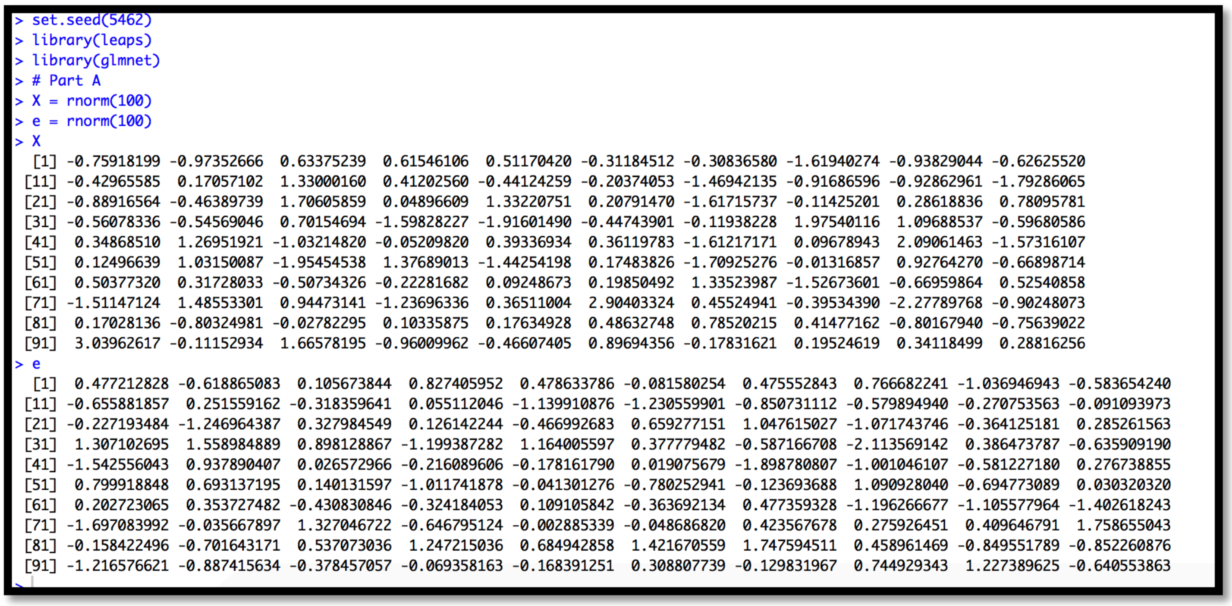
COMP3354 - Statistical Learning
Assignment 2

Name: Pranav Talwar
UID: 3035435462

Chapter 6, Question 8

a) Rcommand:
```
> set.seed(5462)
> library(leaps)
> library(glmnet)
> # Part A
> X = rnorm(100)
> e = rnorm(100)
> X
> e
```

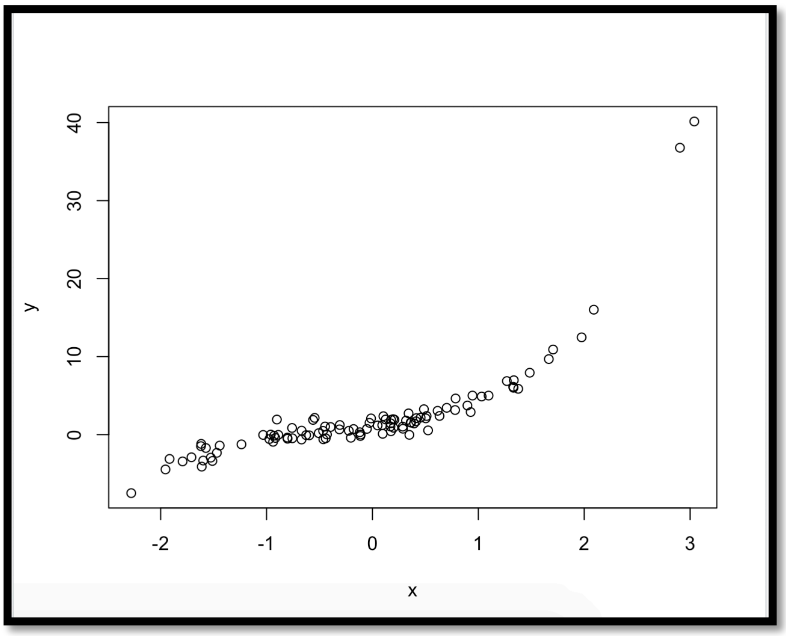
Output:



```
> set.seed(5462)
> library(leaps)
> library(glmnet)
> # Part A
> X = rnorm(100)
> e = rnorm(100)
> X
  [1] -0.75918199 -0.97352666  0.63375239  0.61546106  0.51170420 -0.31184512 -0.30836580 -1.61940274 -0.93829044 -0.62625520
 [11] -0.42965585  0.17057102  1.33000160  0.41202560 -0.44124259 -0.20374053 -1.46942135 -0.91686596 -0.92862961 -1.79286065
 [21] -0.88916564 -0.46389739  1.70605859  0.04896609  1.33220751  0.20791470 -1.61715737 -0.11425201  0.28618836  0.78095781
 [31] -0.56078336 -0.54569046  0.70154694 -1.59828227 -1.91601490 -0.44743901 -0.11938228  1.97540116  1.09688537 -0.59680586
 [41]  0.34868510  1.26951921 -1.03214820 -0.05209820  0.39336934  0.36119783 -1.61217171  0.09678943  2.09061463 -1.57316107
 [51]  0.12496639  1.03150087 -1.95454538  1.37689013 -1.44254198  0.17483826 -1.70925276 -0.01316857  0.92764270 -0.66898714
 [61]  0.50377320  0.31728033 -0.50734326 -0.22281682  0.09248673  0.19850492  1.33523987 -1.52673601 -0.66959864  0.52540858
 [71] -1.51147124  1.48553301  0.94473141 -1.23696336  0.36511004  2.90403324  0.45524941 -0.39534390 -2.27789768 -0.90248073
 [81]  0.17028136 -0.80324981 -0.02782295  0.10335875  0.17634928  0.48632748  0.78520215  0.41477162 -0.80167940 -0.75639022
 [91]  3.03962617 -0.11152934  1.66578195 -0.96009962 -0.46607405  0.89694356 -0.17831621  0.19524619  0.34118499  0.28816256
> e
  [1]  0.477212828 -0.618865083  0.105673844  0.827405952  0.478633786 -0.081580254  0.475552843  0.766682241 -1.036946943 -0.583654240
 [11] -0.655881857  0.251559162 -0.318359641  0.055112046 -1.139910876 -1.230559901 -0.850731112 -0.579894940 -0.270753563 -0.091093973
 [21] -0.227193484 -1.246964387  0.327984549  0.126142244 -0.466992683  0.659277151  1.047615027 -1.071743746 -0.364125181  0.285261563
 [31]  1.307102695  1.558984889  0.898128867 -1.199387282  1.164005597  0.377779482 -0.587166708 -2.113569142  0.386473787 -0.635909190
 [41] -1.542556043  0.937890407  0.026572966 -0.216089606 -0.178161790  0.019075679 -1.898780807 -1.001046107 -0.581227180  0.276738855
 [51]  0.799918848  0.693137195  0.140131597 -1.011741878 -0.041301276 -0.780252941 -0.123693688  1.090928040 -0.694773089  0.030320320
 [61]  0.202723065  0.353727482 -0.430830846 -0.324184053  0.109105842 -0.363692134  0.477359328 -1.196266677 -1.105577964 -1.402618243
 [71] -1.697083992 -0.035667897  1.327046722 -0.646795124 -0.002885339 -0.048686820  0.423567678  0.275926451  0.409646791  1.758655043
 [81] -0.158422496 -0.701643171  0.537073036  1.247215036  0.684942858  1.421670559  1.747594511  0.458961469 -0.849551789 -0.852260876
 [91] -1.216576621 -0.887415634 -0.378457057 -0.069358163 -0.168391251  0.308807739 -0.129831967  0.744929343  1.227389625 -0.640553863
>
```

b) Rcommand:
```
> Y = 1 + X + X*X + X*X*X + e
> data.new = data.frame(y = Y, x = X)
> plot(data.new)
```

Output:

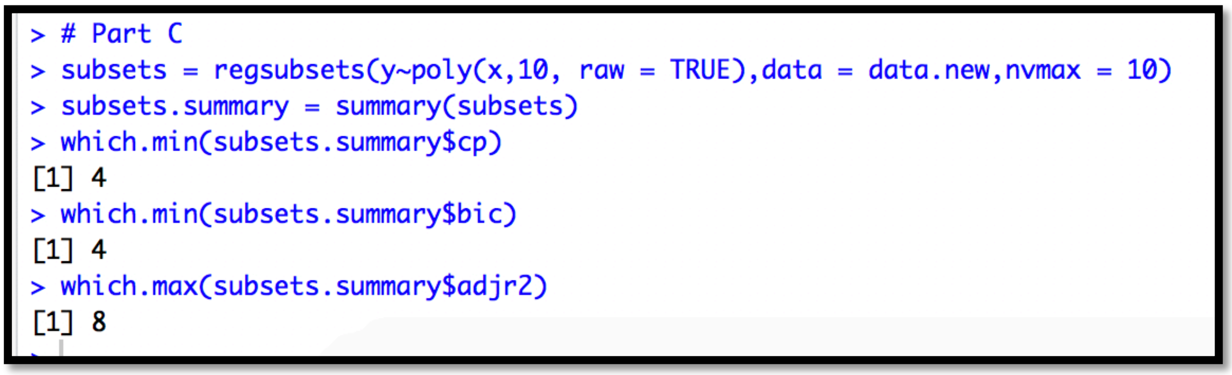


In the following model Y = β0 +β1*X +β2*X2 +β3*X3 +e, the values chosen are 1 for all the betas.

c) Rcommand: > subsets = regsubsets(y~poly(x,10, raw = TRUE),data = data.new,nvmax = 10)

```
> subsets.summary = summary(subsets)
> which.min(subsets.summary$cp)
> which.min(subsets.summary$bic)
> which.max(subsets.summary$adjr2)
```

Output :



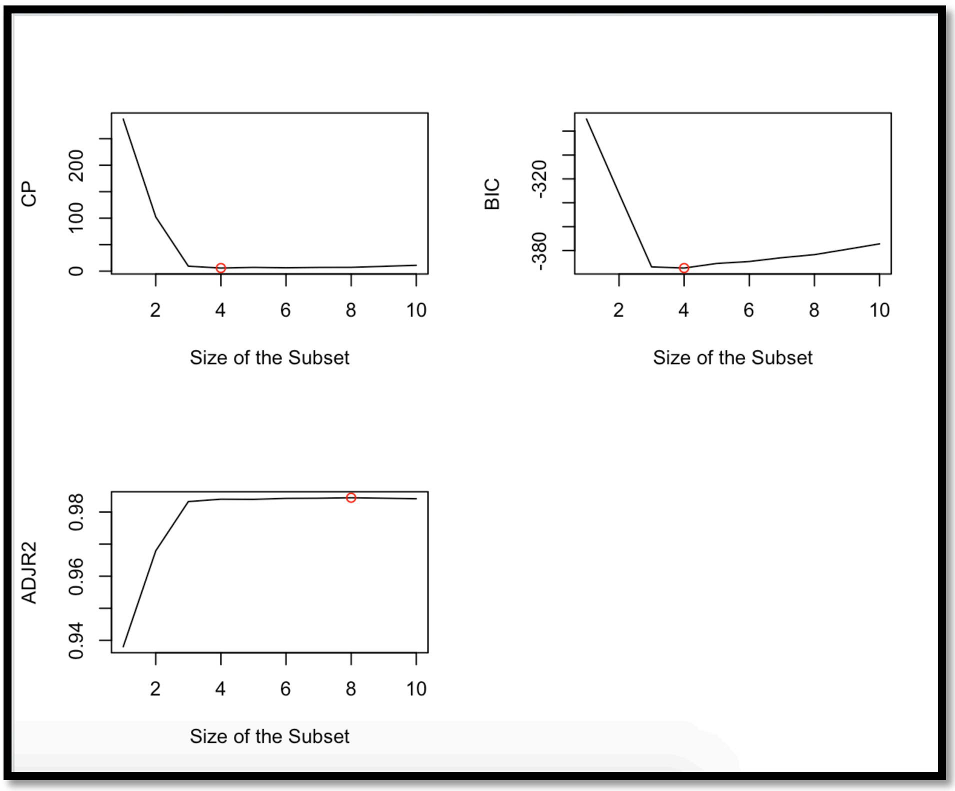
```
> # Part C
> subsets = regsubsets(y~poly(x,10, raw = TRUE),data = data.new,nvmax = 10)
> subsets.summary = summary(subsets)
> which.min(subsets.summary$cp)
[1] 4
> which.min(subsets.summary$bic)
[1] 4
> which.max(subsets.summary$adjr2)
[1] 8
```

According to the best subset selection method, the best models are;

1) CP: Model 4
2) BIC: Model 4
3) ADJR2: Model 8

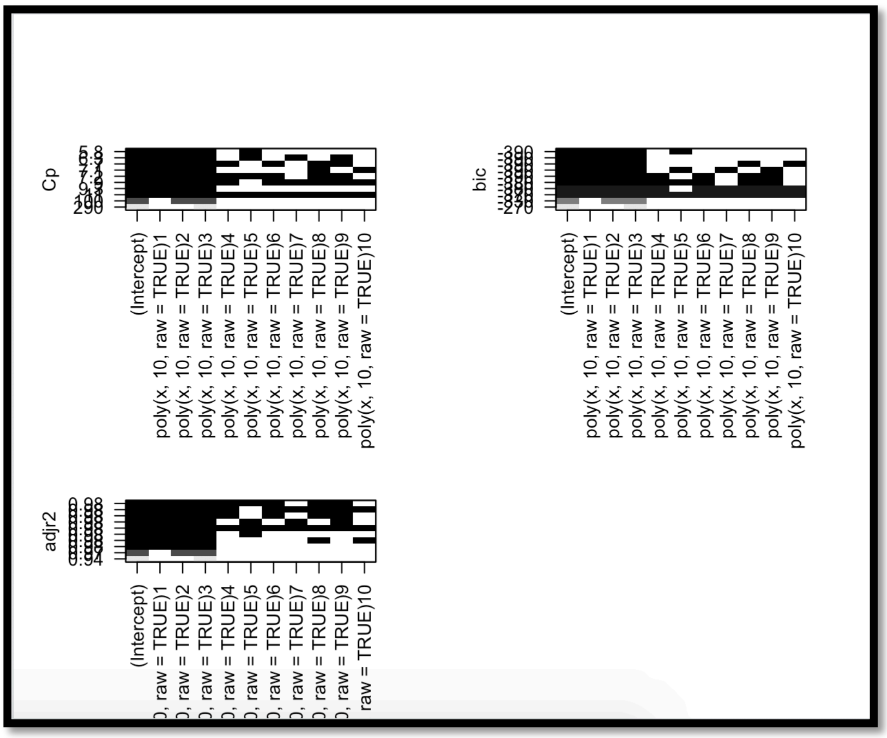
Rcommand:
```
> par(mfrow=c(2,2))
> plot(subsets.summary$cp, xlab = "Size of the Subset", ylab ="CP", type = "l")
>  points(4, subsets.summary$cp[4],col = "red")
> plot(subsets.summary$bic, xlab = "Size of the Subset", ylab = "BIC", type
="l")
> points(4, subsets.summary$bic[4],col = "red")
> plot(subsets.summary$adjr2, xlab = "Size of the Subset", ylab ="ADJR2", type
= "l")
> points(8, subsets.summary$adjr2[8],col = "red")
```

Output:



Rcommand:
```
> par(mfrow=c(2,2))
> plot(subsets, scale="Cp")
> plot(subsets, scale="bic")
> plot(subsets, scale="adjr2")
```

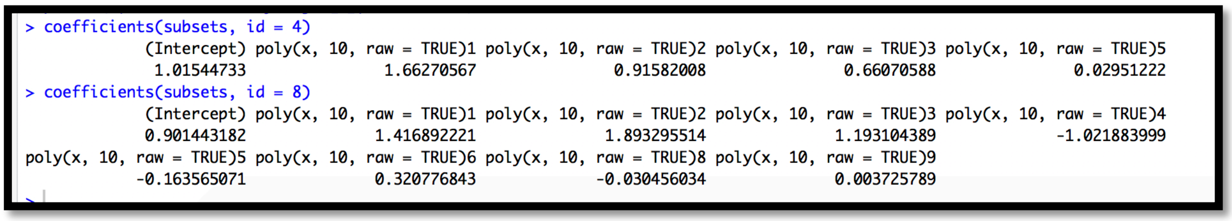
Output:



From these plots we can see that the model 4 is the best when CP and BIC are used in best subset selection. Model 4 has the lowest CP and BIC among all the models. Whereas, Model 8 is the best when ADJR2 is used in the best subset selection method since it has the highest ADJR2 among all the other models.

