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=TR
UE)

# 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:

$$Y_2 = \beta_{2,1}Y_1 + \epsilon_2$$

and

$$Y_3 = \beta_{3,1}Y_1 + \beta_{3,2}Y_2 + \epsilon_3$$

Perform a linear regression to find the coefficients $\beta_{2,1}$, $\beta_{3,1}$ and $\beta_{3,2}$.

```
# 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]
```

```r
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] -0.1272797

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] 0.0004425704

#Coefficent of Y2
beta3_2 <- model_summary$coefficients[[3]]
print(beta3_2)

## [1] -0.1559556
```

Question 1b

Estimate $\sigma_2^2 = var(\epsilon_2)$

```r
# Question 1b
sigma_2_square <- (summary(model_1)$sigma)^2
print(sigma_2_square)

## [1] 0.04925335
```

Question 1c

Estimate $\sigma_3^2 = var(\epsilon_3)$

```r
# Question 1c
sigma_3_square <- (summary(model_2)$sigma)^2
print(sigma_3_square)

## [1] 0.03382979
```

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,]  1.0000000000 0.0000000    0
## [2,]  0.1272796638 1.0000000    0
## [3,] -0.0004425704 0.1559556    1
```

Question 1e

Compute $T\sum T^\intercal$

```
TT <- T%*%vcmat%*%t(T)

print(T)
```

```
##               [,1]      [,2] [,3]
## [1,]  1.0000000000 0.0000000    0
## [2,]  0.1272796638 1.0000000    0
## [3,] -0.0004425704 0.1559556    1
```

Question 1f

Calculate S^{-1} given:

$$S^{-1} = T^\intercal D^{-1} T$$

where D is a 3x3 diagonal matrix, with entries on the main diagonal as σ_1^2, σ_2^2, σ_3^2 and T has already been calculated earlier.

To make S^{-1} as close to $\sum^{-1}$ possible, let us assume:

$$\sum^{-1} = T^\intercal D^{-1} T$$

Based on that we can calculate the value of D as:

$$D = (T \sum^{-1} T^\intercal)^{-1}$$

```
# Calculate the inverse of covariance matrix
vcmat_inv <- solve(vcmat)
round(vcmat_inv)
```

```
##      [,1] [,2] [,3]
## [1,]   10    2    0
## [2,]    2   20    5
## [3,]    0    5   30
```

```
# Sigma square can be derived from estimated D
sigma_1_square <- solve(T%*%vcmat_inv%*%t(T))[1,1]

# Calculate D
D = matrix(c(sigma_1_square, 0, 0, 0, sigma_2_square, 0, 0, 0, sigma_3_square
),nrow = 3)
print(D)
```

```
##           [,1]       [,2]       [,3]
## [1,] 0.1058399 0.00000000 0.00000000
## [2,] 0.0000000 0.04925335 0.00000000
## [3,] 0.0000000 0.00000000 0.03382979
```

```
# Compute S inverse
print(round(t(T)%*%solve(D)%*%T))
```

```
##      [,1] [,2] [,3]
## [1,]   10    3    0
## [2,]    3   21    5
## [3,]    0    5   30
```

```
#Compare with inverse of covariance matrix
print(round(vcmat_inv))
```

```
##      [,1] [,2] [,3]
## [1,]   10    2    0
## [2,]    2   20    5
## [3,]    0    5   30
```

Question 2

Load the dataset from local storage

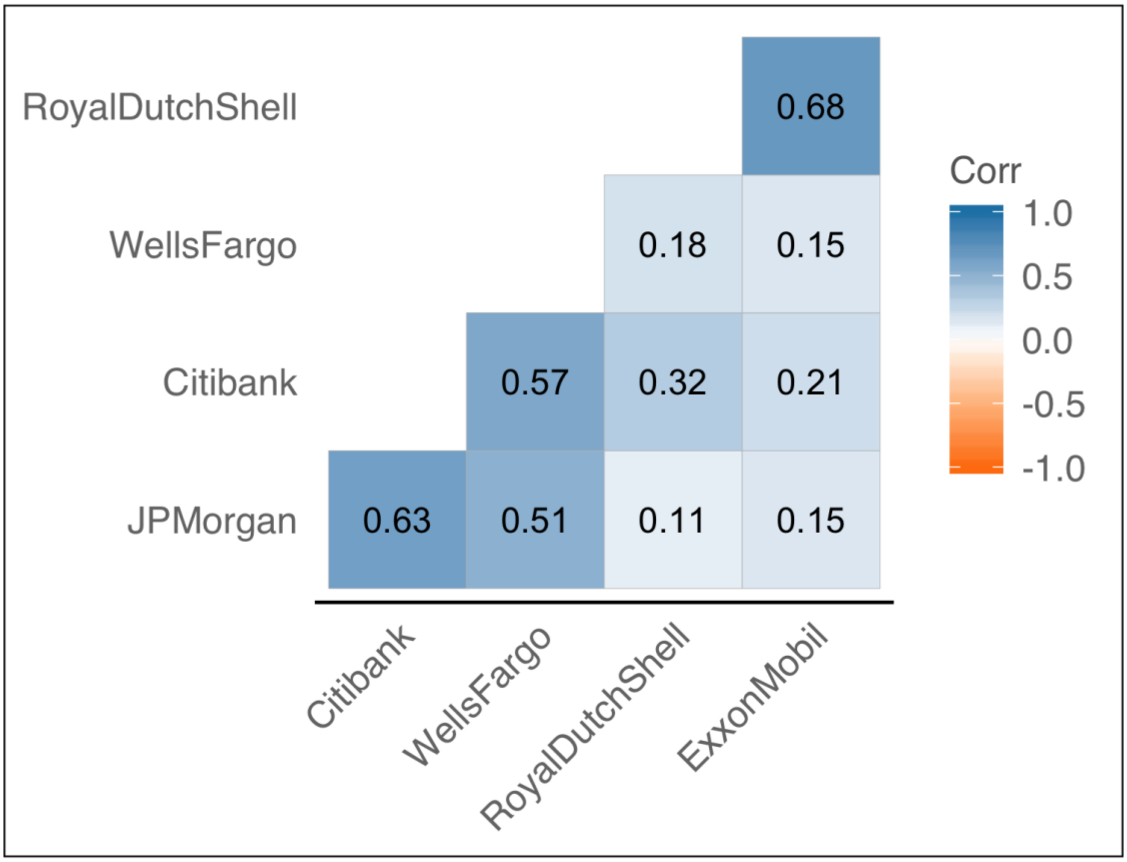
Load the dataset using ‘fread’ from **data.table** package. The stock data consists of weekly returns of five different stocks. The weekly returns of each stock are defined as (current week closing price - previous week closing price)/(previous week closing price) for 103 successive weeks.

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

```
##     JPMorgan            Citibank            WellsFargo
##  Min.   :-0.045867   Min.   :-0.0597924   Min.   :-0.0362141
##  1st Qu.:-0.013564   1st Qu.:-0.0132409   1st Qu.:-0.0080536
##  Median : 0.003363   Median : 0.0017339   Median : 0.0003354
##  Mean   : 0.001063   Mean   : 0.0006554   Mean   : 0.0016261
##  3rd Qu.: 0.016804   3rd Qu.: 0.0140293   3rd Qu.: 0.0100178
##  Max.   : 0.048480   Max.   : 0.0525266   Max.   : 0.0406957
##  RoyalDutchShell       ExxonMobil
##  Min.   :-0.053948   Min.   :-0.063605
##  1st Qu.:-0.014470   1st Qu.:-0.012539
##  Median : 0.006335   Median : 0.005215
##  Mean   : 0.004049   Mean   : 0.004039
##  3rd Qu.: 0.022237   3rd Qu.: 0.021622
##  Max.   : 0.061994   Max.   : 0.078416
```

The observations in the data appear to be independently distributed but the rate of return across stocks are correlated. Generally, one would expect the stocks would move together in response to general economic conditions. The correlation below shows a relationship between JPMorgan, Citibank and WellsFargo (banking stocks) and also relationship between Royal Dutch Shell and ExxonMobil (oil stocks)

```
ggcorrplot(cor(dat, use = "pairwise.complete.obs"), hc.order = FALSE, type =
"lower",
           ggtheme = ggthemes::theme_gdocs,
           colors = c("#ff7f0e", "white", "#1f83b4"),
           lab = TRUE)+
           theme(panel.grid.major=element_blank())
```



Question 2a

Perform factor analysis using principal component analysis method. Looking at the importance of the components, the first two principal components explains 80% of the variance in the data. The proportion of the variance explained by component 3 is less than 0.2 ($1/p$) (p being 5 in this case). A rule of thumb suggests retaining only those components whose variances individually are greater than $1/p$. So, we will retain only the first two principal components.

```
# Principal component analysis method
dat_pc<-princomp(dat)
summary(dat_pc, loadings = TRUE)

## Importance of components:
##                            Comp.1     Comp.2     Comp.3     Comp.4
## Standard deviation     0.03680217 0.02635056 0.01585365 0.01188352
## Proportion of Variance 0.52926066 0.27133298 0.09821584 0.05518400
## Cumulative Proportion  0.52926066 0.80059364 0.89880948 0.95399348
##                            Comp.5
## Standard deviation     0.01085046
## Proportion of Variance 0.04600652
## Cumulative Proportion  1.00000000
##
## Loadings:
##                 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## JPMorgan         0.223  0.625  0.326  0.663  0.118
## Citibank         0.307  0.570 -0.250 -0.414 -0.589
## WellsFargo       0.155  0.345        -0.497  0.780
## RoyalDutchShell  0.639 -0.248 -0.642  0.309  0.148
## ExxonMobil       0.651 -0.322  0.646 -0.216
```

Based on the importance of the components, we have seen that first two components explains most of the variance, hence we are going to perform factor analysis using m=2 factors.

```
fact_pc<-principal(dat, nfactors=2,rotate="none")
print(fact_pc)
```

```
## Principal Components Analysis
## Call: principal(r = dat, nfactors = 2, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                  PC1   PC2   h2   u2 com
## JPMorgan        0.73 -0.44 0.73 0.27 1.6
## Citibank        0.83 -0.28 0.77 0.23 1.2
## WellsFargo      0.73 -0.37 0.67 0.33 1.5
## RoyalDutchShell 0.60  0.69 0.85 0.15 2.0
## ExxonMobil      0.56  0.72 0.83 0.17 1.9
##
##                        PC1  PC2
## SS loadings           2.44 1.41
## Proportion Var        0.49 0.28
## Cumulative Var        0.49 0.77
## Proportion Explained  0.63 0.37
## Cumulative Proportion 0.63 1.00
##
## Mean item complexity =  1.6
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.1
##  with the empirical chi square  19.17  with prob <  1.2e-05
##
## Fit based upon off diagonal values = 0.95
```

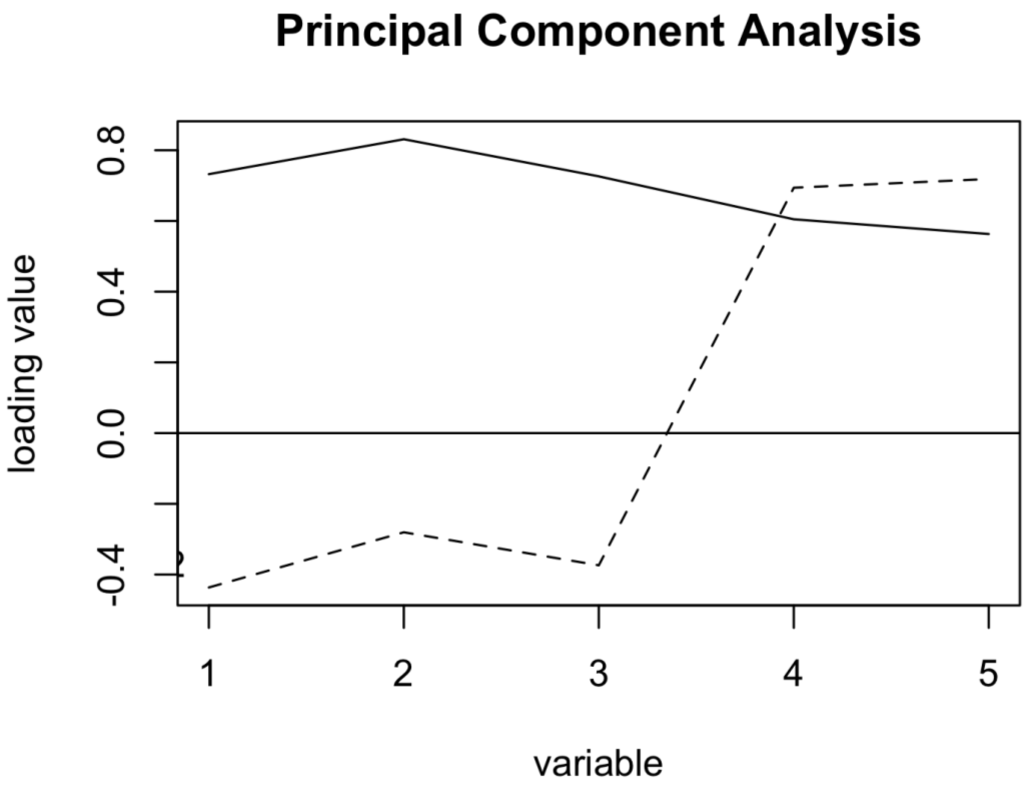
Question 2b

Looking at the factor loadings, all of the stocks load highly on the first factor while the second factor shows contrasting loading of banking stocks (negative low loadings) to the oil stocks (positive high loadings). It is clear that the factor F_1 represents general economic conditions and can be called as *Market Factor*. The second factor F_2 seems to differentiate stocks in different industries and can be called as *Industry Factor*.

So, looking at the factor variances and loadings, it can be summarized that the weekly rates of stocks are determined by general market conditions and activities in the respective industries and some of it is explained by other/residual factors.

The chart below shows the same story where all the variables load highly on the first factor and first 3 variable load negatively on second factor while the last 2 variables load positively on second factor.

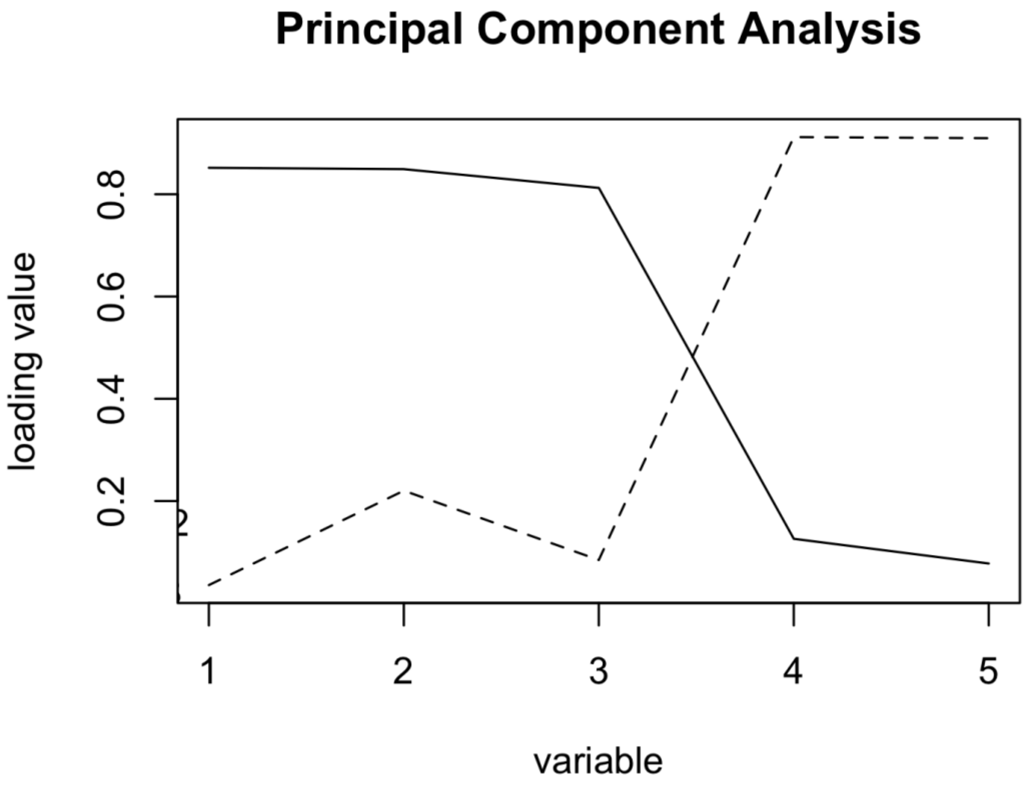
```
factor.plot(fact_pc)
```



Let's look at the factor anlysis with rotated loadings. Here, we are using the same principal components analysis method with **varimax** rotation. We will still use two factors and looking at the loadings, we now see a different pattern. The banking stocks load heavily on the the factor 1 F_1 while the oil stocks load heavily on the second factor F_2.

Factor 1 represent those economic forces that causes the bank stocks to move together while the Factor 2 represents the economic forces that affect the oil stocks. The chart confirms the above findings.

```
fact_pc_rot <-principal(dat, nfactors=2,rotate="varimax")
print(fact_pc_rot)

## Principal Components Analysis
## Call: principal(r = dat, nfactors = 2, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                  RC1  RC2   h2   u2 com
## JPMorgan        0.85 0.04 0.73 0.27 1.0
## Citibank        0.85 0.22 0.77 0.23 1.1
## WellsFargo      0.81 0.08 0.67 0.33 1.0
## RoyalDutchShell 0.13 0.91 0.85 0.15 1.0
## ExxonMobil      0.08 0.91 0.83 0.17 1.0
##
##                        RC1  RC2
## SS loadings           2.13 1.72
## Proportion Var        0.43 0.34
## Cumulative Var        0.43 0.77
## Proportion Explained  0.55 0.45
## Cumulative Proportion 0.55 1.00
##
## Mean item complexity =  1
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.1
##  with the empirical chi square  19.17  with prob <  1.2e-05
##
## Fit based upon off diagonal values = 0.95

factor.plot(fact_pc_rot)
```



Question 2c

Table 9.8 on page 510 of Johnson and Wichern shows the unrotated and rotated factors for the same dataset obtained using the maximum likelihood method. In our previous steps we have used the principal component method for obtaining the factors. Here, we will use another method (without rotation) to calculate the factors. The unrotated factors using the maximum likelihood method shows that the oil stocks load heavily on the the factor 1 (F_1) while the banking stock load heavily on the second factor F_2. Factor 1 represent those economic forces that causes the oil stocks to move together while the Factor 2 represents the economic forces that affect the banking stocks.

```
# Maximum Likelihood method
fact_ml <- factanal(x = dat,factors = 2, rotation = 'none')
print(fact_ml)

##
## Call:
## factanal(x = dat, factors = 2, rotation = "none")
##
## Uniquenesses:
```

```
##        JPMorgan        Citibank      WellsFargo RoyalDutchShell 
##           0.417           0.275           0.542           0.005 
##      ExxonMobil 
##           0.530 
## 
## Loadings:
##                 Factor1 Factor2
## JPMorgan         0.121   0.754 
## Citibank         0.328   0.786 
## WellsFargo       0.188   0.650 
## RoyalDutchShell  0.997         
## ExxonMobil       0.685         
## 
##                Factor1 Factor2
## SS loadings      1.622   1.610
## Proportion Var   0.324   0.322
## Cumulative Var   0.324   0.646
## 
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 1.97 on 1 degree of freedom.
## The p-value is 0.16
```

The rotated factors using the maximum likelihood method are similar to the rotated factors using the principal component method and shows that the banking stocks load heavily on the the factor 1 (F_1)while the oil stock load heavily on the second factor F_2.

```
# Maximum likelihood method with rotation
fact_ml_rot <- factanal(x = dat,factors = 2, rotation = 'varimax')
print(fact_ml_rot)
```

```
## 
## Call:
## factanal(x = dat, factors = 2, rotation = "varimax")
## 
## Uniquenesses:
##        JPMorgan        Citibank      WellsFargo RoyalDutchShell 
##           0.417           0.275           0.542           0.005 
##      ExxonMobil 
##           0.530 
## 
## Loadings:
##                 Factor1 Factor2
## JPMorgan        0.763          
## Citibank        0.819   0.232  
## WellsFargo      0.668   0.108
## RoyalDutchShell 0.113   0.991
## ExxonMobil      0.108   0.677
##
##                 Factor1 Factor2
## SS loadings       1.725   1.507
## Proportion Var    0.345   0.301
## Cumulative Var    0.345   0.646
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 1.97 on 1 degree of freedom.
## The p-value is 0.16
```

Question 3

Load the dataset from local storage

Data Descriptions: Four measurements of male Egyptian skulls from 5 different time periods. Thirty skulls are measured from each time period.
Number of observations: 150
Variable Names:

- MB: Maximal Breadth of Skull
- BH: Basibregmatic Height of Skull
- BL: Basialveolar Length of Skull
- NH: Nasal Height of Skull
- Year: Approximate Year of Skull Formation (negative = B.C., positive = A.D.)

