**LINEAR MODEL SELECTION AND PREDICTION**

**ARVIND PAWAR**

**DHRUVIN KOTHARI**

**NORTHEASTERN UNIVERSITY**

**ALY6015 INTERMEDIATE ANALYTICS**

**INSTRUCTOR NAME- TENGLONG LI**

**DATE- 03/13/2019**



PART- Ⅰ

A. Import the “assignment3.csv” dataset into your R Studio. Split the data into a training dataset and a test dataset, with 75% of observations randomly going to the training data and 25% randomly going to the test data. Please use set.seed(12345) for this step so that we will have reproducible results.

Solution:

> #part1

> setwd("C:/Users/Arvind/Desktop/Intermediate Analytics/Week 3")

> a3 <- read.csv("assignment3.csv")

> set.seed(12345)

> train <- floor(0.75\*nrow(a3))

> train\_ind <-sample(seq\_len(nrow(a3)),size = train)

> trainset <- a3[train\_ind, ]

> testset <- a3[-train\_ind, ]

> dim(trainset)

[1] 186 9

> dim(testset)

[1] 63 9

read.csv() function is used for reading the .csv file. The above data is split into randomly selected 75% training data and rest 25% for test data. We used ISLR library and floor() function for splitting data. Set.seed() function is used to set the seed from random number generator, which is beneficial for creating simulations or random objects that can be reproduced. Used dim() function to know the number of rows and columns.

**B1**. **Let’s start with manual feature reduction. Use linear regression to predict EP (energy production) from the other variables in the dataset (using your training data).**

**Solution:**

In this step we used linear regression function lm() in order to predict Energy Production. We have

> Model <- lm(EP~wind+pressure+humidity+visability+FFMC+DMC+DC+ISI, data = trainset)#build the regression model for EP

> summary(Model)

From the above code, we have created a regression model and named it as Model with all independent variables to predict dependent variable EP(Energy production) using the 75% training dataset and linear regression. The p-value that we got from doing so is, 2.231e-09, which is very less than the significance level of 0.05.

Linear model can be defined as:

Y = b0 + b1X+e,

Y=dependent variable

X=independent variable

B0=intercept

The results that we got from the above code is:

> Model<- lm(EP~wind+pressure+humidity+visability+FFMC+DMC+ISI, data = trainset)#removed DC

> summary(Model)

Now we have to check which variables have strong relationship with dependent variable EP and which model is best fir for prediction. So, we start this by removing one variable at a time based on p-value of the independent variables. The variable that has highest p-value will be removed and then will check the prediction with remaining variables.

Call:

lm(formula = EP ~ wind + pressure + humidity + visability + FFMC +

DMC + ISI, data = trainset)

Residuals:

Min 1Q Median 3Q Max

-25.32 -11.54 -2.82 11.67 40.51

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -796.64289 217.53885 -3.662 0.00033 \*\*\*

wind -1.65624 0.87598 -1.891 0.06029 .

pressure 1.19842 0.16905 7.089 3.05e-11 \*\*\*

humidity 3.22607 3.78344 0.853 0.39498

visability -3.24840 3.49749 -0.929 0.35426

FFMC 0.25830 0.34729 0.744 0.45800

DMC 0.04087 0.02924 1.398 0.16395

ISI 0.14662 0.25465 0.576 0.56549

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.03 on 178 degrees of freedom

Multiple R-squared: 0.2695, Adjusted R-squared: 0.2408

F-statistic: 9.381 on 7 and 178 DF, p-value: 6.989e-10

We can see that, in above output, p-value is 6.989e-10 as we removed DC from above linear regression model. Similarly, we will keep removing variables that don’t have relationship with EP and will observe the p-value.

