2025

Data Analysis Report

PROFESSOR: ANTONIO PUNZO
STUDENT: MUHAMMAD SALMAN SABIR

DATA SET: ROCK

Introduction

This report outlines the analysis I conducted on the "rock" dataset, which is included in the default datasets of R. The report is structured into three chapters. The first chapter provides an overview of the dataset and includes a univariate analysis of the variables, with a focus on specific characteristics. In the second chapter, I describe the Principal Component Analysis (PCA). Finally, in the third chapter, I discuss the Cluster Analysis and performed.

Dataset

The dataset "rock" was collected by BP Research, image analysis by Ronit Katz, U. Oxford. It contains the Measurements of 48 rock samples from a petroleum reservoir. Each core sample was measured for permeability, area of pores, total perimeter of pores, and shape which are important factors in determining the productivity of the reservoir.

The purpose of collecting this dataset was likely to study the physical properties of the rocks in the reservoir and to understand how these properties affect the flow of oil and gas. Overall, the dataset is likely to be useful for researchers and engineers who are interested in developing models to predict reservoir productivity or in designing drilling and production operations for the reservoir.

The dataset contains 48 observations and 4 continuous variables.

Loading the Data
First, we need to load the dataset into R. The dataset is available as an R package, so we can install and load it using the following commands:

```
> install.packages("datasets")
> library(datasets)
> data(rock)
> str(rock)

'data.frame':     48 obs. of  4 variables:
 $ area : int  4990 7002 7558 7352 7943 7979 9333 8209 8393 6425 ...
 $ peri : num  2792 3893 3931 3869 3949 ...
 $ shape: num  0.0903 0.1486 0.1833 0.1171 0.1224 ...
 $ perm : num  6.3 6.3 6.3 6.3 17.1 17.1 17.1 17.1 119 119 ...
```

Univariate Analysis

Univariate analysis is the simplest form of statistical analysis, in which we examine the distribution of a single variable. In our case, we have four variables: area, perimeter, shape, and perm.

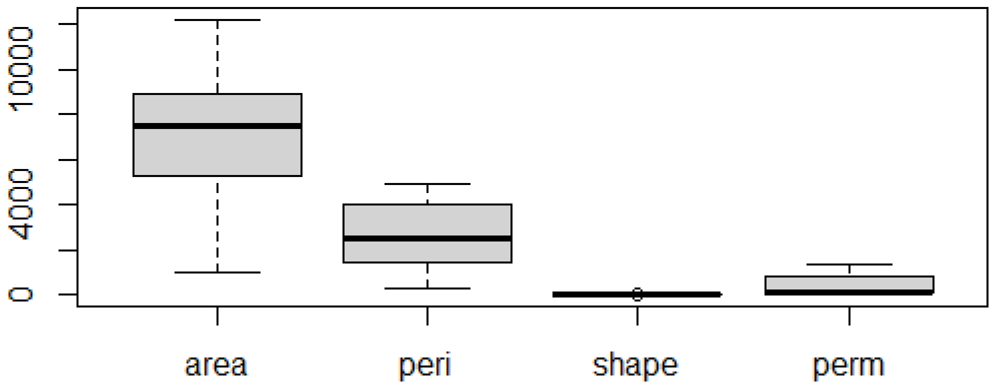
There are 4 variables so here are the details of all the variables.

[1]	area:	area of pores space, in pixels out of 256 by 256 (length x height)
[2]	peri:	perimeter in pixels (adding the length of all 4 sides)
[3]	shape:	perimeter/sqrt(area)
[4]	perm:	permeability in milli-Darcies (passage of liquids or gases)

```
> head(rock)
  area      peri       shape          perm
1 4990      2791.90    0.0903296      6.3
2 7002      3892.60    0.1486220      6.3
3 7558      3930.66    0.1833120      6.3
4 7352      3869.32    0.1170630      6.3
5 7943      3948.54    0.1224170      17.1
6 7979      4010.15    0.1670450      17.1
```

```
> summary(rock)

     Area              peri              shape              perm

 Min. : 1016       Min.   : 308.6     Min.   :0.09033    Min.   :   6.30
 1st Qu.: 5305     1st Qu.:1414.9     1st Qu.:0.16226    1st Qu.:  76.45
 Median : 7487     Median :2536.2     Median :0.19886    Median : 130.50
 Mean   : 7188     Mean   :2682.2     Mean   :0.21811    Mean   : 415.45
 3rd Qu.: 8870     3rd Qu.:3989.5     3rd Qu.:0.26267    3rd Qu.: 777.50
 Max.   :12212     Max.   :4864.2     Max.   :0.46413    Max.   :1300.00
```

