x, 10, raw = T)5  0.001376594
## poly(x, 10, raw = T)6  .
## poly(x, 10, raw = T)7  .
## poly(x, 10, raw = T)8  .
## poly(x, 10, raw = T)9  .
## poly(x, 10, raw = T)10 .
```

Lasso predicts a model using X^1,X^2, X^3, X^4, X^5. All variables except X^4 were chosen with backward, forward subselection example above.

- 4f

```
set.seed(1)
beta7 = 1
```

```r
Y=beta0+beta7*X^7+eps
df=data.frame(y=Y,x=X)
regfitx7=regsubsets(y~poly(x,10,raw=T), data=df, nvmax=10)
regfitx7.summary=summary(regfitx7)
k=which.min(regfitx7.summary$bic)
coefficients(regfitx7,k)

##           (Intercept) poly(x, 10, raw = T)7
##             0.9589402             1.0007705

k=which.min(regfitx7.summary$cp)
coefficients(regfitx7,k)

##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##             1.0704904            -0.1417084             1.0015552

k=which.max(regfitx7.summary$adjr2)
coefficients(regfitx7,k)

##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             1.0762524             0.2914016            -0.1617671
## poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
##            -0.2526527             1.0091338
```

BIC picks the correct 1 variable model with X^7; Cp picks 2 variable model with X^2 and X^7 and Adjusted R^2 picks a 4 variable model with X^1, X^2,X^3 and X^7

```r
xnew=model.matrix(y~poly(x,10,raw=T),data=df)[,-1]
cv.out=cv.glmnet(xnew, Y, alpha=1)
bestlam=cv.out$lambda.min

lasso.pred=predict(lasso.mod,s=bestlam,newx=xnew, type="coefficients")
lasso.pred

## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)            1.725094858
## poly(x, 10, raw = T)1  .
## poly(x, 10, raw = T)2  .
## poly(x, 10, raw = T)3  0.826320444
## poly(x, 10, raw = T)4  .
## poly(x, 10, raw = T)5  0.008857532
## poly(x, 10, raw = T)6  .
## poly(x, 10, raw = T)7  .
## poly(x, 10, raw = T)8  .
## poly(x, 10, raw = T)9  .
## poly(x, 10, raw = T)10 .
```

Lasso picks the 2 variable model with X^3 and X^5. The intercept value is 1.7 as compared to 1.07 in the best subset selection

Q5.

- 5a.

```
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.2.2

set.seed(1)
sum(is.na(College))

## [1] 0

train=sample(1:nrow(College),nrow(College)/2)
test=(-train)
College.train=College[train,]
College.test=College[test,]
```

- 5b.

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

## [1] 1108531
```

RSS= 1108531

- 5c.

```
library(glmnet)
ridge_train=model.matrix(Apps~.,data=College.train)
ridge_test=model.matrix(Apps~.,data=College.test)
grid=10^seq(4,-2,length=100)
ridge.mod=cv.glmnet(ridge_train, College.train[,"Apps"], alpha=0,
lambda=grid)
bestlam=ridge.mod$lambda.min
bestlam

## [1] 0.1873817

ridge.pred=predict(ridge.mod, newx=ridge_test, s=bestlam)
mean((College.test[,"Apps"]-ridge.pred)^2)

## [1] 1108062
```

RSS= 1108062. The test RSS is comparable to the result from least squares fit.

- 5d.

