Assignment 5- Chap 6

Problem 1–Exercise 1

(a) Best subset selection can have the smallest training error since it will consider all the possible model that have minimum training error, whereas the other two methods kidda depands on the first predictors they select and choose the predictors that can minimize the RSS.

(b) Best subset selection is very possible to have the smallest test RSS since it can consider all the possible model that can minimize the training error, whereas the other two methods considers less options than the best subet selection therefore the two method will be hard to outperform the best subset selection.

(c) i. true ii.true iii. false iv. false v. false

Problem 2–Exercise 3

(a) iv Steadily decrease. As s increases from 0, the beta will increase from 0 to their least square estimate values
(b) ii decrease initially then eventually start increasing in a U shape. When s = 0, the test error will be extremely large, since all the estimate is 0, except intercept, then after increase s, the estimate will gradually fit with the true model, but eventually, the test error will increase again, since the model start overfitting the data.
(c) iii steadily increases, as s increase, the estimate will start to fit the dataset, and eventually fit the data with the very high variance.
(d) iv Steadily decrease. As s increases from 0, the bias will become smaller.
(e) v remain constant, irreducible error is always there and constant, no matter how we increase s.

Problem 3–Exercise 8

(a)

```
x = rnorm(100)
noice = rnorm(100)
```

(b)

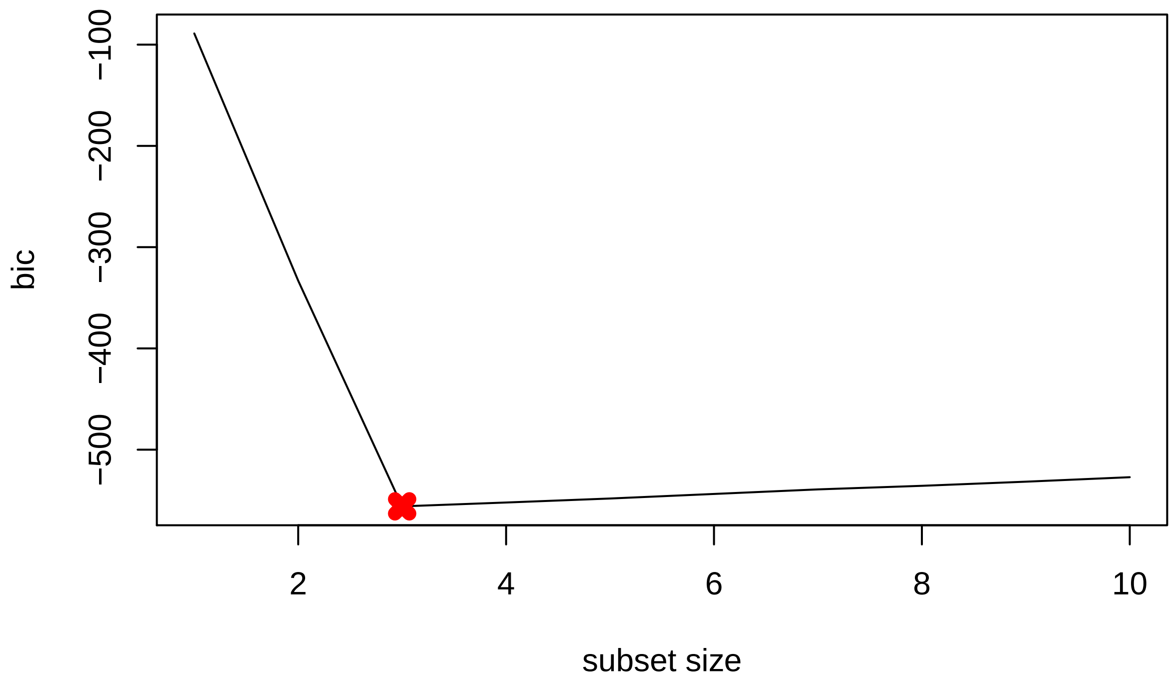
```
beta0 = 5
beta1 = 7
beta2 = -5
beta3 = 0.5
y = beta0 + beta1 * x + beta2 * x^2 + beta3 * x^3 + noice
```

(c)

```
library(leaps)
data= data.frame(y=y, x= x)
bs1 = regsubsets(y ~ poly(x,10,raw=T), data= data, nvmax=10)
bs.summary= summary(bs1)
which.min(bs.summary$bic)
```