```
# Load the dataset
egyptskull <- fread('Data/egyptskull.csv')

# Descriptive statistics
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

# Convert the dependent variable to a factor
egyptskull[, Epoch:= as.factor(Epoch)]
```

Question 3a

Logistic regression is a statistical model used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Logistic regression is the appropriate regression analysis to conduct when the dependent variable is binary. Like all regression analyses, the logistic regression is a predictive analysis. In statistics, the logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.

Question 3b

Classification trees are used to predict membership of cases or objects into classes of a categorical dependent variable from their measurements on one or more predictor variables. Classification tree analysis has traditionally been one of the main techniques used in data mining.

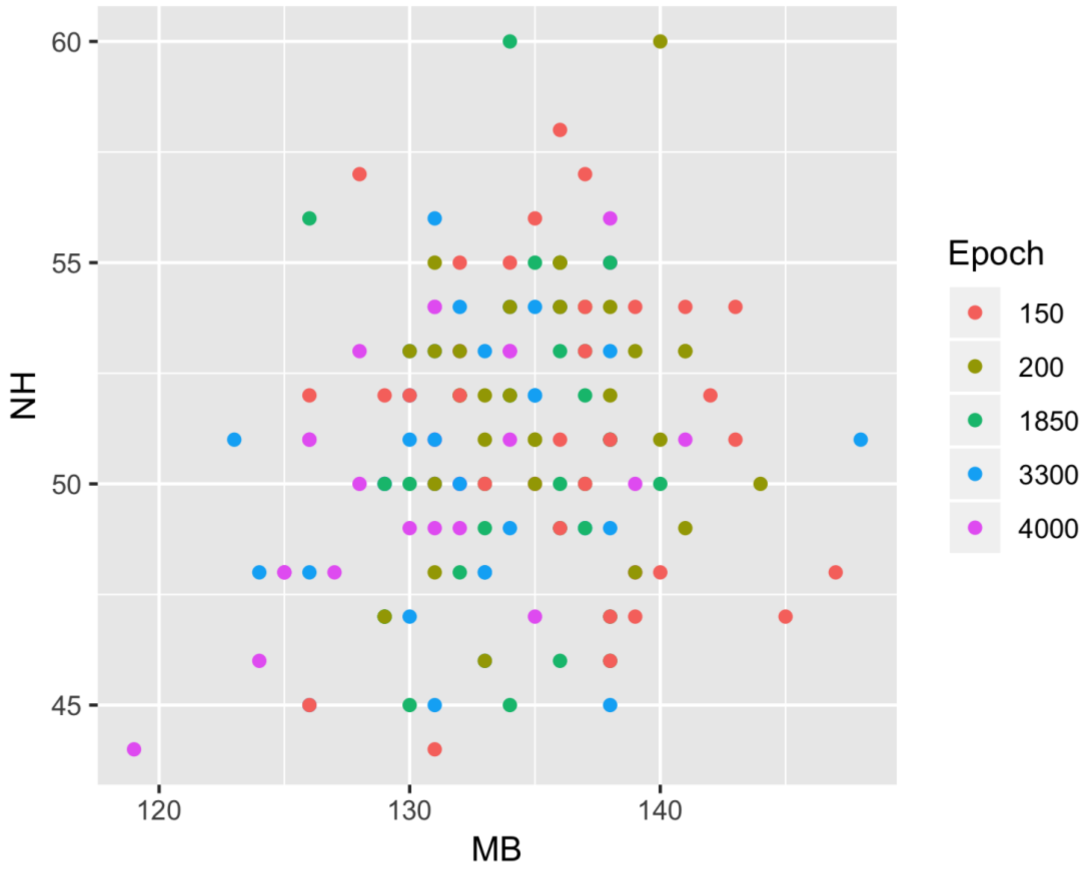
It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

In computer science, Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).

Question 3c

Scatter Plot of Maximal Breadth of Skull with the Nasal Height of Skull.
The scatter plot shows no clear linear relationship and does not have any patterns across different Epochs.

```
ggplot(egyptskull, aes(x=MB, y=NH, 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
```

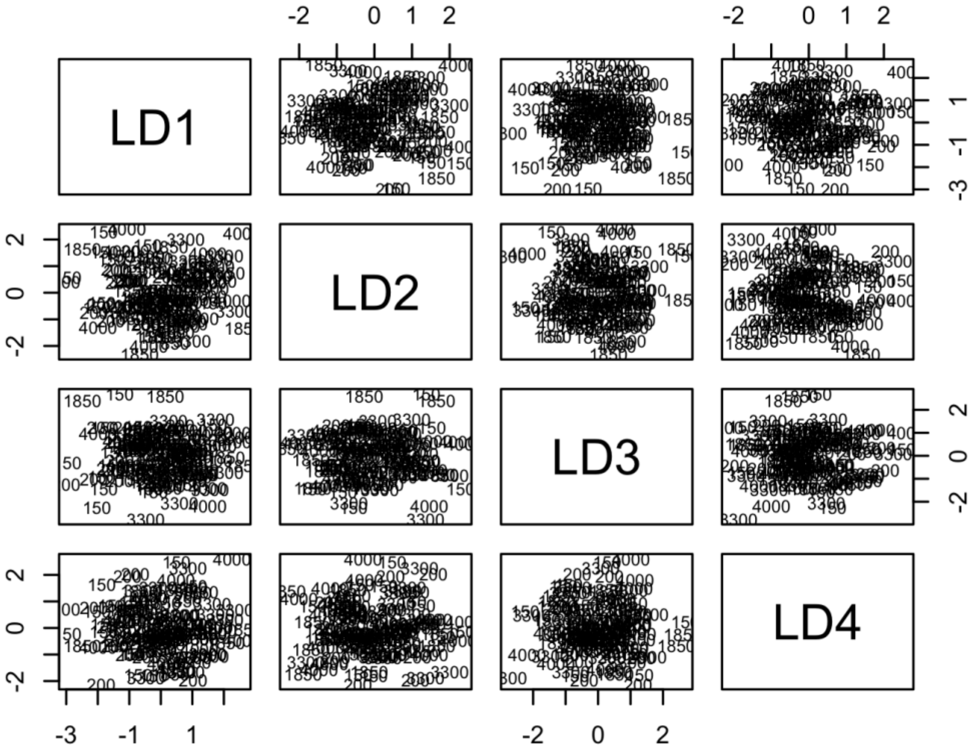
```
egyptskull_test[, .N, by = list(Epoch)]
```

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

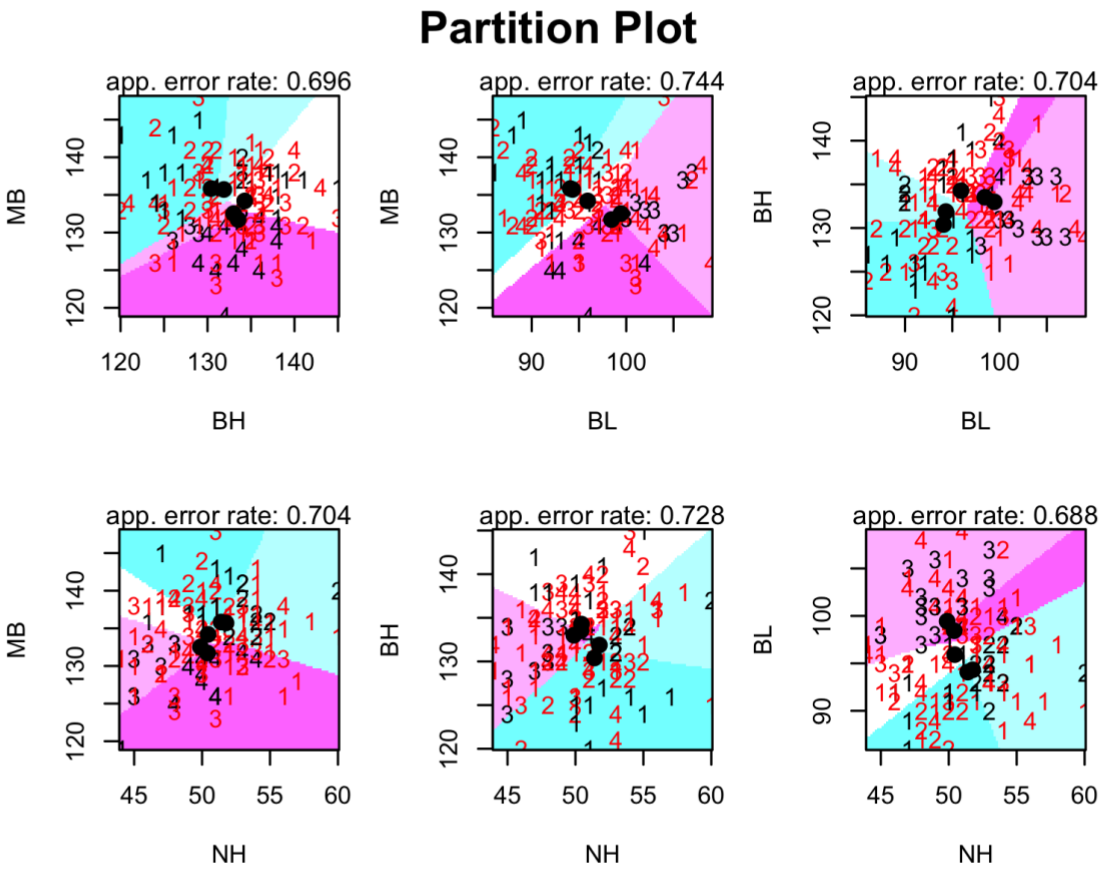
Perform LDA

Looking at the chart, we can see few misclassifications on the training set. The error rate is high being over 70% on training set.

```
#######    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

partimat(as.factor(Epoch) ~ MB+BH+BL+NH, data=egyptskull_train,method="lda")
```



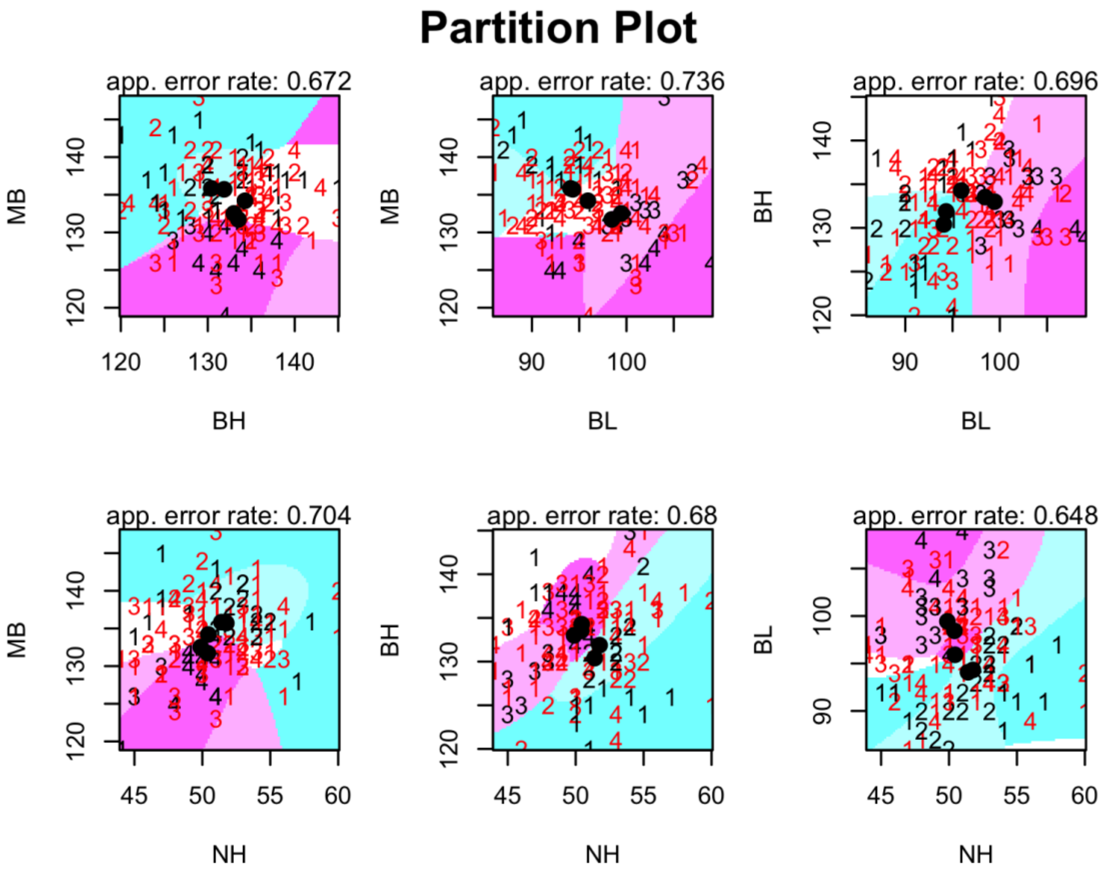
Perform QDA

Looking at the chart, we can see fewer misclassifications on the training set than LDA. The error rate is high being over 65% on training set but better than LDA

```
######## 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

partimat(as.factor(Epoch) ~ MB+BH+BL+NH, data=egyptskull_train,method="qda")
```



Perform Multinomial Logistic

Looking at the variable importance, none of the variable explain the variance in Epoch except for BL (Basialveolar Length of Skull). Even that explains only one class.

```
######  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]]
```

Perform CART

The algorithm of the decision tree models works by repeatedly partitioning the data into multiple sub-spaces, so that the outcomes in each final sub-space is as homogeneous as possible. This approach is technically called recursive partitioning. The produced result consists of a set of rules used for predicting the outcome variable, which can be either:

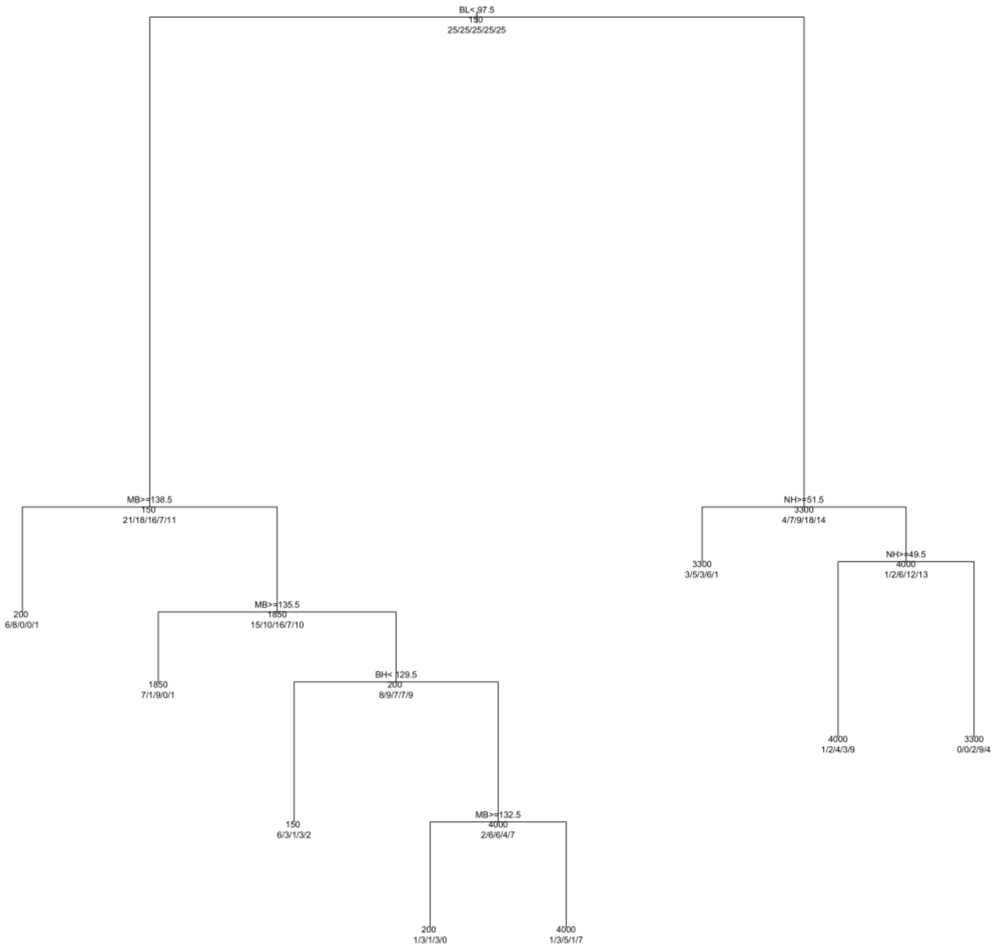
- a continuous variable, for regression trees
- a categorical variable, for classification trees

The decision rules generated by the CART (Classification & Regression Trees) predictive model are generally visualized as a binary tree.

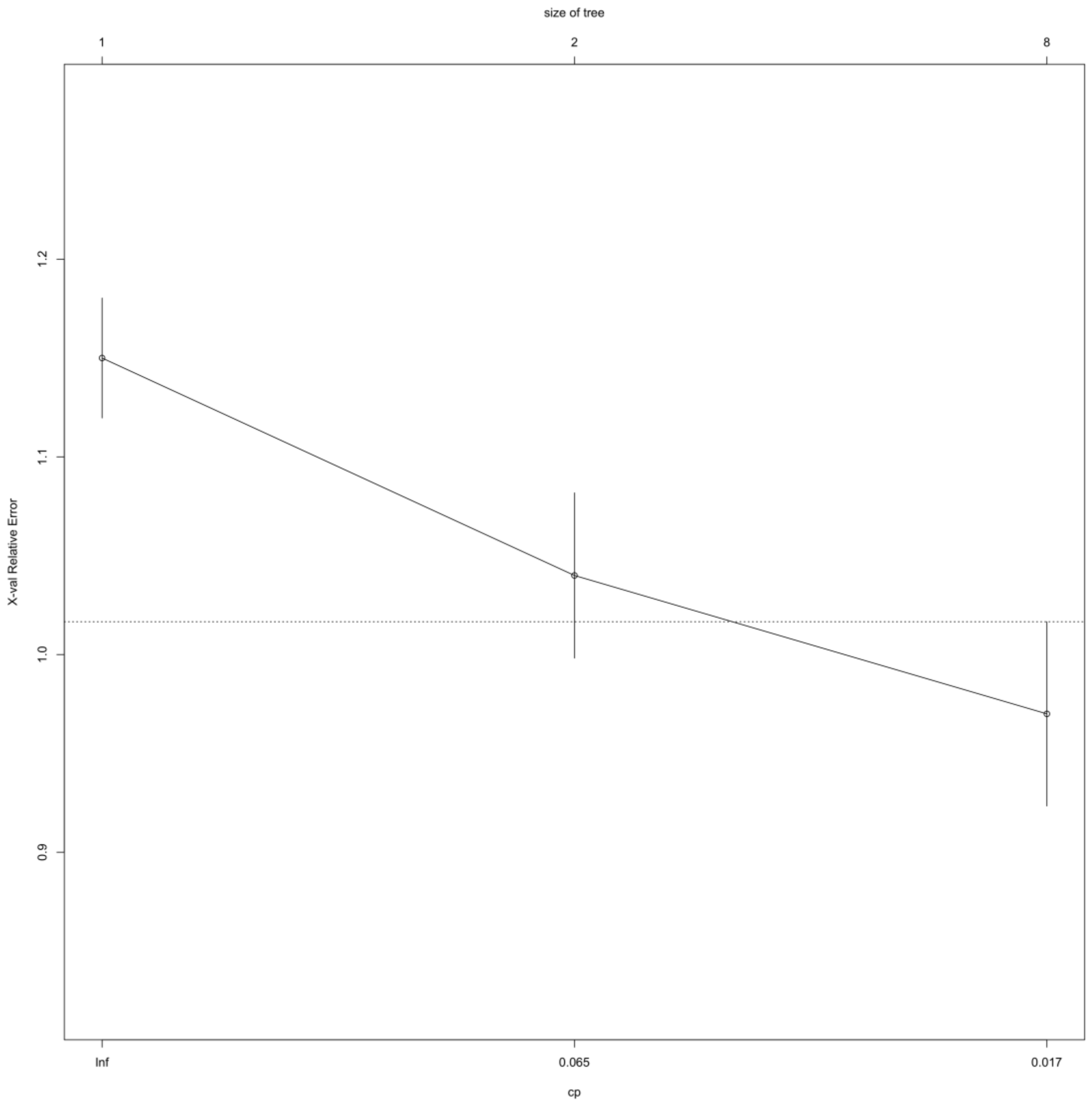
The model's lowest error rate is 0.68 after 7 spilts.

Even after tuning the 'cp' parameter that is pruning the tree at the lowest error rate, the model performance does not improve.