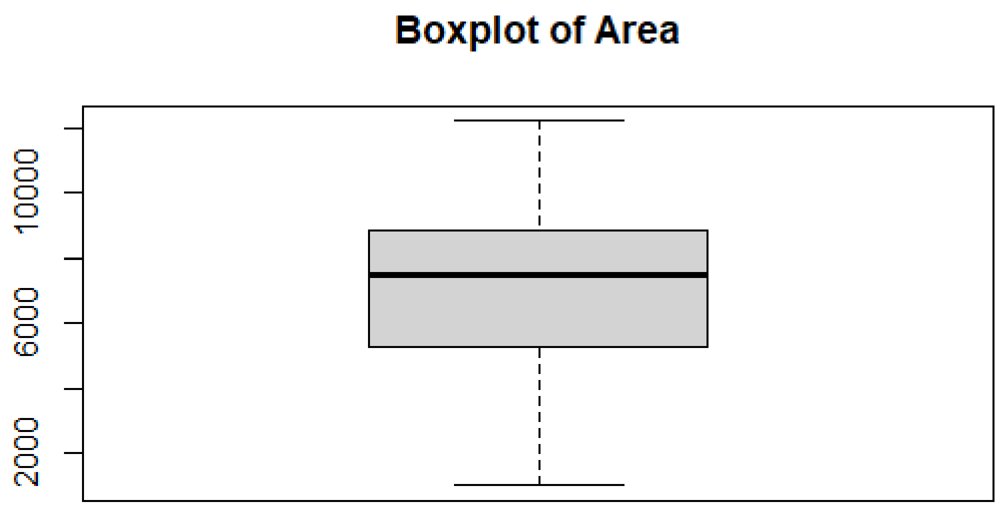
```
> boxplot(rock)
```



1. Area

```
> summary(rock$area)
   Min.      1st Qu.     Median     Mean      3rd Qu.     Max.
   1016      5305        7487       7188      8870        12212

> sd(rock$area)
[1] 2683.849

> var(rock$area)
[1] 7203045
```

```
> boxplot(rock$area, main = "Boxplot of Area")
```



This boxplot represents the summary of the distribution of the Area variable.
The median of area is 7487 which means the 50% of the data lies below 7487 and 50% above it. **As the mean is 7188 which is less than the value of median so it means the data is negatively skewed**. Third quartile (Q3) shows that the 75% of data falls below 8870.

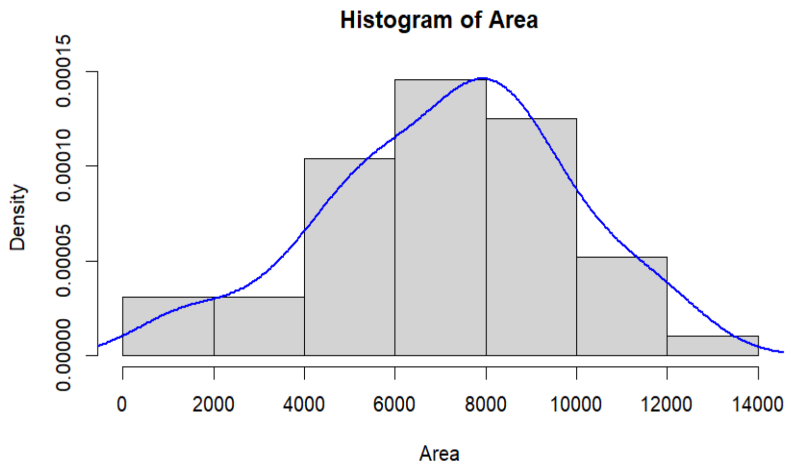
To calculate the skewness I have installed the package
"moments"

```
# Load the moments package
> library(moments)

# Calculate the skewness of variable area
skewness_value <- skewness(rock$area)  # Calculate skewness
print(skewness_value)
[1] -0.3038973
```

It means data is left-skewed.