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

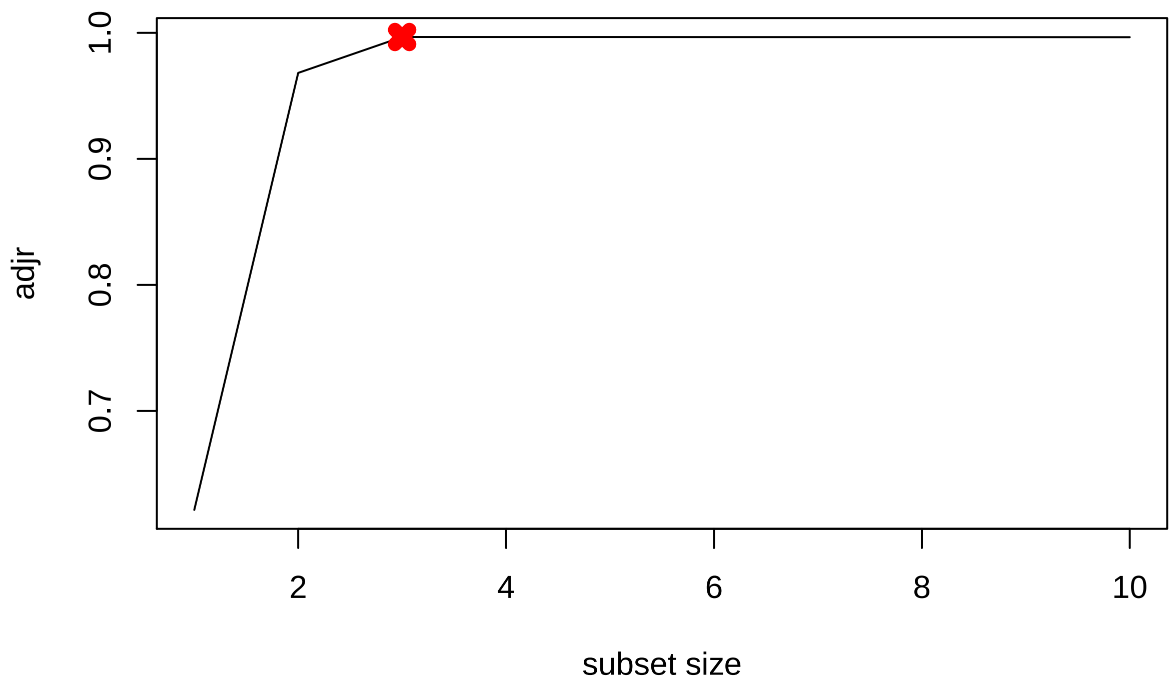
```r
plot(bs.summary$bic, xlab = "subset size", ylab="bic", type = "l")
points(3, bs.summary$bic[3], pch = 4, col = "red", lwd = 7)
```




```r
which.min(bs.summary$adjr2)
```

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

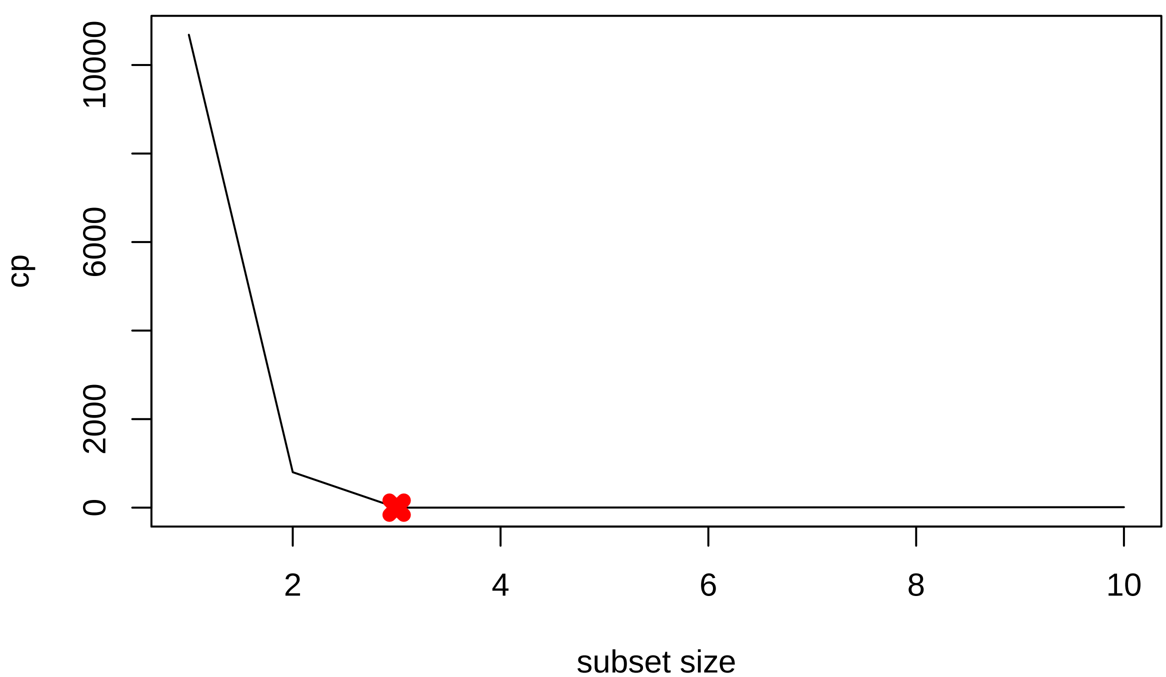
```r
plot(bs.summary$adjr2, xlab = "subset size", ylab="adjr", type = "l")
points(3, bs.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
```




```r
which.min(bs.summary$cp)
```