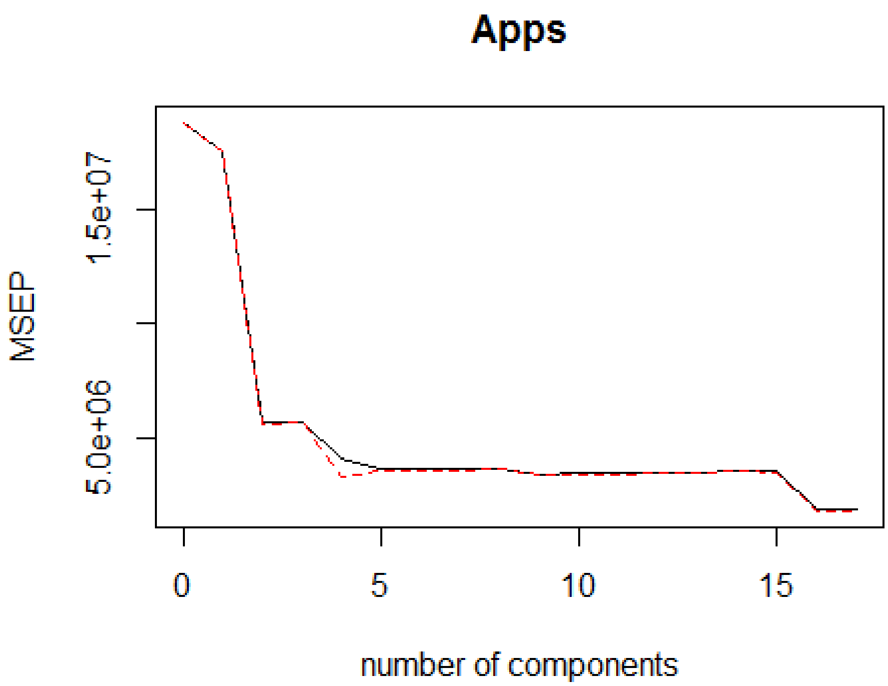
```
lasso.mod=cv.glmnet(ridge_train, College.train[,"Apps"], alpha=1,
lambda=grid)
bestlam=lasso.mod$lambda.min
```

```r
lasso.pred=predict(lasso.mod,newx=ridge_test, s=bestlam)
mean((College.test[,"Apps"]-lasso.pred)^2)

## [1] 1026783

predict(lasso.mod,newx=ridge_test, s=bestlam, type="coefficients")

## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -4.230907e+02
## (Intercept)  .
## PrivateYes  -4.926762e+02
## Accept       1.542260e+00
## Enroll      -4.183196e-01
## Top10perc    4.768619e+01
## Top25perc   -7.845864e+00
## F.Undergrad -5.064600e-03
## P.Undergrad  .
## Outstate    -5.204703e-02
## Room.Board   1.871769e-01
## Books        7.387966e-04
## Personal     .
## PhD         -4.068964e+00
## Terminal    -3.303902e+00
## S.F.Ratio    .
## perc.alumni -2.127554e+00
## Expend       3.204866e-02
## Grad.Rate    2.863551e+00
```

RSS error (1026783) is lower than ridge and least squares

- 5e.

```r
library(pls)

## Warning: package 'pls' was built under R version 3.2.2

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

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



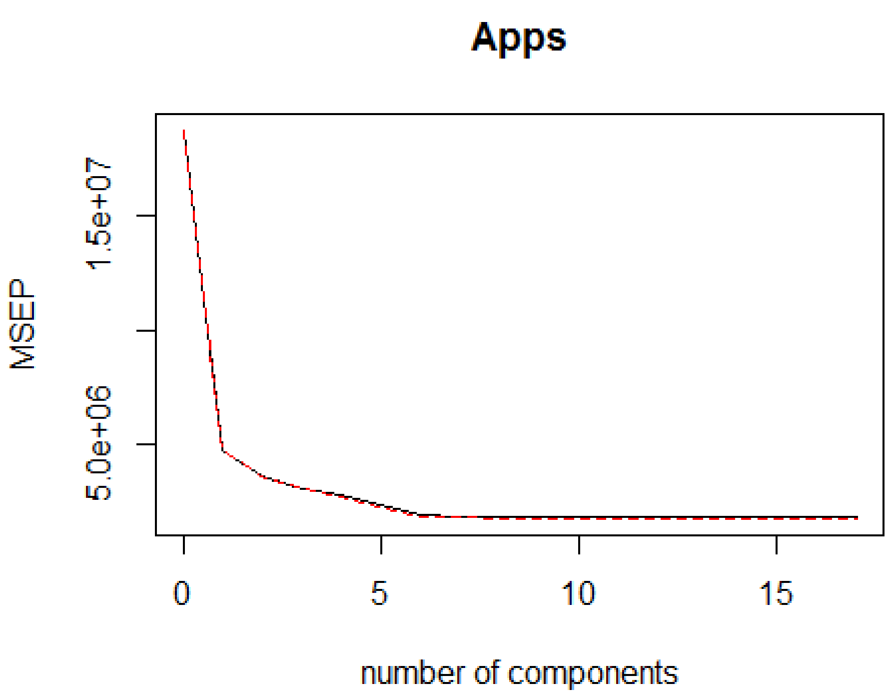