> Model<- lm(EP~wind+pressure+humidity+visability+FFMC+DMC, data = trainset)#removed ISI

> summary(Model)

Call:

lm(formula = EP ~ wind + pressure + humidity + visability + FFMC +

DMC, data = trainset)

Residuals:

Min 1Q Median 3Q Max

-25.382 -11.908 -2.964 11.906 40.258

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -798.0957 217.1177 -3.676 0.000313 \*\*\*

wind -1.6368 0.8737 -1.873 0.062645 .

pressure 1.2008 0.1687 7.118 2.55e-11 \*\*\*

humidity 2.9186 3.7386 0.781 0.436020

visability -2.9619 3.4554 -0.857 0.392499

FFMC 0.3433 0.3138 1.094 0.275407

DMC 0.0436 0.0288 1.514 0.131759

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.01 on 179 degrees of freedom

Multiple R-squared: 0.2681, Adjusted R-squared: 0.2436

F-statistic: 10.93 on 6 and 179 DF, p-value: 2.324e-10

> Model<- lm(EP~wind+pressure+visability+FFMC+DMC, data = trainset)#removed humidity

> summary(Model)

Call:

lm(formula = EP ~ wind + pressure + visability + FFMC + DMC,

data = trainset)

Residuals:

Min 1Q Median 3Q Max

-25.605 -12.110 -3.067 11.378 39.410

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -737.08845 202.34686 -3.643 0.000353 \*\*\*

wind -1.26335 0.73033 -1.730 0.085376 .

pressure 1.22047 0.16661 7.325 7.72e-12 \*\*\*

visability -0.42503 1.17370 -0.362 0.717683

FFMC 0.32755 0.31279 1.047 0.296416

DMC 0.04382 0.02876 1.523 0.129440

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.99 on 180 degrees of freedom

Multiple R-squared: 0.2656, Adjusted R-squared: 0.2453

F-statistic: 13.02 on 5 and 180 DF, p-value: 8.019e-11

> Model<- lm(EP~wind+pressure+FFMC+DMC, data = trainset)#removed visability

> summary(Model)

Call:

lm(formula = EP ~ wind + pressure + FFMC + DMC, data = trainset)

Residuals:

Min 1Q Median 3Q Max

-25.42 -12.35 -3.17 11.87 39.79

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -777.04533 169.20773 -4.592 8.19e-06 \*\*\*

wind -1.06669 0.48716 -2.190 0.0298 \*

pressure 1.22087 0.16621 7.345 6.76e-12 \*\*\*

FFMC 0.32482 0.31194 1.041 0.2991

DMC 0.04469 0.02859 1.563 0.1198

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.95 on 181 degrees of freedom

Multiple R-squared: 0.2651, Adjusted R-squared: 0.2489

F-statistic: 16.32 on 4 and 181 DF, p-value: 1.953e-11

> Model<- lm(EP~wind+pressure+DMC, data = trainset)#removed FFMC

> summary(Model)

Call:

lm(formula = EP ~ wind + pressure + DMC, data = trainset)

Residuals:

Min 1Q Median 3Q Max

-25.398 -12.467 -3.041 11.686 38.025

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -761.57465 168.59319 -4.517 1.12e-05 \*\*\*

wind -1.01213 0.48444 -2.089 0.0381 \*

pressure 1.23135 0.16594 7.420 4.31e-12 \*\*\*

DMC 0.05951 0.02481 2.399 0.0175 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.96 on 182 degrees of freedom

Multiple R-squared: 0.2607, Adjusted R-squared: 0.2485

F-statistic: 21.39 on 3 and 182 DF, p-value: 6.463e-12

> Model<- lm(EP~pressure+DMC, data = trainset)#removed wind

> summary(Model)

Call:

lm(formula = EP ~ pressure + DMC, data = trainset)

Residuals:

Min 1Q Median 3Q Max

-25.952 -11.953 -2.782 11.441 36.214

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -789.06533 169.61721 -4.652 6.29e-06 \*\*\*

pressure 1.22339 0.16741 7.308 8.14e-12 \*\*\*

DMC 0.05300 0.02483 2.134 0.0342 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.09 on 183 degrees of freedom