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

```r
plot(bs.summary$cp, xlab = "subset size", ylab="cp", type = "l")
points(3, bs.summary$cp[3], pch = 4, col = "red", lwd = 7)
```



```r
coefficients(bs1, id=3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             5.1383516             6.9086040            -5.0324623
## poly(x, 10, raw = T)3
##             0.5114084
```

(d)

```r
fs1=regsubsets(y ~ poly(x,10, raw=T), data = data, nvmax = 10, method="forward")
bw1=regsubsets(y ~ poly(x,10,raw=T), data = data, nvmax=10, method = "backward")
fs1.summary=summary(fs1)
bw1.summary=summary(bw1)
par(mfrow = c(3,2))
which.min(fs1.summary$cp)
```

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

```r
plot(fs1.summary$cp, xlab = "subset size", ylab="cp", type = "l")
points(3, fs1.summary$cp[3], pch = 4, col = "red", lwd = 7)
which.min(fs1.summary$bic)
```

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

```r
plot(fs1.summary$bic, xlab = "subset size", ylab="bic", type = "l")
points(3, fs1.summary$bic[3], pch = 4, col = "red", lwd = 7)
which.min(fs1.summary$adjr2)
```

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

```r
plot(fs1.summary$adjr2, xlab = "subset size", ylab="adjr2", type = "l")
points(3, fs1.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
which.min(bw1.summary$cp)
```

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

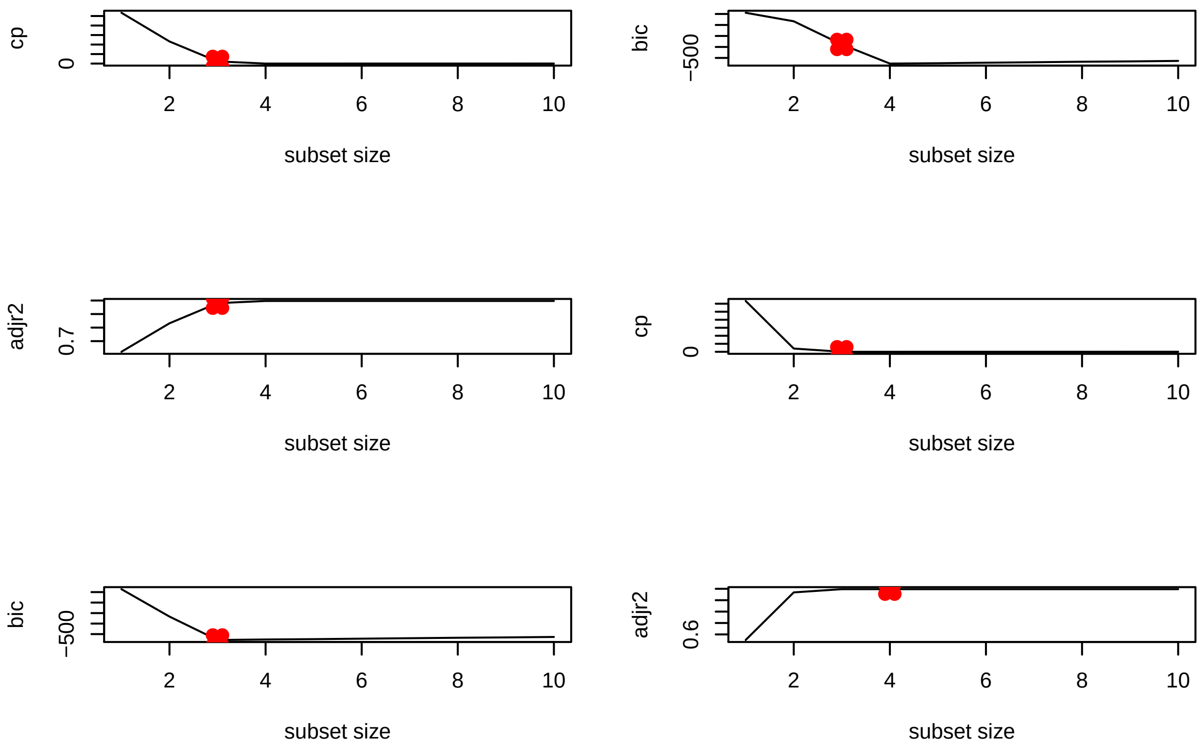
```r
plot(bw1.summary$cp, xlab = "subset size", ylab="cp", type = "l")
points(3, bw1.summary$cp[3], pch = 4, col = "red", lwd = 7)
which.min(bw1.summary$bic)
```

