Multivariate Data Analysis - Assignment 2

## Multivariate Data Analysis Spring 2019 (37459-2019-SPRING-CITY)

### Assignment: 2

### Student Name: Anuj Kapil

### Student Id: 12678708

## Part A

## Question 1

For a given mean vector and covariance matrix, we can simulate random samples from the multivariate normal distribution in R using the ‘mvrnorm’ function from **MASS** package.

# Question 1  
# Mean vector  
mv<-rep(0, 3)  
  
# Cov matrix  
vcmat <- 1/5630 \* matrix(c(575,-60,10,-60,300,-50,10,-50,196),nrow=3,byrow=TRUE)  
  
# Covariance matrix  
print(vcmat)

## [,1] [,2] [,3]  
## [1,] 0.102131439 -0.010657194 0.001776199  
## [2,] -0.010657194 0.053285968 -0.008880995  
## [3,] 0.001776199 -0.008880995 0.034813499

#MVN  
mnd <- mvrnorm(n=1000,mv,vcmat)

### Question 1a

Calculate the least square estimates using R function for Y2 and Y3 where:

and

Peform a linear regression to find the coefficients , and .

# Question 1a  
  
# Convert matrix to a data.table  
mnd\_df <- as.data.frame(as.table(mnd))  
setDT(mnd\_df)  
mnd\_dt <- dcast(mnd\_df, Var1~Var2, value.var = 'Freq')  
mnd\_dt[,Var1:=NULL]  
  
colnames(mnd\_dt) <- c('Y1', 'Y2', 'Y3')  
  
model\_1<-lm(Y2~Y1, data = mnd\_dt)  
  
model\_summary <- summary(model\_1)  
  
# Coefficent of Y1  
beta2\_1 <- model\_summary$coefficients[[2]]  
print(beta2\_1)

## [1] -0.09761641

model\_2<-lm(Y3~Y1+Y2, data = mnd\_dt)  
  
model\_summary <- summary(model\_2)  
  
#Coefficent of Y1  
beta3\_1 <- model\_summary$coefficients[[2]]  
print(beta3\_1)

## [1] -0.04073385

#Coefficent of Y2  
beta3\_2 <- model\_summary$coefficients[[3]]  
print(beta3\_2)

## [1] -0.1806884

### Question 1b

Estimate

# Question 1b  
sigma\_2\_square <- (summary(model\_1)$sigma)^2  
print(sigma\_2\_square)

## [1] 0.05030198

### Question 1c

Estimate

# Question 1c  
sigma\_3\_square <- (summary(model\_2)$sigma)^2  
print(sigma\_3\_square)

## [1] 0.0335324

### Question 1d

Construct the 3x3 matrix from coefficients

T <- matrix(c(1,-1\*beta2\_1,-1\*beta3\_1,0,1,-1\*beta3\_2,0,0,1),nrow = 3)  
print(T)

## [,1] [,2] [,3]  
## [1,] 1.00000000 0.0000000 0  
## [2,] 0.09761641 1.0000000 0  
## [3,] 0.04073385 0.1806884 1

### Question 1e

Compute

TT <- T\*vcmat\*t(T)  
print(TT)

## [,1] [,2] [,3]  
## [1,] 0.1021314 0.00000000 0.0000000  
## [2,] 0.0000000 0.05328597 0.0000000  
## [3,] 0.0000000 0.00000000 0.0348135

### Question 1f

TT\_inv <- solve(TT)  
sigma\_inv <- solve(vcmat)  
  
y <- sigma\_inv[1,1]   
sigma\_1\_square <-y-(T[2,1]^2\*sigma\_2\_square+ T[3,1]\*sigma\_3\_square)

## Question 2

### Load the dataset from local storage

Load the dataset using ‘fread’ from **data.table** package

dat <- fread('Data/stockdata.csv')

### Question 2a

Perform factor analysis

fact <- factanal(x = dat,factors = 2)

### Question 2b

From loading matrix we can see: 1. factor 1 explains JPMorgan in 76% of variance, Citibank in 81% of variance, wellsfargo in 66% of variance, 2. Royaldutchshell in 11% of variance, ExxonMoboil in 10% of variance 3. factor 2 epxlains 23% of Citibank variance, 10% of wellsfargo variance,99% of RotalDutchShell vairnace, 67% of ExxonMobil variance

library('psych')

##   
## Attaching package: 'psych'

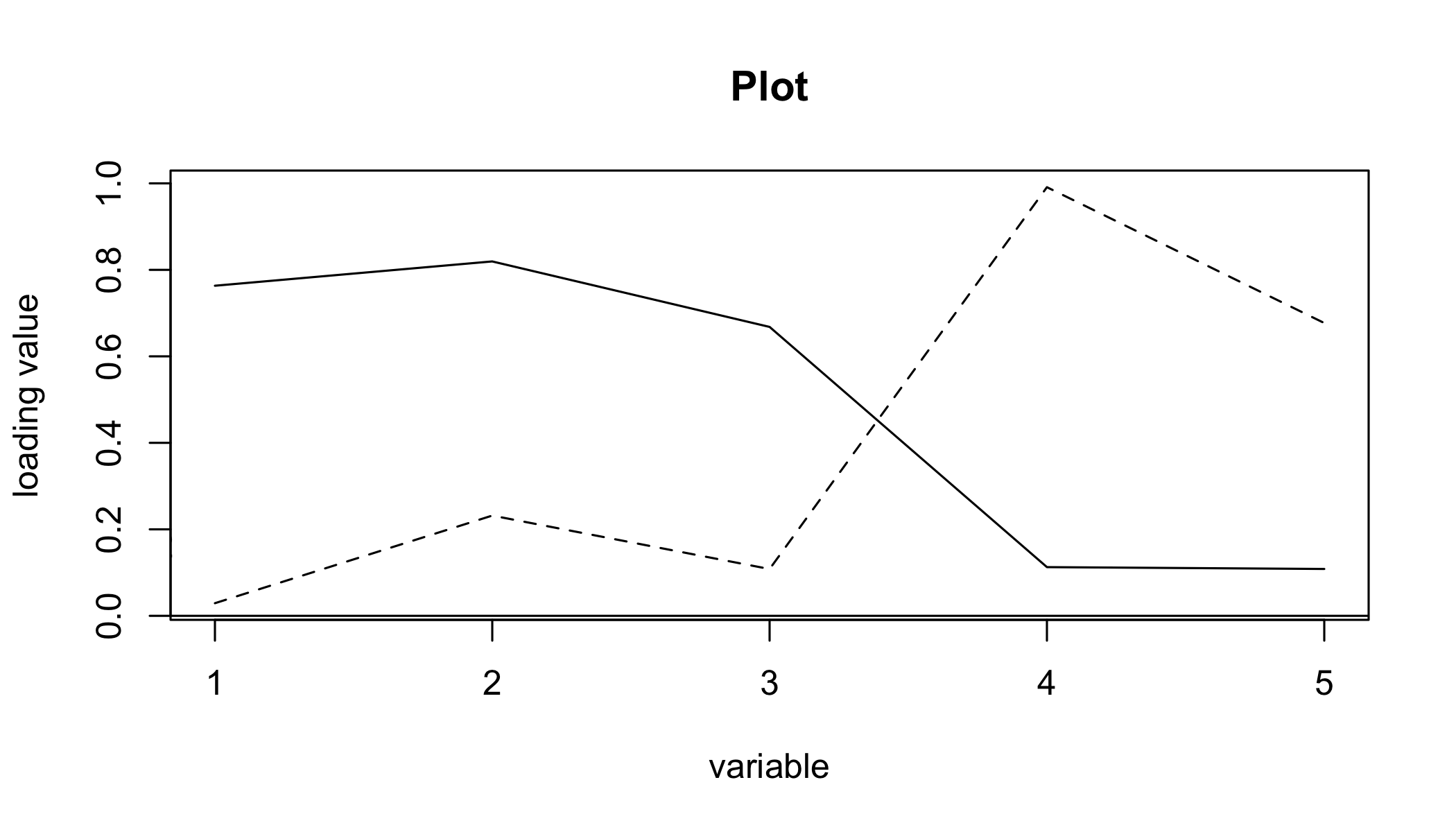
## The following objects are masked from 'package:VGAM':  
##   
## fisherz, logistic, logit

## The following object is masked from 'package:car':  
##   
## logit

## The following object is masked from 'package:lars':  
##   
## error.bars

## The following objects are masked from 'package:ggplot2':  
##   
## %+%, alpha

factor.plot(fact)



Also from the loading plot, we can also see factor2 has influces more on RoyalDutchshell and ExxonMobil factor1 has more influences on Citibank and JPMorgan

### Question 2c

Compare

## Question 3

### Load the dataset from local storage

egyptskull <- fread('Data/egyptskull.csv')  
  
summary(egyptskull)

## MB BH BL NH   
## Min. :119 Min. :120.0 Min. : 81.00 Min. :44.00   
## 1st Qu.:131 1st Qu.:129.0 1st Qu.: 93.00 1st Qu.:49.00   
## Median :134 Median :133.0 Median : 96.00 Median :51.00   
## Mean :134 Mean :132.5 Mean : 96.46 Mean :50.93   
## 3rd Qu.:137 3rd Qu.:136.0 3rd Qu.:100.00 3rd Qu.:53.00   
## Max. :148 Max. :145.0 Max. :114.00 Max. :60.00   
## Epoch   
## Min. : 150   
## 1st Qu.: 200   
## Median :1850   
## Mean :1900   
## 3rd Qu.:3300   
## Max. :4000

egyptskull[, Epoch:= as.factor(Epoch)]