Rcommand: > coefficients(subsets, id = 4)
> coefficients(subsets, id = 8)

Output:



```
> coefficients(subsets, id = 4)
            (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2 poly(x, 10, raw = TRUE)3 poly(x, 10, raw = TRUE)5
             1.01544733               1.66270567               0.91582008               0.66070588               0.02951222
> coefficients(subsets, id = 8)
            (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2 poly(x, 10, raw = TRUE)3 poly(x, 10, raw = TRUE)4
            0.901443182              1.416892221              1.893295514              1.193104389             -1.021883999
poly(x, 10, raw = TRUE)5 poly(x, 10, raw = TRUE)6 poly(x, 10, raw = TRUE)8 poly(x, 10, raw = TRUE)9
           -0.163565071              0.320776843             -0.030456034              0.003725789
```

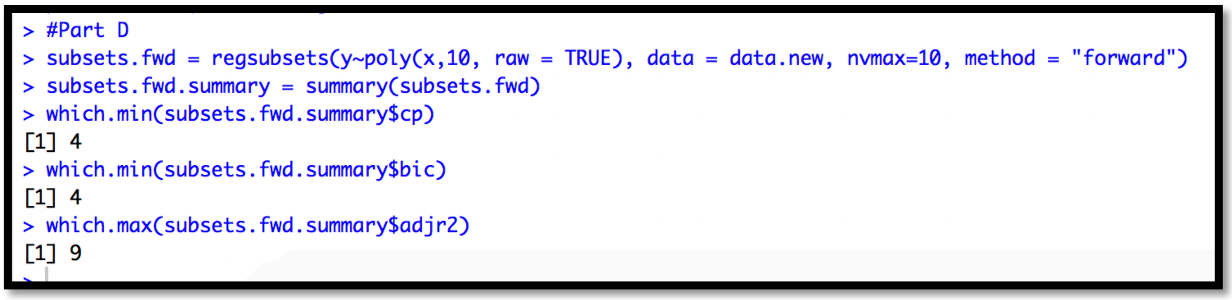
The coefficients of model 4 and model 8 are given above. We can see that model 3 includes the coefficients till X^3 as well as X^5 (very small) whereas model 8 also includes variables X^4, X^6, X^8 and X^10. The coefficients of model 3 are more or less accurate as compared to the real model, whereas the coefficient of X^5 is negligible.

d) **Using Forward Selection**

Rcommand: > subsets.fwd = regsubsets(y~poly(x,10, raw = TRUE), data = data.new, nvmax=10, method = "forward")

> subsets.fwd.summary = summary(subsets.fwd)
> which.min(subsets.fwd.summary$cp)
> which.min(subsets.fwd.summary$bic)
> which.max(subsets.fwd.summary$adjr2)

Output:

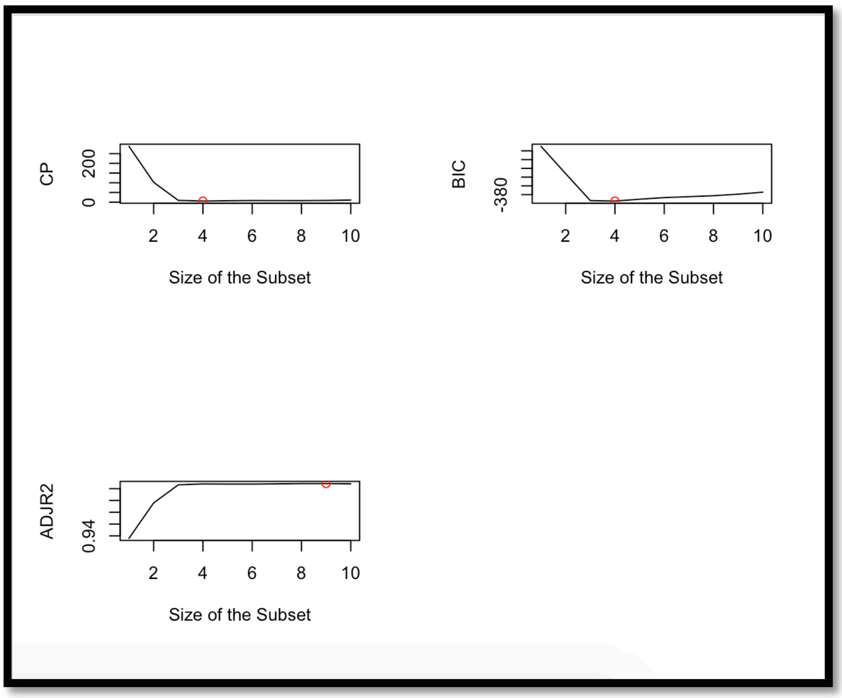


```
> #Part D
> subsets.fwd = regsubsets(y~poly(x,10, raw = TRUE), data = data.new, nvmax=10, method = "forward")
> subsets.fwd.summary = summary(subsets.fwd)
> which.min(subsets.fwd.summary$cp)
[1] 4
> which.min(subsets.fwd.summary$bic)
[1] 4
> which.max(subsets.fwd.summary$adjr2)
[1] 9
```

Rcommand: > par(mfrow=c(2,2))

> plot(subsets.fwd.summary$cp, xlab = "Size of the Subset", ylab ="CP", type = "l")
> points(4, subsets.fwd.summary$cp[4],col = "red")
> plot(subsets.fwd.summary$bic, xlab = "Size of the Subset", ylab = "BIC", type ="l")
> points(4, subsets.fwd.summary$bic[4],col = "red")
> plot(subsets.fwd.summary$adjr2, xlab = "Size of the Subset", ylab ="ADJR2", type = "l")
> points(9, subsets.fwd.summary$adjr2[9],col = "red")

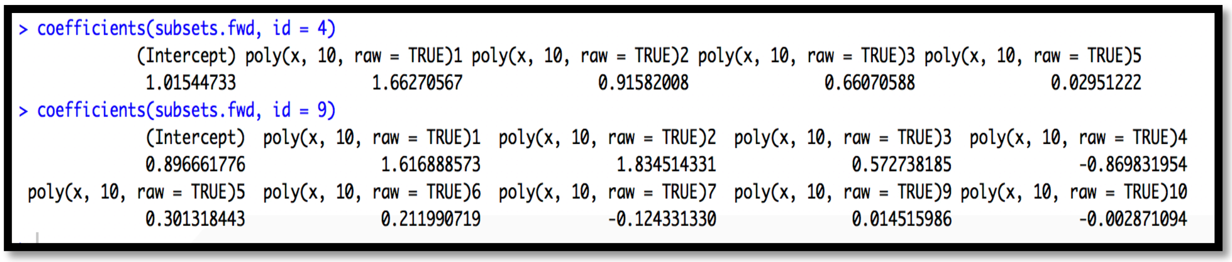
Output:



The best model according to forward selection approach when CP, BIC and ACJR2 are used is model 4 (CP, BIC)and model 9 (ADJR2) as can be seen in the plots given above.