```
######## 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)
```

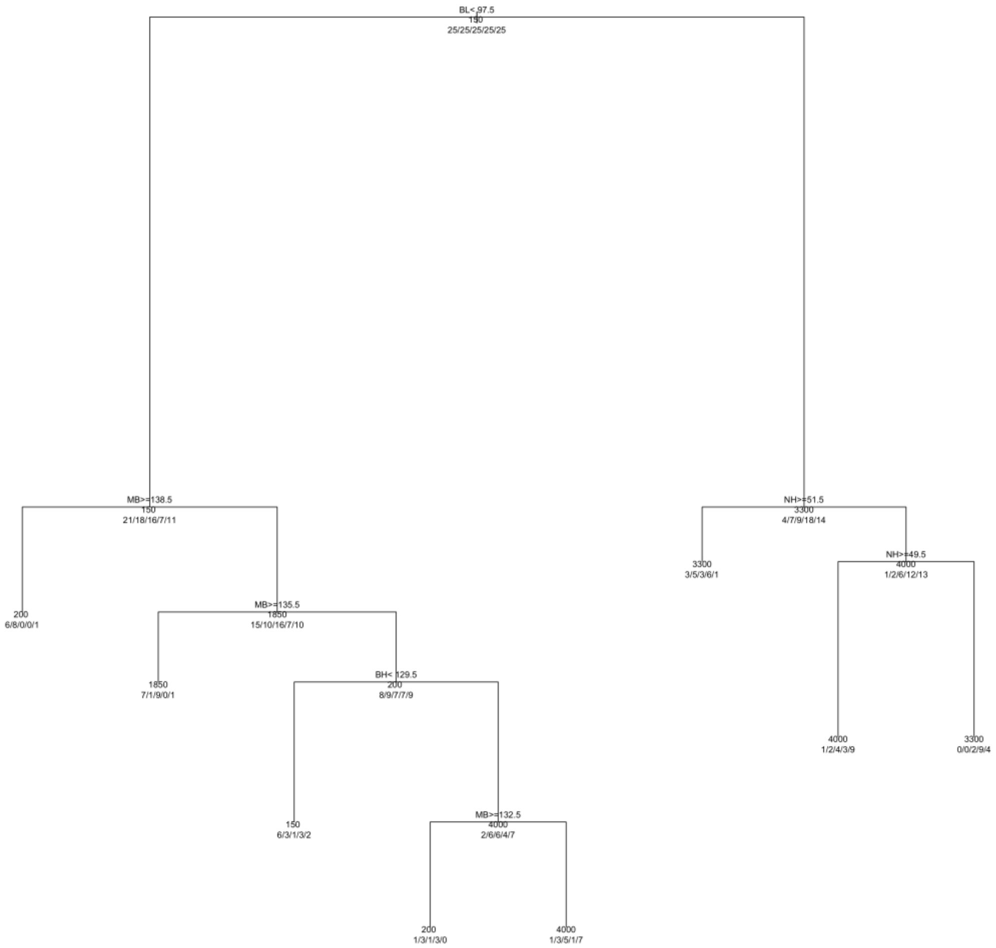


```
egyptskull_test$pred_ct<-predict(model_ct,egyptskull_test,type="vector")
egyptskull_test[, pred_ct:= c(4000, 3300, 1850, 200, 150)[pred_ct]]

model_ct$cptable

##     CP nsplit rel error xerror       xstd
## 1 0.14      0      1.00   1.15 0.03033150
## 2 0.03      1      0.86   1.04 0.04179952
## 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)
```

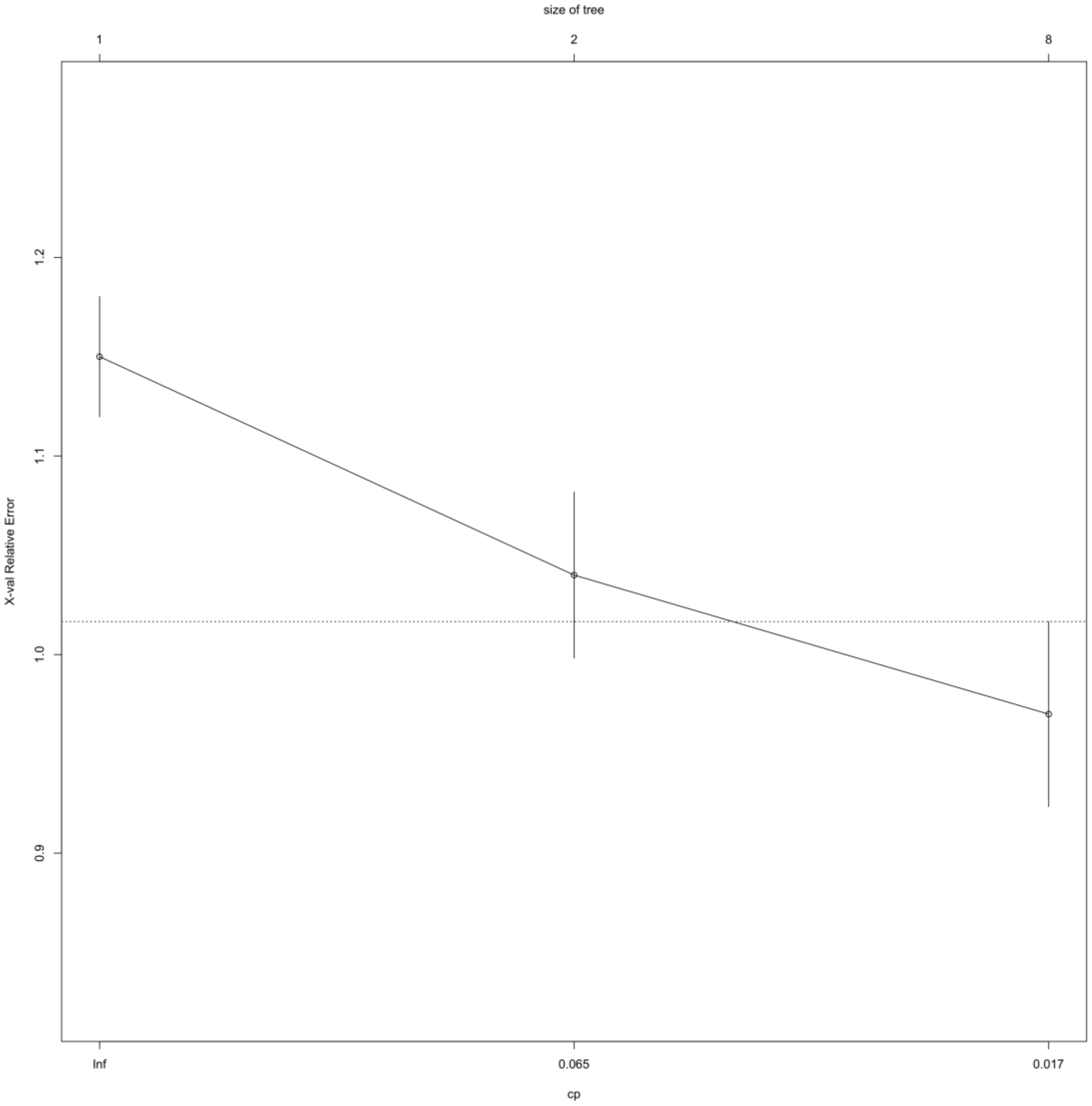
```
## Call:
## rpart(formula = Epoch ~ MB + BH + BL + NH, data = egyptskull_train,
##     method = "class")
##   n= 125
##
##     CP nsplit rel error xerror       xstd
## 1 0.14      0      1.00   1.15 0.03033150
## 2 0.03      1      0.86   1.04 0.04179952
## 3 0.01      7      0.68   0.97 0.04661330
##
## 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)
```



```
plotcp(model_ct_fit)
```



```
egyptskull_test$pred_ct_fit<-predict(model_ct_fit,egyptskull_test,type="vecto
r")
egyptskull_test[, pred_ct_fit:= c(4000, 3300, 1850, 200, 150)[pred_ct_fit]]
```

Build a neural network

Building the network with 5 hidden layers. Adjusting the hidden layers to a lower or a higher number does not lower the error rate on the test set.

```
##### Nnet
model_nnet<-nnet(Epoch ~ MB+BH+BL+NH, data = egyptskull_train,size=5,decay=0.
1)

## # weights:  55
## initial  value 253.566013
## iter  10 value 201.182014
## iter  20 value 196.610503
## iter  30 value 179.910660
## iter  40 value 176.177831
## iter  50 value 175.673082
## iter  60 value 175.137878
## iter  70 value 173.330391
## iter  80 value 172.904557
## iter  90 value 172.660001
## iter 100 value 171.828923
## final  value 171.828923
## stopped after 100 iterations
```

```
summary(model_nnet)
```

```
## a 4-5-5 network with 55 weights
## options were - softmax modelling  decay=0.1
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1
##   0.04   0.54  -0.25  -0.03  -0.63
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2
##   0.00   0.57  -1.13   1.10  -0.61
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3
##   0.14   0.26  -0.60  -0.95   2.64
##  b->h4 i1->h4 i2->h4 i3->h4 i4->h4
##  -0.04  -1.58   0.41   1.42   0.31
##  b->h5 i1->h5 i2->h5 i3->h5 i4->h5
##   0.00  -0.29   0.21   0.01   0.15
##  b->o1 h1->o1 h2->o1 h3->o1 h4->o1 h5->o1
##   0.95  -1.64   0.85   1.21  -1.74  -0.02
##  b->o2 h1->o2 h2->o2 h3->o2 h4->o2 h5->o2
##  -0.85   0.54   0.10   1.22  -0.33   0.22
##  b->o3 h1->o3 h2->o3 h3->o3 h4->o3 h5->o3
##   0.66   0.31  -0.86  -0.88  -0.47   0.22
##  b->o4 h1->o4 h2->o4 h3->o4 h4->o4 h5->o4
##   0.16  -0.01   0.22  -1.16   0.80   0.07
##  b->o5 h1->o5 h2->o5 h3->o5 h4->o5 h5->o5
##  -0.93   0.80  -0.30  -0.38   1.74  -0.49
```

```
egyptskull_test$pred_nnet<-predict(model_nnet,egyptskull_test,type="class")
```

Question 3e

Multinomial Logistic Regression error rate on the test set is the least. Confusion matrix shows the least amount of misclassifications in Multinomial Logistic Regression. Hence, the best fit model is Multinomial Logistic Regression.

```
# LDA
#confusion matrix
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

# error rate
mean(egyptskull_test$lda_predict != egyptskull_test$Epoch)

## [1] 0.64

# QDA
#confusion matrix
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

# error rate
mean(egyptskull_test$qda_predict != egyptskull_test$Epoch)

## [1] 0.72

# Multinomial
#confusion matrix
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
```

```
# error rate
mean(egyptskull_test$pred_mnl != egyptskull_test$Epoch)

## [1] 0.6

# CART
#confusion matrix
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

# error rate
mean(egyptskull_test$pred_ct_fit != egyptskull_test$Epoch)

## [1] 0.96

# Nnet
#confusion matrix
table(egyptskull_test$Epoch,egyptskull_test$pred_nnet)

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

# error rate
mean(egyptskull_test$pred_nnet != egyptskull_test$Epoch)

## [1] 0.72
```

Question 3f

The lowest error rate on the test set is using the Multinomial Logistic Regression. The error rate is .6. So, we are going to use this model for new predictions.

```
####### 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]]
print(egyptskull_val)
```

```
##     MB  BH  BL NH pred_mnl
## 1: 128 143 103 50     4000
## 2: 129 126  91 50      150
## 3: 130 127  99 45     3300
## 4: 130 131  98 53     4000
## 5: 134 124  91 55      150
## 6: 130 130 104 49     3300
## 7: 134 139 101 49     3300
## 8: 136 133  91 49     1850
```

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
#
```

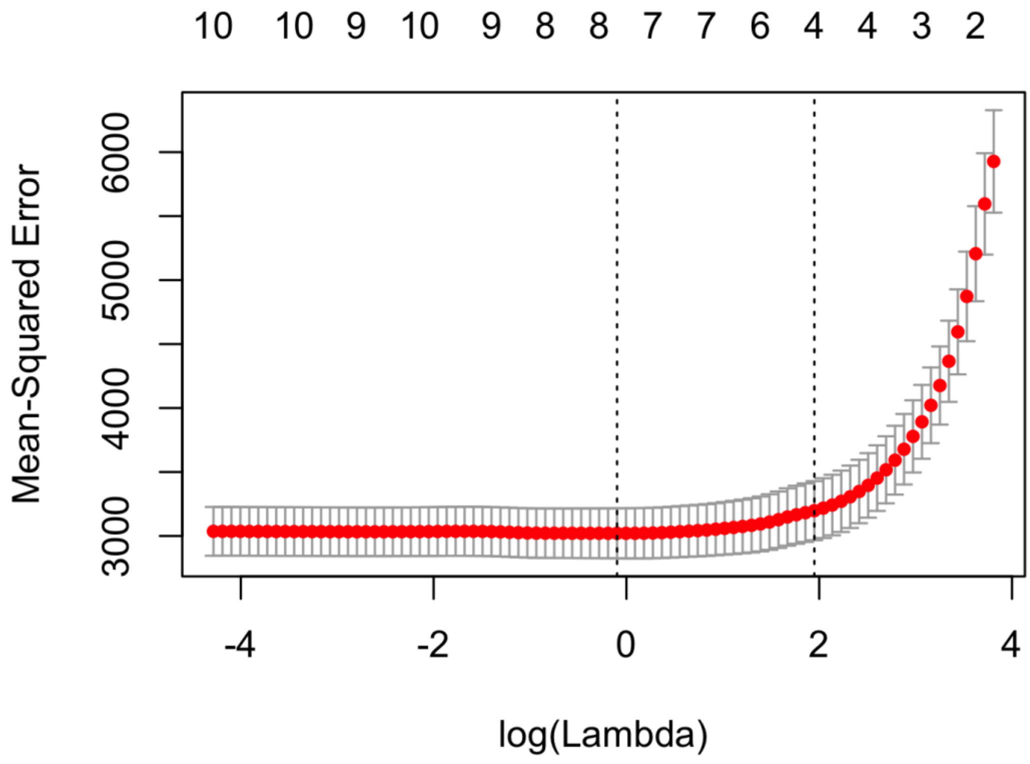
```
diabetes_orig <- fread('Data/diabetes.csv')
dim(diabetes_orig)

## [1] 442  11
```

Question 1a

Least Absolute Shrinkage and Selection Operator

Using k-fold cross validation to find the optimal value of lamda.
lambda.min is the ideal choice for lambda for getting the best fit for LASSO.
A rule of thumb is to use lamba.1se for the optimal fit for LASSO

```
Xmatrix <- as.matrix(diabetes_orig[,1:10])
yVector <- diabetes_orig$Y

cvfit <- cv.glmnet(Xmatrix , yVector)

plot(cvfit)
```



```
cvfit$lambda.min
```

```
## [1] 0.9073702
```

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

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) -236.3565156
## AGE            .
## SEX          -19.0272081
## BMI            5.6298337
## BP             1.0263724
## S1            -0.1475826
## S2             .
## S3            -0.8191866
## S4             0.1267917
## S5            46.9738478
## S6             0.2295918
```

```
cvfit$lambda.1se
```

```
## [1] 7.025438
```

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

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) -213.074728
## AGE            .
## SEX            .
## BMI            5.377448
## BP             0.622048
## S1             .
## S2             .
## S3            -0.380388
## S4             .
## S5            39.521973
## S6             .
```

Trying out different values of lamda.
lambda = 1
Calculating the beta coefficients and r-square

```
LASSOfit_1 <- glmnet(Xmatrix , yVector , lambda=1)
coef(LASSOfit_1)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -235.4791517
## AGE            .        
## SEX          -18.6905954
## BMI            5.6247065
## BP             1.0199542
## S1            -0.1398245
## S2             .        
## S3            -0.8228253
## S4             .        
## S5            46.7986688
## S6             0.2231709
```

```
betaHat_1 <- as.numeric(LASSOfit_1$beta)
betaHat_1
```

```
##  [1]   0.0000000 -18.6905954   5.6247065   1.0199542  -0.1398245
##  [6]   0.0000000  -0.8228253   0.0000000  46.7986688   0.2231709
```

```
y <- yVector
y_hat_cv <- predict(LASSOfit_1, Xmatrix)
rsq_lasso_cv <- cor(y, y_hat_cv)^2
rsq_lasso_cv
```

```
##             s0
## [1,] 0.5137702
```

```
LASSOfit_1$dev.ratio
```

```
## [1] 0.513285
```

lambda = 2
Calculating the beta coefficients and r-square

```
LASSOfit_2 <- glmnet(Xmatrix , yVector , lambda=2)
coef(LASSOfit_2)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -228.72044434
## AGE            .         
## SEX          -15.17631734
## BMI            5.57814487
## BP             0.95395397
## S1            -0.07849061
## S2                .
## S3              -0.77856545
## S4                .
## S5              44.35993112
## S6               0.14724351
```

```
betaHat_2 <- as.numeric(LASSOfit_2$beta)
betaHat_2
```

```
##  [1]   0.00000000 -15.17631734   5.57814487   0.95395397  -0.07849061
##  [6]   0.00000000  -0.77856545   0.00000000  44.35993112   0.14724351
```

```
y <- yVector
y_hat_cv <- predict(LASSOfit_2, Xmatrix)
rsq_lasso_cv <- cor(y, y_hat_cv)^2
rsq_lasso_cv
```

```
##             s0
## [1,] 0.5111147
```

```
LASSOfit_2$dev.ratio
```

```
## [1] 0.5093864
```

lambda = 4
Calculating the beta coefficients and r-square

```
LASSOfit_4 <- glmnet(Xmatrix , yVector , lambda=2)
coef(LASSOfit_4)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -228.72044434
## AGE                .
## SEX            -15.17631734
## BMI              5.57814487
## BP               0.95395397
## S1              -0.07849061
## S2                .
## S3              -0.77856545
## S4                .
## S5              44.35993112
## S6               0.14724351
```

```
betaHat_4 <- as.numeric(LASSOfit_4$beta)
betaHat_4
```

```
##  [1]   0.00000000 -15.17631734   5.57814487   0.95395397  -0.07849061
##  [6]   0.00000000  -0.77856545   0.00000000  44.35993112   0.14724351
```

```
y <- yVector
y_hat_cv <- predict(LASSOfit_4, Xmatrix)
rsq_lasso_cv <- cor(y, y_hat_cv)^2
rsq_lasso_cv
```

```
##             s0
## [1,] 0.5111147
```

```
LASSOfit_4$dev.ratio
```

```
## [1] 0.5093864
```

lambda = 0.5
Calculating the beta coefficients and r-square

```
LASSOfit_0.5 <- glmnet(Xmatrix , yVector , lambda=0.5)
coef(LASSOfit_0.5)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) -247.1746352
## AGE             .
## SEX          -20.6284070
## BMI            5.6586905
## BP             1.0615003
## S1            -0.2213937
## S2              .
## S3            -0.6647105
## S4             2.4375027
## S5            47.7908075
## S6             0.2532147
```

```
betaHat_0.5 <- as.numeric(LASSOfit_0.5$beta)
betaHat_0.5
```

```
##  [1]   0.0000000 -20.6284070   5.6586905   1.0615003  -0.2213937
##  [6]   0.0000000  -0.6647105   2.4375027  47.7908075   0.2532147
```

```
y <- yVector
y_hat_cv <- predict(LASSOfit_0.5, Xmatrix)
rsq_lasso_cv <- cor(y, y_hat_cv)^2
rsq_lasso_cv
```