### Question 3a

Logistics Regression

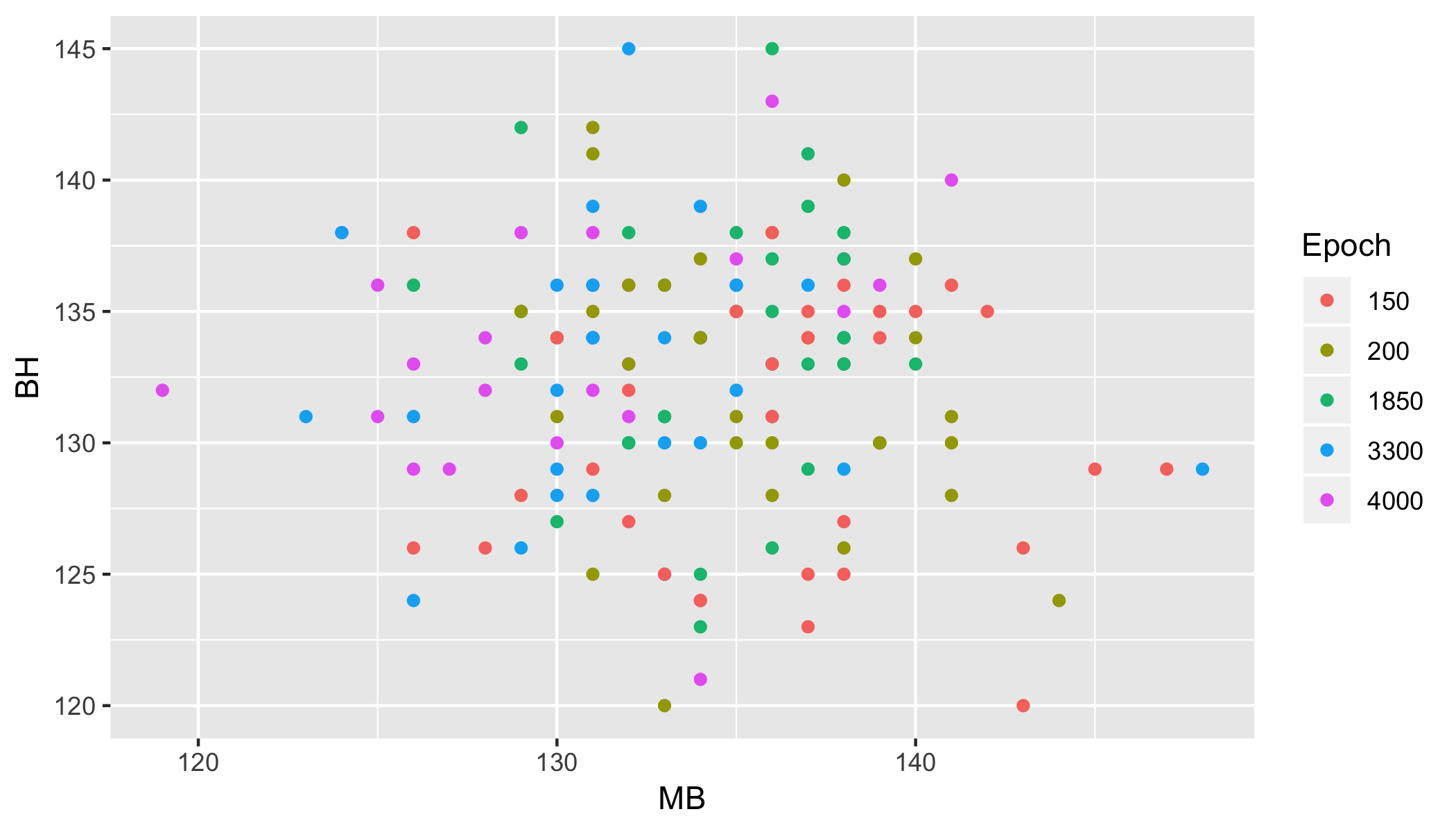
### Question 3b

Classification Trees

### Question 3c

Scatter Plot

ggplot(egyptskull, aes(x=MB, y=BH, group=Epoch))+  
 geom\_point(aes(color=Epoch))



### Question 3d

Split the dataset in to train and test datasets. For multionomial regression, we need to create 5 different response variables to denote the five levels of Epoch categories.

egyptskull[, Epoch\_1:=ifelse(Epoch == 4000, 1, 0)]  
egyptskull[, Epoch\_2:=ifelse(Epoch == 3300, 1, 0)]  
egyptskull[, Epoch\_3:=ifelse(Epoch == 1850, 1, 0)]  
egyptskull[, Epoch\_4:=ifelse(Epoch == 200, 1, 0)]  
egyptskull[, Epoch\_5:=ifelse(Epoch == 150, 1, 0)]  
  
  
egyptskull\_train <- egyptskull[,.SD[1:25], by = list(Epoch)]  
egyptskull\_test <- egyptskull[,.SD[26:30], by = list(Epoch)]  
  
egyptskull\_train[, .N, by = list(Epoch)]

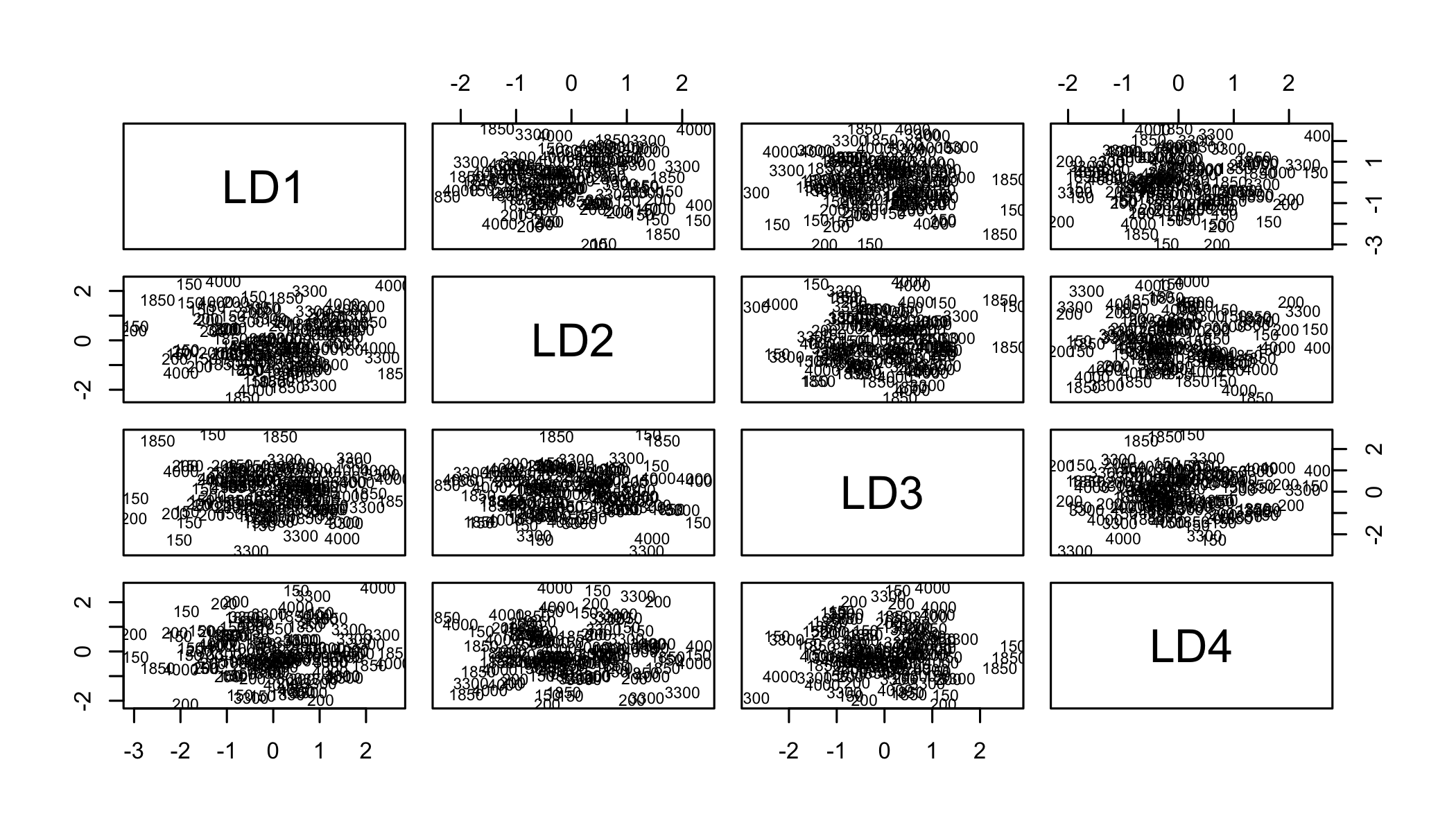
## Epoch N  
## 1: 4000 25  
## 2: 3300 25  
## 3: 1850 25  
## 4: 200 25  
## 5: 150 25

egyptskull\_test[, .N, by = list(Epoch)]

## Epoch N  
## 1: 4000 5  
## 2: 3300 5  
## 3: 1850 5  
## 4: 200 5  
## 5: 150 5

Perform LDA

####### LDA  
  
model\_lda <- lda(Epoch ~ MB+BH+BL+NH, data=egyptskull\_train)  
plot(model\_lda)



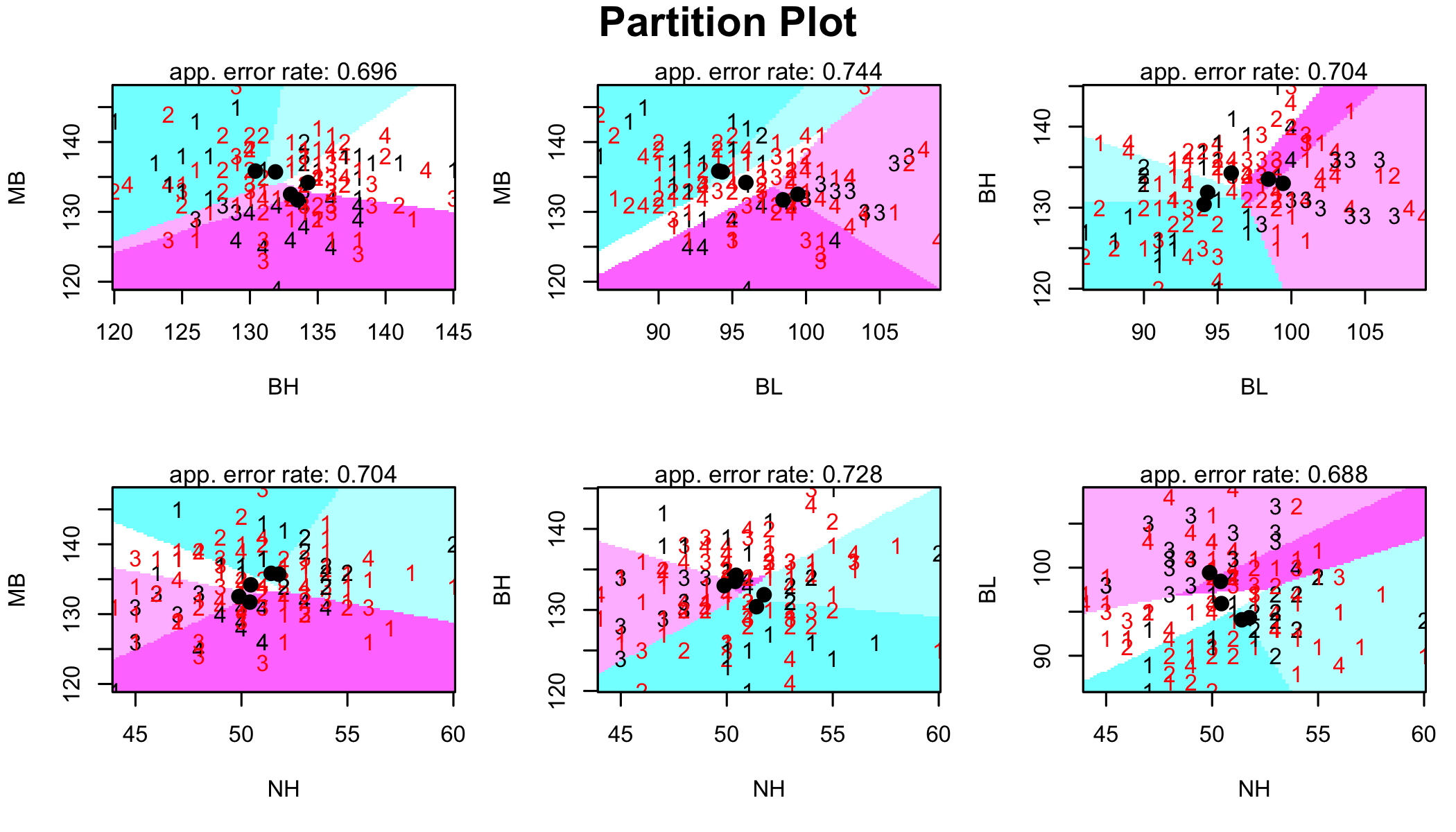
egyptskull\_test$lda\_predict<-predict(model\_lda,egyptskull\_test[,2:5])$class  
table(egyptskull\_test$Epoch,egyptskull\_test$lda\_predict)