We can visualize the Area variable in more detailed way using the Histogram.

```
> hist(rock$area, freq=F, main="Histogram of Area", xlab="Area")
> lines(density(rock$area), col="blue", lwd=2)
```



Histogram of Area
Density
0.00000
0.00005
0.00010
0.00015
0
2000
4000
6000
8000
10000
12000
14000
Area

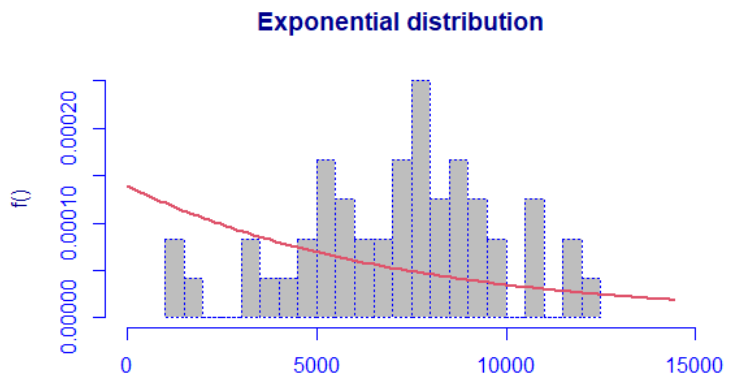
We can see from the histogram that it has moderate tail on the left hand side and the values of area is bit more concentrated on the right side. It can also be verified from the skewness method which resulted in the negative value.

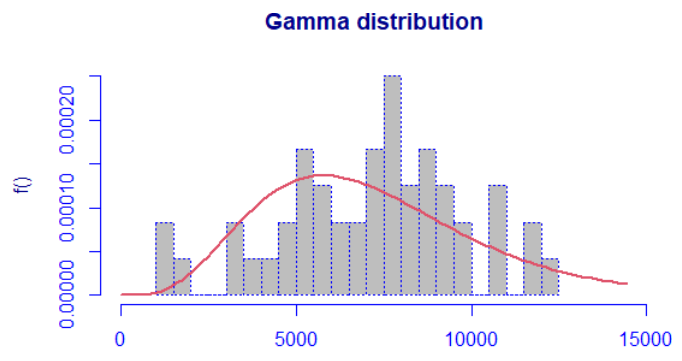
Here we are fitting some of the most commonly used distributions to our dataset variable - Area.

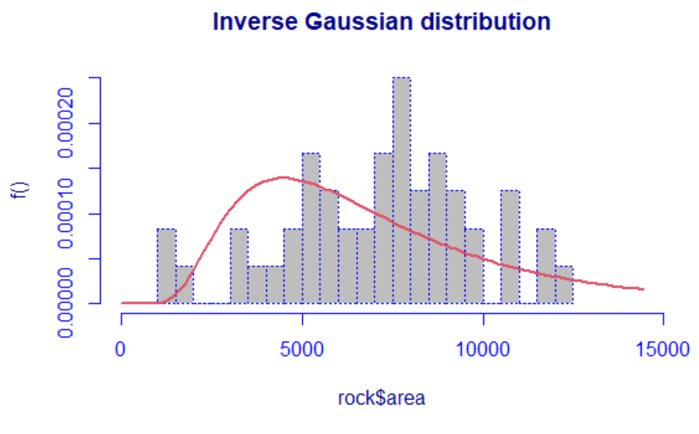
The histDist() function is part of the **gamlss** package in R. Install and Load the **gamlss** Package**:**

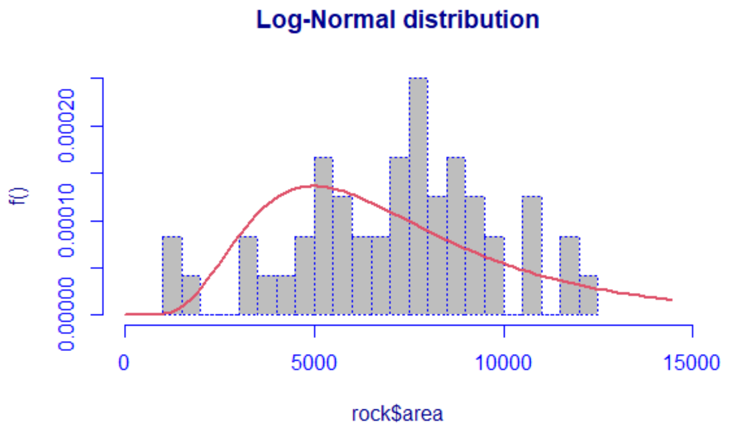
```
install.packages("gamlss")
library(gamlss)
```