Rcommand: > coefficients(subsets.fwd, id = 4)
> coefficients(subsets.fwd, id = 9)

Output:



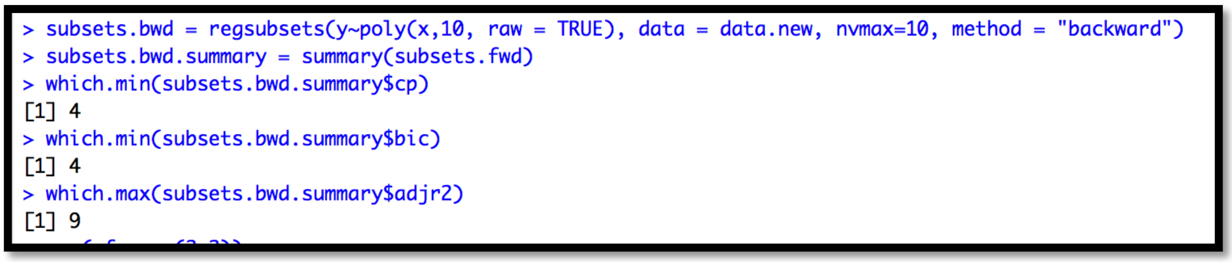
```
> coefficients(subsets.fwd, id = 4)
            (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2 poly(x, 10, raw = TRUE)3 poly(x, 10, raw = TRUE)5
             1.01544733               1.66270567               0.91582008               0.66070588               0.02951222
> coefficients(subsets.fwd, id = 9)
             (Intercept)  poly(x, 10, raw = TRUE)1  poly(x, 10, raw = TRUE)2  poly(x, 10, raw = TRUE)3  poly(x, 10, raw = TRUE)4
             0.896661776               1.616888573               1.834514331               0.572738185              -0.869831954
 poly(x, 10, raw = TRUE)5  poly(x, 10, raw = TRUE)6  poly(x, 10, raw = TRUE)7  poly(x, 10, raw = TRUE)9 poly(x, 10, raw = TRUE)10
             0.301318443               0.211990719              -0.124331330               0.014515986              -0.002871094
```

The coefficients of model 4 and model 9 are given above. The coefficients of model 4 are the same as given by best subset selection method, whereas the coefficients for model 9 contain all the variables up till X^10 with the exception of X^8.

<u>Using Backward selection</u>

Rcommand: > subsets.bwd = regsubsets(y~poly(x,10, raw = TRUE), data = data.new, nvmax=10, method = "backward")
> subsets.bwd.summary = summary(subsets.fwd)
> which.min(subsets.bwd.summary$cp)
> which.min(subsets.bwd.summary$bic)
> which.max(subsets.bwd.summary$adjr2)

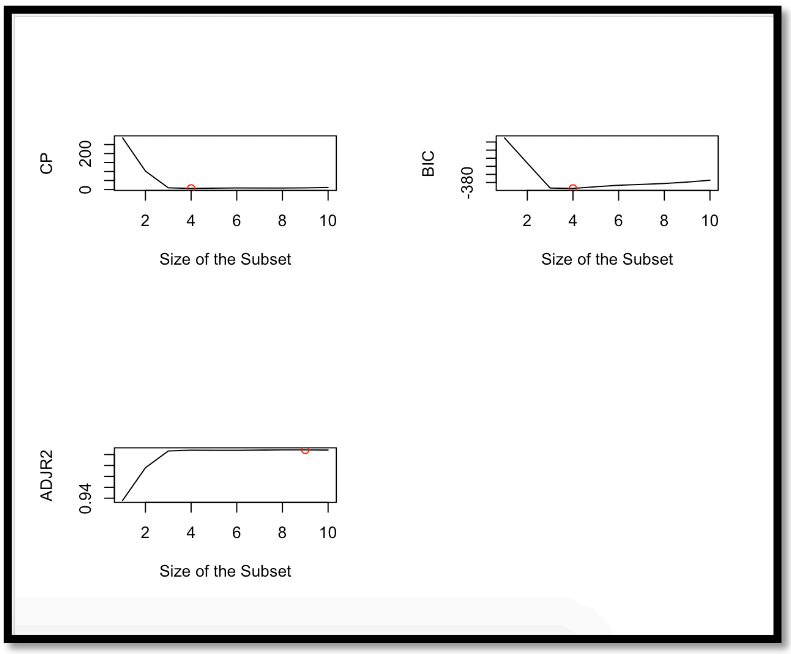
Output:



```
> subsets.bwd = regsubsets(y~poly(x,10, raw = TRUE), data = data.new, nvmax=10, method = "backward")
> subsets.bwd.summary = summary(subsets.fwd)
> which.min(subsets.bwd.summary$cp)
[1] 4
> which.min(subsets.bwd.summary$bic)
[1] 4
> which.max(subsets.bwd.summary$adjr2)
[1] 9
```

```
Rcommand: > par(mfrow=c(2,2))
                  > plot(subsets.bwd.summary$cp, xlab = "Size of the Subset", ylab ="CP", type
          = "l")
                  > points(4, subsets.bwd.summary$cp[4],col = "red")
                  > plot(subsets.bwd.summary$bic, xlab = "Size of the Subset", ylab = "BIC",
type ="l")
                  > points(4, subsets.bwd.summary$bic[4],col = "red")
                  > plot(subsets.bwd.summary$adjr2, xlab = "Size of the Subset", ylab
          ="ADJR2", type = "l")
                 > points(9, subsets.bwd.summary$adjr2[9],col = "red")
```

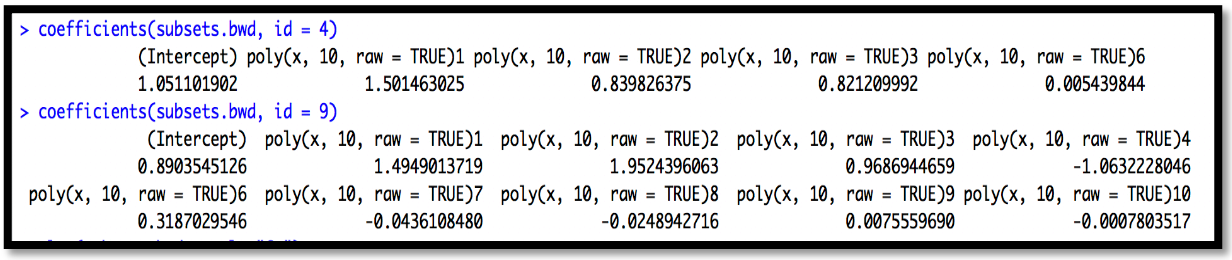
Output:



The best model according to backward selection when CP, BIC and ADJR2 are used is model 4 (CP, BIC) and model 9 (ADJR2).

Rcommand: > coefficients(subsets.bwd, id = 4)
> coefficients(subsets.bwd, id = 9)

Output:

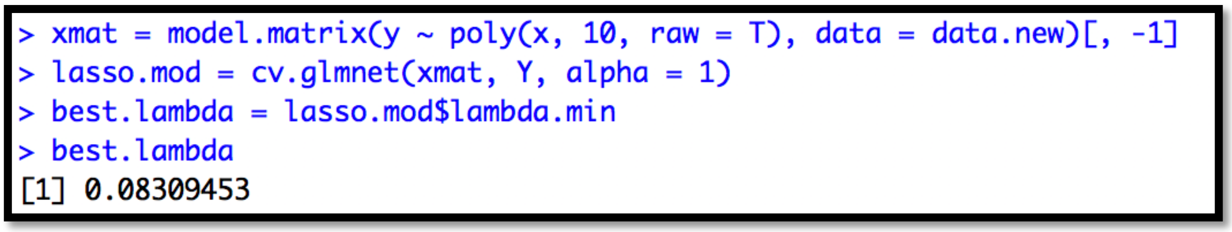


```
> coefficients(subsets.bwd, id = 4)
          (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2 poly(x, 10, raw = TRUE)3 poly(x, 10, raw = TRUE)6
          1.051101902              1.501463025              0.839826375              0.821209992              0.005439844
> coefficients(subsets.bwd, id = 9)
           (Intercept)  poly(x, 10, raw = TRUE)1  poly(x, 10, raw = TRUE)2  poly(x, 10, raw = TRUE)3  poly(x, 10, raw = TRUE)4
          0.8903545126              1.4949013719              1.9524396063              0.9686944659             -1.0632228046
 poly(x, 10, raw = TRUE)6  poly(x, 10, raw = TRUE)7  poly(x, 10, raw = TRUE)8  poly(x, 10, raw = TRUE)9 poly(x, 10, raw = TRUE)10
          0.3187029546             -0.0436108480             -0.0248942716              0.0075559690             -0.0007803517
```

The coefficients of model 4 and model 9 are given above. The coefficients of model 4 are close to what was given by best subset selection method and forward selection method except it has a X^6 term in the model selected by it, whereas the coefficients for model 9 contain all the variables uptill X^10 with the exception of X^5.