Multiple R-squared: 0.243, Adjusted R-squared: 0.2347

F-statistic: 29.37 on 2 and 183 DF, p-value: 8.675e-12

> Reduced\_Model<- lm(EP~pressure, data = trainset)#removed DMC

> summary(Model)

Call:

lm(formula = EP ~ pressure + DMC, data = trainset)

Residuals:

Min 1Q Median 3Q Max

-25.952 -11.953 -2.782 11.441 36.214

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -789.06533 169.61721 -4.652 6.29e-06 \*\*\*

pressure 1.22339 0.16741 7.308 8.14e-12 \*\*\*

DMC 0.05300 0.02483 2.134 0.0342 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.09 on 183 degrees of freedom

Multiple R-squared: 0.243, Adjusted R-squared: 0.2347

F-statistic: 29.37 on 2 and 183 DF, p-value: 8.675e-12

Now we have removed the weakly correlated variables and we can say that variable pressure is highly correlated with dependent variable EP. Only pressure variable has p-value 8.827e-1, which is very less than 0.05 significance level. Also, R-square value is 0.243 that means 24.3% variation. The value of R-squared is better if it is greater. Above p-value, more variation and higher R-squared value can state that the reduced model is a best fit for this dataset.

**B2**. **Use the anova() function to see if the saturated and reduced models are significantly different from one another. Record results of this test.**

**Solution:**

ANOVA- In short can be called as Analysis of variance, it is a statistical technique for testing if their population means are all equal.

> anova(Reduced\_Model, Model)

Analysis of Variance Table

Model 1: EP ~ pressure

Model 2: EP ~ wind + pressure + humidity + visability + FFMC + DMC + DC +

ISI

Res.Df RSS Df Sum of Sq F Pr(>F)

1 184 42735

2 177 40231 7 2504.3 1.574 0.1458

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

**B3**. **Using your saturated model and your reduced model (separately) predict values of Y using your test data. Calculate R2 for each model to assess prediction error.**

> install.packages("leaps")

> library(leaps)

> predict\_model<-predict(Model,newdata = testset)

> head(predict\_model)#using saturated model

3 5 6 7 8 17

452.9374 453.4993 463.4489 454.0606 463.6000 459.5900

> Observed\_Values <- testset$EP

> head(Observed\_Values)

[1] 465.86 471.23 457.98 440.74 439.14 463.11

> #calculate the the R\_Squared Value and RMSE for Model

> SSE<-sum((Observed\_Values-predict\_model)^2)

> SST<-sum((Observed\_Values-mean(Observed\_Values))^2)

> r\_squared<-1-SSE/SST

> r\_squared

[1] 0.1712093

> RMSE<-sqrt(mean((Observed\_Values-predict\_model)^2))

> RMSE

[1] 16.49502

> RSS <- sum((predict\_model - Observed\_Values)^2)

> RSS

[1] 17141.4

> MAE <- mean(abs(Observed\_Values - predict\_model))

> MAE

[1] 13.60792

> #predict reduced model

> predict\_reduced<- predict(Reduced\_Model, newdata=testset)

> head(predict\_reduced)

3 5 6 7 8 17

455.3763 455.5365 463.7414 454.0951 460.6861 462.4971

> #calculate the the R\_Squared Value and RMSE for Model

> SSE1<-sum((Observed\_Values-predict\_reduced)^2)

> SST1<-sum((Observed\_Values-mean(Observed\_Values))^2)

> r\_squared1<-1-SSE1/SST1

> r\_squared1

[1] 0.3127864

> RMSE1<-sqrt(mean((Observed\_Values-predict\_reduced)^2))

> RMSE1

[1] 15.02022

> RSS1 <- sum((predict\_reduced - Observed\_Values)^2)

> RSS1

[1] 14213.24

> MAE1 <- mean(abs(Observed\_Values - predict\_reduced))

> MAE1

[1] 12.79585

Parameters Saturated Model Reduced Model

R-square 0.1712 0.3127

RMSE 16.495 15.020

MAE 13.607 12.795

RSS 17141.4 14213.24

From above result we can conclude that our reduced model is better than the first Model.