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

```r
plot(bw1.summary$bic, xlab = "subset size", ylab="bic", type = "l")
points(3, bw1.summary$bic[3], pch = 4, col = "red", lwd = 7)
which.min(bw1.summary$adjr2)
```

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

```r
plot(bw1.summary$adjr2, xlab = "subset size", ylab="adjr2", type = "l")
points(4, bw1.summary$adjr2[4], pch = 4, col = "red", lwd = 7)
```




```r
coefficients(fs1, id=3)
```

```
##             (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##            4.387670e+00           9.070917e+00          -4.327996e+00
## poly(x, 10, raw = T)10
##           -3.299527e-05
```

```r
coefficients(bw1, id=3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             5.1383516             6.9086040            -5.0324623
## poly(x, 10, raw = T)3
##             0.5114084
```

Both of these two methods pick 3 variable model except beackward stepwise with adjusted R2 pick 4 variable. As the id=3 predictors show, both of the method pick the correct predictors and the coefficient of each predictor is quiet close to the true relationship.

(e)

```r
library(glmnet)
```

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

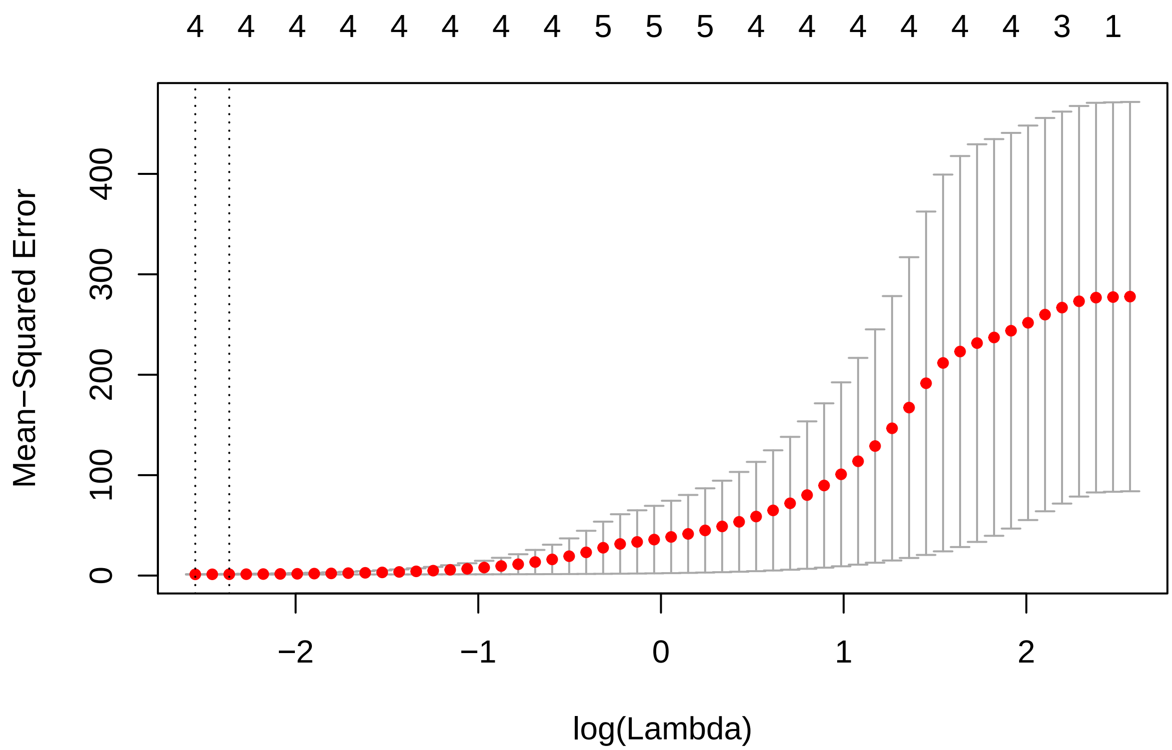
```
## Loading required package: foreach

## Loaded glmnet 2.0-18
```

```r
xmatrix= model.matrix(y ~ poly(x, 10, raw = T), data= data)[, -1]
lasso1.cv=cv.glmnet(xmatrix, y, alpha=1)
best=lasso1.cv$lambda.min
best
```

```
## [1] 0.07814304
```

```r
plot(lasso1.cv)
```




```r
lasso1=glmnet(xmatrix, y, alpha=1)
predict(lasso1, s=best, type="coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)            5.087028064
## poly(x, 10, raw = T)1  6.859914913
## poly(x, 10, raw = T)2 -4.965025286
## poly(x, 10, raw = T)3  0.506051128
## poly(x, 10, raw = T)4 -0.003136705
## poly(x, 10, raw = T)5  .
## 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 .
```

As you can see, the last six estimates have been shinked to 0 (x5~x10), and the estimate of x4 is very close to 0.