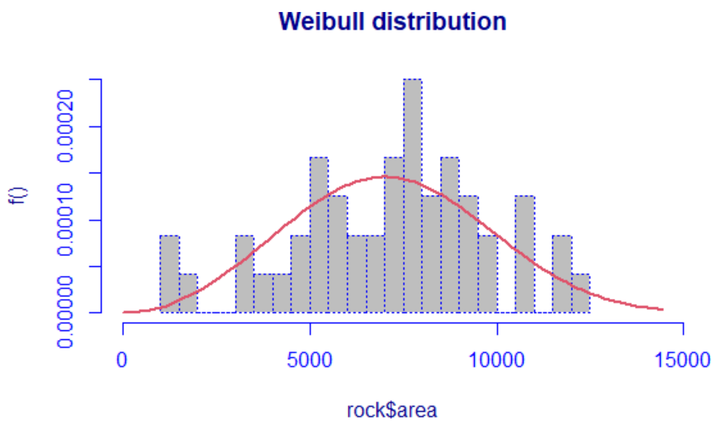
```
> fit.EXP <- histDist(rock$area, family=EXP, nbins = 24, main="Exponential distribution")
> fit.GA <- histDist(rock$area, family=GA, nbins = 24, main="Gamma distribution")
> fit.IG <- histDist(rock$area, family=IG, nbins = 24, main="Iinverse Gaussian distribution")
> fit.LOGNO <- histDist(rock$area, family=LOGNO, nbins = 24, main="Log-Normal distribution")
> fit.WEI <- histDist(rock$area, family=WEI, nbins = 24, main="Weibull distribution")
```











```
> data.frame(row.names = c("Exponential", "Gamma", "Inverse Gaussian", "Log-Normal", "Weibull"),
+           AIC=c(AIC(fit.EXP), AIC(fit.GA), AIC(fit.IG), AIC(fit.LOGNO), AIC(fit.WEI)),
+           SBC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc)
+           )
                     AIC         BIC
Exponential          950.4925    952.3637
Gamma                907.9637    911.7062
Inverse Gaussian     924.5491    928.2915
Log-Normal           919.2915    923.0339
Weibull              898.3867    902.1291
```

The Akaike information criterion and the Bayesian information criterion are techniques utilized for assessing and contrasting statistical models. Their objective is to determine which model best fits our distribution. Consequently, we can conclude that the **Weibull model** is the optimal model for our data as it has the lowest value for AIC and BIC.

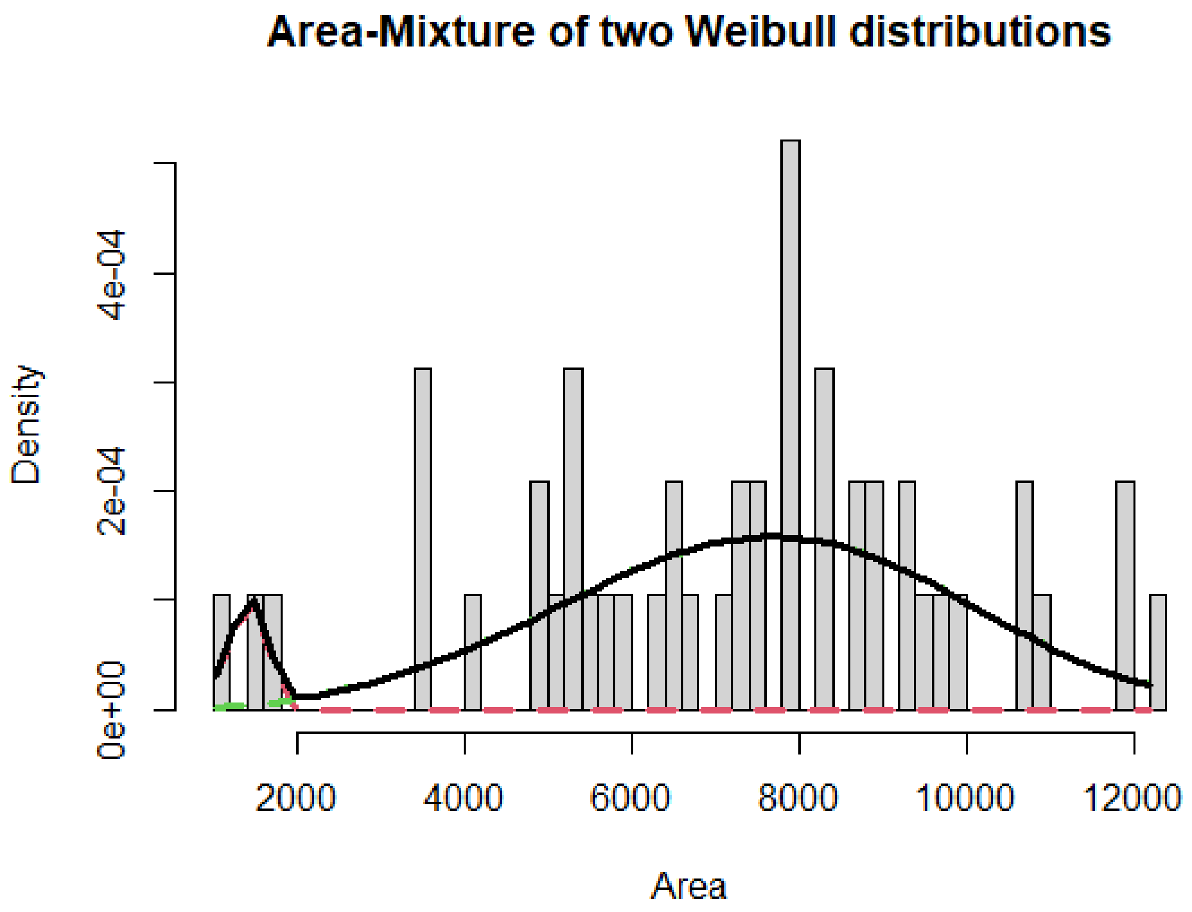
Additionally, we can utilize the **Likelihood-Ratio Test** to compare two nested models.

```
> LR.test(fit.IG,fit.LOGNO)

 Likelihood Ratio Test for nested GAMLSS models.
 (No check whether the models are nested is performed).

 Null model: deviance= 920.5491 with  2 deg. of freedom
 Altenative model: deviance= 915.2915 with  2 deg. of freedom

 LRT = 5.257644 with 0 deg. of freedom and p-value= 0
```