**C.** **Use the stepAIC() function to implement backward selection (starting with the full model). Does this approach give you the same reduced model you found above?**

**Solution:**

We have performed the stepAIC() function with backward direction.

> install.packages("ISLR")

> library(ISLR)

> fit\_model <- lm(EP~., data = a3)

> Step\_AIC <- stepAIC(fit\_model, direction = "backward")

Start: AIC=1365.3

EP ~ wind + pressure + humidity + visability + FFMC + DMC + DC +

ISI

Df Sum of Sq RSS AIC

- ISI 1 0.0 55732 1363.3

- DMC 1 13.6 55746 1363.4

- DC 1 75.5 55808 1363.6

- humidity 1 166.8 55899 1364.0

- FFMC 1 174.3 55907 1364.1

- visability 1 237.7 55970 1364.4

<none> 55732 1365.3

- wind 1 494.1 56226 1365.5

- pressure 1 17552.4 73285 1431.5

Step: AIC=1363.3

EP ~ wind + pressure + humidity + visability + FFMC + DMC + DC

Df Sum of Sq RSS AIC

- DMC 1 14.3 55747 1361.4

- DC 1 76.5 55809 1361.7

- humidity 1 168.7 55901 1362.1

- FFMC 1 219.7 55952 1362.3

- visability 1 240.7 55973 1362.4

<none> 55732 1363.3

- wind 1 494.5 56227 1363.5

- pressure 1 17556.2 73289 1429.5

Step: AIC=1361.37

EP ~ wind + pressure + humidity + visability + FFMC + DC

Df Sum of Sq RSS AIC

- DC 1 73.1 55820 1359.7

- humidity 1 173.6 55920 1360.1

- FFMC 1 206.0 55953 1360.3

- visability 1 241.0 55988 1360.4

<none> 55747 1361.4

- wind 1 501.0 56248 1361.6

- pressure 1 17571.0 73318 1427.6

Step: AIC=1359.7

EP ~ wind + pressure + humidity + visability + FFMC

Df Sum of Sq RSS AIC

- humidity 1 185.6 56005 1358.5

- visability 1 258.4 56078 1358.8

- FFMC 1 370.7 56191 1359.3

<none> 55820 1359.7

- wind 1 533.0 56353 1360.1

- pressure 1 17512.0 73332 1425.6

Step: AIC=1358.52

EP ~ wind + pressure + visability + FFMC

Df Sum of Sq RSS AIC

- visability 1 92.7 56098 1356.9

- wind 1 347.6 56353 1358.1

- FFMC 1 355.6 56361 1358.1

<none> 56005 1358.5

- pressure 1 18092.7 74098 1426.2

Step: AIC=1356.93

EP ~ wind + pressure + FFMC

Df Sum of Sq RSS AIC

- wind 1 299.8 56398 1356.3

- FFMC 1 348.2 56446 1356.5

<none> 56098 1356.9

- pressure 1 18139.4 74238 1424.7

Step: AIC=1356.26

EP ~ pressure + FFMC

Df Sum of Sq RSS AIC

- FFMC 1 302.9 56701 1355.6

<none> 56398 1356.3

- pressure 1 18199.3 74597 1423.9

Step: AIC=1355.59

EP ~ pressure

Df Sum of Sq RSS AIC

<none> 56701 1355.6

- pressure 1 19067 75767 1425.8

PART-Ⅱ

**Using the same dataset from above, we’re now going to utilize Lasso to fit a model with all X variables to predict our outcome variable, energy production.**

1. **Transform your training and testing datasets into a matrix of Xs and vector for Y, such that you get the following four objects: training matrix of Xs, training vector of Y, test matrix of Xs, test vector of Y.**