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

Output:

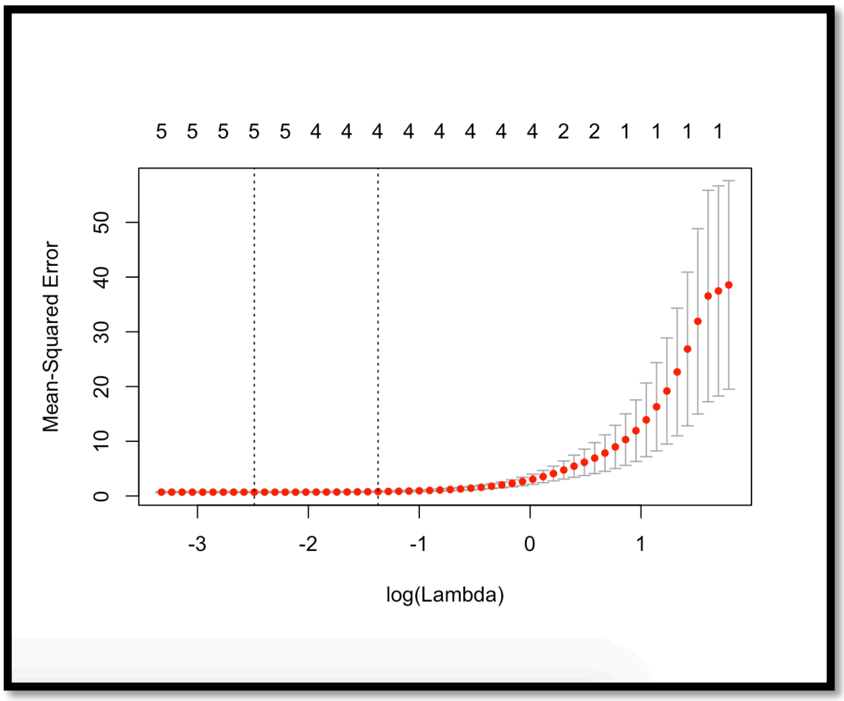


```
> xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.new)[, -1]
> lasso.mod = cv.glmnet(xmat, Y, alpha = 1)
> best.lambda = lasso.mod$lambda.min
> best.lambda
[1] 0.08309453
```

The best value of lambda as given by the cross-validation method is 0.08309453.

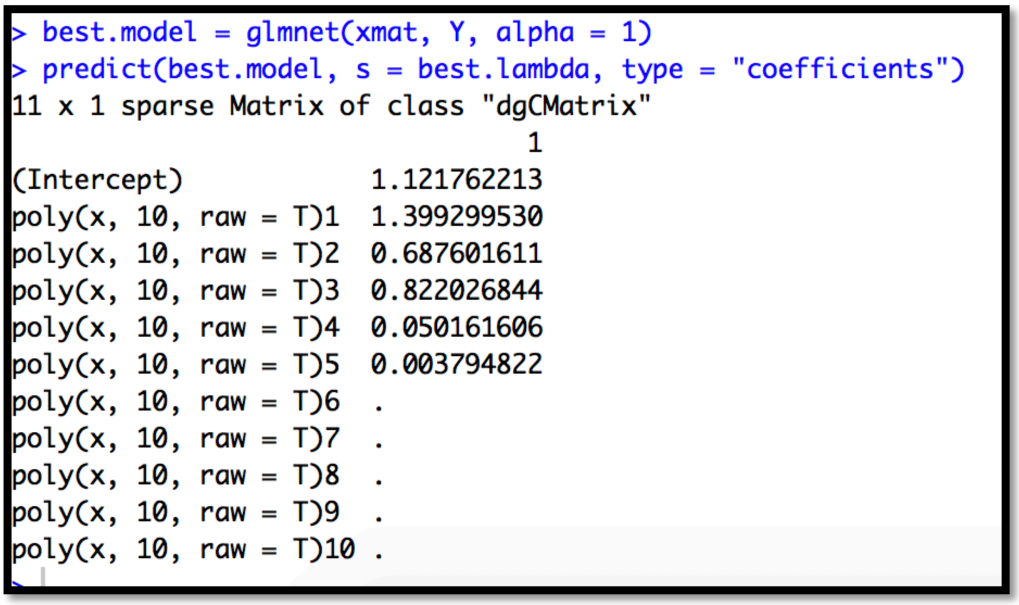
Rcommand: > plot(lasso.mod)

Output:



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

Output:

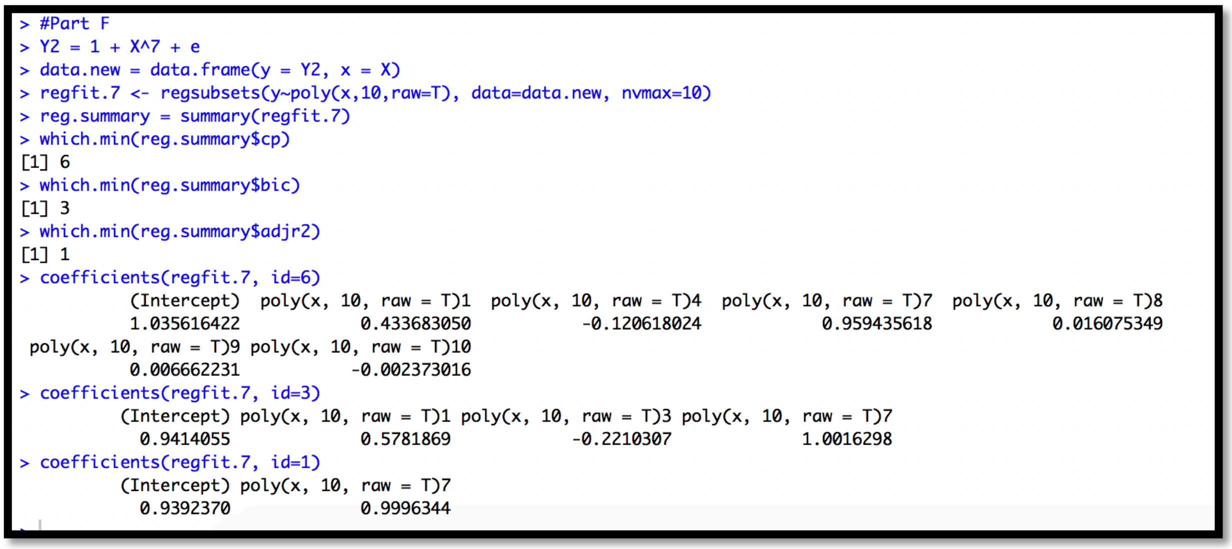


```
> best.model = glmnet(xmat, Y, alpha = 1)
> predict(best.model, s = best.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                                1
(Intercept)           1.121762213
poly(x, 10, raw = T)1  1.399299530
poly(x, 10, raw = T)2  0.687601611
poly(x, 10, raw = T)3  0.822026844
poly(x, 10, raw = T)4  0.050161606
poly(x, 10, raw = T)5  0.003794822
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 also picks X^5 and X^4 (insignificant coefficient). The values of the other coefficients are close to the real values.

f) Rcommand: > Y2 = 1 + X^7 + e
> data.new = data.frame(y = Y2, x = X)
> regfit.7 <- regsubsets(y~poly(x,10,raw=T), data=data.new, nvmax=10)
> reg.summary = summary(regfit.7)
> which.min(reg.summary$cp)
> which.min(reg.summary$bic)
> which.min(reg.summary$adjr2)
> coefficients(regfit.7, id=6)
> coefficients(regfit.7, id=3)
> coefficients(regfit.7, id=1)

Output:

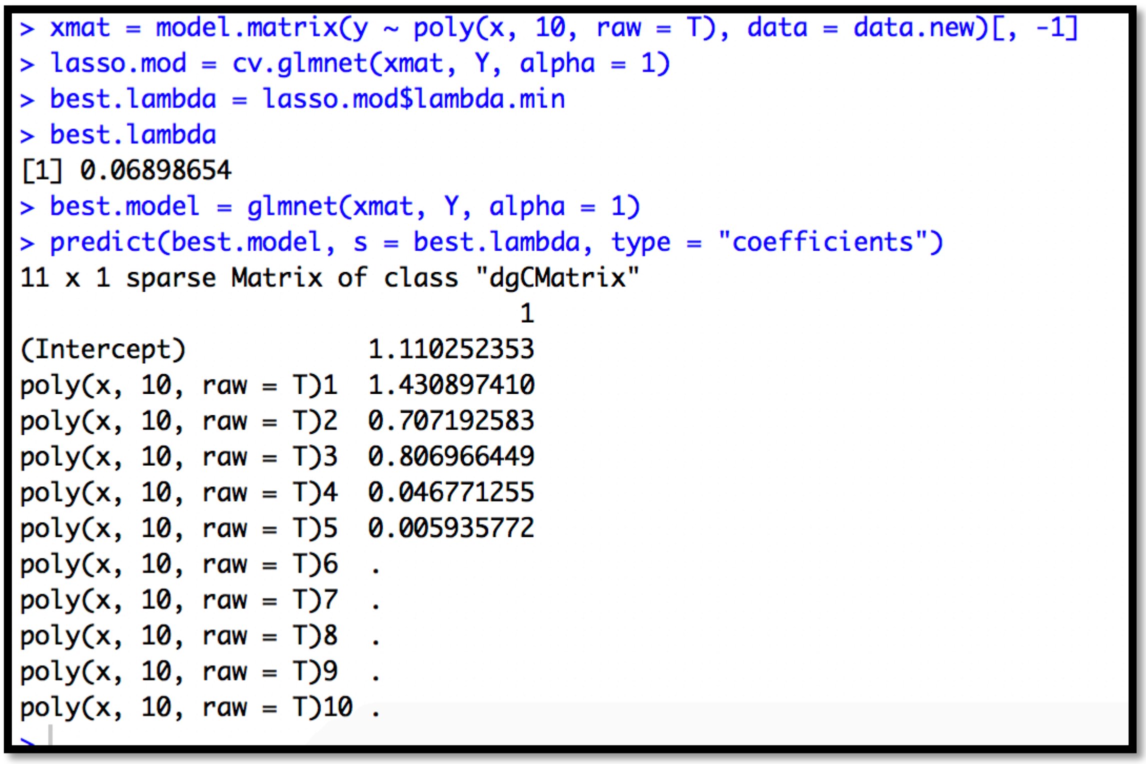