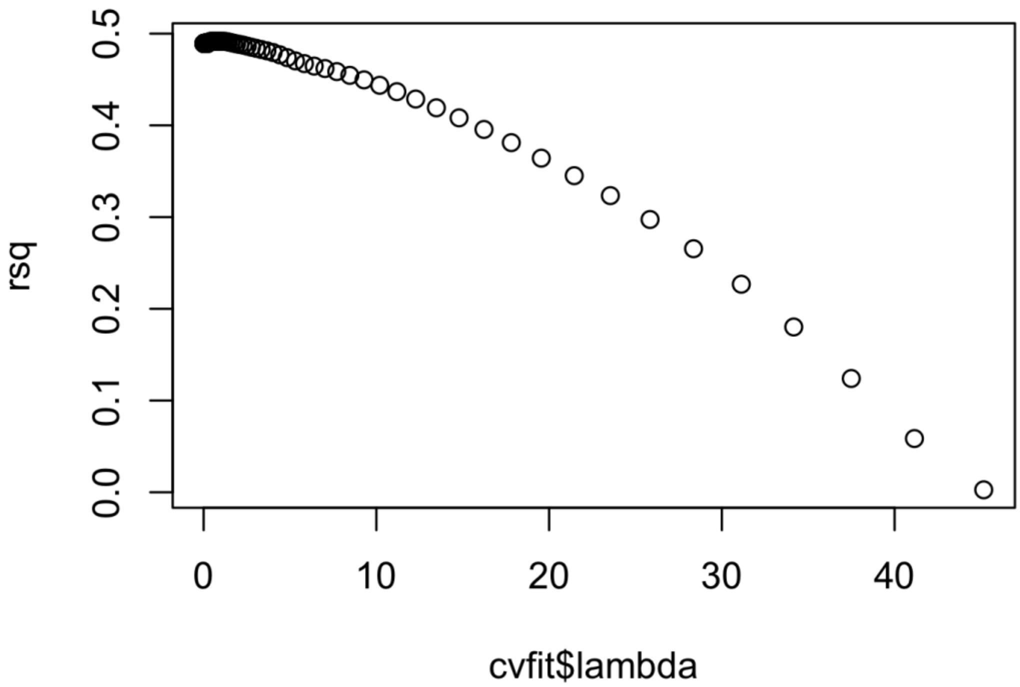
```
##             s0
## [1,] 0.5150459

LASSOfit_0.5$dev.ratio

## [1] 0.5149112
```

lambda = 0.05
Calculating the beta coefficients and r-square

```
LASSOfit_0.05 <- glmnet(Xmatrix , yVector , lambda=0.05)
coef(LASSOfit_0.05)

## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -306.12734656
## AGE           -0.02730111
## SEX          -22.63842274
## BMI            5.61662308
## BP             1.10846558
## S1            -0.81669090
## S2             0.50656843
## S3             0.02630060
## S4             5.29636953
## S5            61.93246459
## S6             0.27876848

betaHat_0.05 <- as.numeric(LASSOfit_0.05$beta)
betaHat_0.05

##  [1]  -0.02730111 -22.63842274   5.61662308   1.10846558  -0.81669090
##  [6]   0.50656843   0.02630060   5.29636953  61.93246459   0.27876848

y <- yVector
y_hat_cv <- predict(LASSOfit_0.05, Xmatrix)
rsq_lasso_cv <- cor(y, y_hat_cv)^2
rsq_lasso_cv

##             s0
## [1,] 0.5174911

LASSOfit_0.05$dev.ratio

## [1] 0.517488
```

Calculating the max r-square from the k-fold cross validation.
lambda.min comes out to be the ideal choice which yeilds the maximum r-squared

```
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: 0.9073702 0.4917177
```

lambda = 0.0735996
Calculating the beta coefficients and r-square

```
LASSOfit_0.07 <- glmnet(Xmatrix , yVector , lambda=0.0735996)
coef(LASSOfit_0.07)

## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -303.24112690
## AGE           -0.02432023
## SEX          -22.49809373
## BMI            5.62627039
## BP             1.10591890
## S1            -0.77818737
## S2             0.46707556
## S3             .
## S4             5.37361802
## S5            60.89521694
## S6             0.27696570
```

```
betaHat_0.07 <- as.numeric(LASSOfit_0.07$beta)
betaHat_0.07
```

```
##  [1]  -0.02432023 -22.49809373   5.62627039   1.10591890  -0.77818737
##  [6]   0.46707556   0.00000000   5.37361802  60.89521694   0.27696570
```

```
y <- yVector
y_hat_cv <- predict(LASSOfit_0.07, Xmatrix)
rsq_lasso_cv <- cor(y, y_hat_cv)^2
rsq_lasso_cv
```

```
##             s0
## [1,] 0.5174173
```

```
LASSOfit_0.07$dev.ratio
```

```
## [1] 0.5174113
```

Question 1b

With the higher values of lambda, we are getting lesser variables that explain most of the variance in the disease progression (dependent variable). However, the r-squared gets lower with the lesser variables. The best r-square is achieved at lambda = 0.0735996 and includes almost all the variables that explain most of the variance. It should be noted that the key variables are still the same as found through manual experiments of trying linear regressions with different combinations of the variable. At lambda = 0.0735996, the most significant variable is S5, followed by SEX, BMI, S4 and BP. The remaining variables explains very little of the variance in the disease progression (dependent variable).

Question 1c

Variable significance from LASSO model is similar to the ones obtained from multiple linear regression. However, comparing the beta coefficients from LASSO model with the coefficients from multiple linear regression, the coefficients from LASSO model are slightly higher than obtained from multiple linear regression. The r-squared from LASSO model is slightly higher at 0.5174173 as compared to 0.5062 from multiple linear regression. In a way, LASSO model is better as it presents a similar fit and provides an automated way of finding beta co-efficents whereas multiple linear regression requires manual trial and error experiments.

```
diabetes_orig[, SEX:=as.factor(SEX)]
model <- lm(Y~S2+S4+BMI+BP+S5+SEX, data = diabetes_orig)
summary(model)

##
## Call:
## lm(formula = Y ~ S2 + S4 + BMI + BP + S5 + SEX, data = diabetes_orig)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -147.812  -38.973   -3.339   39.018  153.135
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -293.4056    28.2133 -10.400  < 2e-16 ***
## S2            -0.4265     0.1164  -3.664 0.000279 ***
## S4            13.2872     3.5137   3.782 0.000178 ***
## BMI            6.1300     0.6990   8.770  < 2e-16 ***
## BP             1.1309     0.2186   5.173 3.53e-07 ***
## S5            39.0179     6.9215   5.637 3.11e-08 ***
## SEX2         -19.4930     5.7303  -3.402 0.000731 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.54 on 435 degrees of freedom
## Multiple R-squared:  0.5062, Adjusted R-squared:  0.4994
## F-statistic: 74.33 on 6 and 435 DF,  p-value: < 2.2e-16
```

Question 2

Load the dataset

Seeds dataset contains data about four varieties of wheat capturing the hedonic characteristics of each variety of wheat. The data of wheat seeds is gathered from UCI website which is a great dataset repository. The numbers of samples of wheat seeds are 210 from three wheat classes Kama, Rosa and Canadian are collected for classification process. Seven geometrical or morphological features of seeds are considered on the basis of which seeds are classified into three classes of wheat.

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

## [1] 210   8
```

Question 2a

Estimate of covariance matrix using sample covariance method:

$$Q = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T$$

```
seeds_vcmat <- cov(seeds[,1:7])
print(seeds_vcmat)

##                                area   perimeter   compactness
## area                     8.46635078  3.77844320  0.0418225658
## perimeter                3.77844320  1.70552820  0.0163319511
## compactness              0.04182257  0.01633195  0.0005583493
## length_of_kernel         1.22470367  0.56266555  0.0038518256
## width_of_kernel          1.06691136  0.46606493  0.0067977190
## asymmetry_coefficient   -1.00435584 -0.42676598 -0.0117765562
## length_of_kernel_groove  1.23513290  0.57175254  0.0026342068
##                         length_of_kernel width_of_kernel
## area                         1.224703671     1.066911361
## perimeter                    0.562665550     0.466064932
## compactness                  0.003851826     0.006797719
## length_of_kernel             0.196305245     0.143991709
## width_of_kernel              0.143991709     0.142668202
## asymmetry_coefficient       -0.114289956    -0.146542890
## length_of_kernel_groove      0.203125110     0.139068229
##                         asymmetry_coefficient length_of_kernel_groove
## area                             -1.004355845             1.235132905
## perimeter                        -0.426765980             0.571752539
## compactness                      -0.011776556             0.002634207
## length_of_kernel                 -0.114289956             0.203125110
## width_of_kernel                  -0.146542890             0.139068229
## asymmetry_coefficient             2.260684046            -0.008187052
## length_of_kernel_groove          -0.008187052             0.241553081
```

Question 2b

Another method is maximum likelihood estimate:

$$Q = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T$$

Simple cases, where observations are complete, can be dealt with by using the sample covariance matrix. The sample covariance matrix (SCM) is an unbiased and efficient estimator of the covariance matrix. If the random variable has normal distribution, the sample covariance matrix has Wishart distribution and a slightly differently scaled version of it is the maximum likelihood estimate.

If the sample size n is small and the number of considered variables p is large, the above empirical estimators of covariance and correlation are very unstable. Specifically, it is possible to furnish estimators that improve considerably upon the maximum likelihood estimate in terms of mean squared error. As an alternative, many methods have been suggested to improve the estimation of the covariance matrix. All of these approaches rely on the concept of shrinkage. This is implicit in Bayesian methods and in penalized maximum likelihood methods and explicit in the Stein-type shrinkage approach.

Question 2c

i.) A zero entry in the precision matrix (the inverse of the covariance matrix) means the corresponding variables are independent given all the other variables. Finding the covariance matrix that fits the data and has a conveniently large number of zero entries in it's inverse matrix is known as Covariance Selection. Zeros in the inverse covariance matrix are desirable both for computational and conceptual reasons: they indicate conditional independence between variables, making the model smaller. Conditional independence constraints describe the sparsity pattern of the inverse covariance matrix $\sum^{-1}$, zeros showing the conditional independence between variables. So, will choose the maximum likelihood estimate method. Maximizing the log-likelihood with respect to $\sum^{-1}$ leads to the maximum likelihood estimate $\widehat{\sum}^{-1} = S^{-1}$ which isn't usually sparse (here S is the sample covariance obtained from the data). Also when $p > n$, S will be singular and so the maximum likelihood estimate cannot be computed.

ii.) This is the similar case which we had in Part A Q1 (f) where two of the off-diagnol entries in the inverse of the covariance matrix are zero and we calculated S^{-1} to be as close to $\sum^{-1}$

Question 3

Research Question

To build a multiclass classifier for automatically classifying different varieties of seeds into 3 classes of wheat. Manual apporach uses various geometrical or morphological features of seeds to identify the variety of wheat. Mathematical algorithms can be utilized to perform a qualitative research. Investigation will require data collection and analysis, and the methodology for this will include experimenting with 4 different mathematical methods for building a classifier.

Required Dataset

Seeds dataset contains data about 3 varieties of wheat capturing the hedonic characteristics of each variety of wheat. The numbers of samples of wheat seeds are 210 from three wheat classes Kama, Rosa and Canadian are collected for classification process. Seven geometrical or morphological features of seeds are considered on the basis of which seeds can be classified into three classes of wheat.

Data Acquisition

The dataset has been gathered from UCI Machine Learning Repository:

http://archive.ics.uci.edu/ml/datasets/seeds

Research Objective

To build a multiclass classifier for classifying the seven geometrical or morphological features of seeds into 3 classes of wheat.

Machine Learning is widely used in the field of agriculture for differentiating the varieties of various crops and for identifying their quality as well.

A machine vision system is an alternate to the manual inspection, in the field of biological sciences to analyze biological products. To classify the varieties of various food crops and for identifying their quality as well, the machine vision is broadly used in the field of agriculture. Machine algorithms can be used to identify different varieties of wheat seeds to classify them according to their quality.

In biology and agronomy crop seed characteristics are very significant aspects. Machine vision technology is developed to quantify the features, the quality precise examination and graduation of the crop seeds. A novel scheme is presented to extract and quantify some of features having worth biologically.

Methodology

The dataset has been gathered from UCI Machine Learning Repository. Data will be analysed for descriptive statistics, pre-processing and analyzing the univariate and multivariate distribution. The data will be checked to understand the linear relationship among various variables. Four different classification techniques will be used to fit a model with the best performance. LDA, QDA, Multinomial Logistic regression and CART Decision tress. The dataset will be split into training and test datasets which will be used to train the models and test the prediction performance of each of the models. The model performance will be compared to answer our research question of building an optimal classifier for classfying the 3 varieties of wheat.

Descriptive Statistics

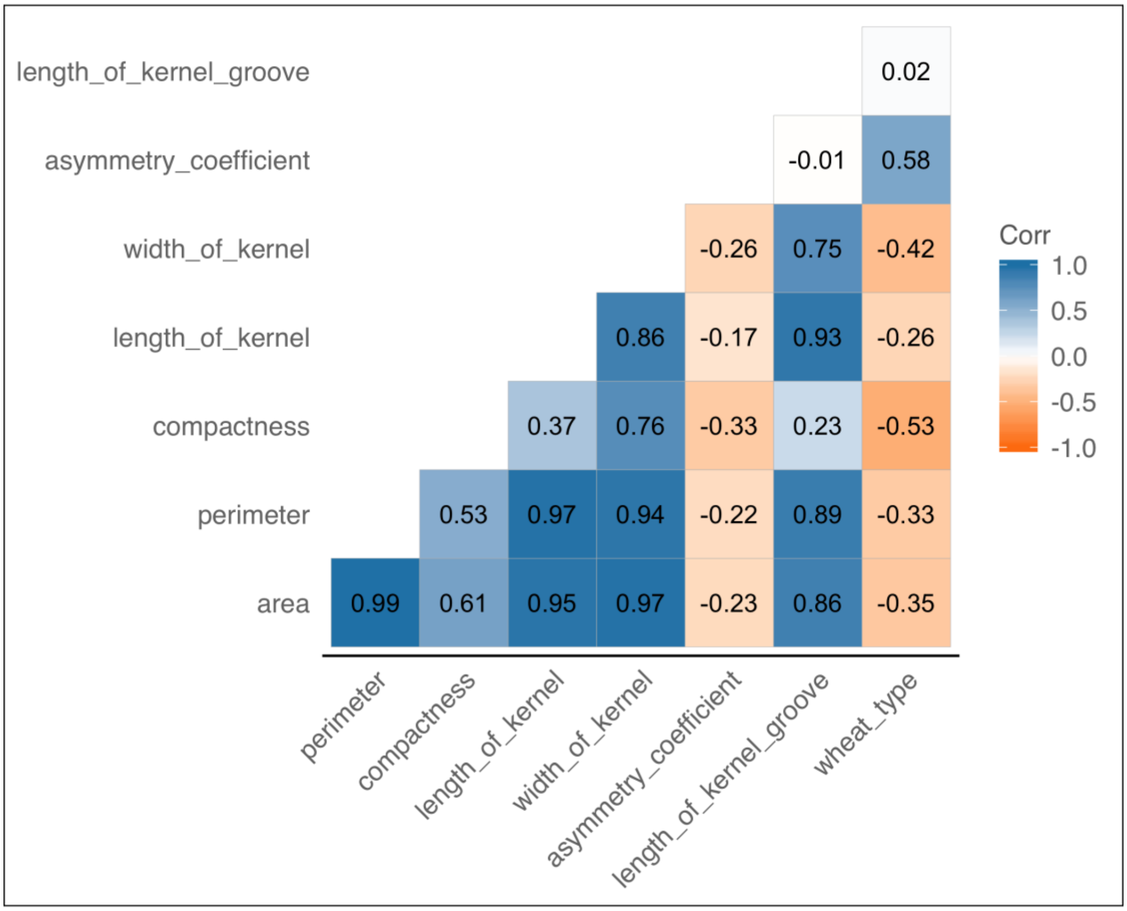
No missing observations identified in the dataset. The data has 210 observation with 7 features (independent variable) and 1 class of seeds variabe (dependent variable).

```
summary(seeds)

##       area         perimeter      compactness     length_of_kernel
##  Min.   :10.59   Min.   :12.41   Min.   :0.8081   Min.   :4.899   
##  1st Qu.:12.27   1st Qu.:13.45   1st Qu.:0.8569   1st Qu.:5.262   
##  Median :14.36   Median :14.32   Median :0.8734   Median :5.524   
##  Mean   :14.85   Mean   :14.56   Mean   :0.8710   Mean   :5.629   
##  3rd Qu.:17.30   3rd Qu.:15.71   3rd Qu.:0.8878   3rd Qu.:5.980   
##  Max.   :21.18   Max.   :17.25   Max.   :0.9183   Max.   :6.675   
##  width_of_kernel asymmetry_coefficient length_of_kernel_groove
##  Min.   :2.630   Min.   :0.7651        Min.   :4.519          
##  1st Qu.:2.944   1st Qu.:2.5615        1st Qu.:5.045          
##  Median :3.237   Median :3.5990        Median :5.223          
##  Mean   :3.259   Mean   :3.7002        Mean   :5.408          
##  3rd Qu.:3.562   3rd Qu.:4.7687        3rd Qu.:5.877          
##  Max.   :4.033   Max.   :8.4560        Max.   :6.550          
##    wheat_type
##  Min.   :1   
##  1st Qu.:1   
##  Median :2   
##  Mean   :2   
##  3rd Qu.:3   
##  Max.   :3   

dim(seeds)

## [1] 210   8
```

Multicollinearity Analysis

The visualization shows that the independent variables are highly correlated with each other which may pose some complexities in building the models. ‘asymmetry_coefficient’ is the only variable that is correlated with the dependent variable ‘wheat_type’. From the chart, it is concluded that there is high linear relationship among independent variables.

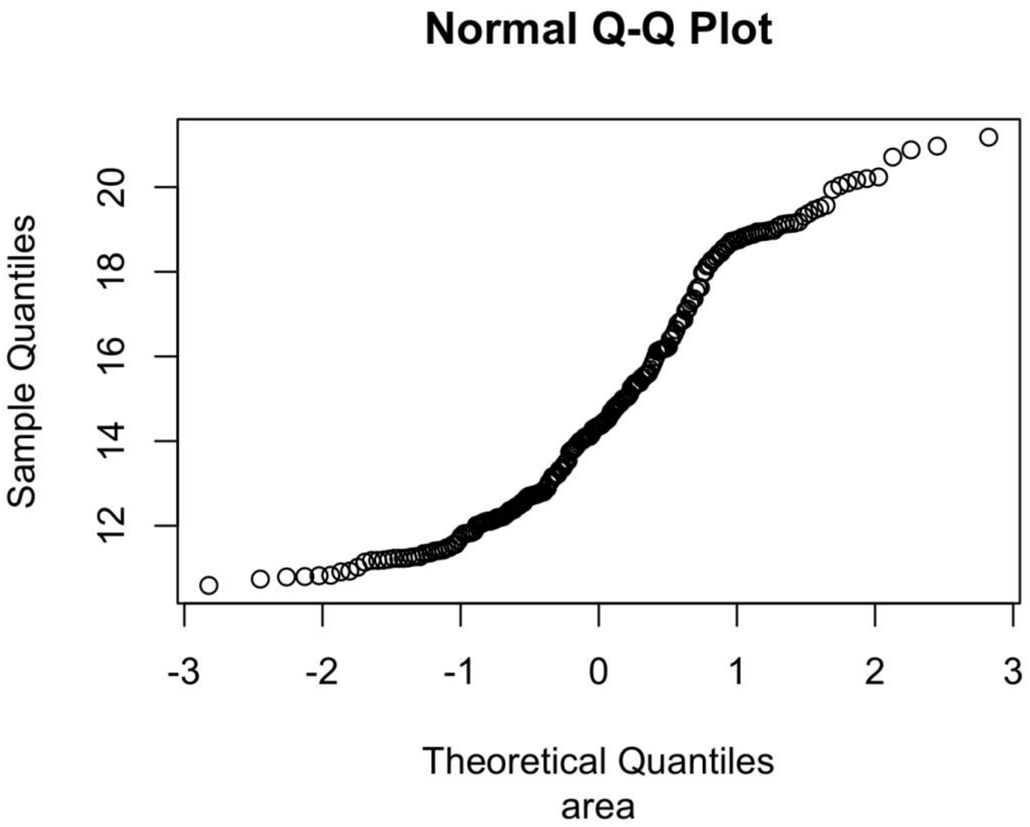
```
#Corr Plot
corr <- cor(seeds, use = "pairwise.complete.obs")