##   
## 150 200 1850 3300 4000  
## 150 3 0 1 1 0  
## 200 1 1 2 1 0  
## 1850 1 1 2 0 1  
## 3300 0 3 0 2 0  
## 4000 0 2 0 2 1

mean(egyptskull\_test$lda\_predict != egyptskull\_test$Epoch)

## [1] 0.64

partimat(as.factor(Epoch) ~ MB+BH+BL+NH, data=egyptskull\_train,method="lda")



Perform QDA

######## QDA  
  
model\_qda<-qda(Epoch ~ MB+BH+BL+NH, data=egyptskull\_train)  
model\_qda

## Call:  
## qda(Epoch ~ MB + BH + BL + NH, data = egyptskull\_train)  
##   
## Prior probabilities of groups:  
## 150 200 1850 3300 4000   
## 0.2 0.2 0.2 0.2 0.2   
##   
## Group means:  
## MB BH BL NH  
## 150 135.84 130.40 94.08 51.40  
## 200 135.72 131.88 94.32 51.76  
## 1850 134.20 134.28 95.92 50.44  
## 3300 132.52 133.00 99.44 49.88  
## 4000 131.72 133.52 98.44 50.40

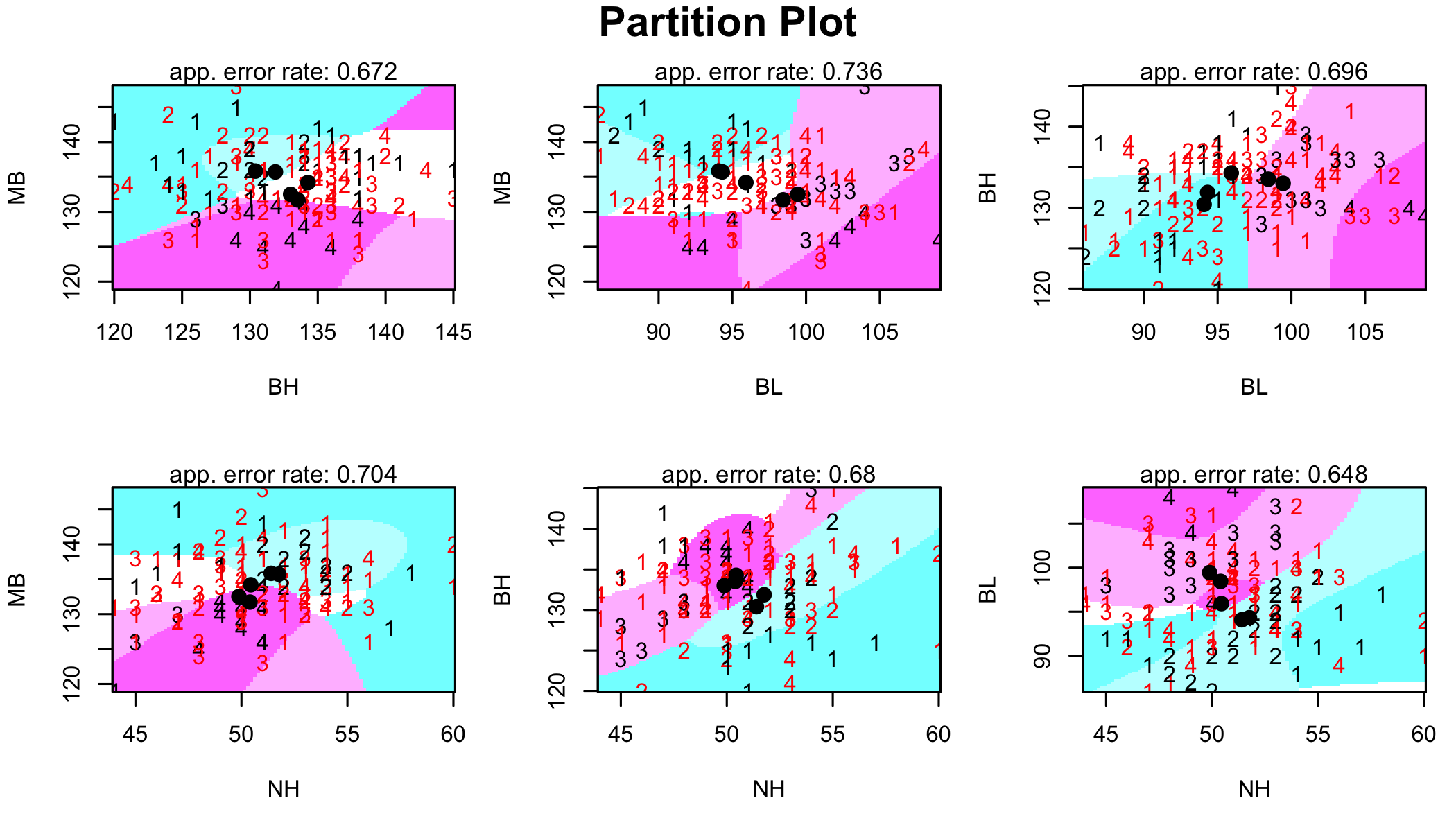
egyptskull\_test$qda\_predict<-predict(model\_qda,egyptskull\_test[,2:5])$class  
table(egyptskull\_test$Epoch,egyptskull\_test$qda\_predict)

##   
## 150 200 1850 3300 4000  
## 150 2 1 0 1 1  
## 200 1 1 1 1 1  
## 1850 1 2 1 1 0  
## 3300 0 1 0 0 4  
## 4000 0 1 0 1 3

mean(egyptskull\_test$qda\_predict != egyptskull\_test$Epoch)

## [1] 0.72

partimat(as.factor(Epoch) ~ MB+BH+BL+NH, data=egyptskull\_train,method="qda")



Perform Multinomial Logistic

###### Multinomial Logistic   
  
model\_mnl<-vglm(formula = cbind(Epoch\_1,Epoch\_2,Epoch\_3,Epoch\_4,Epoch\_5) ~ MB+BH+BL+NH, family = multinomial, data = egyptskull\_train)  
summary(model\_mnl)

##   
## Call:  
## vglm(formula = cbind(Epoch\_1, Epoch\_2, Epoch\_3, Epoch\_4, Epoch\_5) ~   
## MB + BH + BL + NH, family = multinomial, data = egyptskull\_train)  
##   
## Pearson residuals:  
## Min 1Q Median 3Q Max  
## log(mu[,1]/mu[,5]) -3.318 -0.4637 -0.2831 -0.09212 5.238  
## log(mu[,2]/mu[,5]) -3.112 -0.4613 -0.2458 -0.07941 3.680  
## log(mu[,3]/mu[,5]) -2.784 -0.4446 -0.2796 -0.14198 5.502  
## log(mu[,4]/mu[,5]) -2.243 -0.4046 -0.2508 -0.10041 3.086  
##   
## Coefficients:   
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept):1 -1.325060 13.283864 -0.100 0.92054   
## (Intercept):2 -6.528734 13.415429 -0.487 0.62650   
## (Intercept):3 -10.099703 13.091198 NA NA   
## (Intercept):4 -6.996331 12.680035 -0.552 0.58111   
## MB:1 -0.184026 0.071829 NA NA   
## MB:2 -0.135660 0.070781 NA NA   
## MB:3 -0.065723 0.067484 NA NA   
## MB:4 -0.003269 0.063397 -0.052 0.95888   
## BH:1 0.101965 0.067863 1.502 0.13297   
## BH:2 0.081905 0.068726 1.192 0.23336   
## BH:3 0.161953 0.065986 2.454 0.01411 \*   
## BH:4 0.055233 0.058324 0.947 0.34364   
## BL:1 0.185242 0.072970 2.539 0.01113 \*   
## BL:2 0.238083 0.073697 3.231 0.00124 \*\*  
## BL:3 0.052184 0.070169 0.744 0.45706   
## BL:4 -0.010218 0.068163 -0.150 0.88084   
## NH:1 -0.105379 0.103191 -1.021 0.30715   
## NH:2 -0.179985 0.107504 NA NA   
## NH:3 -0.146265 0.098590 NA NA   
## NH:4 0.022474 0.087339 0.257 0.79693   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Names of linear predictors: log(mu[,1]/mu[,5]), log(mu[,2]/mu[,5]),   
## log(mu[,3]/mu[,5]), log(mu[,4]/mu[,5])  
##   
## Residual deviance: 352.3384 on 480 degrees of freedom  
##   
## Log-likelihood: -176.1692 on 480 degrees of freedom  
##   
## Number of Fisher scoring iterations: 5   
##   
## Warning: Hauck-Donner effect detected in the following estimate(s):  
## '(Intercept):3', 'MB:1', 'MB:2', 'MB:3', 'NH:2', 'NH:3'  
##   
##   
## Reference group is level 5 of the response

predictions<-predict(model\_mnl,newdata=egyptskull\_test[,2:5],type="response")  
egyptskull\_test$pred\_mnl<-apply(predictions,1,function(i) which.max(i) )  
  
egyptskull\_test[, pred\_mnl:= c(4000, 3300, 1850, 200, 150)[pred\_mnl]]  
egyptskull\_test[, unique(Epoch)]

## [1] 4000 3300 1850 200 150   
## Levels: 150 200 1850 3300 4000

print(table(egyptskull\_test$Epoch,egyptskull\_test$pred\_mnl))