```
summary(pcr.fit)

## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6
comps
## CV            4335     4184     2372     2376     2009     1896
1896
## adjCV         4335     4184     2368     2374     1811     1883
1887
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13
comps
## CV        1897     1912     1847      1848      1854      1857
1860
## adjCV     1888     1904     1835      1837      1845      1848
1851
##        14 comps  15 comps  16 comps  17 comps
## CV         1879      1887      1353      1355
## adjCV      1886      1856      1335      1337
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
```

```
## X       31.216    57.68    64.73    70.55    76.33    81.30    85.01
## Apps     6.976    71.47    71.58    83.32    83.44    83.45    83.46
##        8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 
comps
## X        88.40    91.16     93.36     95.38     96.94     97.96     
98.76
## Apps     83.47    84.53     84.86     84.98     84.98     84.99     
85.24
##        15 comps  16 comps  17 comps
## X         99.40     99.87    100.00
## Apps      90.87     93.93     93.97

pcr.pred=predict(pcr.fit, College.test, ncomp=16)
mean((College.test[,"Apps"]- data.frame(pcr.pred))^2)

## [1] 1166897
```

Test RSS using 16 components (1166897) is higher than lasso, ridge and least squares

- 5f

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




```
summary(pls.fit)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6
comps
## CV            4335     2176     1889     1748     1663     1517
1364
## adjCV         4335     2171     1884     1738     1631     1483
1345
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13
comps
## CV        1353     1332     1327      1324      1325      1324
1323
## adjCV     1334     1315     1311      1309      1309      1309
1307
##        14 comps  15 comps  16 comps  17 comps
## CV         1322      1322      1323      1323
## adjCV      1306      1307      1307      1307
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       26.91    43.08    63.26    65.16    68.50    73.75    76.10
## Apps    76.64    83.93    87.14    91.90    93.49    93.85    93.91
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14
comps
## X       79.03    81.76     85.41     89.03     91.38     93.31
95.43
## Apps    93.94    93.96     93.96     93.96     93.97     93.97
93.97
##       15 comps  16 comps  17 comps
## X        97.41     98.78    100.00
## Apps     93.97     93.97     93.97
```

```r
pls.pred=predict(pls.fit, College.test, ncomp=14)
mean((College.test[,"Apps"]- data.frame(pls.pred))^2)
```

```
## [1] 1112475
```

Test RSS using 14 components (1112475) is higher than lasso, ridge and least squares and smaller than pcr

- 5g. Lasso gave the best results. Fit using Ridge and Least Squares was similar to lasso and the Test RSS was comparable.The Test RSS using Principal components regression and partial least squares were within the range of 10% Test RSS reported by Lasso.

Q6

- 6a.

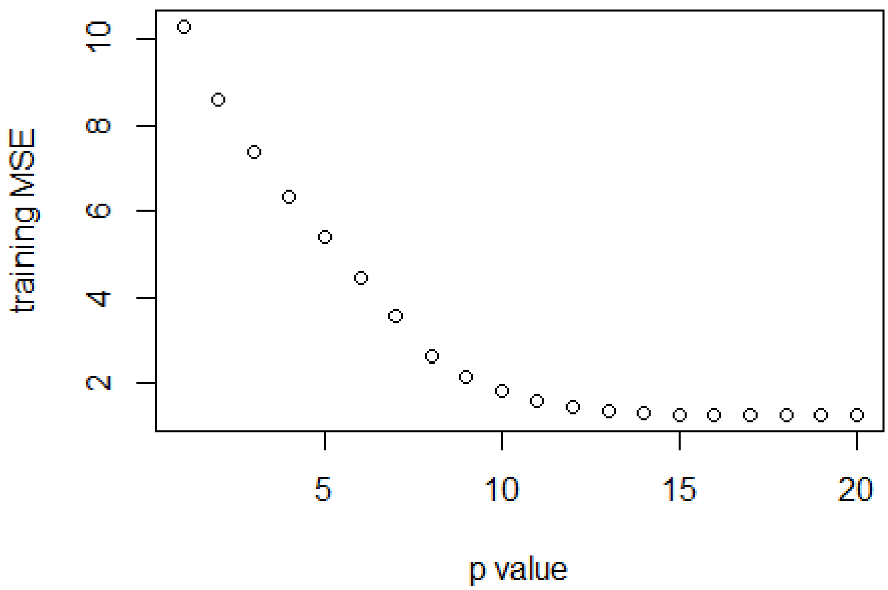
```
set.seed(1)
library(leaps)
n=100
p=20
x=matrix(rnorm(n*p), nrow=n, ncol=p)
beta=rnorm(p)
eps=rnorm(p)