ggcorrplot(corr, hc.order = FALSE, type = "lower",
           ggtheme = ggthemes::theme_gdocs,
           colors = c("#ff7f0e", "white", "#1f83b4"),
           lab = TRUE)+
           theme(panel.grid.major=element_blank())
```



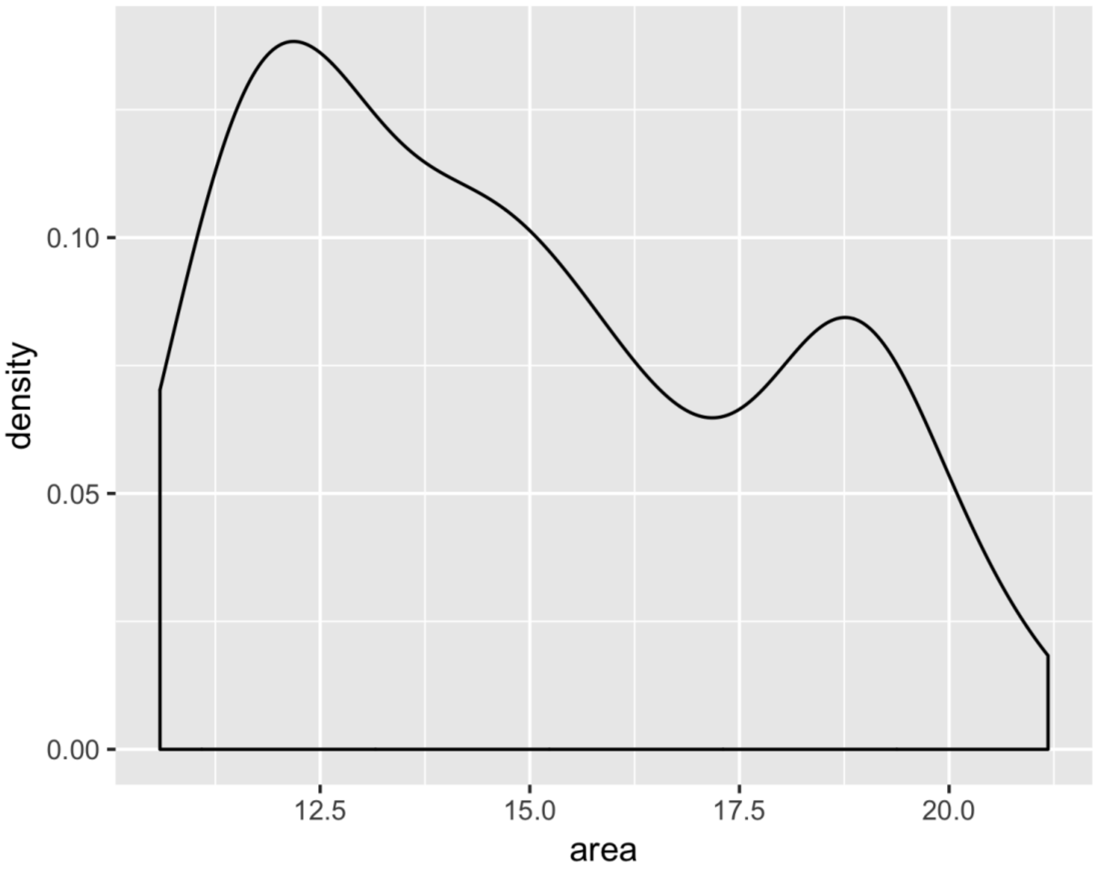
Univariate Analysis

Shapiro Wilk test rejects the null hypothesis all variables being univariate normal. Hence, none of the variables are normally distributed.

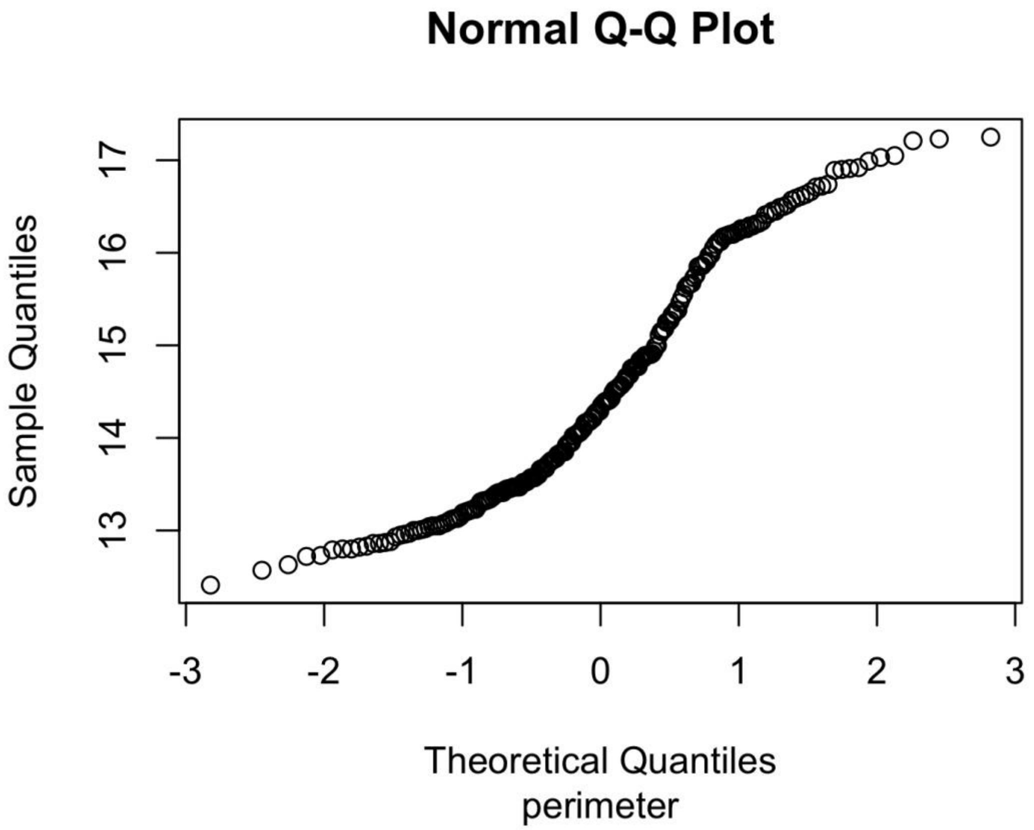
```
options(scipen = 999)
# Normal Q-Q plot for area
qqnorm(seeds$area, sub = colnames(seeds)[1])
```



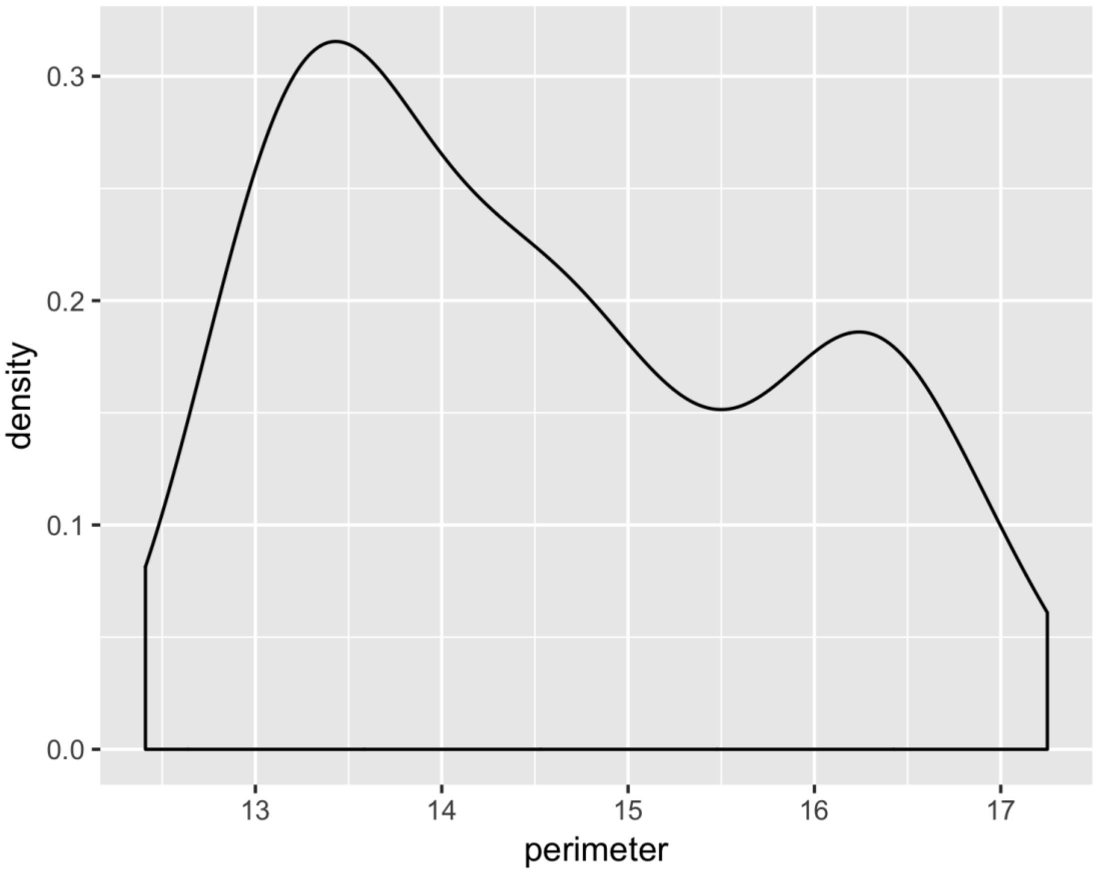
```
ggplot(seeds, aes(x=area)) + geom_density()
```



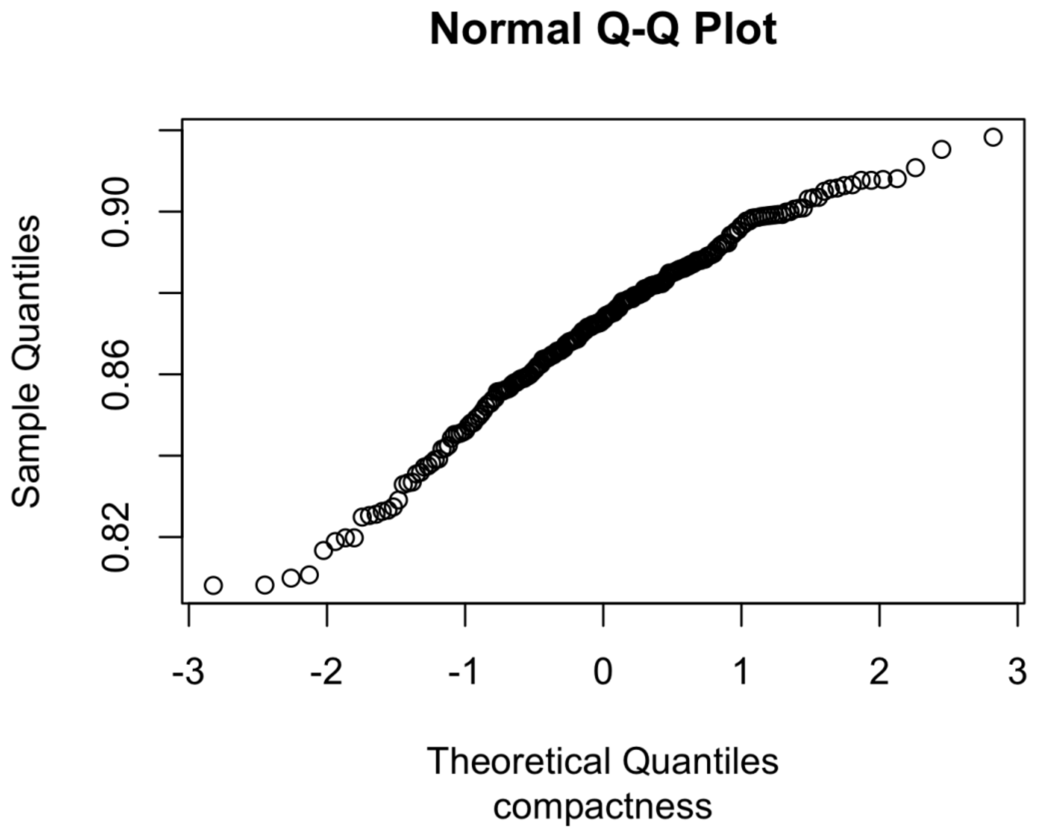
```
# Shapiro Wilk test for variable area
shapiro.test(seeds$area)

##
##  Shapiro-Wilk normality test
##
## data:  seeds$area
## W = 0.93259, p-value = 0.00000002948

# Normal Q-Q plot for perimeter
qqnorm(seeds$perimeter, sub = colnames(seeds)[2])
```



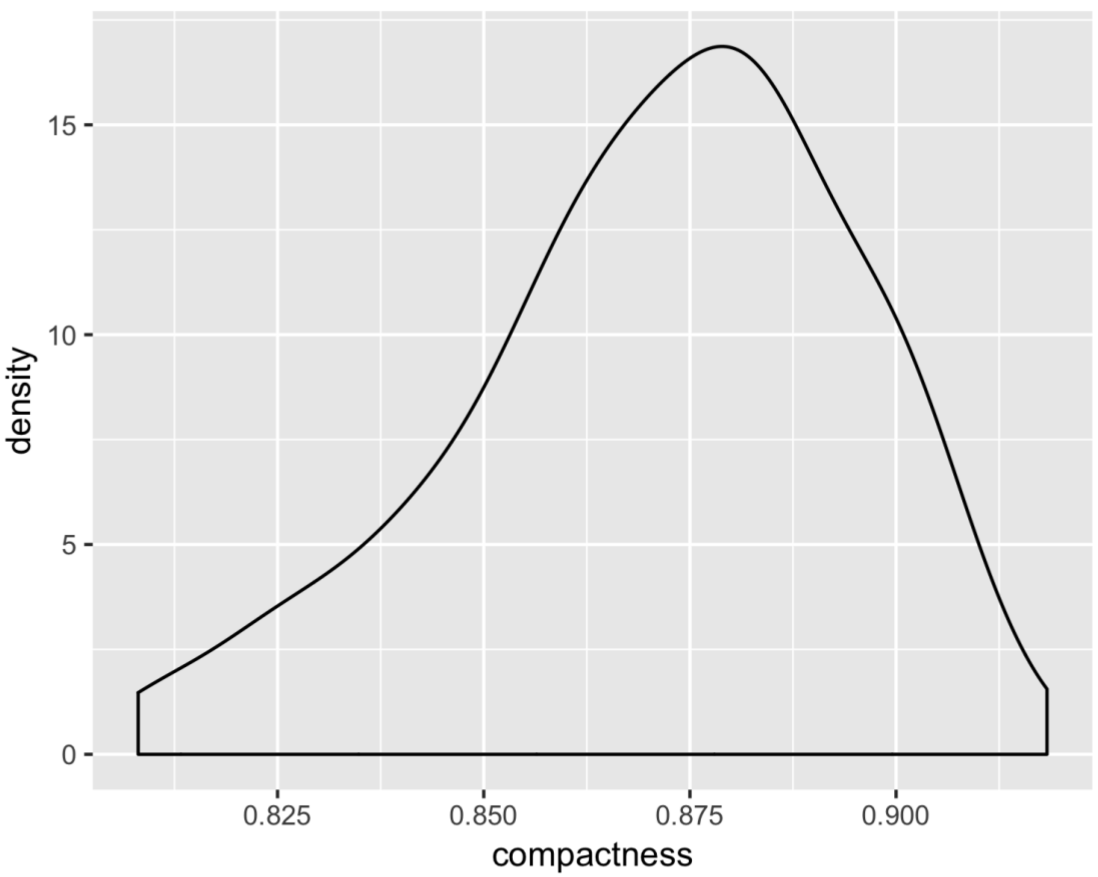
```
ggplot(seeds, aes(x=perimeter)) + geom_density()
```



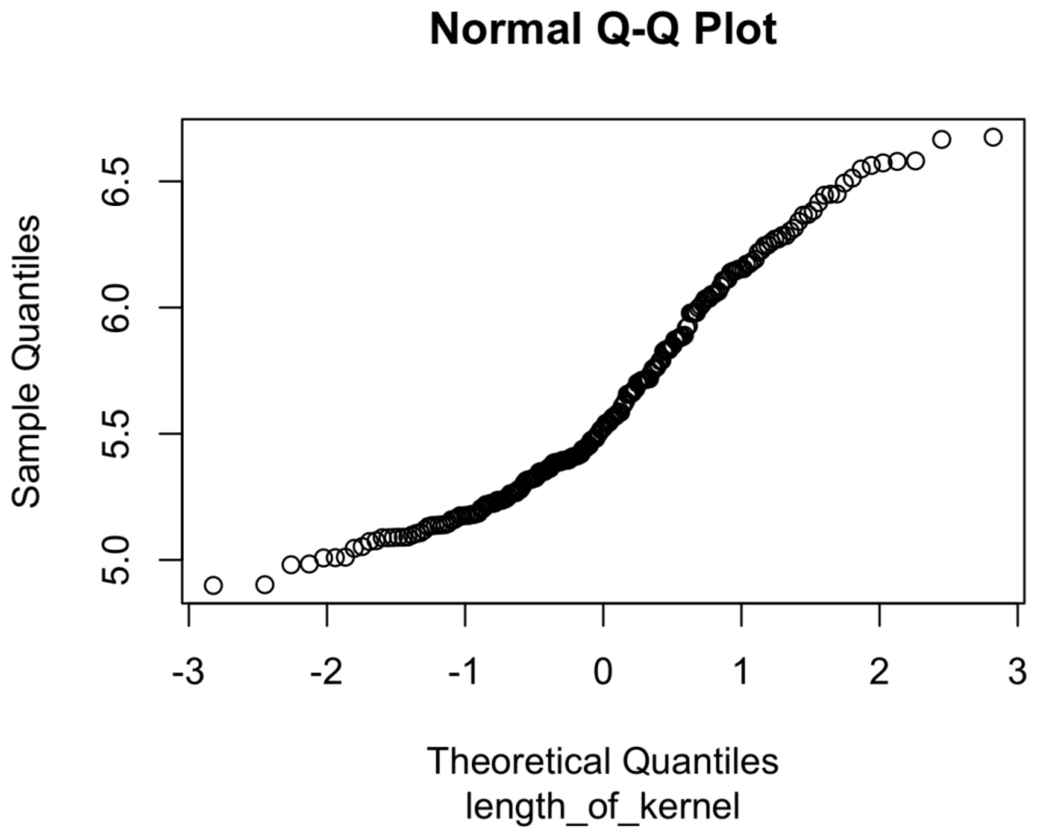
```
# Shapiro Wilk test for variable perimeter
shapiro.test(seeds$perimeter)