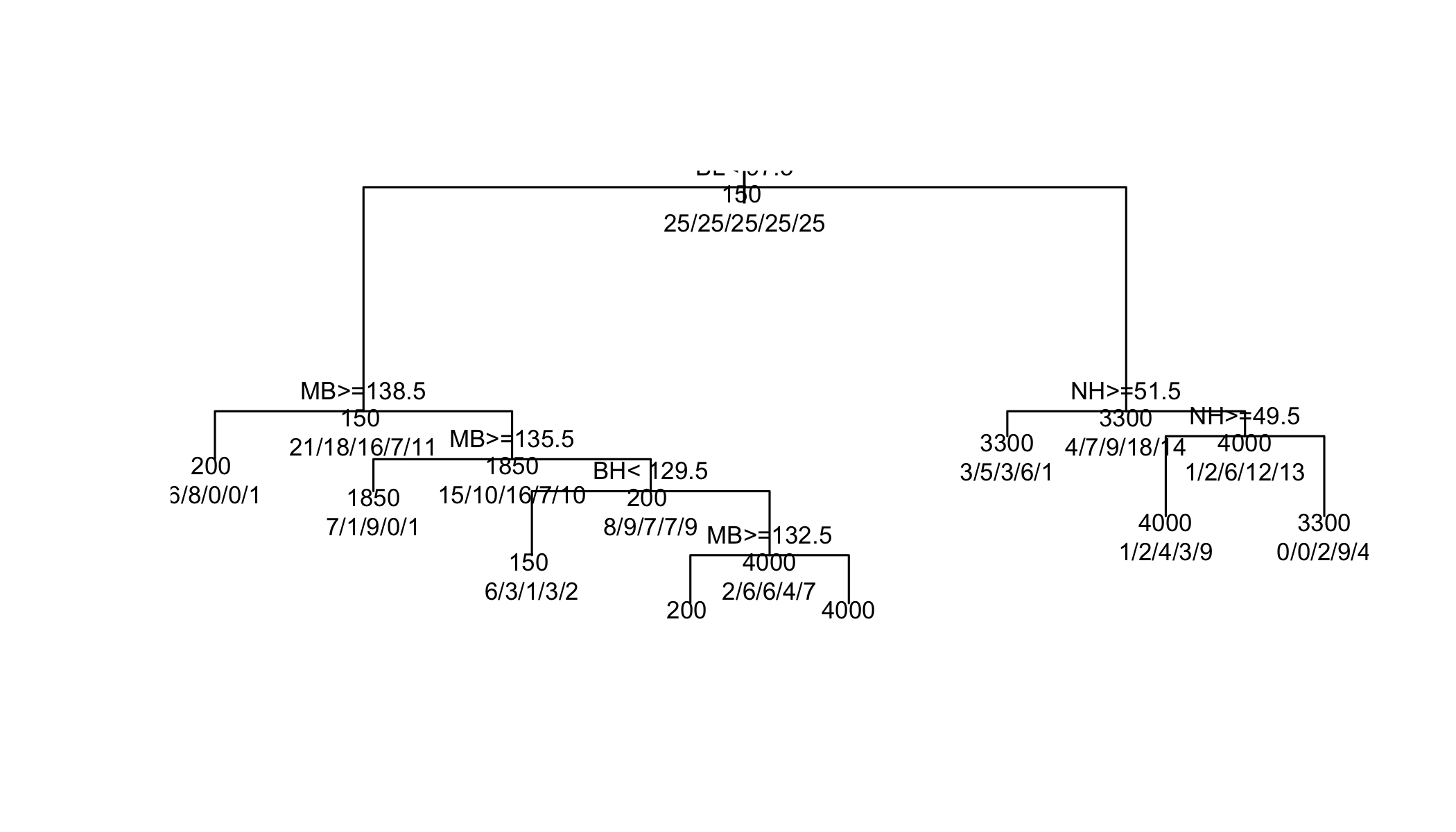
##   
## 150 200 1850 3300 4000  
## 150 3 0 1 1 0  
## 200 1 1 2 1 0  
## 1850 1 1 2 0 1  
## 3300 0 3 0 2 0  
## 4000 0 1 0 2 2

mean(egyptskull\_test$pred\_mnl != egyptskull\_test$Epoch)

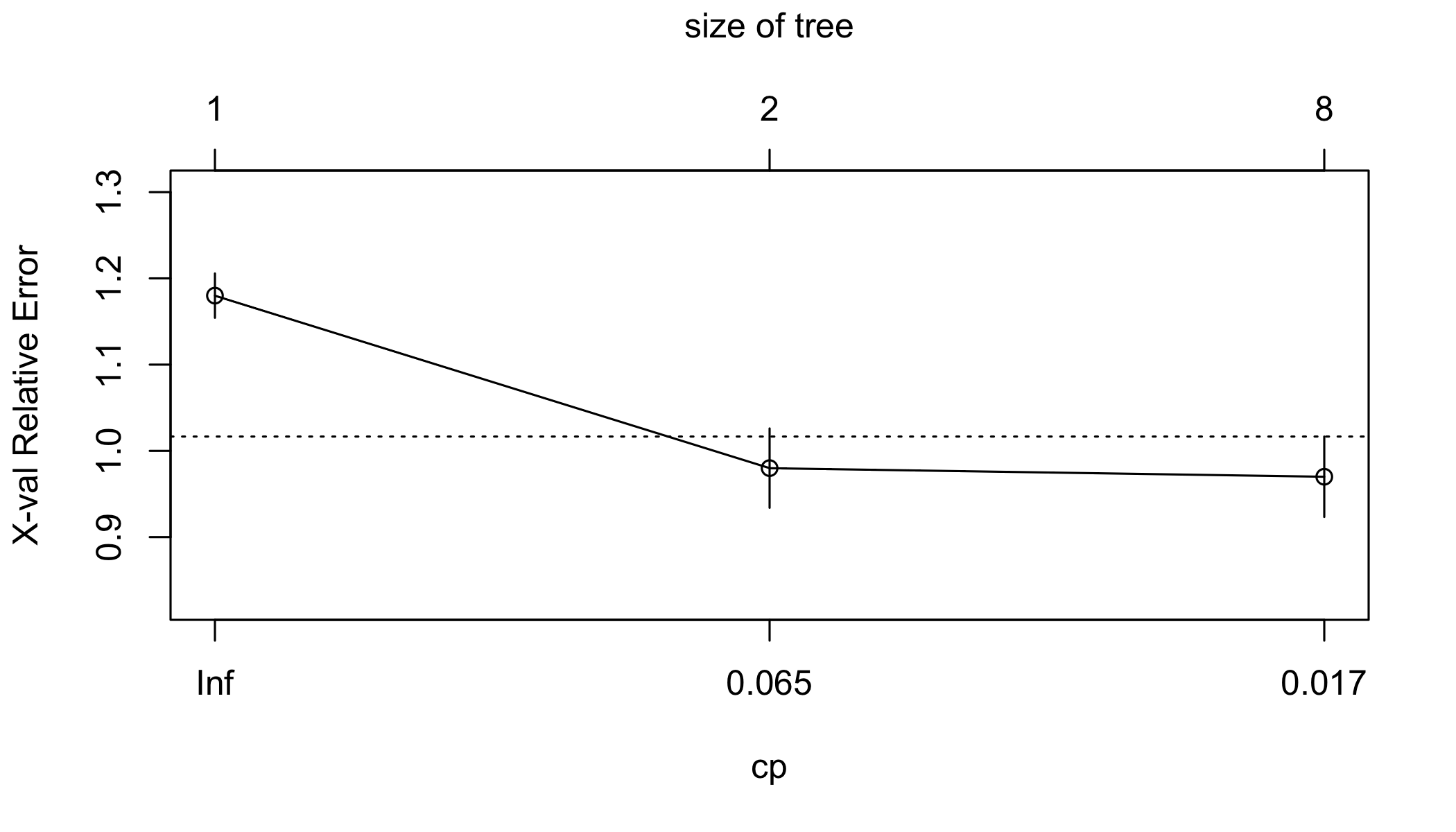
## [1] 0.6

Perform CART

######## CART  
  
model\_ct <- rpart(Epoch ~ MB+BH+BL+NH, data = egyptskull\_train, method="class")  
plot(model\_ct)  
text(model\_ct, use.n=TRUE, all=TRUE, cex=.7)



plotcp(model\_ct)



egyptskull\_test$pred\_ct<-predict(model\_ct,egyptskull\_test,type="vector")  
egyptskull\_test[, pred\_ct:= c(4000, 3300, 1850, 200, 150)[pred\_ct]]  
table(egyptskull\_test$Epoch,egyptskull\_test$pred\_ct)

##   
## 150 200 1850 3300 4000  
## 150 0 1 2 1 1  
## 200 3 0 2 0 0  
## 1850 0 2 1 1 1  
## 3300 4 1 0 0 0  
## 4000 1 4 0 0 0

mean(egyptskull\_test$pred\_ct != egyptskull\_test$Epoch)

## [1] 0.96

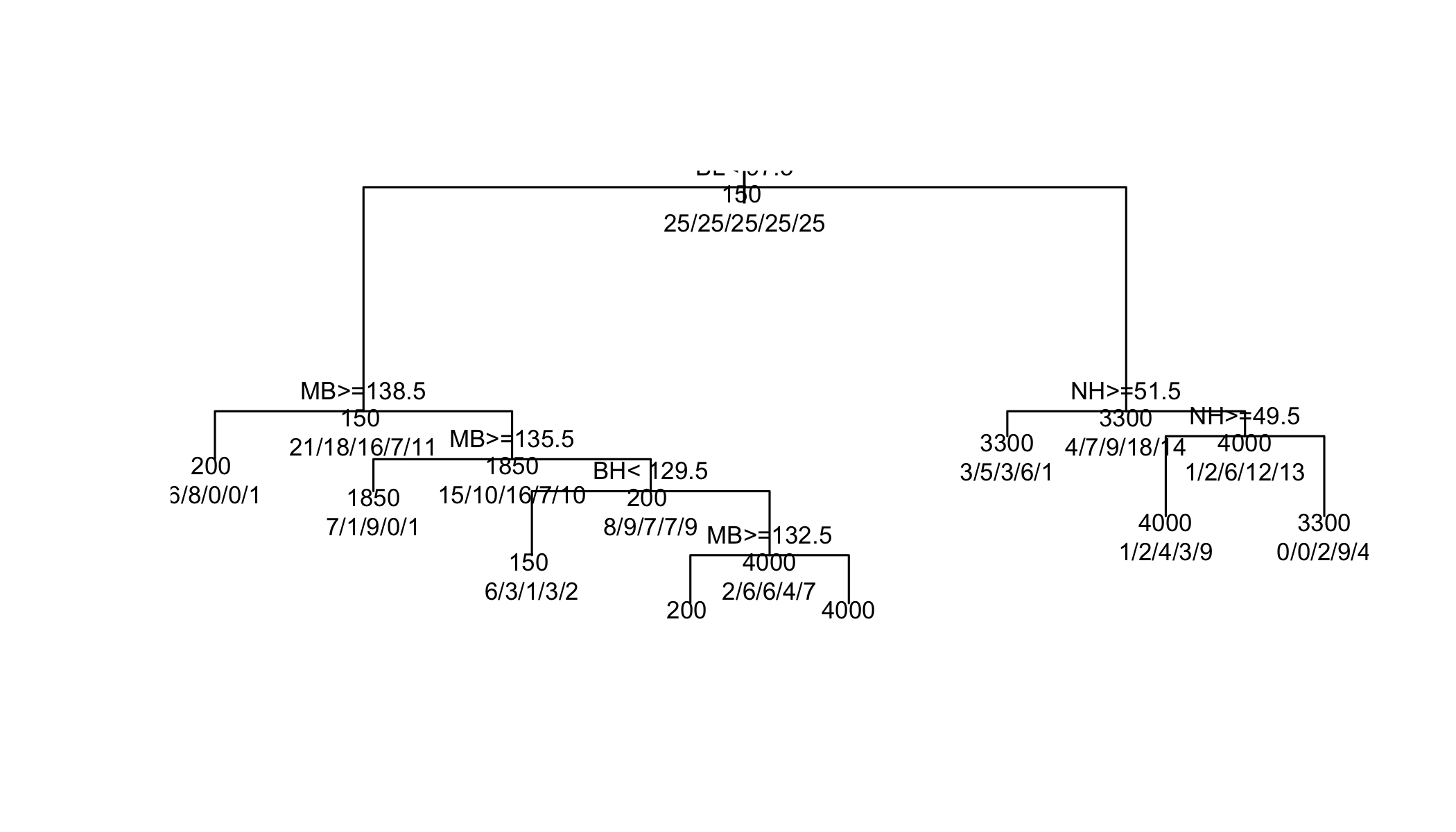
model\_ct$cptable

## CP nsplit rel error xerror xstd  
## 1 0.14 0 1.00 1.18 0.02570603  
## 2 0.03 1 0.86 0.98 0.04600869  
## 3 0.01 7 0.68 0.97 0.04661330

model\_ct\_fit<- prune(model\_ct, cp=model\_ct$cptable[which.min(model\_ct$cptable[,"xerror"]),"CP"])  
  
summary(model\_ct\_fit)