**Solution:**

> #part2

> #a6

> x\_train<-model.matrix(EP~.,trainset)[,-1]

> y\_train<-trainset$EP

> x\_test<-model.matrix(EP~.,testset)[,-1]

> y\_test<-testset$EP

> head(x\_train)

wind pressure humidity visability FFMC DMC DC ISI

180 36.71 1017.25 97.82 75.60 88.8 147.3 614.5 9.0

218 37.20 1013.07 97.25 75.33 88.6 91.8 709.9 7.1

188 36.71 1013.19 97.70 75.60 90.9 126.5 686.5 7.0

248 37.50 1011.33 96.88 75.08 92.2 91.6 503.6 9.6

112 36.08 1016.25 98.98 77.24 91.7 33.3 77.5 9.0

41 35.19 1014.36 100.09 78.05 79.5 60.6 366.7 1.5

> head(y\_train)

[1] 484.98 479.09 449.61 440.95 468.87 439.81

> head(x\_test)

wind pressure humidity visability FFMC DMC DC ISI

3 25.36 1013.32 100.15 80.18 90.6 43.7 686.9 6.7

5 25.36 1013.45 100.14 80.18 89.3 51.3 102.2 9.6

6 25.36 1020.11 100.13 80.18 92.3 85.3 488.0 14.7

7 25.36 1012.28 100.13 80.18 92.3 88.9 495.6 8.5

8 25.88 1017.63 100.13 79.74 91.5 145.4 608.2 10.7

17 34.03 1019.10 100.11 79.74 91.7 35.8 80.8 7.8

> head(y\_test)

[1] 465.86 471.23 457.98 440.74 439.14 463.11

> #b

> install.packages("glmnet")

> library(glmnet)