##
##  Shapiro-Wilk normality test
##
## data:  seeds$perimeter
## W = 0.93616, p-value = 0.00000005902

# Normal Q-Q plot for compactness
qqnorm(seeds$compactness, sub = colnames(seeds)[3])
```



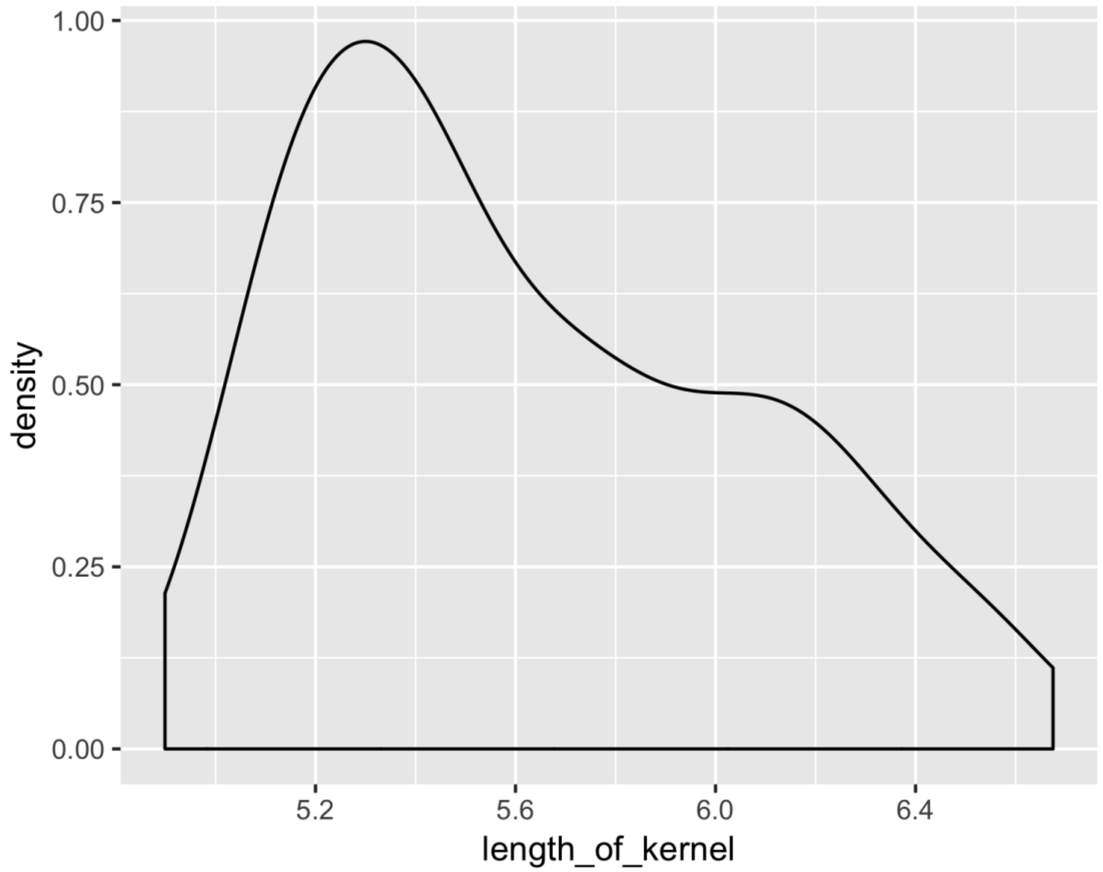
```
ggplot(seeds, aes(x=compactness)) + geom_density()
```



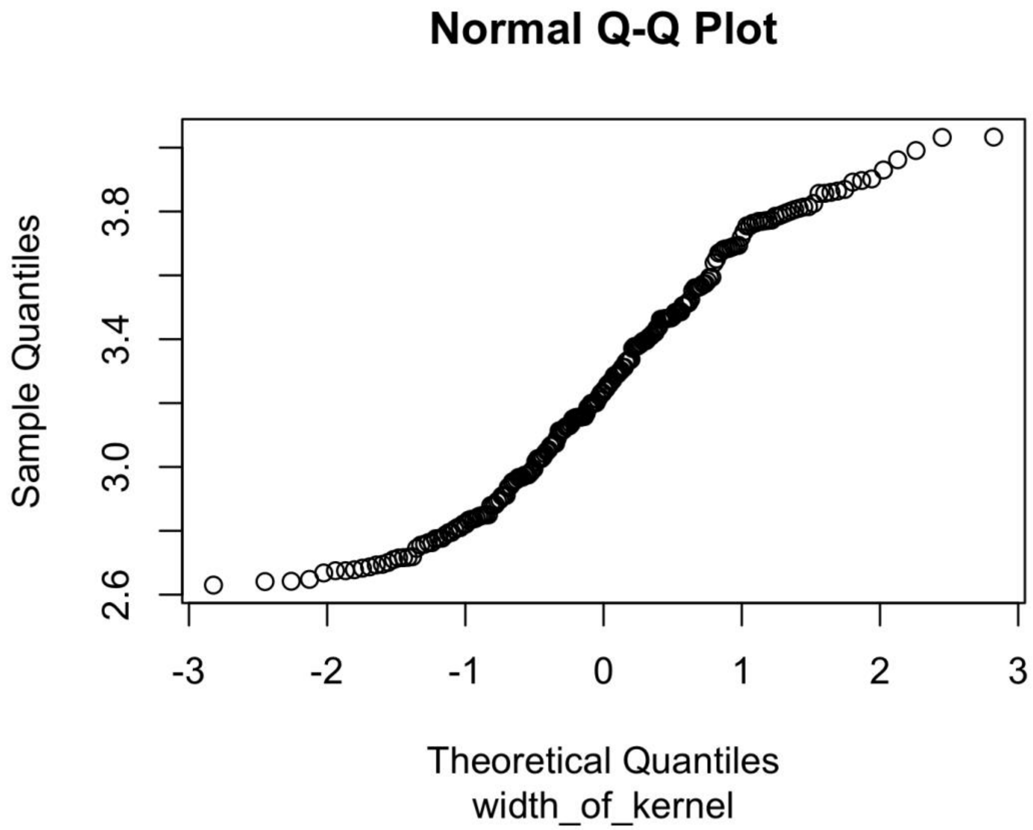
```
# Shapiro Wilk test for variable compactness
shapiro.test(seeds$compactness)

##
##  Shapiro-Wilk normality test
##
## data:  seeds$compactness
## W = 0.97304, p-value = 0.0004696

# Normal Q-Q plot for length_of_kernel
qqnorm(seeds$length_of_kernel, sub = colnames(seeds)[4])
```



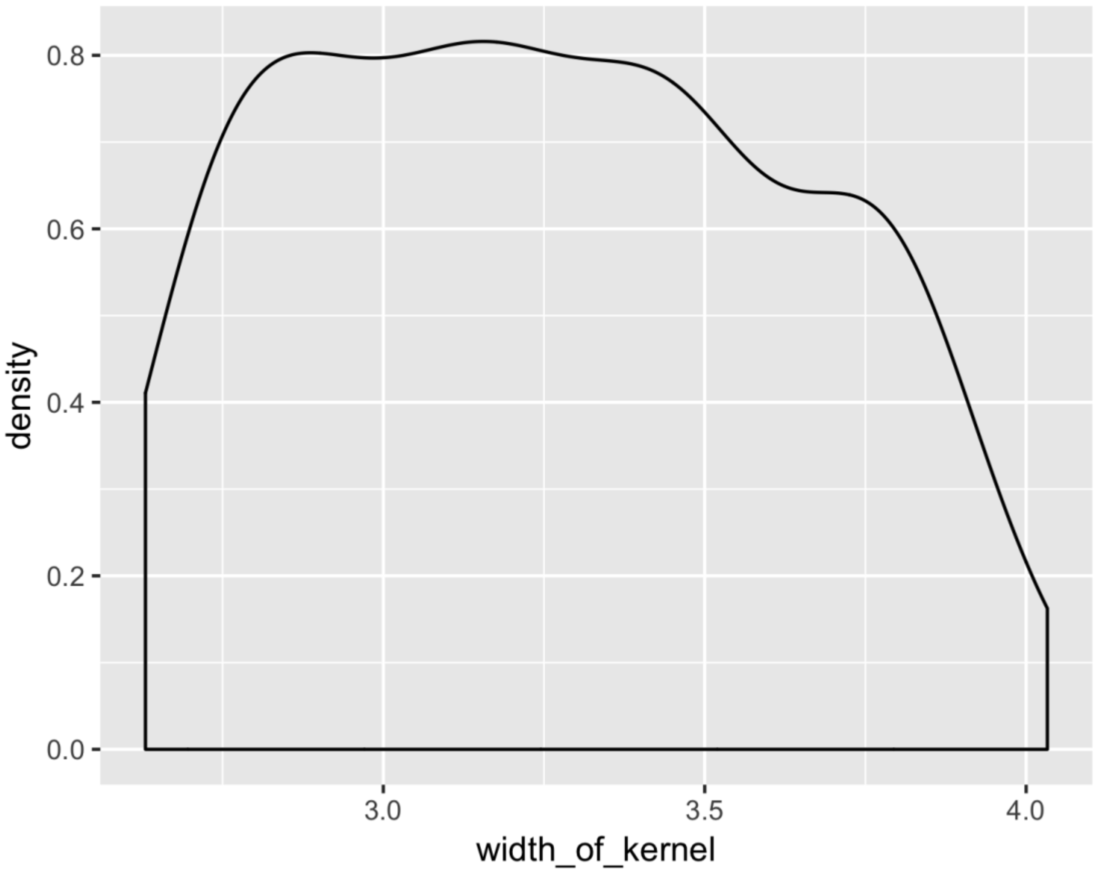
```
ggplot(seeds, aes(x=length_of_kernel)) + geom_density()
```



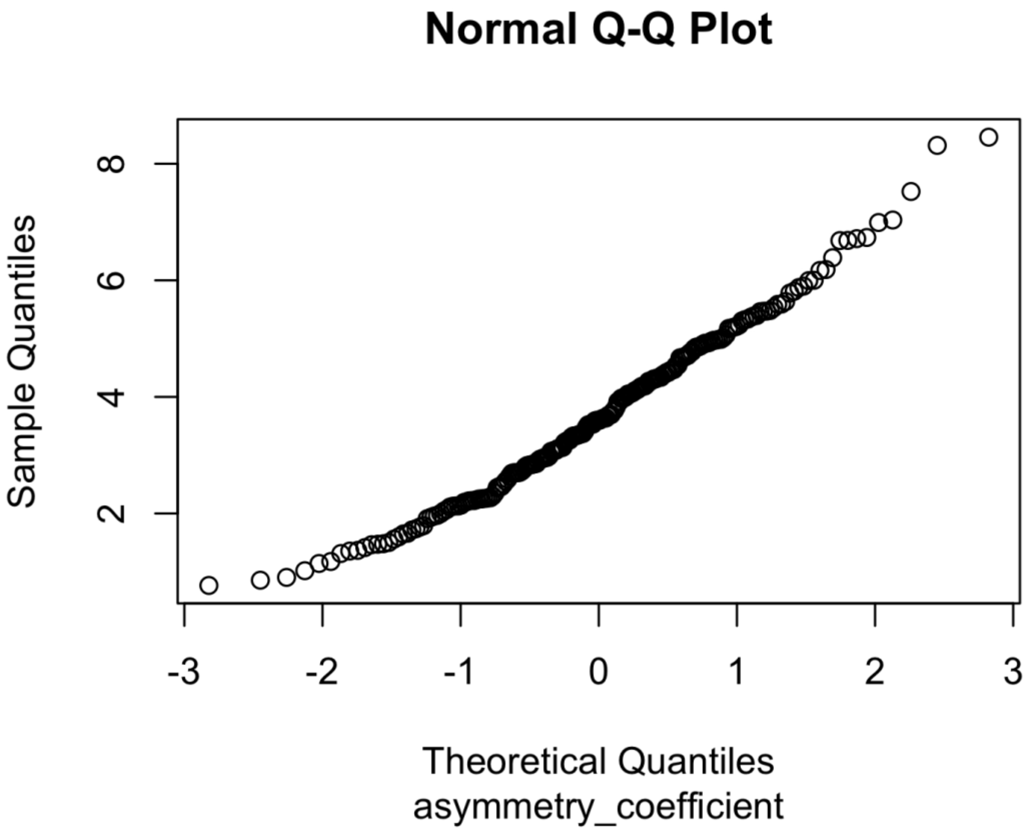
```
# Shapiro Wilk test for variable length_of_kernel
shapiro.test(seeds$length_of_kernel)

##
##  Shapiro-Wilk normality test
##
## data:  seeds$length_of_kernel
## W = 0.9438, p-value = 0.0000002828

# Normal Q-Q plot for width_of_kernel
qqnorm(seeds$width_of_kernel, sub = colnames(seeds)[5])
```



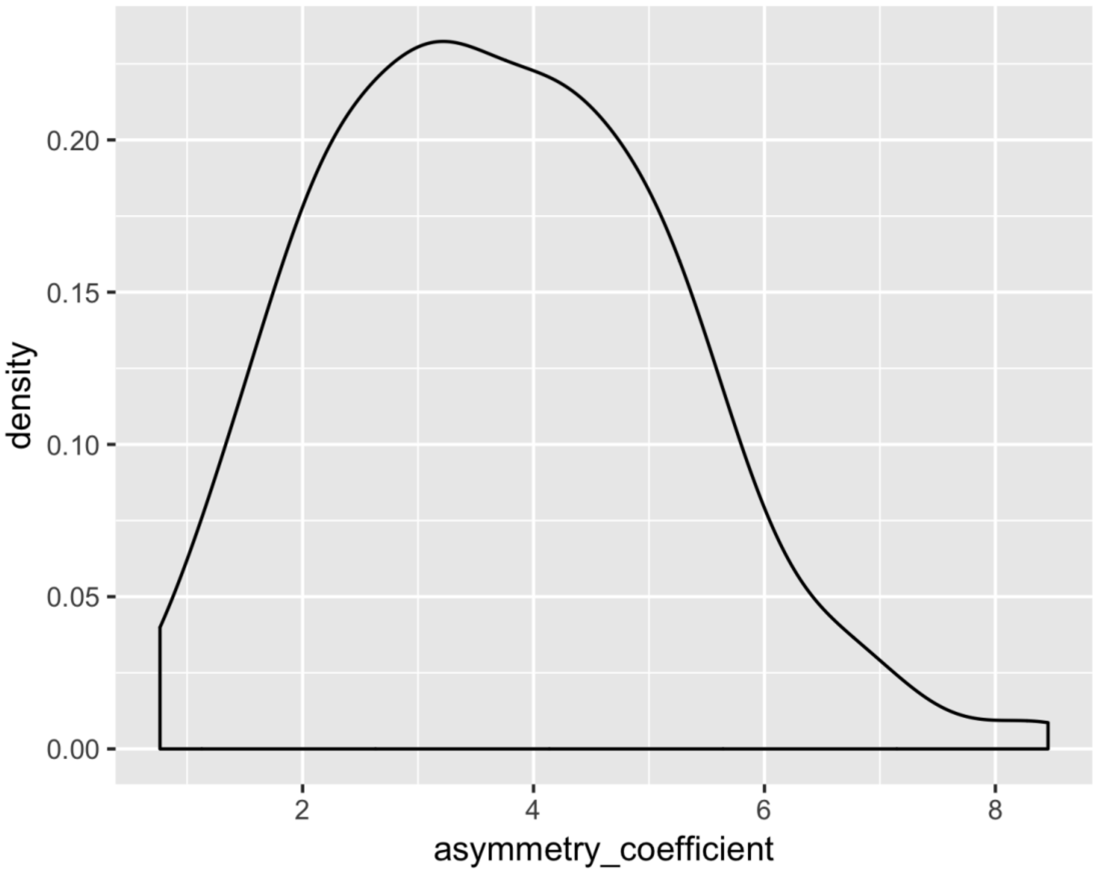
```
ggplot(seeds, aes(x=width_of_kernel)) + geom_density()
```



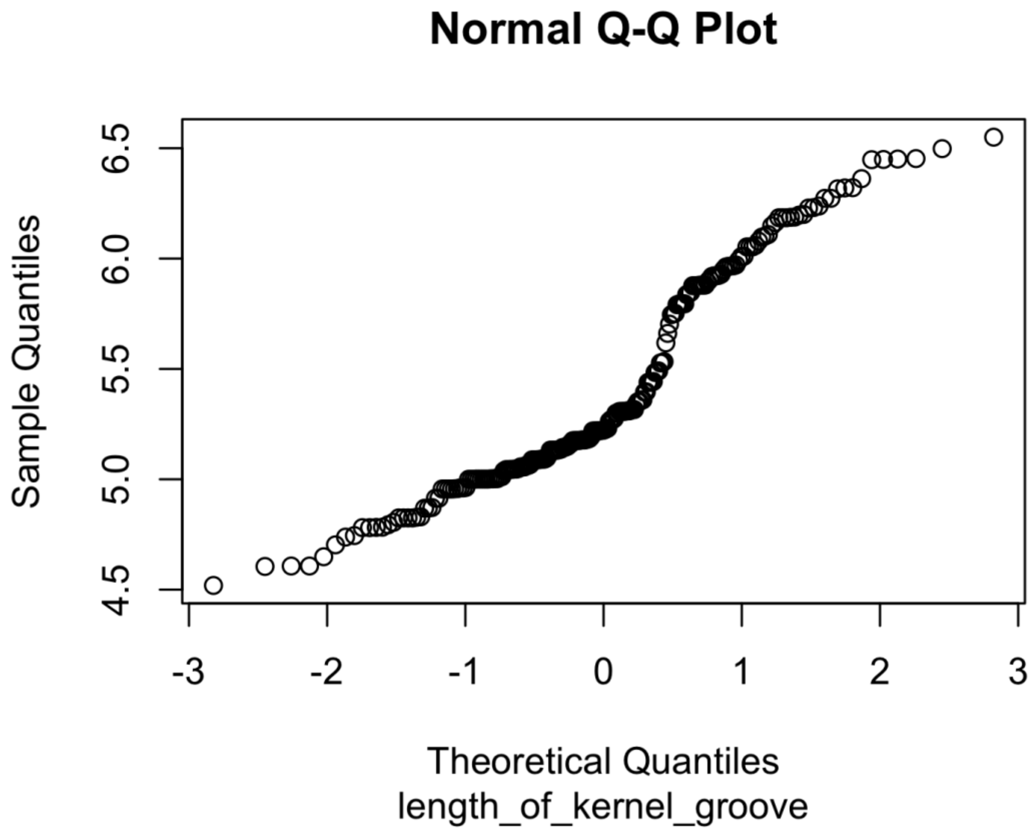
```
# Shapiro Wilk test for variable width_of_kernel
shapiro.test(seeds$width_of_kernel)

##
##  Shapiro-Wilk normality test
##
## data:  seeds$width_of_kernel
## W = 0.96062, p-value = 0.00001445

# Normal Q-Q plot for asymmetry_coefficient
qqnorm(seeds$asymmetry_coefficient, sub = colnames(seeds)[6])
```



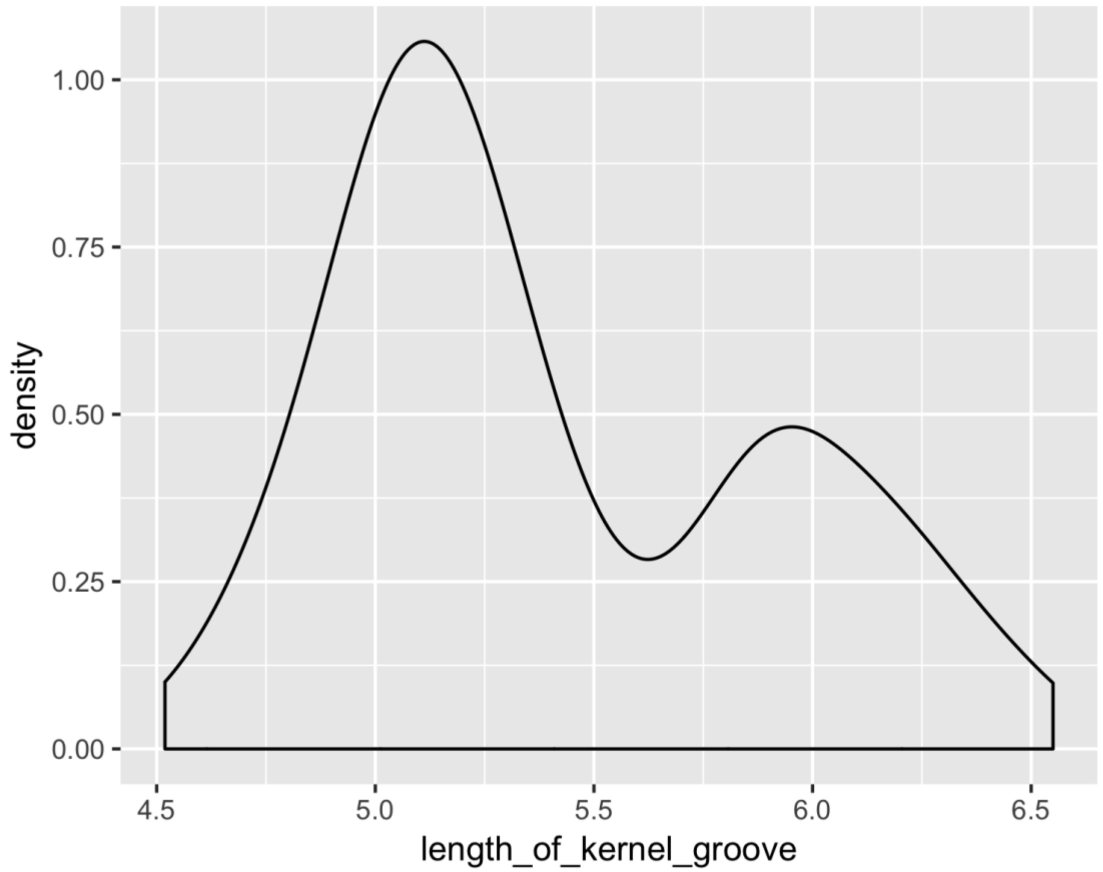
```
ggplot(seeds, aes(x=asymmetry_coefficient)) + geom_density()
```