## Call:  
## rpart(formula = Epoch ~ MB + BH + BL + NH, data = egyptskull\_train,   
## method = "class")  
## n= 125   
##   
## CP nsplit rel error xerror xstd  
## 1 0.14 0 1.00 1.18 0.02570603  
## 2 0.03 1 0.86 0.98 0.04600869  
## 3 0.01 7 0.68 0.97 0.04661330  
##   
## Variable importance  
## MB BL NH BH   
## 38 24 23 15   
##   
## Node number 1: 125 observations, complexity param=0.14  
## predicted class=150 expected loss=0.8 P(node) =1  
## class counts: 25 25 25 25 25  
## probabilities: 0.200 0.200 0.200 0.200 0.200   
## left son=2 (73 obs) right son=3 (52 obs)  
## Primary splits:  
## BL < 97.5 to the left, improve=4.122761, (0 missing)  
## MB < 135.5 to the right, improve=2.933333, (0 missing)  
## BH < 127.5 to the left, improve=2.449634, (0 missing)  
## NH < 51.5 to the right, improve=1.866667, (0 missing)  
## Surrogate splits:  
## BH < 138.5 to the left, agree=0.640, adj=0.135, (0 split)  
## MB < 124.5 to the right, agree=0.592, adj=0.019, (0 split)  
##   
## Node number 2: 73 observations, complexity param=0.03  
## predicted class=150 expected loss=0.7123288 P(node) =0.584  
## class counts: 21 18 16 7 11  
## probabilities: 0.288 0.247 0.219 0.096 0.151   
## left son=4 (15 obs) right son=5 (58 obs)  
## Primary splits:  
## MB < 138.5 to the right, improve=3.0044720, (0 missing)  
## BH < 129.5 to the left, improve=2.4397140, (0 missing)  
## BL < 90.5 to the left, improve=1.2458510, (0 missing)  
## NH < 53.5 to the right, improve=0.8067706, (0 missing)  
##   
## Node number 3: 52 observations, complexity param=0.03  
## predicted class=3300 expected loss=0.6538462 P(node) =0.416  
## class counts: 4 7 9 18 14  
## probabilities: 0.077 0.135 0.173 0.346 0.269   
## left son=6 (18 obs) right son=7 (34 obs)  
## Primary splits:  
## NH < 51.5 to the right, improve=2.0485170, (0 missing)  
## BL < 98.5 to the left, improve=1.7383390, (0 missing)  
## BH < 129.5 to the left, improve=0.8667263, (0 missing)  
## MB < 134.5 to the right, improve=0.7785146, (0 missing)  
## Surrogate splits:  
## BH < 135.5 to the right, agree=0.731, adj=0.222, (0 split)  
##   
## Node number 4: 15 observations  
## predicted class=200 expected loss=0.4666667 P(node) =0.12  
## class counts: 6 8 0 0 1  
## probabilities: 0.400 0.533 0.000 0.000 0.067   
##   
## Node number 5: 58 observations, complexity param=0.03  
## predicted class=1850 expected loss=0.7241379 P(node) =0.464  
## class counts: 15 10 16 7 10  
## probabilities: 0.259 0.172 0.276 0.121 0.172   
## left son=10 (18 obs) right son=11 (40 obs)  
## Primary splits:  
## MB < 135.5 to the right, improve=2.8471260, (0 missing)  
## BH < 129.5 to the left, improve=2.3860150, (0 missing)  
## NH < 54.5 to the right, improve=1.1392830, (0 missing)  
## BL < 90.5 to the right, improve=0.7244507, (0 missing)  
## Surrogate splits:  
## BH < 136.5 to the right, agree=0.724, adj=0.111, (0 split)  
##   
## Node number 6: 18 observations  
## predicted class=3300 expected loss=0.6666667 P(node) =0.144  
## class counts: 3 5 3 6 1  
## probabilities: 0.167 0.278 0.167 0.333 0.056   
##   
## Node number 7: 34 observations, complexity param=0.03  
## predicted class=4000 expected loss=0.6176471 P(node) =0.272  
## class counts: 1 2 6 12 13  
## probabilities: 0.029 0.059 0.176 0.353 0.382   
## left son=14 (19 obs) right son=15 (15 obs)  
## Primary splits:  
## NH < 49.5 to the right, improve=2.1636740, (0 missing)  
## BL < 101.5 to the left, improve=1.0882350, (0 missing)  
## BH < 129.5 to the left, improve=0.8051665, (0 missing)  
## MB < 136.5 to the right, improve=0.5690045, (0 missing)  
## Surrogate splits:  
## BH < 130.5 to the right, agree=0.618, adj=0.133, (0 split)  
## BL < 103.5 to the left, agree=0.618, adj=0.133, (0 split)  
## MB < 130.5 to the right, agree=0.588, adj=0.067, (0 split)  
##   
## Node number 10: 18 observations  
## predicted class=1850 expected loss=0.5 P(node) =0.144  
## class counts: 7 1 9 0 1  
## probabilities: 0.389 0.056 0.500 0.000 0.056   
##   
## Node number 11: 40 observations, complexity param=0.03  
## predicted class=200 expected loss=0.775 P(node) =0.32  
## class counts: 8 9 7 7 9  
## probabilities: 0.200 0.225 0.175 0.175 0.225   
## left son=22 (15 obs) right son=23 (25 obs)  
## Primary splits:  
## BH < 129.5 to the left, improve=1.473333, (0 missing)  
## BL < 92.5 to the left, improve=1.250384, (0 missing)  
## MB < 128.5 to the right, improve=0.861039, (0 missing)  
## NH < 53.5 to the right, improve=0.587500, (0 missing)  
## Surrogate splits:  
## NH < 46.5 to the left, agree=0.70, adj=0.200, (0 split)  
## BL < 91.5 to the left, agree=0.65, adj=0.067, (0 split)  
##   
## Node number 14: 19 observations  
## predicted class=4000 expected loss=0.5263158 P(node) =0.152  
## class counts: 1 2 4 3 9  
## probabilities: 0.053 0.105 0.211 0.158 0.474   
##   
## Node number 15: 15 observations  
## predicted class=3300 expected loss=0.4 P(node) =0.12  
## class counts: 0 0 2 9 4  
## probabilities: 0.000 0.000 0.133 0.600 0.267   
##   
## Node number 22: 15 observations  
## predicted class=150 expected loss=0.6 P(node) =0.12  
## class counts: 6 3 1 3 2  
## probabilities: 0.400 0.200 0.067 0.200 0.133   
##   
## Node number 23: 25 observations, complexity param=0.03  
## predicted class=4000 expected loss=0.72 P(node) =0.2  
## class counts: 2 6 6 4 7  
## probabilities: 0.080 0.240 0.240 0.160 0.280   
## left son=46 (8 obs) right son=47 (17 obs)  
## Primary splits:  
## MB < 132.5 to the right, improve=1.8600000, (0 missing)  
## BL < 95.5 to the left, improve=0.9473016, (0 missing)  
## NH < 49.5 to the right, improve=0.8600000, (0 missing)  
## BH < 133.5 to the right, improve=0.3917460, (0 missing)  
## Surrogate splits:  
## BL < 96.5 to the right, agree=0.76, adj=0.250, (0 split)  
## NH < 53.5 to the right, agree=0.72, adj=0.125, (0 split)  
##   
## Node number 46: 8 observations  
## predicted class=200 expected loss=0.625 P(node) =0.064  
## class counts: 1 3 1 3 0  
## probabilities: 0.125 0.375 0.125 0.375 0.000   
##   
## Node number 47: 17 observations  
## predicted class=4000 expected loss=0.5882353 P(node) =0.136  
## class counts: 1 3 5 1 7  
## probabilities: 0.059 0.176 0.294 0.059 0.412

plot(model\_ct\_fit)  
text(model\_ct\_fit, use.n=TRUE, all=TRUE, cex=.7)



egyptskull\_test$pred\_ct\_fit<-predict(model\_ct\_fit,egyptskull\_test,type="vector")  
egyptskull\_test[, pred\_ct\_fit:= c(4000, 3300, 1850, 200, 150)[pred\_ct\_fit]]  
table(egyptskull\_test$Epoch,egyptskull\_test$pred\_ct\_fit)

##   
## 150 200 1850 3300 4000  
## 150 0 1 2 1 1  
## 200 3 0 2 0 0  
## 1850 0 2 1 1 1  
## 3300 4 1 0 0 0  
## 4000 1 4 0 0 0

mean(egyptskull\_test$pred\_ct\_fit != egyptskull\_test$Epoch)

## [1] 0.96

Build a neural network

##### Nnet  
  
model\_nnet<-nnet(Epoch ~ MB+BH+BL+NH, data = egyptskull\_train,size=5,decay=0.1)

## # weights: 55  
## initial value 213.681209   
## iter 10 value 200.785758  
## iter 20 value 190.364591  
## iter 30 value 185.053526  
## iter 40 value 184.349741  
## iter 50 value 182.130107  
## iter 60 value 174.984570  
## iter 70 value 169.690900  
## iter 80 value 167.716483  
## iter 90 value 166.743568  
## iter 100 value 166.479654  
## final value 166.479654   
## stopped after 100 iterations

egyptskull\_test$pred\_nnet<-predict(model\_nnet,egyptskull\_test,type="class")  
table(egyptskull\_test$Epoch,egyptskull\_test$pred\_nnet)

##   
## 150 1850 200 3300 4000  
## 150 1 0 2 2 0  
## 200 1 2 1 1 0  
## 1850 2 1 0 2 0  
## 3300 1 0 2 2 0  
## 4000 1 1 0 1 2

mean(egyptskull\_test$pred\_nnet != egyptskull\_test$Epoch)

## [1] 0.72

### Question 3e

# LDA  
table(egyptskull\_test$Epoch,egyptskull\_test$lda\_predict)

##   
## 150 200 1850 3300 4000  
## 150 3 0 1 1 0  
## 200 1 1 2 1 0  
## 1850 1 1 2 0 1  
## 3300 0 3 0 2 0  
## 4000 0 2 0 2 1

mean(egyptskull\_test$lda\_predict != egyptskull\_test$Epoch)

## [1] 0.64

# QDA  
table(egyptskull\_test$Epoch,egyptskull\_test$qda\_predict)

##   
## 150 200 1850 3300 4000  
## 150 2 1 0 1 1  
## 200 1 1 1 1 1  
## 1850 1 2 1 1 0  
## 3300 0 1 0 0 4  
## 4000 0 1 0 1 3

mean(egyptskull\_test$qda\_predict != egyptskull\_test$Epoch)

## [1] 0.72

# Multinomial  
print(table(egyptskull\_test$Epoch,egyptskull\_test$pred\_mnl))

##   
## 150 200 1850 3300 4000  
## 150 3 0 1 1 0  
## 200 1 1 2 1 0  
## 1850 1 1 2 0 1  
## 3300 0 3 0 2 0  
## 4000 0 1 0 2 2

mean(egyptskull\_test$pred\_mnl != egyptskull\_test$Epoch)