(f)

```
beta7 = 8
beta0 = 5
y = beta0 + beta7* x^7 + noice
data=data.frame(y=y, x=x)
best.f=regsubsets(y~ poly(x, 10, raw=T), data = data, nvmax = 10)
bs.f= summary(best.f)
which.min(bs.f$bic)
```

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

```
which.min(bs.f$adjr2)
```

```
## [1] 10
```

```
which.min(bs.f$cp)
```

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

```
coefficients(best.f, id=1)
```

```
##           (Intercept) poly(x, 10, raw = T)7
##              5.087714              8.000027
```

```
coefficients(best.f, id=10)
```

```
##            (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2
##           4.9808755598          -0.4984937461           0.9725848531
##  poly(x, 10, raw = T)3  poly(x, 10, raw = T)4  poly(x, 10, raw = T)5
##           0.6244838643          -1.1034990087          -0.2490019278
##  poly(x, 10, raw = T)6  poly(x, 10, raw = T)7  poly(x, 10, raw = T)8
##           0.4068521244           8.0337153983          -0.0552708992
##  poly(x, 10, raw = T)9 poly(x, 10, raw = T)10
##          -0.0009677929           0.0022428131
```

BIC and CP for best model selection is quiet accurate, select only one variable model and the estimate of it is very close, where as the adjr2 select 10 variable model, but as you can see, the estimate of x7 and intercept is quiet close and the other estimate is close to 0, therefore, for adjr2, it only includes too many variables but the accuracy is still good.

```
xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data)[, -1]
lasso.cvf=cv.glmnet(xmat, y ,alpha=1)
best.lamdaf=lasso.cvf$lambda.min
best.lassof=glmnet(xmat, y , alpha = 1)
predict(best.lassof, s= best.lamdaf, type = "coefficients")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)            -11.871330
## 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    7.766822
## poly(x, 10, raw = T)8    .
## poly(x, 10, raw = T)9    .
## poly(x, 10, raw = T)10   .
```

Lasso shrinks the x1~x6 and x8~x10 to 0, which is quiet accurate, and the estimate of the remaining two variables is close to the true relationship.

Problem 4–Exercise 9

(a)

```r
library(ISLR)
data("College")
train= sample(1:dim(College)[1], dim(College)[1]/2)
test = -train
traindata=College[train, ]
testdata=College[test, ]
```

(b)

```r
lr=lm(Apps~., data= traindata)
lr.pred = predict(lr, testdata)
mean((testdata[,"Apps"]-lr.pred)^2)
```

```
## [1] 1156748
```

test error is 1406014

(c)

```r
train.mat= model.matrix(Apps~., data = traindata)
test.mat=model.matrix(Apps~., data = testdata)
grid = 10 ^ seq(4, -2, length=100)
ridge9 = cv.glmnet(train.mat, traindata[, "Apps"], alpha=0, lambda = grid, thresh=1e-12)
lambda.rigdge=ridge9$lambda.min
lambda.rigdge
```

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

```r
ridge99 = cv.glmnet(train.mat, traindata[, "Apps"], alpha=0, lambda = grid, thresh=1e-12)
ridge.pre= predict(ridge99, newx=test.mat, s=lambda.rigdge)
mean((testdata[, "Apps"]- ridge.pre)^2)
```

```
## [1] 1156727
```

test error is 1469542

(d)

```r
train.mat= model.matrix(Apps~., data = traindata)
test.mat=model.matrix(Apps~., data = testdata)
lasso9= cv.glmnet(train.mat, traindata[, "Apps"], alpha = 1, lambda = grid, thresh=1e-12)
lasso99= glmnet(train.mat, traindata[, "Apps"], alpha = 1, lambda = grid, thresh=1e-12)
lasso.pred=predict(lasso99, newx = test.mat, s = lasso9$lambda.min)
mean((testdata[,"Apps"]-lasso.pred)^2)
```

```
## [1] 1152585
```

```r
predict(lasso99, s=lasso9$lambda.min, type="coefficients")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -587.82052640
## (Intercept)    .
## PrivateYes  -427.08822327
## Accept         1.74450552
## Enroll        -1.03762795
## Top10perc     55.07864930
## Top25perc    -19.72860625
## F.Undergrad   -0.02502991
## P.Undergrad    0.11154659
## Outstate      -0.12460861
## Room.Board     0.13547393
## Books          0.15251113
## Personal       0.12469880
## PhD           -9.48443251
## Terminal       2.33418045
## S.F.Ratio     15.23810099
## perc.alumni    6.85790295
## Expend         0.08758917
## Grad.Rate      7.88437846
```