```
# Shapiro Wilk test for variable asymmetry_coefficient
shapiro.test(seeds$asymmetry_coefficient)

##
##  Shapiro-Wilk normality test
##
## data:  seeds$asymmetry_coefficient
## W = 0.98362, p-value = 0.01544

# Normal Q-Q plot for length_of_kernel_groove
qqnorm(seeds$length_of_kernel_groove, sub = colnames(seeds)[7])
```



```
ggplot(seeds, aes(x=length_of_kernel_groove)) + geom_density()
```

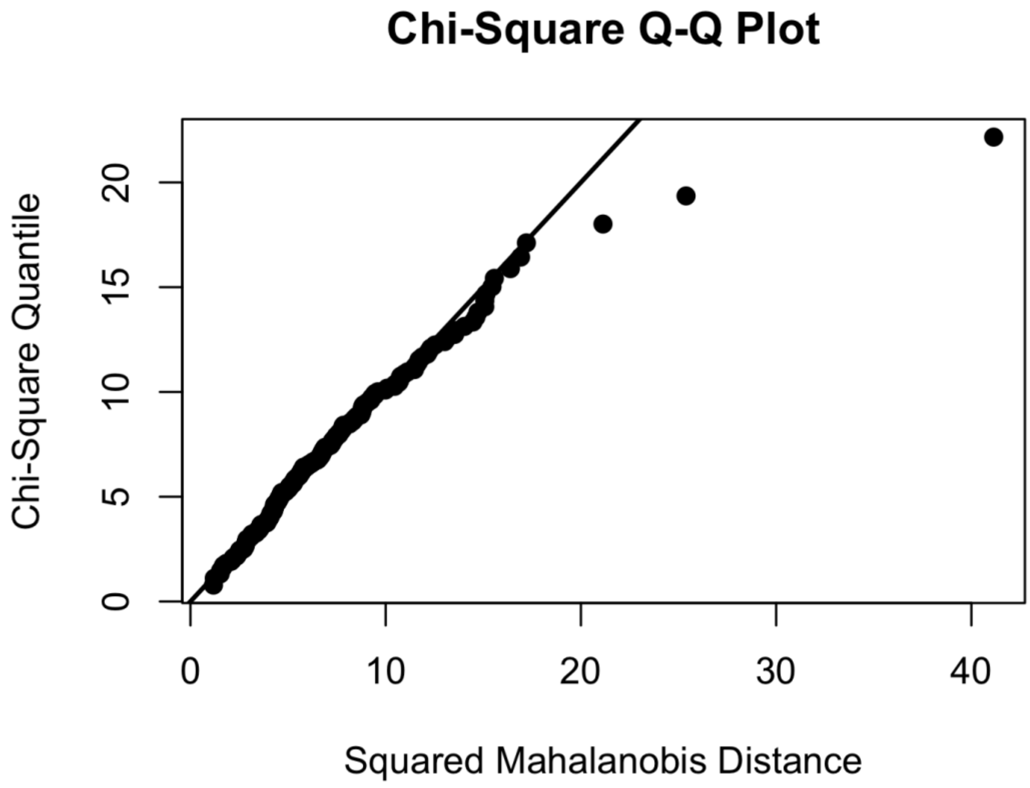


```
# Shapiro Wilk test for variable length_of_kernel_groove
shapiro.test(seeds$length_of_kernel_groove)

##
##  Shapiro-Wilk normality test
##
## data:  seeds$length_of_kernel_groove
## W = 0.92494, p-value = 0.000000007142
```

Multivariate Normality test

The data is not multivariate normal as per the royston test.

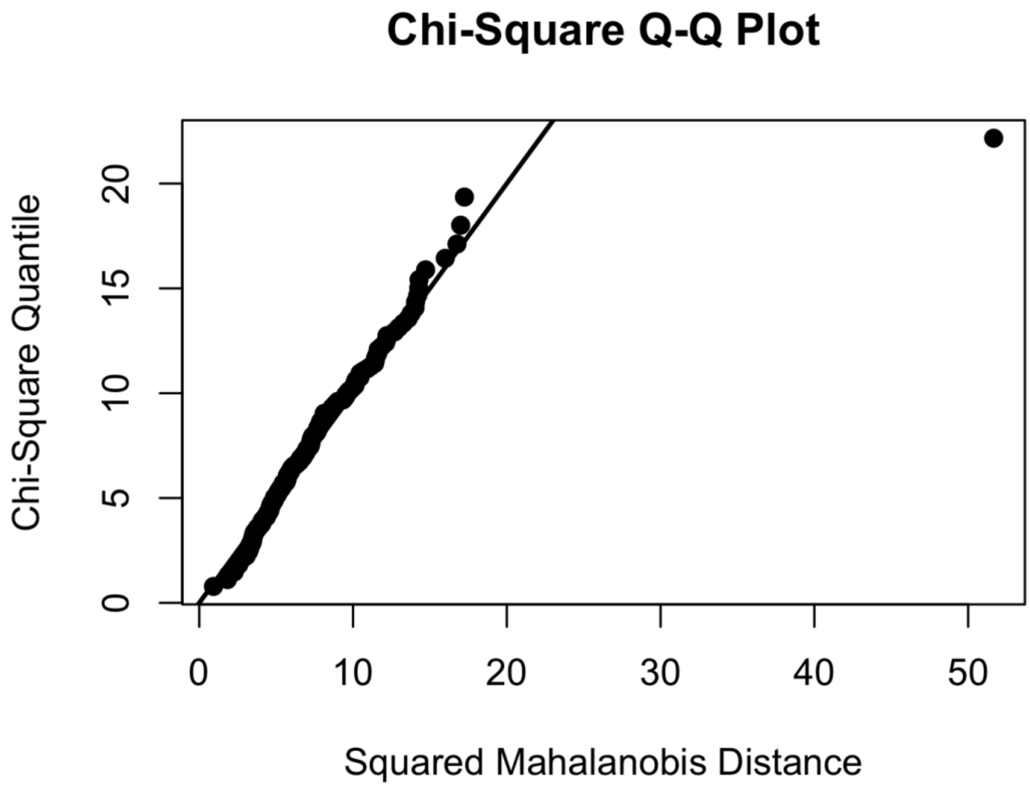
```
mvtest <- mvn(seeds[,1:7], mvnTest='royston', multivariatePlot='qq')
```



```
mvtest$multivariateNormality

##      Test        H                p value MVN
## 1 Royston 64.13045 0.00000000000006038104  NO

mvtest$univariateNormality

##           Test                Variable Statistic   p value Normality
## 1 Shapiro-Wilk          area              0.9326  <0.001      NO
## 2 Shapiro-Wilk        perimeter           0.9362  <0.001      NO
## 3 Shapiro-Wilk       compactness          0.9730  0.0005      NO
## 4 Shapiro-Wilk    length_of_kernel        0.9438  <0.001      NO
## 5 Shapiro-Wilk     width_of_kernel        0.9606  <0.001      NO
## 6 Shapiro-Wilk  asymmetry_coefficient     0.9836  0.0154      NO
## 7 Shapiro-Wilk length_of_kernel_groove    0.9249  <0.001      NO

mvtest$Descriptives

##                           n       Mean    Std.Dev    Median     Min     Max
## area                    210 14.8475238 2.90969943 14.35500 10.5900 21.1800
## perimeter               210 14.5592857 1.30595873 14.32000 12.4100 17.2500
## compactness             210  0.8709986 0.02362942  0.87345  0.8081  0.9183
## length_of_kernel        210  5.6285333 0.44306348  5.52350  4.8990  6.6750
## width_of_kernel         210  3.2586048 0.37771444  3.23700  2.6300  4.0330
## asymmetry_coefficient   210  3.7002010 1.50355713  3.59900  0.7651  8.4560
## length_of_kernel_groove 210  5.4080714 0.49148050  5.22300  4.5190  6.5500
##                             25th      75th        Skew    Kurtosis
## area                    12.27000 17.305000  0.3941946 -1.1052328
## perimeter               13.45000 15.715000  0.3810678 -1.1269312
## compactness              0.85690  0.887775 -0.5302931 -0.1923636
## length_of_kernel         5.26225  5.979750  0.5179985 -0.8164443
## width_of_kernel          2.94400  3.561750  0.1324647 -1.1182220
## asymmetry_coefficient    2.56150  4.768750  0.3959475 -0.1210795
## length_of_kernel_groove  5.04500  5.877000  0.5538958 -0.8697756
```

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
```

```
##      Test        H              p value MVN
## 1 Royston 57.67264 0.000000000001674206  NO
```

```
mvtest_trans$univariateNormality
```

```
##           Test                   Variable Statistic   p value Normality
## 1 Shapiro-Wilk          area^0.02             0.9450  <0.001      NO
## 2 Shapiro-Wilk       perimeter^0.05           0.9428  <0.001      NO
## 3 Shapiro-Wilk      compactness^0.26          0.9697  0.0002      NO
## 4 Shapiro-Wilk   length_of_kernel^-0.24       0.9540  <0.001      NO
## 5 Shapiro-Wilk    width_of_kernel^-0.31       0.9612  <0.001      NO
## 6 Shapiro-Wilk asymmetry_coefficient^0.54     0.9941  0.5774      YES
## 7 Shapiro-Wilk length_of_kernel_groove^0.1    0.9354  <0.001      NO
```

```
mvtest_trans$Descriptives
```

```
##                               n      Mean    Std.Dev     Median
## area^0.02                   210  2.7671146 0.20630365  2.7506817
## perimeter^0.05              210  2.8574139 0.10111683  2.8428006
## compactness^0.26            210 -0.1359227 0.02632398 -0.1329469
## length_of_kernel^-0.24      210  1.4141363 0.05121080  1.4040953
## width_of_kernel^-0.31       210  0.9840812 0.08095530  0.9855554
## asymmetry_coefficient^0.54  210  1.8217361 0.84214289  1.8470037
## length_of_kernel_groove^0.1 210  1.8396677 0.10631577  1.8025705
##                                    Min         Max      25th       75th
## area^0.02                    2.4276846  3.16712622  2.583743  2.9503021
## perimeter^0.05               2.6803006  3.05582763  2.771514  2.9489351
## compactness^0.26            -0.2072624 -0.08429163 -0.151368 -0.1172097
## length_of_kernel^-0.24       1.3230648  1.52736424  1.371650  1.4564393
## width_of_kernel^-0.31        0.8362288  1.13358251  0.918514  1.0511577
## asymmetry_coefficient^0.54  -0.2492668  4.01729291  1.226098  2.4545992
## length_of_kernel_groove^0.1  1.6321132  2.07427491  1.761514  1.9433641
##                                    Skew   Kurtosis
## area^0.02                    0.18426120 -1.2312979
## perimeter^0.05               0.28789644 -1.1876224
## compactness^0.26            -0.57712885 -0.1248961
## length_of_kernel^-0.24       0.38694406 -0.9553874
## width_of_kernel^-0.31       -0.06443810 -1.1457415
## asymmetry_coefficient^0.54  -0.07441285 -0.4131909
## length_of_kernel_groove^0.1  0.45214139 -0.9401847
```

Data Preprocessing

Split the dataset in to train and test datasets. For multionomial regression, we need to create 3 different response variables to denote the three levels of wheat categories.

```
# Convert the dependent variable to a factor
seeds[, wheat_type:= as.factor(wheat_type)]

seeds[, wheat_type_1:=ifelse(wheat_type == 1, 1, 0)]
seeds[, wheat_type_2:=ifelse(wheat_type == 2, 1, 0)]
seeds[, wheat_type_3:=ifelse(wheat_type == 3, 1, 0)]

seeds_train <- seeds[,.SD[1:55], by = list(wheat_type)]
seeds_test <- seeds[,.SD[56:70], by = list(wheat_type)]

seeds_train[, .N, by = list(wheat_type)]
```

```
##    wheat_type  N
## 1:          1 55
## 2:          2 55
## 3:          3 55
```

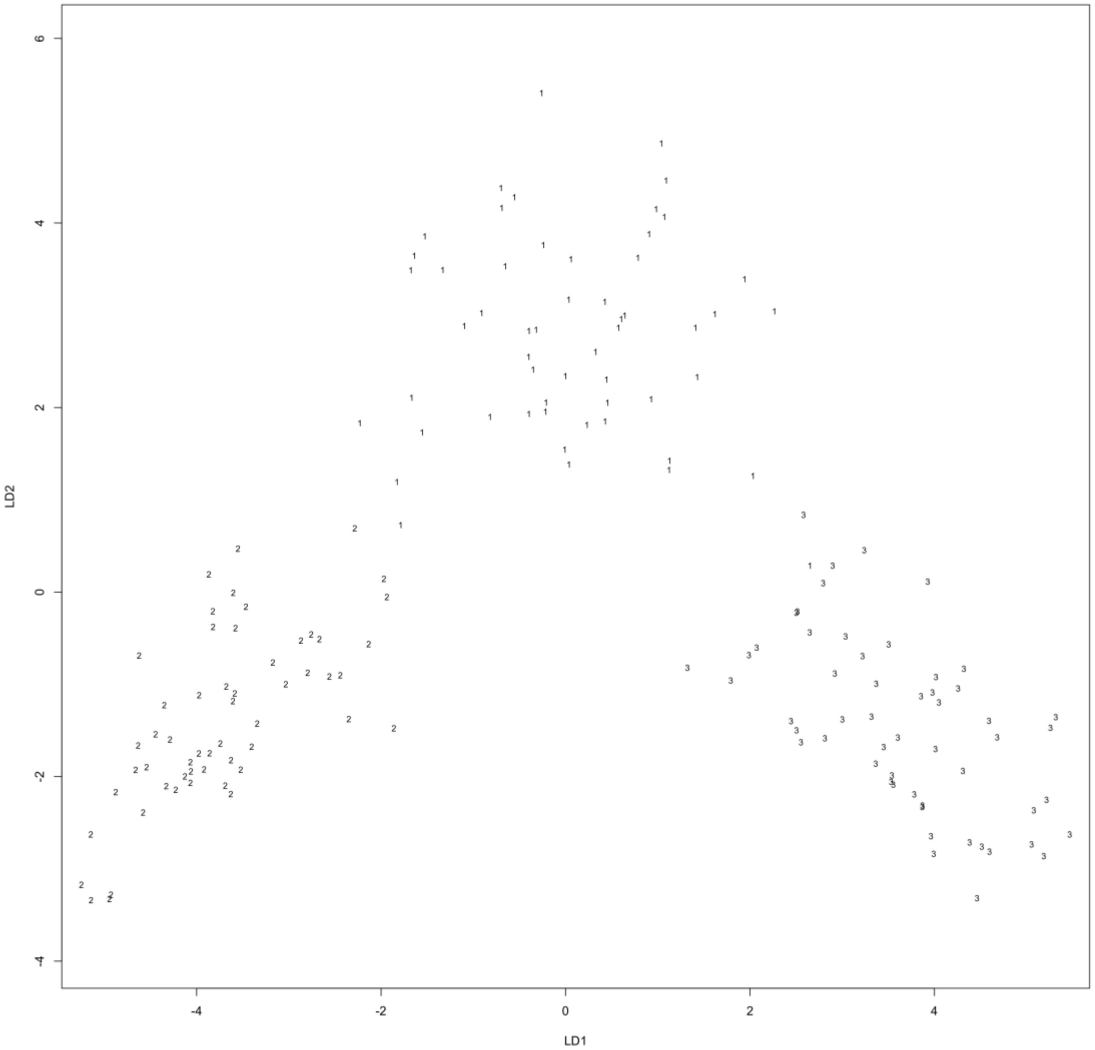
```
seeds_test[, .N, by = list(wheat_type)]
```

```
##    wheat_type  N
## 1:          1 15
## 2:          2 15
## 3:          3 15
```

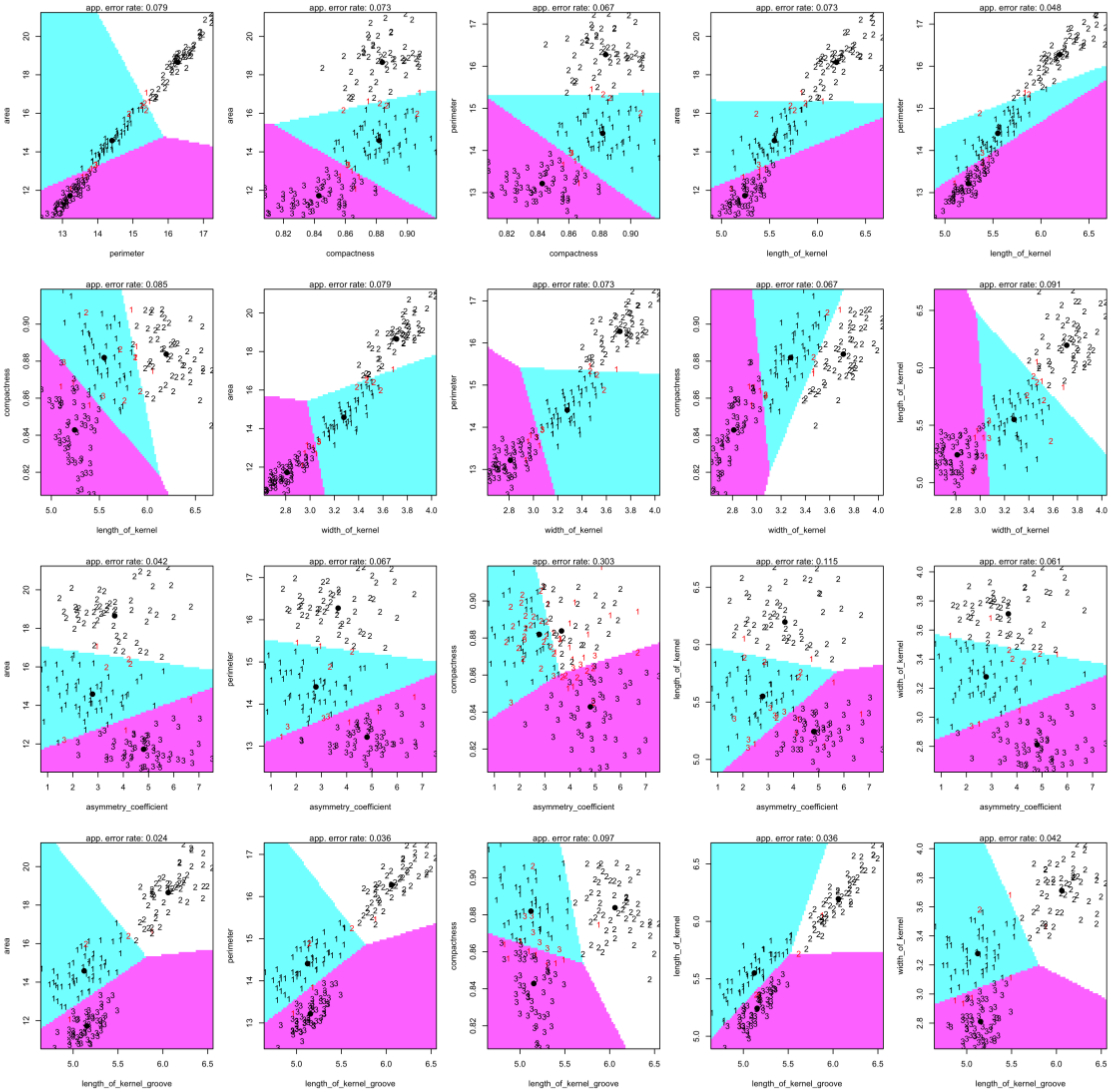
LDA

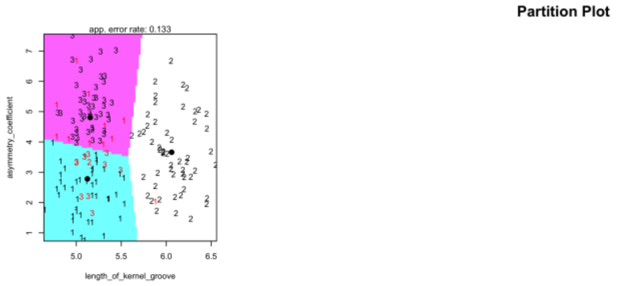
Performing a Linear discriminant analysis on the dataset. Looking at the chart, we can see very few misclassifications on the test set. The error rate is quite low being 18% on the test set, which is very good.

```
#######   LDA
model_lda <- lda(wheat_type ~ area+perimeter+compactness+length_of_kernel+wid
th_of_kernel+asymmetry_coefficient+length_of_kernel_groove, data=seeds_train)
plot(model_lda)
```



```
seeds_test$lda_predict<-predict(model_lda,seeds_test[,2:8])$class

partimat(as.factor(wheat_type) ~ area+perimeter+compactness+length_of_kernel+
width_of_kernel+asymmetry_coefficient+length_of_kernel_groove, data=seeds_tra
in,method="lda")
```


app. error rate: 0.079
area
perimeter
app. error rate: 0.073
area
compactness
app. error rate: 0.067
perimeter
compactness
app. error rate: 0.073
area
length_of_kernel
app. error rate: 0.048
perimeter
length_of_kernel
app. error rate: 0.085
compactness
length_of_kernel
app. error rate: 0.079
area
width_of_kernel
app. error rate: 0.073
perimeter
width_of_kernel
app. error rate: 0.067
compactness
width_of_kernel
app. error rate: 0.091
length_of_kernel
width_of_kernel
app. error rate: 0.042
area
asymmetry_coefficient
app. error rate: 0.067
perimeter
asymmetry_coefficient
app. error rate: 0.303
compactness
asymmetry_coefficient
app. error rate: 0.115
length_of_kernel
asymmetry_coefficient
app. error rate: 0.061
width_of_kernel
asymmetry_coefficient
app. error rate: 0.024
area
length_of_kernel_groove
app. error rate: 0.036
perimeter
length_of_kernel_groove
app. error rate: 0.097
compactness
length_of_kernel_groove
app. error rate: 0.036
length_of_kernel
length_of_kernel_groove
app. error rate: 0.042
width_of_kernel
length_of_kernel_groove