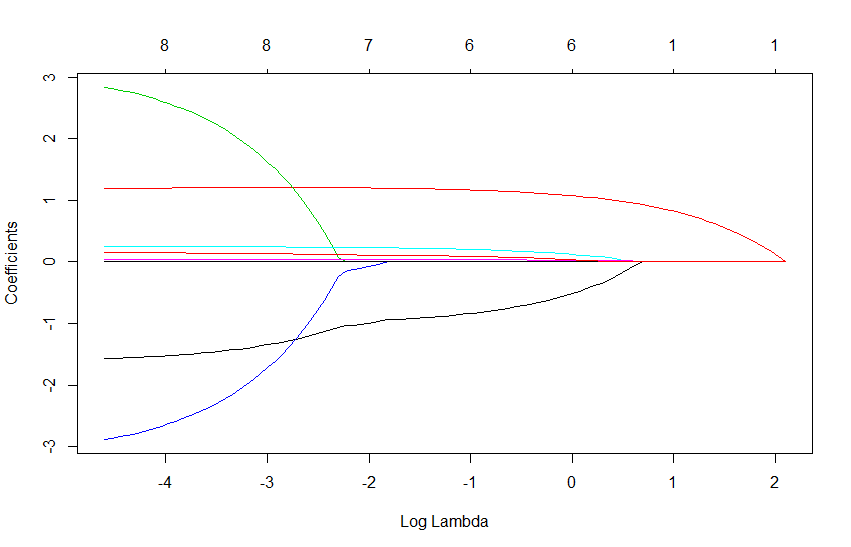
Loading required package: Matrix

Loading required package: foreach

Loaded glmnet 2.0-16

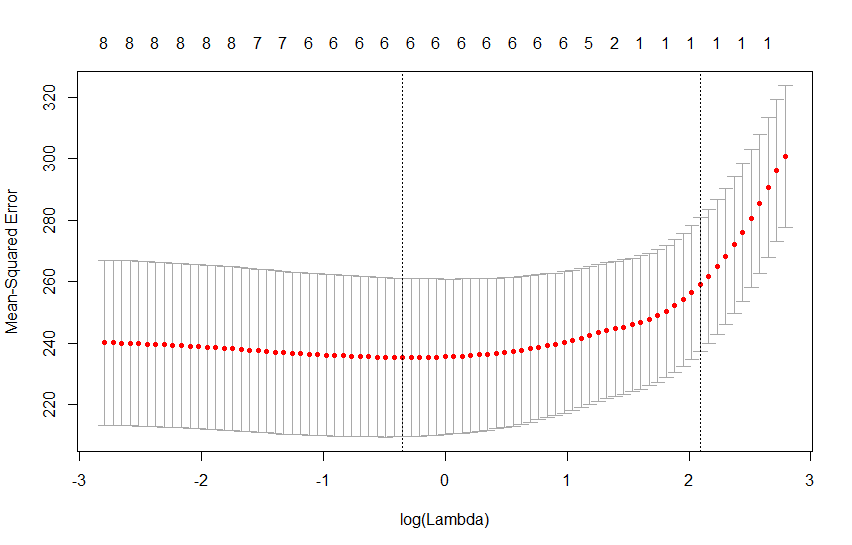
> Lasso\_model<-glmnet(x\_train,y\_train,alpha=1,nlambda = 100,lambda.min.ratio = 0.001)

> plot(Lasso\_model,xvar = "lambda")



> Cross\_Validation<-cv.glmnet(x\_train,y\_train,nlambda = 100,alpha=0.5,lambda.min.ratio=0.001)

> plot(Cross\_Validation)



> best\_lambda<-Cross\_Validation$lambda.min

> predict\_y<-predict(Lasso\_model, s = best\_lambda, newx = x\_test)

> predict\_y

1

3 461.4051

5 461.3947

6 471.0873

7 461.9110

8 469.4150

17 461.3628

19 450.6061

22 453.0118

25 460.4402

27 463.5215

30 450.2886

35 457.4324

39 449.1007

43 453.7446

46 458.2853

47 449.4699

48 454.4933

50 453.1566

61 461.2276

62 452.1577

65 463.5008

69 452.4493

71 462.2707

72 461.5841

81 456.1277

86 463.9228

95 456.1782

103 456.9257

105 448.7384

110 455.8084

118 446.0192

119 448.9852

133 452.1746

134 462.2365

137 465.5908

139 447.5697

140 458.2008

142 463.5972

145 459.5774

149 453.5586

153 455.9370

157 464.0277

170 462.6953

172 444.5343

178 445.8579

184 462.5010

189 460.7282

190 465.7647

191 448.9948

195 461.0545

197 435.8018

200 441.3846

205 464.9455

206 454.5581

213 448.3617

223 458.4699

225 453.9005

236 453.6959

237 446.9339

240 448.4003

245 449.4727

247 462.8001

249 467.6659

> #c

> Observe\_y<-y\_test

> Observe\_y

[1] 465.86 471.23 457.98 440.74 439.14 463.11 442.84 436.36 455.55 442.72 434.65 451.70 480.72 429.50 452.82 438.08 451.47 460.43 460.76

[20] 478.82 452.50 437.96 478.06 484.94 436.44 464.65 448.52 428.68 471.42 481.89 433.99 445.97 443.43 461.14 472.67 433.19 482.52 462.09

[39] 472.68 441.33 472.41 474.10 478.47 431.69 438.63 474.07 455.56 478.29 438.36 438.41 427.71 431.73 476.37 446.77 440.73 479.48 494.67

[58] 443.85 433.44 479.32 434.22 444.66 474.55

**C. Compare predicted Ys and observed Ys using R2 (and/or your favorite prediction error metric). How well does the Lasso approach perform, and compare to the approaches from above?**

**Solution:**

> sse<-sum((observed\_y-predict\_y)^2)

> sst<-sum((observed\_y-mean(observed\_y))^2)

> rsq<-1-SSE/SST

> rsq #

[1] 0.2092551

PART-Ⅲ

Lastly, let’s try PCR with the same training and testing datasets from the energy dataset used above.