test error is 1405995

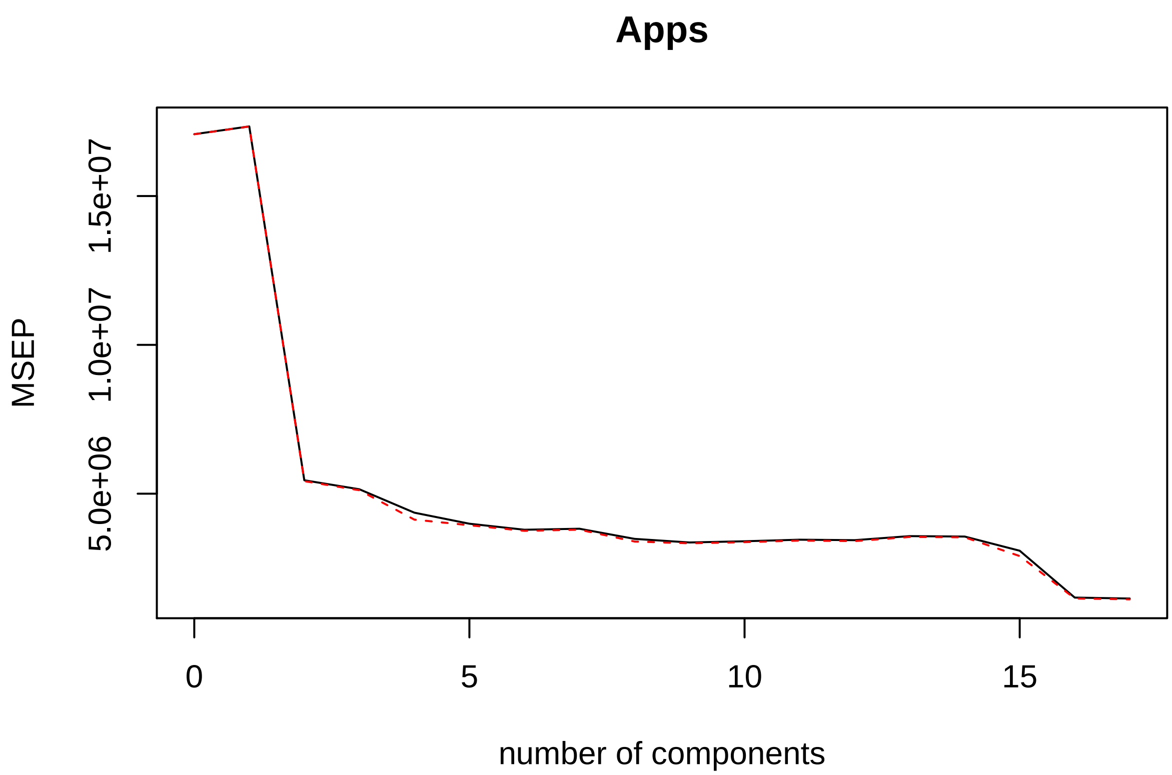
(e)

```r
library(pls)
```

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

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

```
pcr.fit=pcr(Apps~., data=traindata, 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            4132     4164     2335     2269     2088     1998     1946
## adjCV         4132     4164     2329     2262     2032     1985     1937
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1955     1866     1833      1844      1859      1854      1891
## adjCV     1947     1842     1826      1836      1851      1846      1883
##        14 comps  15 comps  16 comps  17 comps
## CV         1887      1756      1228      1216
## adjCV      1880      1704      1217      1205
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
```

```
## X      32.08950    57.57    64.79    70.43    75.77    80.66    84.44
## Apps    0.06418    70.04    71.75    78.96    79.19    80.50    80.50
##        8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X        87.72    90.80     92.99     95.07     96.80     97.87     98.80
## Apps     82.77    82.79     82.81     82.81     82.97     83.00     83.17
##        15 comps  16 comps  17 comps
## X         99.45     99.89    100.00
## Apps      92.08     93.52     93.69
```