beta[5]=0
beta[9]=0
beta[12]=0

y=x %*% beta + eps
```

- 6b.

```
test=sample(1:nrow(x),nrow(x)/10)
train=(-test)
x.train=x[train,]
x.test=x[test,]
y.train=y[train,]
y.test=y[test,]
```

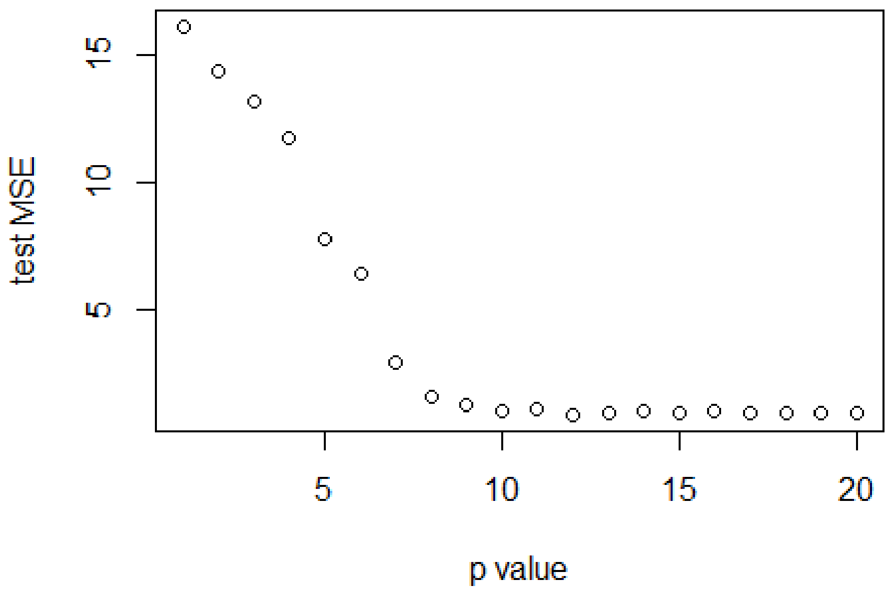
- 6c.

```
df=data.frame(x=x,y=y)
regfit =regsubsets(y~.,data=df, nvmax=p)
val.errors=rep(NA,p)

train.mat=model.matrix(y~.,data=df[train,])
test.mat=model.matrix(y~.,data=df[test,])

for(i in 1:p) {
 coefi=coef(regfit,id=i)
 pred=train.mat[,names(coefi)]%*%coefi
 val.errors[i]=mean((df$y[train]-pred)^2)
}

plot(val.errors, xlab="p value", ylab="training MSE")
```



- 6d.

```
val.errors=rep(NA,p)
for(i in 1:p) {
 coefi=coef(regfit,id=i)
 pred=test.mat[,names(coefi)]%*%coefi
 val.errors[i]=mean((df$y[test]-pred)^2)
}

plot(val.errors, xlab="p value", ylab="test MSE")
```



- 6e.