## [1] 0.6

# CART  
table(egyptskull\_test$Epoch,egyptskull\_test$pred\_ct\_fit)

##   
## 150 200 1850 3300 4000  
## 150 0 1 2 1 1  
## 200 3 0 2 0 0  
## 1850 0 2 1 1 1  
## 3300 4 1 0 0 0  
## 4000 1 4 0 0 0

mean(egyptskull\_test$pred\_ct\_fit != egyptskull\_test$Epoch)

## [1] 0.96

# Nnet  
table(egyptskull\_test$Epoch,egyptskull\_test$pred\_nnet)

##   
## 150 1850 200 3300 4000  
## 150 1 0 2 2 0  
## 200 1 2 1 1 0  
## 1850 2 1 0 2 0  
## 3300 1 0 2 2 0  
## 4000 1 1 0 1 2

mean(egyptskull\_test$pred\_nnet != egyptskull\_test$Epoch)

## [1] 0.72

### Question 3f

####### Predict  
egyptskull\_val <- data.table(rbind(c(128, 143, 103, 50)   
 , c(129, 126, 91, 50)  
 , c(130, 127, 99, 45)  
 , c(130, 131, 98, 53)  
 , c(134, 124, 91, 55)  
 , c(130, 130, 104, 49)  
 , c(134, 139, 101, 49)  
 , c(136, 133, 91, 49)  
 ))  
  
names(egyptskull\_val) <- names(egyptskull)[1:4]  
  
# Use multinomial  
  
predictions<-predict(model\_mnl,newdata=egyptskull\_val,type="response")  
  
egyptskull\_val$pred\_mnl<-apply(predictions,1,function(i) which.max(i) )  
  
egyptskull\_val[, pred\_mnl:= c(4000, 3300, 1850, 200, 150)[pred\_mnl]]

## Part B

## Question 1

### Load the dataset from web

#url <- 'https://web.stanford.edu/~hastie/Papers/LARS/diabetes.data'  
#diabetes\_orig <- fread(url, sep = '\t')  
  
#fwrite(diabetes\_orig, 'Data/diabetes.csv')  
  
#   
# data(diabetes)  
# Xmatrix <- diabetes$x  
# yVector <- diabetes$y  
#   
  
diabetes\_orig <- fread('Data/diabetes.csv')  
dim(diabetes\_orig)

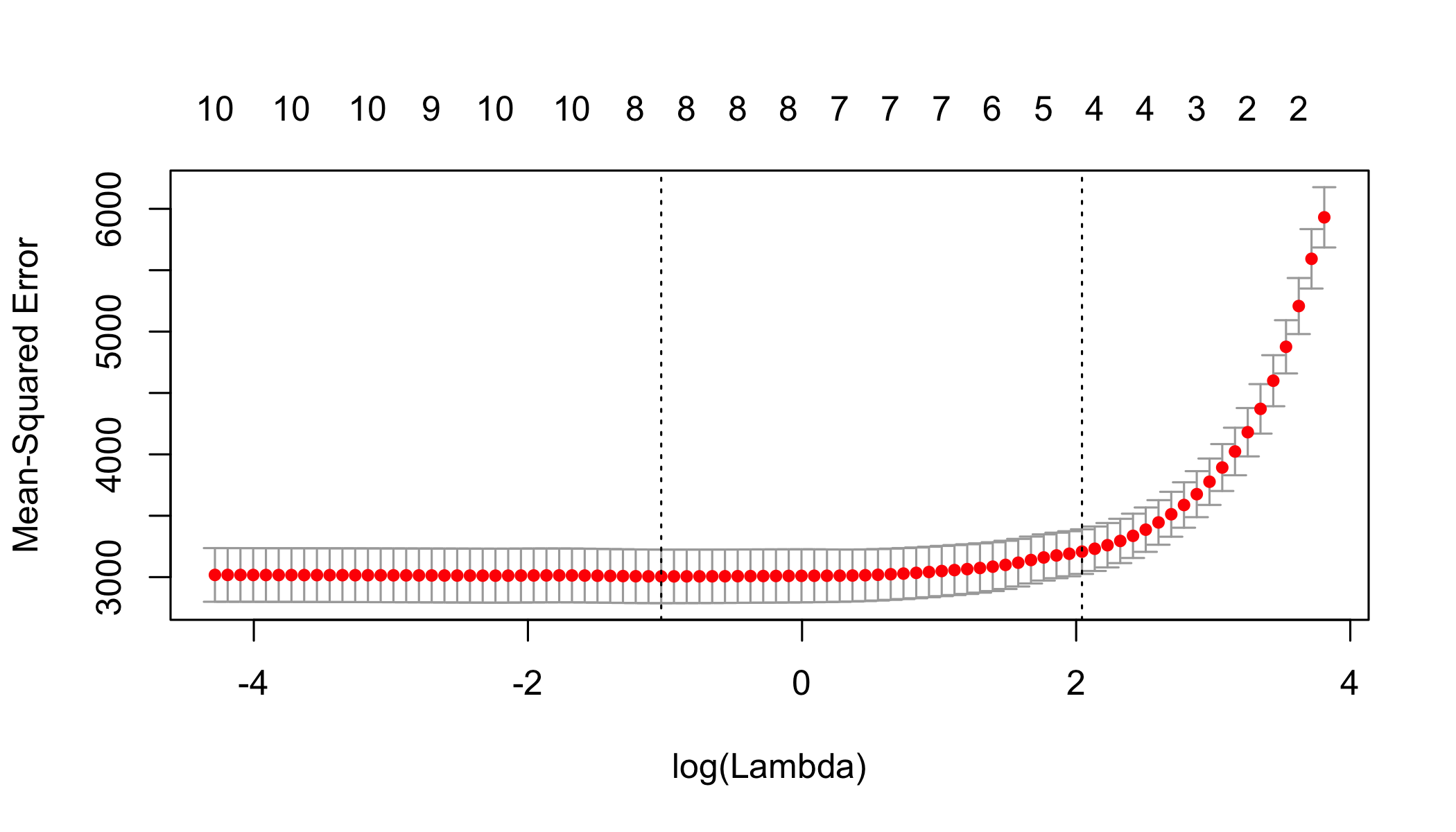
## [1] 442 11

### Question 1a

Xmatrix <- as.matrix(diabetes\_orig[,1:10])  
yVector <- diabetes\_orig$Y  
  
dim(Xmatrix)

## [1] 442 10

cvfit <- cv.glmnet(Xmatrix , yVector)  
  
plot(cvfit)



log(cvfit$lambda.min)

## [1] -1.027542

coef(cvfit, s = "lambda.min")

## 11 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -250.9863338  
## AGE .   
## SEX -21.1816522  
## BMI 5.6696782  
## BP 1.0736394  
## S1 -0.2473766  
## S2 .   
## S3 -0.6099868  
## S4 3.2497703  
## S5 48.0714331  
## S6 0.2616137

log(cvfit$lambda.1se)

## [1] 2.042571

coef(cvfit, s = "lambda.1se")

## 11 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -208.1840571  
## AGE .   
## SEX .   
## BMI 5.3182206  
## BP 0.5921749  
## S1 .   
## S2 .   
## S3 -0.3478550  
## S4 .   
## S5 39.0650267  
## S6 .

LASSOfit <- glmnet(Xmatrix , yVector)  
summary(LASSOfit)

## Length Class Mode   
## a0 88 -none- numeric  
## beta 880 dgCMatrix S4   
## df 88 -none- numeric  
## dim 2 -none- numeric  
## lambda 88 -none- numeric  
## dev.ratio 88 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 3 -none- call   
## nobs 1 -none- numeric

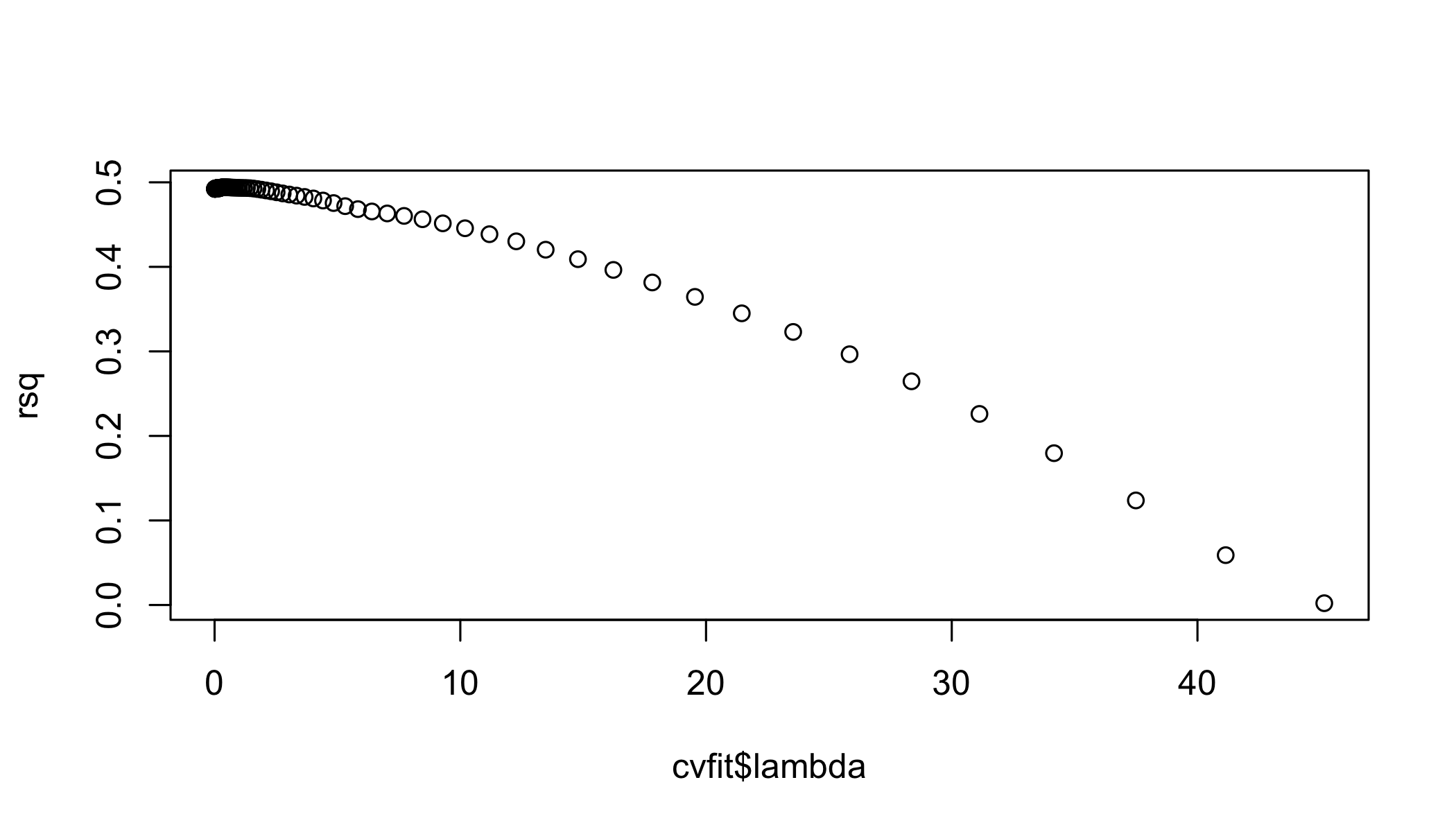
LASSOfit$lambda

## [1] 45.16003002 41.14813742 37.49265031 34.16190659 31.12705697  
## [6] 28.36181502 25.84222954 23.54647710 21.45467297 19.54869896  
## [11] 17.81204642 16.22967331 14.78787387 13.47415991 12.27715268  
## [16] 11.18648427 10.19270784 9.28721577 8.46216512 7.71040969  
## [21] 7.02543815 6.40131759 5.83264218 5.31448632 4.84236200  
## [26] 4.41217991 4.02021401 3.66306928 3.33765230 3.04114447  
## [31] 2.77097757 2.52481156 2.30051426 2.09614292 1.90992736  
## [36] 1.74025468 1.58565525 1.44479000 1.31643884 1.19949004  
## [41] 1.09293065 0.99583771 0.90737023 0.82676196 0.75331471  
## [46] 0.68639230 0.62541510 0.56985495 0.51923061 0.47310359  
## [51] 0.43107437 0.39277891 0.35788552 0.32609195 0.29712284  
## [56] 0.27072727 0.24667660 0.22476253 0.20479525 0.18660180  
## [61] 0.17002461 0.15492010 0.14115742 0.12861739 0.11719137  
## [66] 0.10678041 0.09729434 0.08865098 0.08077547 0.07359960  
## [71] 0.06706121 0.06110368 0.05567540 0.05072935 0.04622269  
## [76] 0.04211640 0.03837489 0.03496577 0.03185951 0.02902920  
## [81] 0.02645032 0.02410055 0.02195952 0.02000870 0.01823118  
## [86] 0.01661157 0.01513585 0.01379122

betaHat <- as.numeric(LASSOfit$beta)  
  