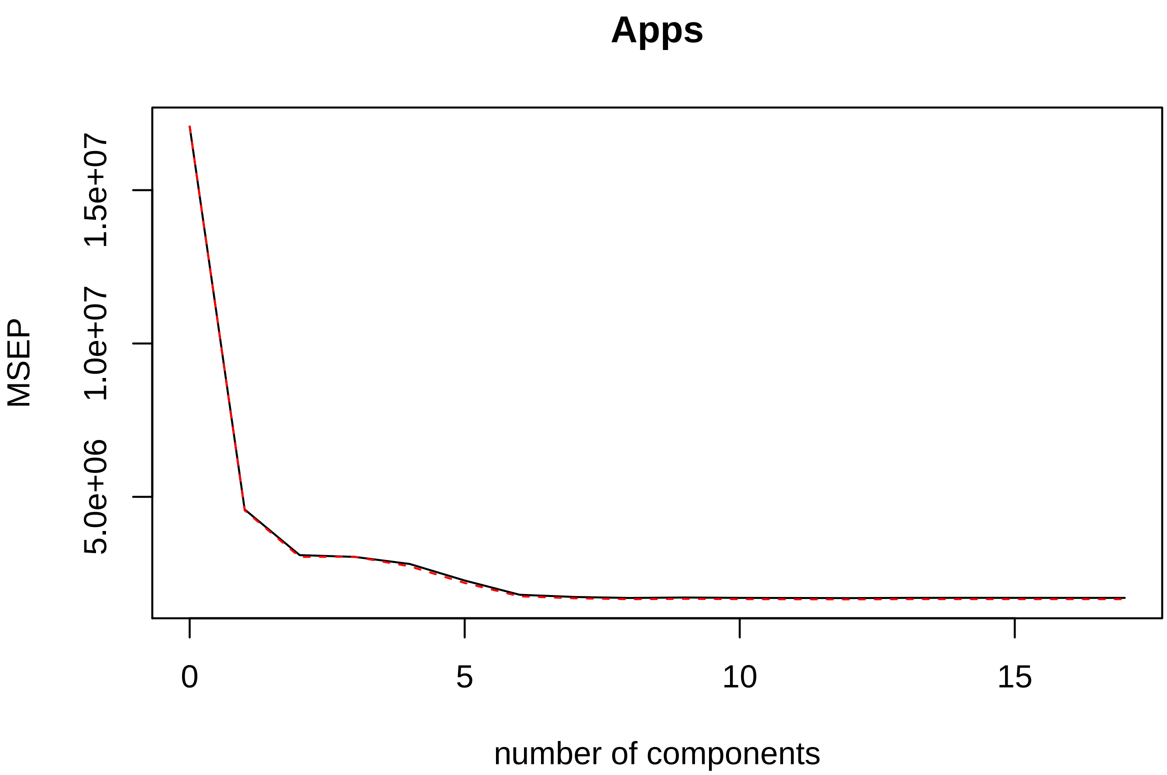
```
pcr.pred= predict(pcr.fit, testdata, ncomp=10)
mean((testdata[,"Apps"]- pcr.pred)^2)
```

```
## [1] 1322150
```

n=10, test error is 2797760

(f)

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




```
summary(plsr.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            4132     2142     1761     1744     1677     1507     1344
## adjCV         4132     2135     1744     1745     1653     1481     1325
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1317     1305     1310      1306      1305      1303      1306
## adjCV     1301     1289     1294      1290      1289      1288      1290
##        14 comps  15 comps  16 comps  17 comps
## CV         1307      1306      1306      1306
## adjCV      1291      1290      1290      1290
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       25.32    31.59    60.33    66.69    70.72    74.35    78.23
## Apps    75.41    85.19    86.16    89.99    92.68    93.47    93.54
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       80.54    81.91     86.37     88.16     90.20     92.27     94.10
## Apps    93.59    93.64     93.65     93.67     93.68     93.69     93.69
##       15 comps  16 comps  17 comps
## X        96.85     97.94    100.00
## Apps     93.69     93.69     93.69
```