**A. Fit a PCR model to your training dataset using the pcr() function. Use the validationplot() function either set to RMSEP, MSEP, and/or R2 to determine how many principle components should be used in the final model. How many principal components did you choose?**

**Solution:**

> #part 3

> install.packages("pls")

> library(pls)

> set.seed (1000)

> pcr\_model <- pcr(EP~., data = a3, scale = TRUE, validation = "CV")

> summary(pcr\_model)

Data: X dimension: 249 8

Y dimension: 249 1

Fit method: svdpc

Number of components considered: 8

VALIDATION: RMSEP

Cross-validated using 10 random segments.

(Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps

CV 17.51 17.57 17.62 15.83 15.40 15.42 15.43 15.47 15.51

adjCV 17.51 17.56 17.60 15.81 15.38 15.40 15.40 15.45 15.48

TRAINING: % variance explained

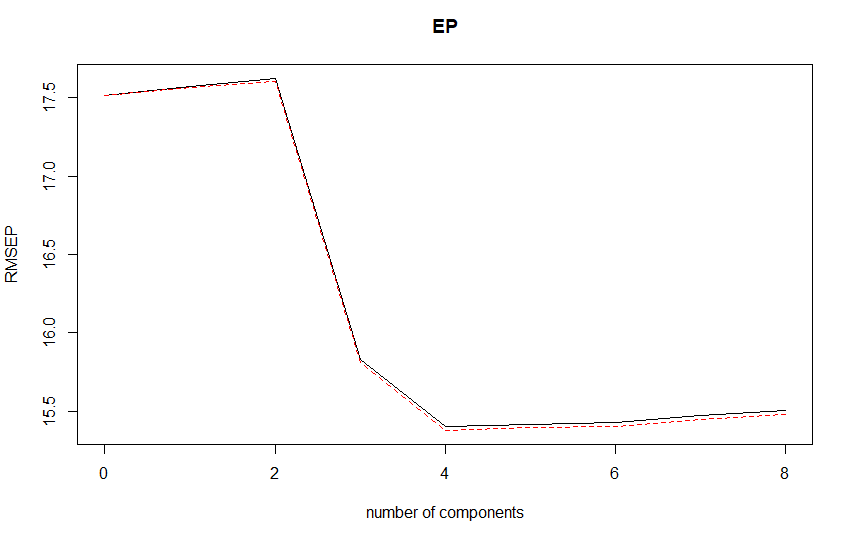
1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps

X 33.2166 62.170 75.10 86.03 91.89 96.82 99.64 100.00

EP 0.1251 1.897 20.31 25.79 25.94 26.10 26.15 26.44

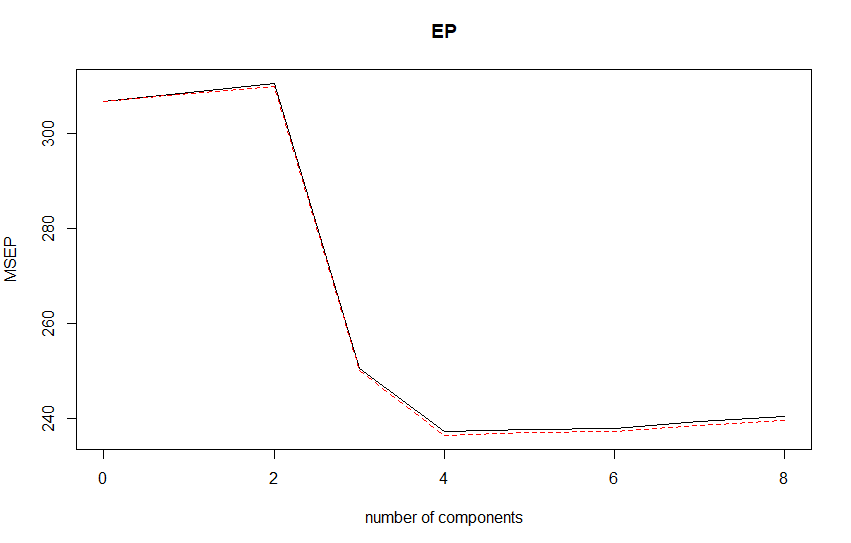
> ##RMSEP

> validationplot(pcr\_model,val.type = c("RMSEP"))