```
> #Part F
> Y2 = 1 + X^7 + e
> data.new = data.frame(y = Y2, x = X)
> regfit.7 <- regsubsets(y~poly(x,10,raw=T), data=data.new, nvmax=10)
> reg.summary = summary(regfit.7)
> which.min(reg.summary$cp)
[1] 6
> which.min(reg.summary$bic)
[1] 3
> which.min(reg.summary$adjr2)
[1] 1
> coefficients(regfit.7, id=6)
           (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)4  poly(x, 10, raw = T)7  poly(x, 10, raw = T)8
           1.035616422            0.433683050           -0.120618024            0.959435618            0.016075349
 poly(x, 10, raw = T)9 poly(x, 10, raw = T)10
           0.006662231           -0.002373016
> coefficients(regfit.7, id=3)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
            0.9414055             0.5781869            -0.2210307             1.0016298
> coefficients(regfit.7, id=1)
          (Intercept) poly(x, 10, raw = T)7
            0.9392370             0.9996344
```

Here we can see that the model selected by the best subset selection method is model 6 (CP), model 3 (BIC) and model 1 (ADJR2). As we can see from the coefficients from the various models generated, model 1 is the best since its selects accurate coefficients and imitates the original model. Whereas the model 3, chooses X^3 coefficient as well, whereas model 6 is completely inaccurate and chooses a large number of extra variables.

Rcommand: > xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.new)[, -1]
> lasso.mod = cv.glmnet(xmat, Y, alpha = 1)
> best.lambda = lasso.mod$lambda.min
> best.lambda
> best.model = glmnet(xmat, Y, alpha = 1)
> predict(best.model, s = best.lambda, type = "coefficients")

Output:



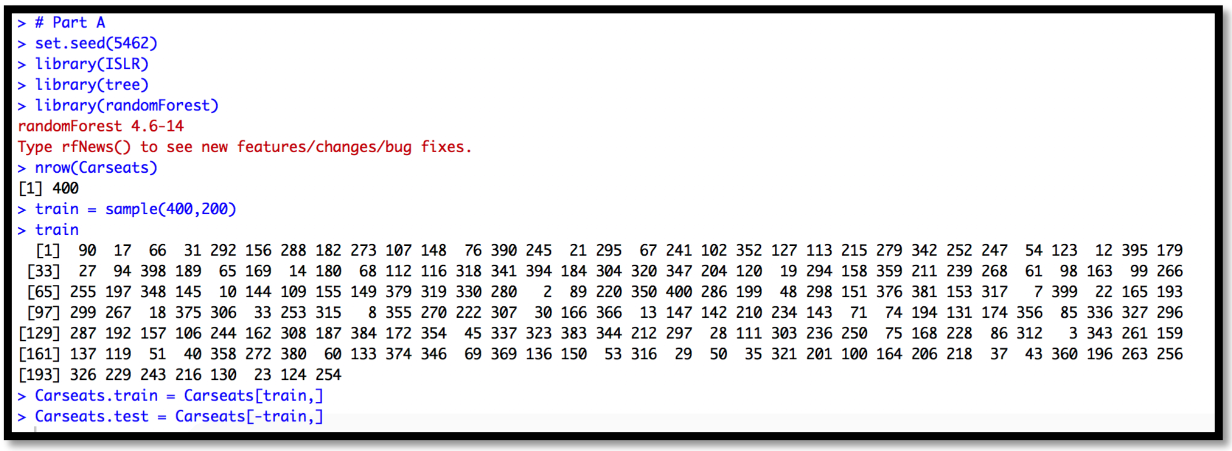
```
> xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.new)[, -1]
> lasso.mod = cv.glmnet(xmat, Y, alpha = 1)
> best.lambda = lasso.mod$lambda.min
> best.lambda
[1] 0.06898654
> best.model = glmnet(xmat, Y, alpha = 1)
> predict(best.model, s = best.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                                1
(Intercept)           1.110252353
poly(x, 10, raw = T)1  1.430897410
poly(x, 10, raw = T)2  0.707192583
poly(x, 10, raw = T)3  0.806966449
poly(x, 10, raw = T)4  0.046771255
poly(x, 10, raw = T)5  0.005935772
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 .
>
```

The lambda generated from cross validation comes out to be 0.06898654. The model generated from lasso method is not so accurate as the one generated by best subset selection (ADJR2), since it picks variables X to X^5.
Hence after using the best subset selection method and lasso method, the best model was selected by best subdset selection (adjr2) as it was closest to the real model.

Chapter 8, Question 8

a) Rcommand:
```
>set.seed(5462)
> library(ISLR)
> library(tree)
>library(randomForest)
>nrow(Carseats)
> train = sample(400,200)
> train
> Carseats.train = Carseats[train,]
> Carseats.test = Carseats[-train,]
```

Output:

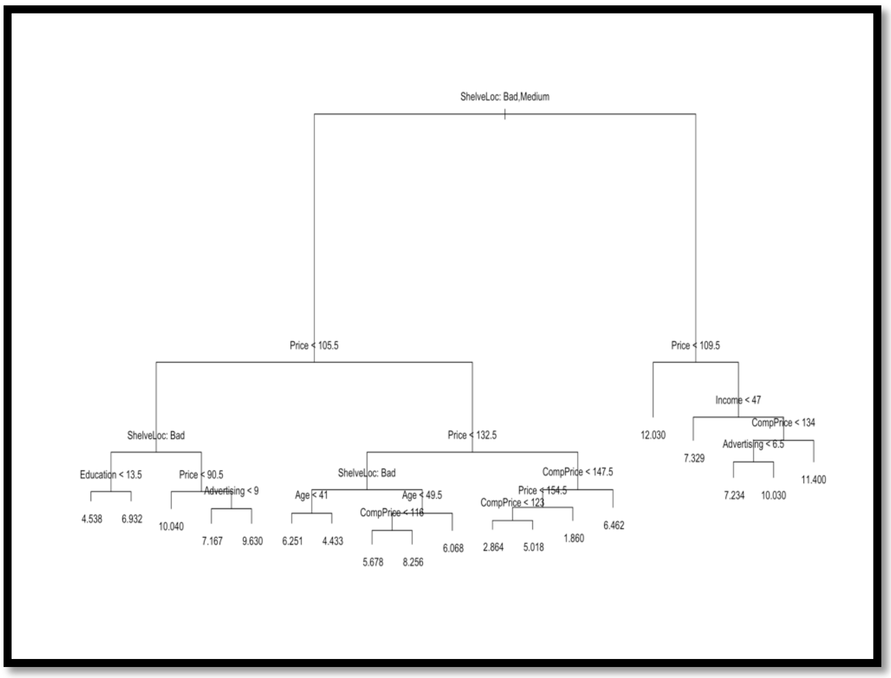