LASSOfit <- glmnet(Xmatrix , yVector , lambda=1.2)  
summary(LASSOfit)

## Length Class Mode   
## a0 1 -none- numeric  
## beta 10 dgCMatrix S4   
## df 1 -none- numeric  
## dim 2 -none- numeric  
## lambda 1 -none- numeric  
## dev.ratio 1 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 4 -none- call   
## nobs 1 -none- numeric

betaHat <- as.numeric(LASSOfit$beta)  
  
rsq <- 1 - cvfit$cvm/var(yVector)  
  
plot(cvfit$lambda,rsq)



lambda\_rsq <- data.table(cbind(lambda=cvfit$lambda,rsq))  
  
lambda\_rsq[rsq==max(rsq)]

## lambda rsq  
## 1: 0.3578855 0.4942198

LASSOfit <- glmnet(Xmatrix , yVector , lambda=1)  
betaHat <- as.numeric(LASSOfit$beta)

r-squared 49.93 better fit similar co-oefficents automated way of finding co-efficents

## Question 2

### Load the dataset

Seeds dataset contains data about four varities wheat capturing the hedonic characteristics of each variety of wheat.

seeds <- fread('Data/seeds\_dataset.txt', sep = "\t")

## Warning in fread("Data/seeds\_dataset.txt", sep = "\t"): Stopped early  
## on line 8. Expected 8 fields but found 10. Consider fill=TRUE and  
## comment.char=. First discarded non-empty line: <<14.11 14.1 0.8911 5.42  
## 3.302 2.7 5 1>>

col\_names\_seeds <- c("area", "perimeter", "compactness", "length\_of\_kernel", "width\_of\_kernel", "asymmetry\_coefficient", "length\_of\_kernel\_groove", "wheat\_type")  
names(seeds) <- col\_names\_seeds

### Question 2a

Estimate of covariance matrix using sample covariance method:

seeds\_vcmat <- cov(seeds[,1:7])

### Question 2b

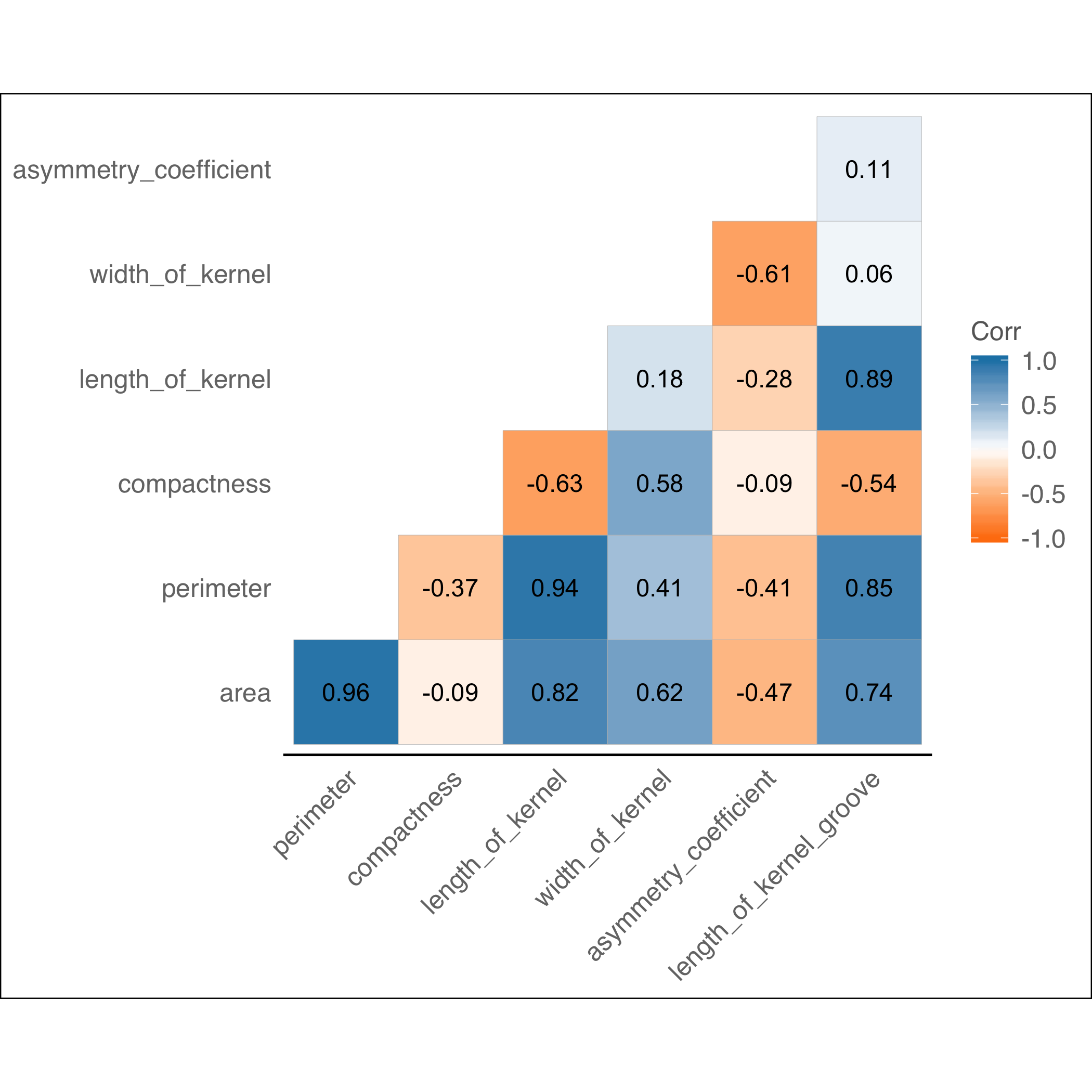
Another method is maximum likelihood estimate:

Not very different

### Question 2c

### Multicollinearity Analysis

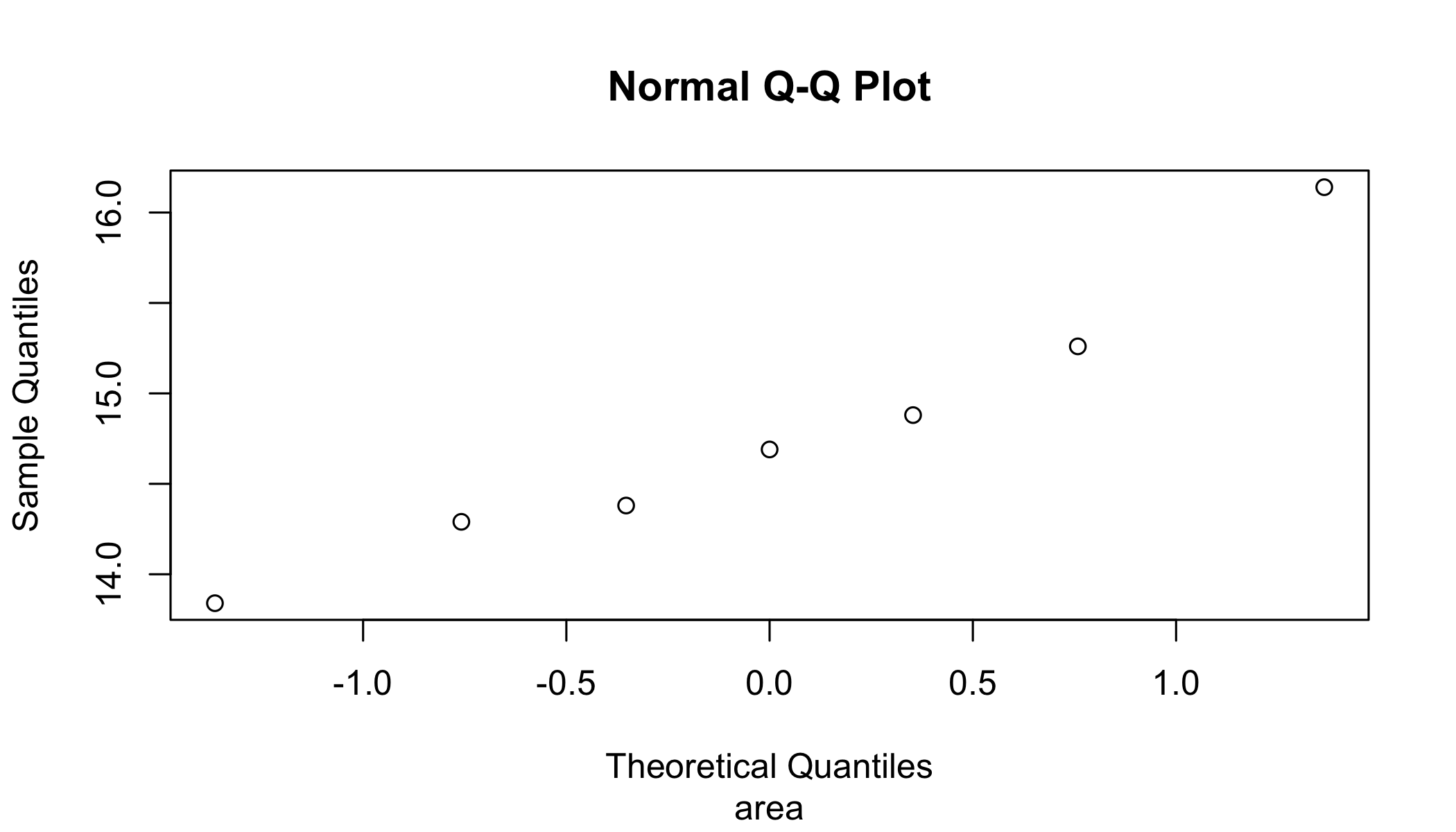
#Corr Plot  
cor\_mat <- seeds[,1:7]  
corr <- cor(cor\_mat, use = "pairwise.complete.obs")  
  
ggcorrplot(corr, hc.order = FALSE, type = "lower",  
 ggtheme = ggthemes::theme\_gdocs,  
 colors = c("#ff7f0e", "white", "#1f83b4"),  
 lab = TRUE)+  
 theme(panel.grid.major=element\_blank())