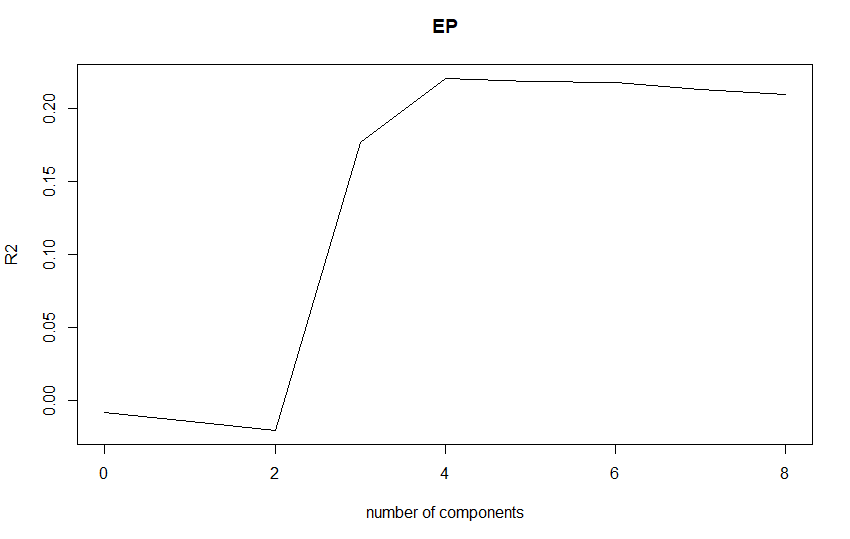
> ##MSEP

> validationplot(pcr\_model,val.type = c("MSEP"))



> ##R2

> validationplot(pcr\_model,val.type = c("R2"))



Let’s compare both PCR model and Linear regression with saturated model using the mean squared error.

> ##5 comm

> pcr\_model\_train<-pcr(EP~.,data=trainset,scale=TRUE,validation='CV')

> pcr\_pred <- predict(pcr\_model, testset, ncomp = 5)

> mean((pcr\_pred - testset$EP)^2)

[1] 230.948

> summary(pcr\_pred)

Min. 1st Qu. Median Mean 3rd Qu. Max.

435.0 450.3 456.2 456.0 462.9 469.3

> lrm <- lm(EP~., data = trainset)

> lrm\_pred <- predict(lrm, testset)

> mean((lrm\_pred - testset$EP)^2)

[1] 272.0857

The Mean Squared Error of Linear Regression is 272.0857 which is similar to PCR. It only reduces our full model from 8 variables to 4 components. As compared to Linear and Lasso Regression, PCR may not be the approach.

References:

* R Multiple Regression. (2019). www.tutorialspoint.com. Retrieved 13 March 2019, from <https://www.tutorialspoint.com/r/r_multiple_regression.htm>
* Ridge Regression and the Lasso. (2017). R-bloggers. Retrieved 13 March 2019, from <https://www.r-bloggers.com/ridge-regression-and-the-lasso/>
* Alice, M., & Alice, M. (2016). Performing Principal Components Regression (PCR) in R. MilanoR. Retrieved 13 March 2019, from <http://www.milanor.net/blog/performing-principal-components-regression-pcr-in-r/>
* stepAIC function | R Documentation. (2019). Rdocumentation.org. Retrieved 13 March 2019, from <https://www.rdocumentation.org/packages/MASS/versions/7.3-51.1/topics/stepAIC>