```
> # Part A
> set.seed(5462)
> library(ISLR)
> library(tree)
> library(randomForest)
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
> nrow(Carseats)
[1] 400
> train = sample(400,200)
> train
  [1]  90  17  66  31 292 156 288 182 273 107 148  76 390 245  21 295  67 241 102 352 127 113 215 279 342 252 247  54 123  12 395 179
 [33]  27  94 398 189  65 169  14 180  68 112 116 318 341 394 184 304 320 347 204 120  19 294 158 359 211 239 268  61  98 163  99 266
 [65] 255 197 348 145  10 144 109 155 149 379 319 330 280   2  89 220 350 400 286 199  48 298 151 376 381 153 317   7 399  22 165 193
 [97] 299 267  18 375 306  33 253 315   8 355 270 222 307  30 166 366  13 147 142 210 234 143  71  74 194 131 174 356  85 336 327 296
[129] 287 192 157 106 244 162 308 187 384 172 354  45 337 323 383 344 212 297  28 111 303 236 250  75 168 228  86 312   3 343 261 159
[161] 137 119  51  40 358 272 380  60 133 374 346  69 369 136 150  53 316  29  50  35 321 201 100 164 206 218  37  43 360 196 263 256
[193] 326 229 243 216 130  23 124 254
> Carseats.train = Carseats[train,]
> Carseats.test = Carseats[-train,]
```

__**The data is split into a training and a testing set on a 1:1 ratio.**__

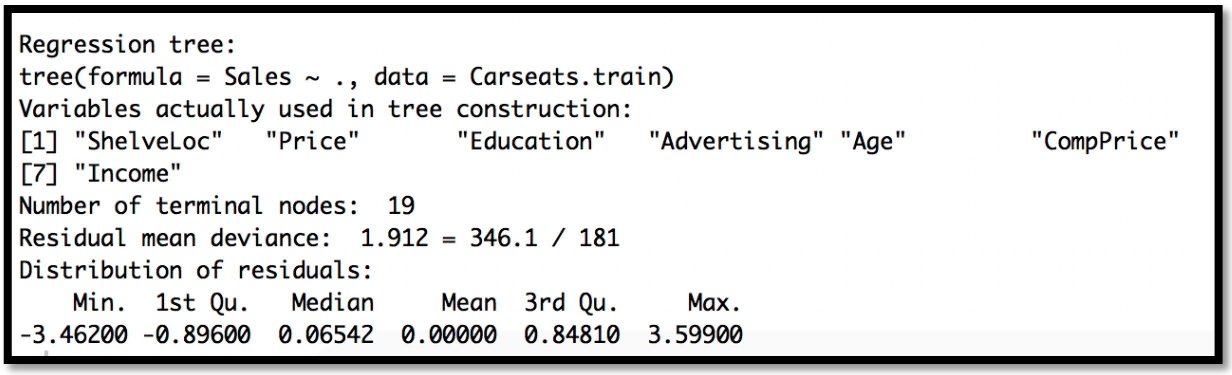
b) Rcommand:
```
>carseats.tree = tree(Sales~.,data=Carseats.train)
>plot(carseats.tree)
>text(carseats.tree, pretty = 0)
```

Output:



Rcommand: summary(carseats.tree)

Output:

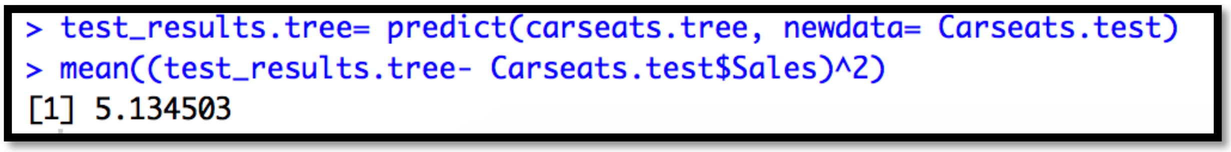


```
Regression tree:
tree(formula = Sales ~ ., data = Carseats.train)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Education"   "Advertising" "Age"         "CompPrice"
[7] "Income"
Number of terminal nodes:  19
Residual mean deviance:  1.912 = 346.1 / 181
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.46200 -0.89600  0.06542  0.00000  0.84810  3.59900
```

<u>The tree produced has 19 terminal nodes.</u>

Rcommand: >test_results.tree= predict(carseats.tree, newdata= Carseats.test)
>mean((test_results.tree- Carseats.test$Sales)^2)

Output:

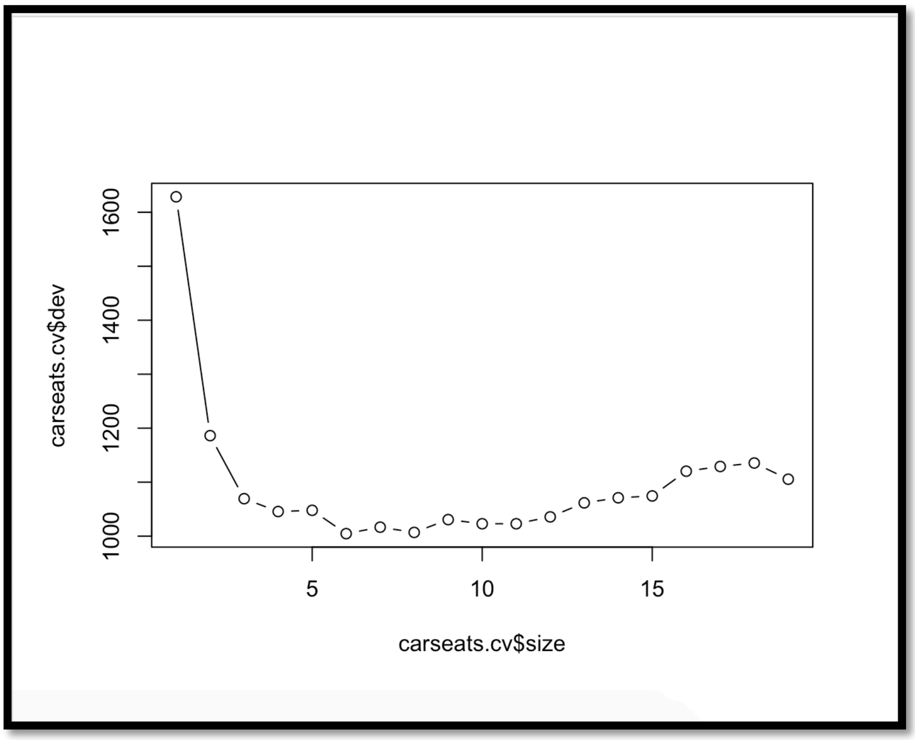