```
which.min(val.errors)

## [1] 12
```

12 parameter model has the smallest test mse

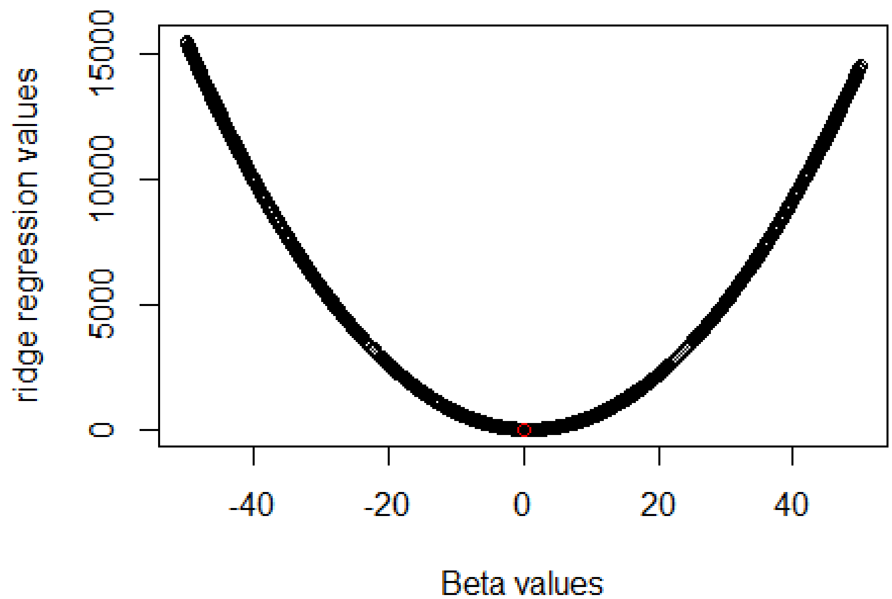
- 6f.

```
coef(regfit,12)

## (Intercept)         x.1         x.2         x.3         x.4
x.6
## -0.03109649 -0.88260240 -1.86261823  1.73345042  0.58266354
0.71656900
##         x.8        x.13        x.15        x.16        x.17
x.18
## -1.44000805  0.40443299  1.04729141  1.26597138 -0.69084672 -
0.83238264
##        x.20
##  1.38086014
```

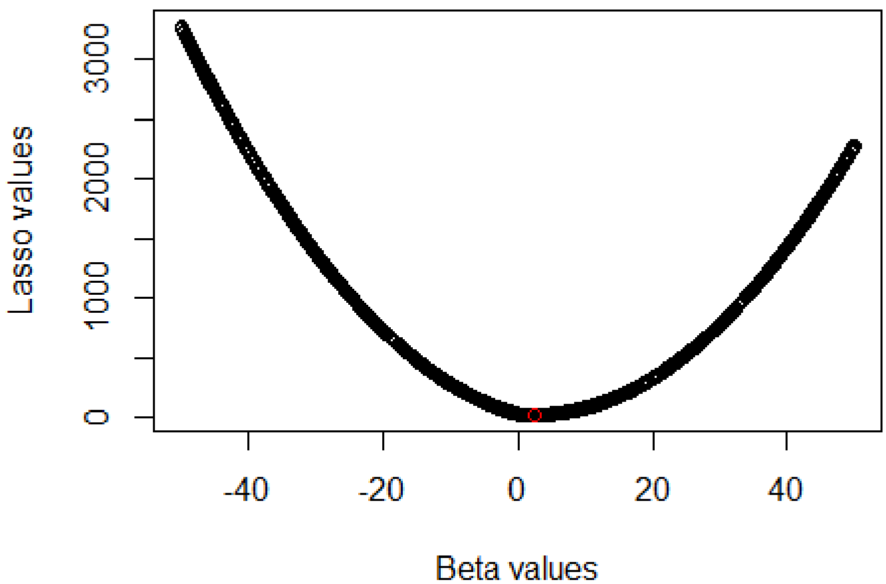
- 6g.

Q7.

```
lambda=5
beta=seq(-50,50,0.1)
y=5
f=(y-beta)^2 + lambda*(beta^2)
plot(beta,f, xlab="Beta values", ylab="ridge regression values")
beta_r=1/(1+lambda)
new_f=(y-beta_r)^2+lambda*(beta_r^2)
points(beta_r,new_f,col="red")
```




```
lambda=5
beta=seq(-50,50,0.1)
y=5
f=(y-beta)^2 + lambda*(abs(beta))
plot(beta,f, xlab="Beta values", ylab="Lasso values")
beta_r=y-lambda/2
new_f=(y-beta_r)^2+lambda*abs(beta_r)
points(beta_r,new_f,col="red")
```



Lasso values
0
1000
2000
3000
-40
-20
0
20
40
Beta values