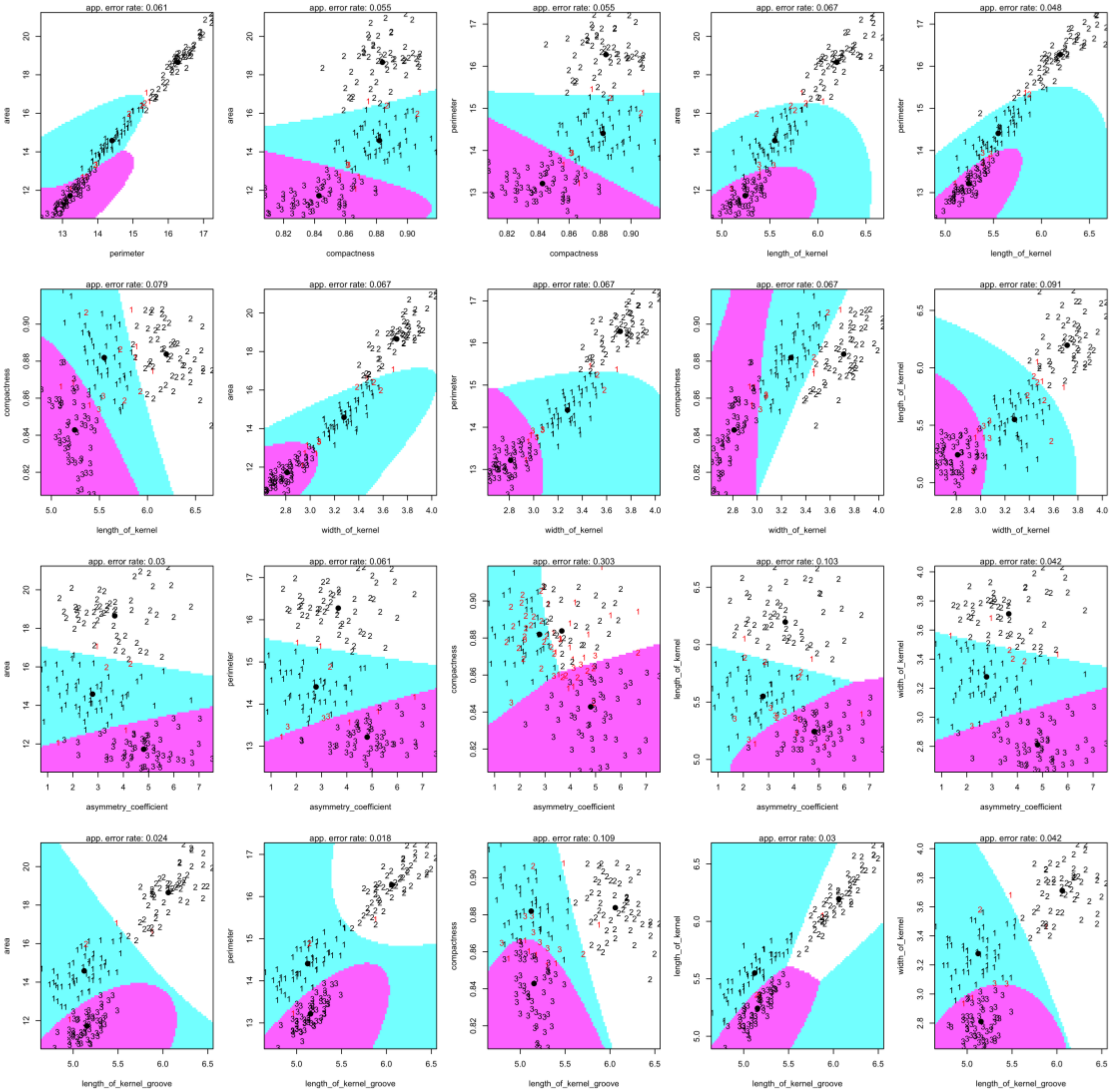


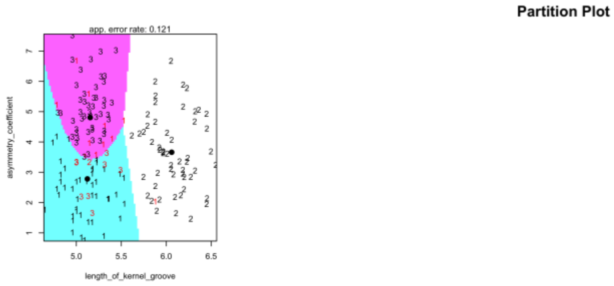
```
# LDA
#confusion matrix
table(seeds_test$wheat_type,seeds_test$lda_predict)

##
##       1  2  3
##   1 13  0  2
##   2  1 14  0
##   3  5  0 10

# error rate
mean(seeds_test$lda_predict != seeds_test$wheat_type)

## [1] 0.1777778
```

QDA

Performing a Quadratic discriminant analysis on the dataset. Looking at the chart, we can see more misclassifications on the test set than LDA. The error rate is higher being 27% on the test set. Linear discrimination works better on this dataset than quadratic in separating the 3 classes of wheat.

```
######## QDA
model_qda<-qda(wheat_type ~ area+perimeter+compactness+length_of_kernel+width
_of_kernel+asymmetry_coefficient+length_of_kernel_groove, data=seeds_train)
model_qda

## Call:
## qda(wheat_type ~ area + perimeter + compactness + length_of_kernel +
##     width_of_kernel + asymmetry_coefficient + length_of_kernel_groove,
##     data = seeds_train)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##       area perimeter compactness length_of_kernel width_of_kernel
## 1 14.58491  14.40764   0.8819309         5.550309        3.278236
## 2 18.65145  16.27600   0.8838018         6.196527        3.711400
## 3 11.72309  13.21582   0.8428309         5.242709        2.810673
##   asymmetry_coefficient length_of_kernel_groove
## 1              2.780713                5.122764
## 2              3.660345                6.059491
## 3              4.805891                5.153982

seeds_test$qda_predict<-predict(model_qda,seeds_test[,2:8])$class

partimat(wheat_type ~ area+perimeter+compactness+length_of_kernel+width_of_ke
rnel+asymmetry_coefficient+length_of_kernel_groove, data=seeds_train, method=
"qda")
```


app. error rate: 0.061
area
perimeter
app. error rate: 0.055
area
compactness
app. error rate: 0.055
perimeter
compactness
app. error rate: 0.067
area
length_of_kernel
app. error rate: 0.048
perimeter
length_of_kernel
app. error rate: 0.079
compactness
length_of_kernel
app. error rate: 0.067
area
width_of_kernel
app. error rate: 0.067
perimeter
width_of_kernel
app. error rate: 0.067
compactness
width_of_kernel
app. error rate: 0.091
length_of_kernel
width_of_kernel
app. error rate: 0.03
area
asymmetry_coefficient
app. error rate: 0.061
perimeter
asymmetry_coefficient
app. error rate: 0.303
compactness
asymmetry_coefficient
app. error rate: 0.103
length_of_kernel
asymmetry_coefficient
app. error rate: 0.042
width_of_kernel
asymmetry_coefficient
app. error rate: 0.024
area
length_of_kernel_groove
app. error rate: 0.018
perimeter
length_of_kernel_groove
app. error rate: 0.109
compactness
length_of_kernel_groove
app. error rate: 0.03
length_of_kernel
length_of_kernel_groove
app. error rate: 0.042
width_of_kernel
length_of_kernel_groove



```
# QDA
#confusion matrix
table(seeds_test$wheat_type,seeds_test$qda_predict)

##
##      1  2  3
##   1 12  0  3
##   2  5 10  0
##   3  4  0 11

# error rate
mean(seeds_test$qda_predict != seeds_test$wheat_type)

## [1] 0.2666667
```

Multinomial logistic

Residual deviance of the model is low which indicates a good fit. The error rate is 0.13 on the test dataset, which is the best so far.

```
model_mnl<-vglm(formula = cbind(wheat_type_1,wheat_type_2,wheat_type_3) ~ are
a+perimeter+compactness+length_of_kernel+width_of_kernel+asymmetry_coefficien
t+length_of_kernel_groove, family = multinomial, data = seeds_train)
summary(model_mnl)
```

```
##
## Call:
## vglm(formula = cbind(wheat_type_1, wheat_type_2, wheat_type_3) ~
##     area + perimeter + compactness + length_of_kernel + width_of_kernel +
##         asymmetry_coefficient + length_of_kernel_groove, family = multinom
ial,
##     data = seeds_train)
##
## Pearson residuals:
##                        Min         1Q      Median         3Q    Max
## log(mu[,1]/mu[,3]) -0.2507 -0.0004539 -0.00001467 0.00006344 0.2725
## log(mu[,2]/mu[,3]) -0.1274 -0.0008345 -0.00005503 0.00003206 0.1407
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept):1             -615.7521  2260.9680  -0.272    0.785
## (Intercept):2              353.9362  3002.1203   0.118    0.906
## area:1                     -18.2152    80.2874  -0.227    0.821
## area:2                      15.5862   105.5402   0.148    0.883
## perimeter:1                 53.9929   163.4872      NA       NA
## perimeter:2                  2.9887   225.8194   0.013    0.989
## compactness:1              351.1986  1373.9877      NA       NA
## compactness:2             -282.5456  1718.4672      NA       NA
## length_of_kernel:1           3.4585    72.2886      NA       NA
## length_of_kernel:2         -57.9704    76.0597      NA       NA
## width_of_kernel:1            1.1885   116.8446   0.010    0.992
## width_of_kernel:2           -7.3878   154.7941      NA       NA
## asymmetry_coefficient:1     -3.1679     2.9930  -1.058    0.290
## asymmetry_coefficient:2     -0.2879     3.7118  -0.078    0.938
## length_of_kernel_groove:1  -39.6800    49.8832  -0.795    0.426
## length_of_kernel_groove:2   -5.1394    43.7949      NA       NA
##
## Names of linear predictors: log(mu[,1]/mu[,3]), log(mu[,2]/mu[,3])
##
```

```
## Residual deviance: 0.903 on 314 degrees of freedom
##
## Log-likelihood: -0.4515 on 314 degrees of freedom
##
## Number of Fisher scoring iterations: 17
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## 'perimeter:1', 'compactness:1', 'compactness:2', 'length_of_kernel:1', 'le
ngth_of_kernel:2', 'width_of_kernel:2', 'length_of_kernel_groove:2'
##
##
## Reference group is level  3  of the response
```

```
predictions<-predict(model_mnl,newdata=seeds_test[,2:8],type="response")
seeds_test$pred_mnl<-apply(predictions,1,function(i) which.max(i) )

#confusion matrix
print(table(seeds_test$wheat_type,seeds_test$pred_mnl))
```

```
##
##      1  2  3
##   1 14  0  1
##   2  0 14  1
##   3  4  0 11
```

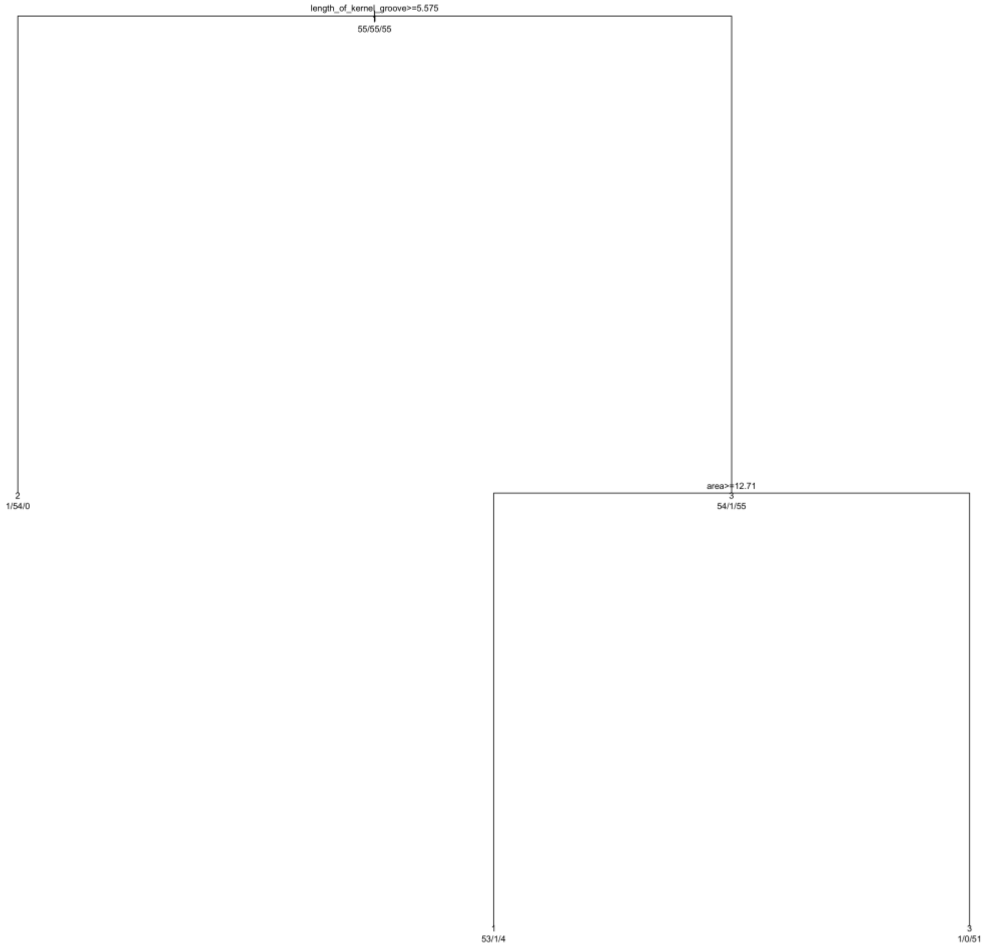
```
# error rate
mean(seeds_test$pred_mnl != seeds_test$wheat_type)
```

```
## [1] 0.1333333
```

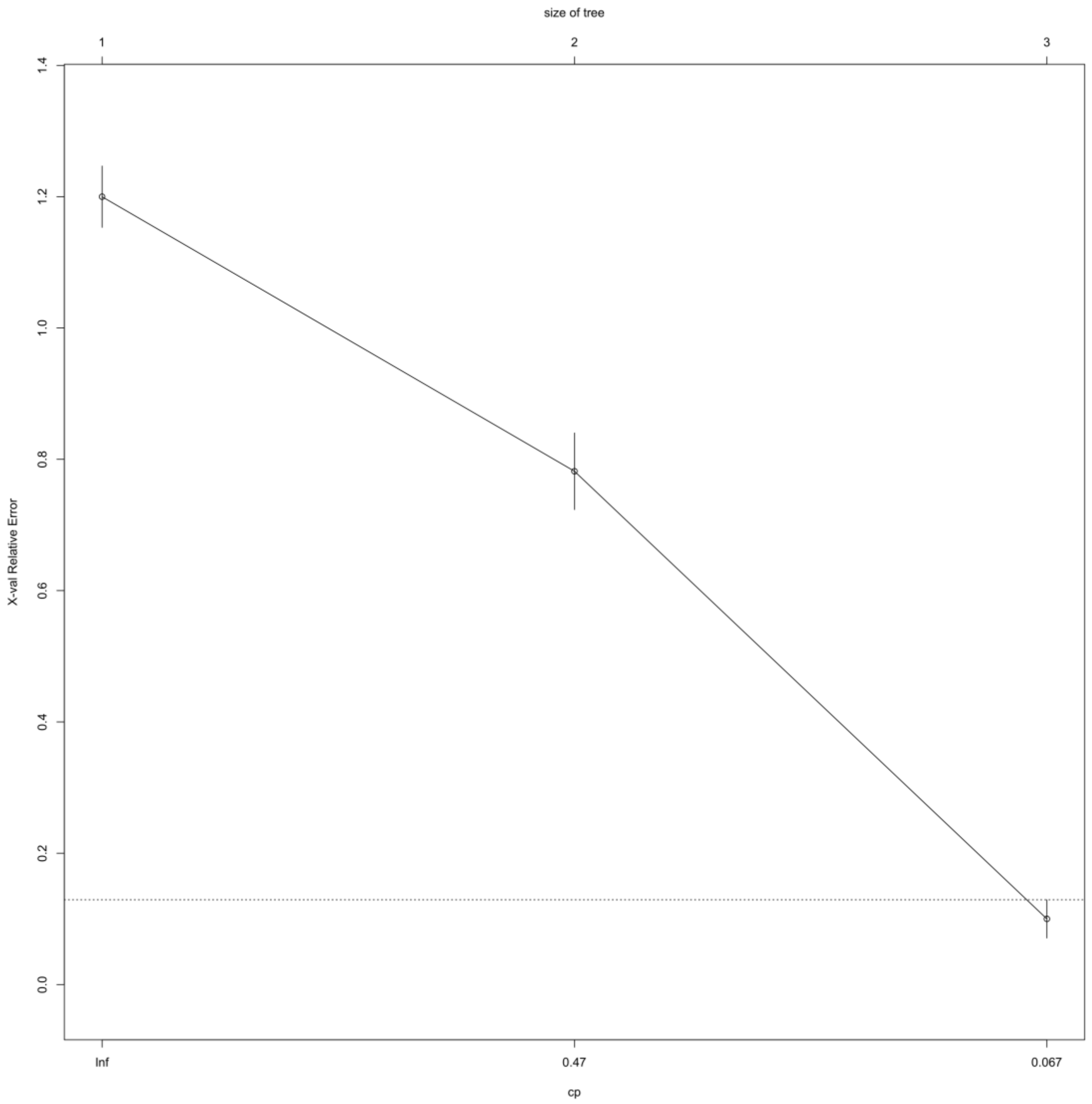
CART

The model’s lowest error rate is 0.06363636 after 2 spilts but the error rate on the test set is 0.22. The model seems to be overfitting.

```
######## CART
model_ct <- rpart(wheat_type ~ area+perimeter+compactness+length_of_kernel+wi
dth_of_kernel+asymmetry_coefficient+length_of_kernel_groove, data = seeds_tra
in, method="class")
plot(model_ct)
text(model_ct, use.n=TRUE, all=TRUE, cex=.7)
```



```
plotcp(model_ct)
```



```
seeds_test$pred_ct<-predict(model_ct,seeds_test,type="vector")
#confusion matrix
table(seeds_test$wheat_type,seeds_test$pred_ct)

##
##       1  2  3
##    1 11  0  4
##    2  1 14  0
##    3  5  0 10

# error rate
mean(seeds_test$pred_ct != seeds_test$wheat_type)

## [1] 0.2222222
```

Conclusion

Based on the model's perfomance of all four models, the Multinomial logistic regression model is the best fitting model and is the recommend classifier for the research question. The seven geometrical or morphological features are apt for classifying the variety of wheat. It is recommend that at least these seven geometrical or morphological features be collected for classifying the variety of seeds. Further research can be done by analyzing more environmental factor that can impact the geometrical or morphological features of the seeds. For example, taking the seeds samples from different feilds and harvested using different methods. Capturing the enivromental and harvesting methods data and analysing it for classification may yeild in better classification models and generalizing the techniques over the population.