```
> test_results.tree= predict(carseats.tree, newdata= Carseats.test)
> mean((test_results.tree- Carseats.test$Sales)^2)
[1] 5.134503
```

The test MSE comes out to be 5.134503.

c) Rcommand: > carseats.cv = cv.tree(carseats.tree)
>plot(carseats.cv$size, carseats.cv$dev, type="b")

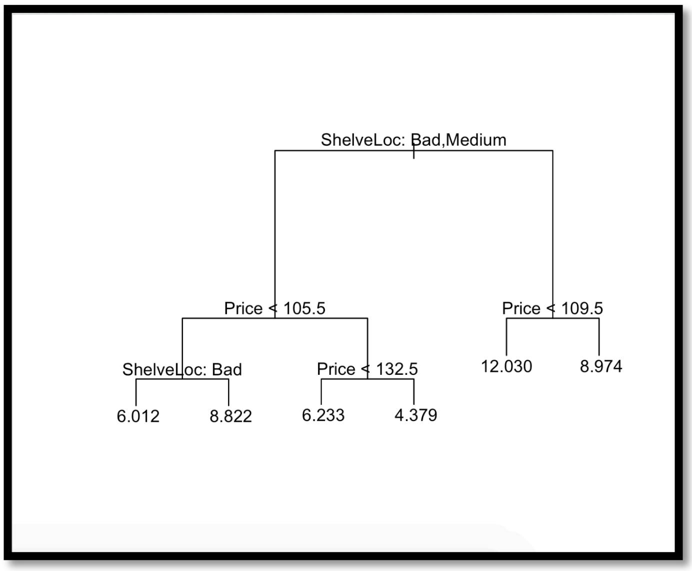
Output:



<u>As we can see, the deviance of is the lowest when the size is 6, and hence we can use 6 to prune the tree.</u>

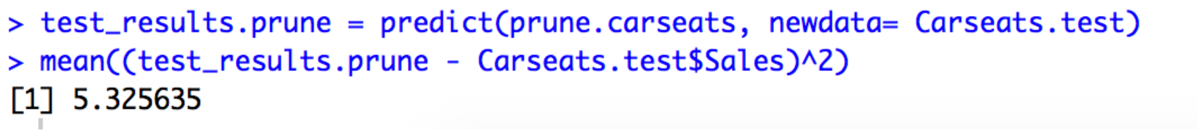
Rcommand: > prune.carseats = prune.tree(carseats.tree, best = 6)
> plot(prune.carseats)
> text(prune.carseats, pretty = 0)

Output:



Rcommand: > test_results.prune = predict(prune.carseats, newdata= Carseats.test)
>mean((test_results.prune - Carseats.test$Sales)^2)

Output:

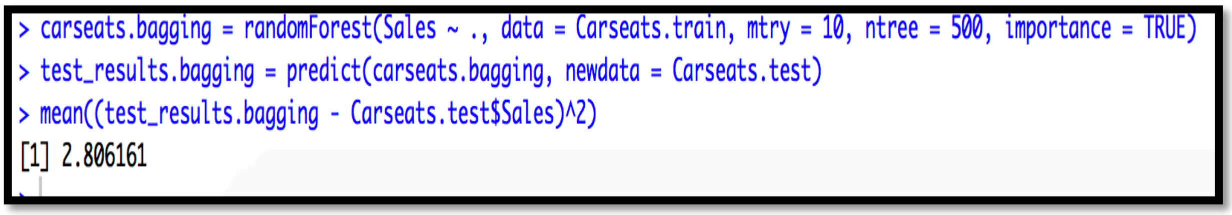


```
> test_results.prune = predict(prune.carseats, newdata= Carseats.test)
> mean((test_results.prune - Carseats.test$Sales)^2)
[1] 5.325635
```

<u>The test MSE in the case of pruning the tree comes out to be 5.325635 which is higher than the case in which the tree was not pruned. Hence, the test MSE increases.</u>

d) Rcommand: > carseats.bagging = randomForest(Sales ~ ., data = Carseats.train, mtry = 10,ntree = 500, importance = TRUE)
> test_results.bagging = predict(carseats.bagging, newdata = Carseats.test)
> mean((test_results.bagging - Carseats.test$Sales)^2)

Output:

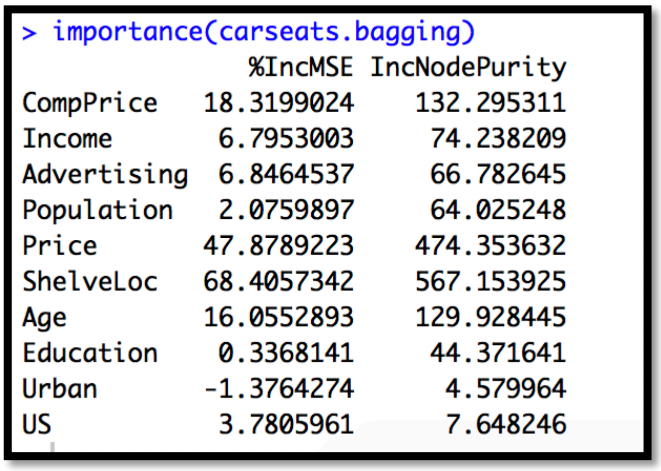


```
> carseats.bagging = randomForest(Sales ~ ., data = Carseats.train, mtry = 10, ntree = 500, importance = TRUE)
> test_results.bagging = predict(carseats.bagging, newdata = Carseats.test)
> mean((test_results.bagging - Carseats.test$Sales)^2)
[1] 2.806161
```

The test MSE when the bagging approach is used comes out to be **<u>2.806161.</u>**

Rcommand: > importance(carseats.bagging)

Output:

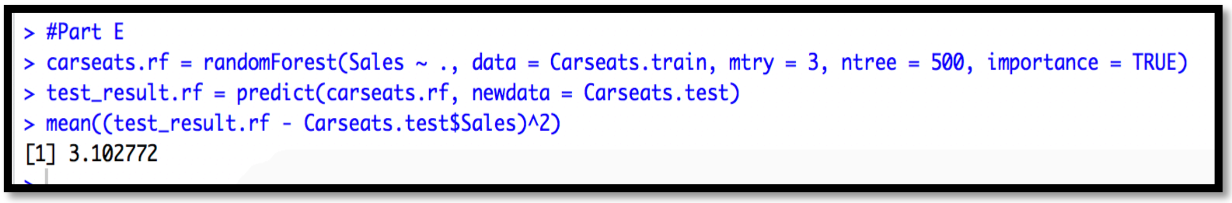


```
> importance(carseats.bagging)
              %IncMSE IncNodePurity
CompPrice   18.3199024    132.295311
Income       6.7953003     74.238209
Advertising  6.8464537     66.782645
Population   2.0759897     64.025248
Price       47.8789223    474.353632
ShelveLoc   68.4057342    567.153925
Age         16.0552893    129.928445
Education    0.3368141     44.371641
Urban       -1.3764274      4.579964
US           3.7805961      7.648246
```

Using the importance() function we can determine that the most important variables are **"Price", "Shelveloc" and "CompPrice".**

e) Rcommand: > carseats.rf = randomForest(Sales ~ ., data = Carseats.train, mtry = 3, ntree = 500, importance = TRUE)
> test_result.rf = predict(carseats.rf, newdata = Carseats.test)
> mean((test_result.rf - Carseats.test$Sales)^2)

Output:

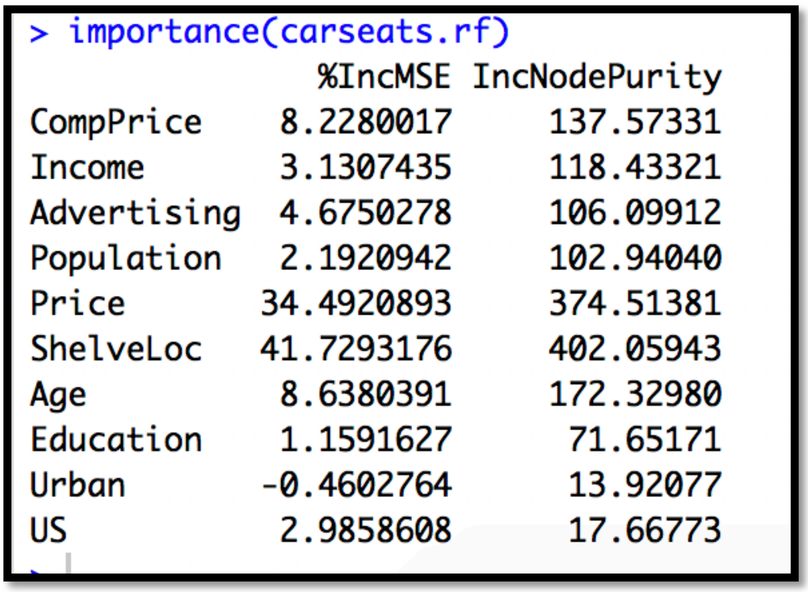


```
> #Part E
> carseats.rf = randomForest(Sales ~ ., data = Carseats.train, mtry = 3, ntree = 500, importance = TRUE)
> test_result.rf = predict(carseats.rf, newdata = Carseats.test)
> mean((test_result.rf - Carseats.test$Sales)^2)
[1] 3.102772
```

The test MSE in this case where Random Forest approach is used comes out to be 3.102772. The test error rate becomes worse when the value of m changes. **The test MSE increases when the value of m is decreased from 10 to 3.**

Rcommand: > importance(carseats.rf)

Output:



```
> importance(carseats.rf)
               %IncMSE IncNodePurity
CompPrice    8.2280017     137.57331
Income       3.1307435     118.43321
Advertising  4.6750278     106.09912
Population   2.1920942     102.94040
Price       34.4920893     374.51381
ShelveLoc   41.7293176     402.05943
Age          8.6380391     172.32980
Education    1.1591627      71.65171
Urban       -0.4602764      13.92077
US           2.9858608      17.66773
```

The most important variables according to the importance() function come out to be "**Price" and "ShelveLoc"** when the Random Forest approach is used.