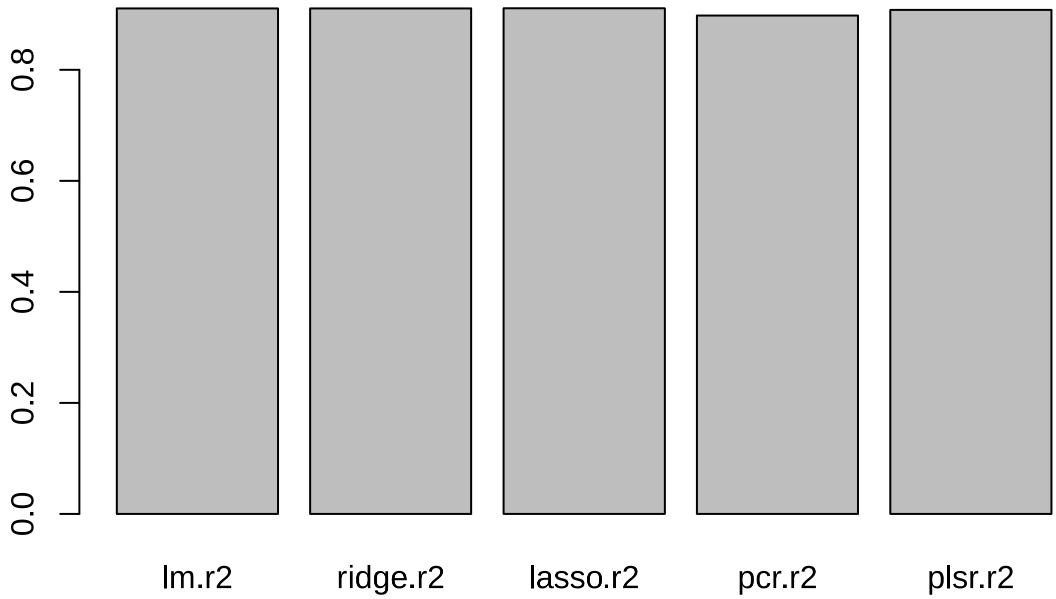
```r
plsr.pred=predict(plsr.fit, testdata, ncomp=10)
mean((testdata[, "Apps"]- plsr.pred)^2)
```

```
## [1] 1190800
```

n=10, test error is 1345153

(g)

```r
testavg=mean(testdata[, "Apps"])
lm.r2= 1-mean((testdata[, "Apps"]-lr.pred)^2)/mean((testdata[, "Apps"]-testavg)^2)
ridge.r2= 1-mean((testdata[, "Apps"]-ridge.pre)^2)/mean((testdata[, "Apps"]-testavg)^2)
lasso.r2= 1-mean((testdata[, "Apps"]-lasso.pred)^2)/mean((testdata[, "Apps"]-testavg)^2)
pcr.r2= 1-mean((testdata[, "Apps"]-pcr.pred)^2)/mean((testdata[, "Apps"]-testavg)^2)
plsr.r2= 1-mean((testdata[, "Apps"]-plsr.pred)^2)/mean((testdata[, "Apps"]-testavg)^2)
barplot(c(lm.r2,ridge.r2,lasso.r2,pcr.r2,plsr.r2),names.arg=c("lm.r2","ridge.r2","lasso.r2","pcr.r2","pl
```



Use R squared to account for the results of the models, all model except pcr got a 90% R squared, which means thats all the model except can get a 90% accuracy (variance explained), and pcr model, otherwise, predicts poorly, which only has around 70% variance explained.

Problem 5–Exercise 11

(a)

```
library(glmnet)
library(MASS)
data(Boston)
lasso.x=model.matrix(crim~. -1, data = Boston)
lasso.y=Boston$crim
cv.lasso=cv.glmnet(lasso.x,lasso.y)
coef(cv.lasso)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept) 1.7799525
## zn          .
## indus       .
## chas        .
## nox         .
## rm          .
## age         .
## dis         .
## rad         0.1920089
## tax         .
## ptratio     .
## black       .
## lstat       .
## medv        .
```

```r
sqrt(cv.lasso$cvm[cv.lasso$lambda == cv.lasso$lambda.min])
```

```
## [1] 6.572957
```

```r
cv.ridge=cv.glmnet(lasso.x,lasso.y,alpha=0)
coef(cv.ridge)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  2.190805375
## zn          -0.002514410
## indus        0.021067968
## chas        -0.102372822
## nox          1.323474252
## rm          -0.106109537
## age          0.004452629
## dis         -0.066139399
## rad          0.029778399
## tax          0.001383946
## ptratio      0.049720162
## black       -0.001713827
## lstat        0.024367321
## medv        -0.016059212
```

```r
sqrt(cv.ridge$cvm[cv.lasso$lambda == cv.lasso$lambda.min])
```

```
## [1] 6.982779
```

```r
pcr.fit = pcr(crim~. , data=Boston, scale= T, validation="CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 506 13
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8.61    7.172    7.168    6.763    6.751    6.755    6.764
## adjCV         8.61    7.171    7.167    6.758    6.744    6.751    6.759
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.756    6.652    6.669     6.660     6.660     6.603     6.539
## adjCV    6.750    6.646    6.662     6.652     6.651     6.593     6.529
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.45    95.40     97.04     98.46     99.52     100.0
## crim    42.47    42.55     42.78     43.04     44.13      45.4
```

n=13, the model has the lowest cv and adjcv.

(b) I would choose lasso model as my chosen model, since from MSR, it has the second best performance. And it is much simpler than the other two models

(c) no, only contain one feature and one intercept. Since it is lasso, the other parameters have been shinked to 0.