 High linear relationship among variables

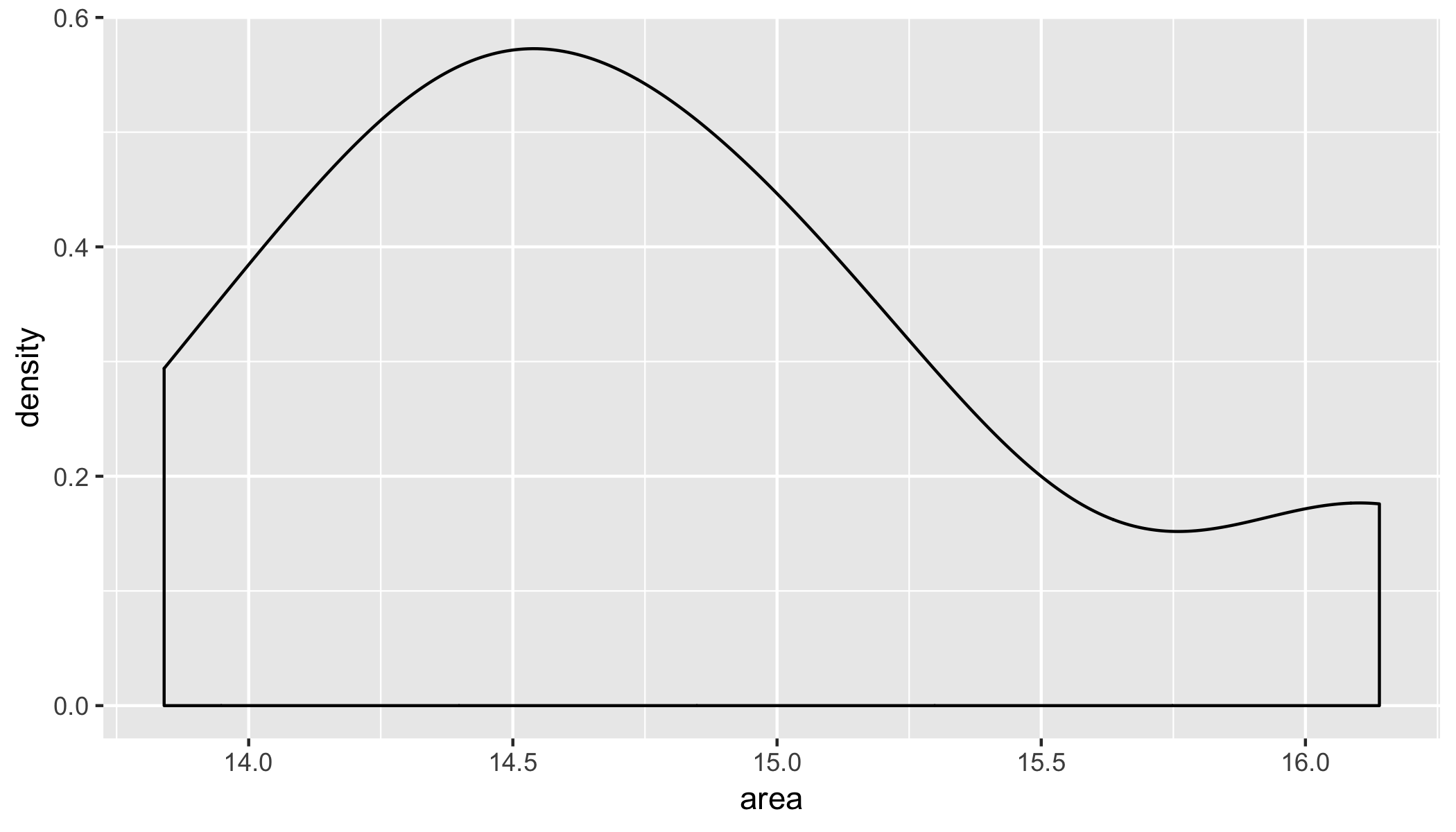
### Univariate Analysis

Shapiro Wilk test rejects the null hypothesis of sample seeds$area being univariate normal.

# Normal Q-Q plot for area  
qqnorm(seeds$area, sub = colnames(seeds)[1])



ggplot(seeds, aes(x=area)) + geom\_density()

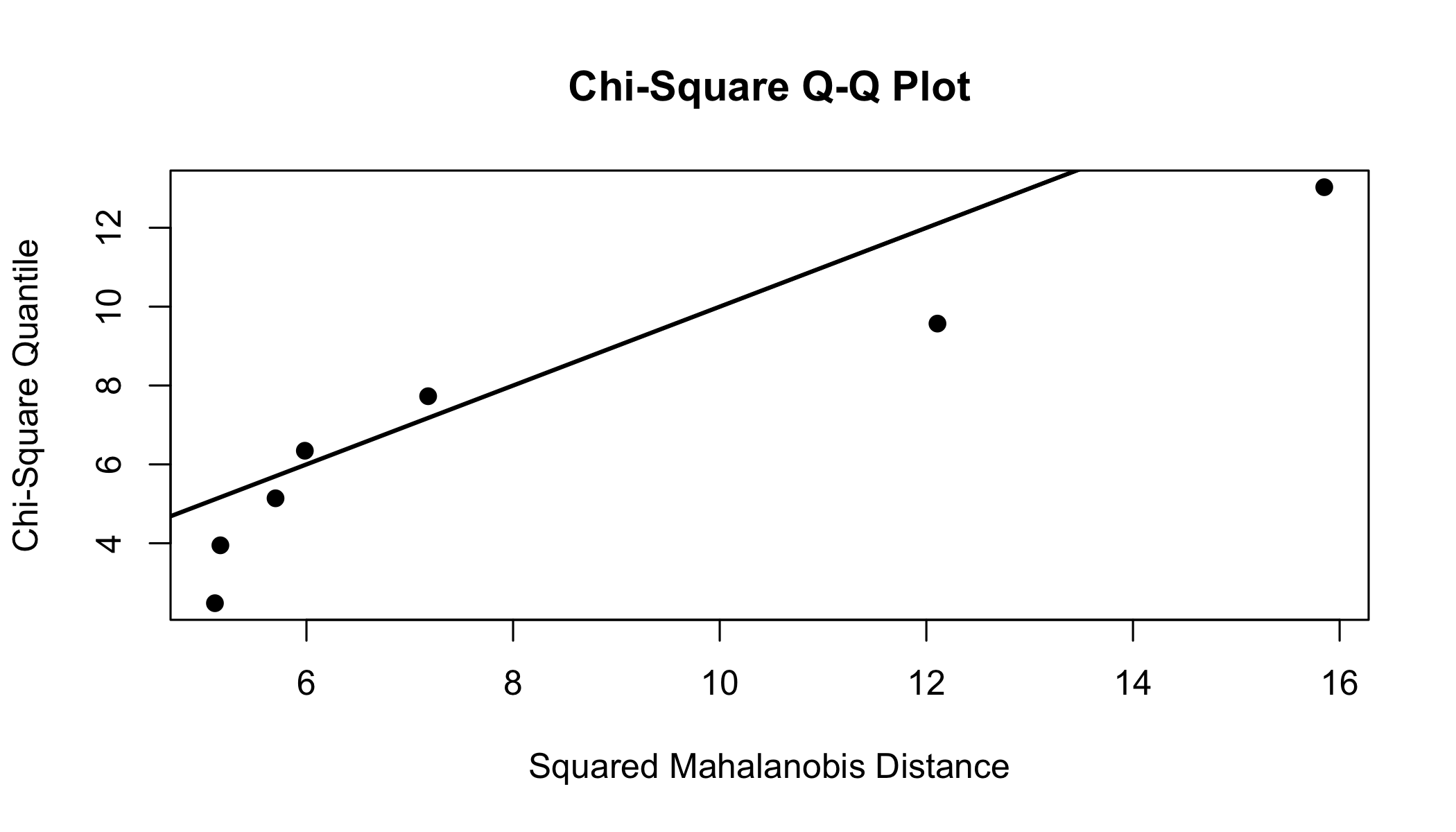


# Shapiro Wilk test for variable x2  
shapiro.test(seeds$area)

##   
## Shapiro-Wilk normality test  
##   
## data: seeds$area  
## W = 0.95557, p-value = 0.78

### Multivariate Normality test

mvtest <- mvn(seeds[,1:7], mvnTest='royston', multivariatePlot='qq')



mvtest$multivariateNormality

## Test H p value MVN  
## 1 Royston 4.724365 0.3576887 YES

mvtest$univariateNormality

## Test Variable Statistic p value Normality  
## 1 Shapiro-Wilk area 0.9556 0.7800 YES   
## 2 Shapiro-Wilk perimeter 0.9584 0.8052 YES   
## 3 Shapiro-Wilk compactness 0.9178 0.4524 YES   
## 4 Shapiro-Wilk length\_of\_kernel 0.9374 0.6150 YES   
## 5 Shapiro-Wilk width\_of\_kernel 0.7913 0.0336 NO   
## 6 Shapiro-Wilk asymmetry\_coefficient 0.9585 0.8054 YES   
## 7 Shapiro-Wilk length\_of\_kernel\_groove 0.8529 0.1305 YES

mvtest$Descriptives

## n Mean Std.Dev Median Min Max  
## area 7 14.7828571 0.75074883 14.6900 13.840 16.140  
## perimeter 7 14.4471429 0.38878933 14.4900 13.940 14.990  
## compactness 7 0.8901429 0.01292863 0.8951 0.871 0.905  
## length\_of\_kernel 7 5.5055714 0.17722571 5.5540 5.291 5.763  
## width\_of\_kernel 7 3.3562857 0.09758366 3.3330 3.259 3.562  
## asymmetry\_coefficient 7 2.2285714 0.85127333 2.2590 1.018 3.586  
## length\_of\_kernel\_groove 7 5.0222857 0.18075371 4.9560 4.805 5.220  
## 25th 75th Skew Kurtosis  
## area 14.3350 15.07000 0.535148711 -1.05730053  
## perimeter 14.1500 14.70500 0.080748023 -1.77155932  
## compactness 0.8805 0.89945 -0.194615818 -1.80852975  
## length\_of\_kernel 5.3550 5.61050 0.085448841 -1.78955168  
## width\_of\_kernel 3.3120 3.35800 1.168832594 -0.02434668  
## asymmetry\_coefficient 1.7880 2.58050 0.041070325 -1.31958973  
## length\_of\_kernel\_groove 4.8905 5.19700 0.009256081 -2.01608255

### Transform to near normal

trans<-powerTransform(seeds[,1:7])  
seeds\_trans <- seeds[,1:7]  
seeds\_trans<-bcPower(seeds\_trans,trans$lambda)  
mvtest\_trans <- mvn(seeds\_trans, mvnTest='royston', multivariatePlot='qq')  
mvtest\_trans$multivariateNormality  
mvtest\_trans$univariateNormality  
mvtest\_trans$Descriptives