In this analysis, we are assessing two different distributions, the Inverse Gaussian distribution, which represents the null hypothesis, and the Log-Normal distribution which represents the alternative hypothesis. The obtained p-value equals 0, which is lower than the pre-determined significance level of 0.05. Therefore, we can reject the null hypothesis (Inverse Gaussian), which suggested that the Inverse Gaussian distribution was the inappropriate model for our data. We accept the alternative hypothesis that the Log-Normal distribution is a better fit. This decision is also supported by the AIC and BIC values, which are lower for the Log-Normal distribution compared to the Inverse Gaussian distribution.

Here we are trying to have mixture of two Weibull Distributions:

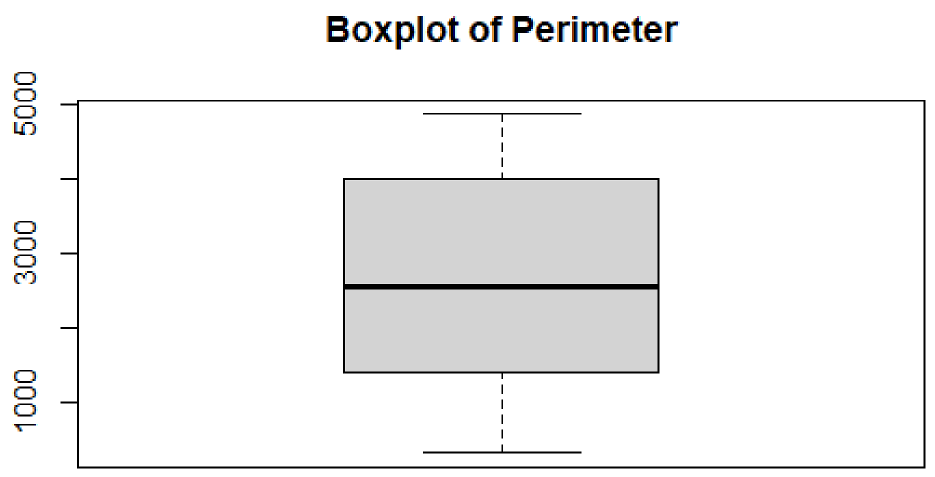
```
> Area.mix.WEI <- gamlssMXfits(data=rock, n = 5, rock$area~1, family = WEI, K = 2)
model= 1
model= 2
model= 3
model= 4
model= 5

> Area.mix.WEI$aic
[1] 896.1547
> Area.mix.WEI$sbc
[1] 905.5107

> mu.hat1 <- exp(Area.mix.WEI[["models"]][[1]][["mu.coefficients"]])
> sigma.hat1 <- exp(Area.mix.WEI[["models"]][[1]][["sigma.coefficients"]])
> mu.hat2 <- exp(Area.mix.WEI[["models"]][[2]][["mu.coefficients"]])
> sigma.hat2 <- exp(Area.mix.WEI[["models"]][[2]][["sigma.coefficients"]])
> hist(rock$area, breaks = 50,freq = FALSE, xlab = "Area" ,main="Area-Mixture of two Weibull distributions")
>
lines(seq(min(rock$area),max(rock$area),length=length(rock$area)),Area.mix.WEI[["prob"]][1]*dWEI(seq(mi
n(rock$area), max(rock$area),length=length(rock$area)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
>
lines(seq(min(rock$area),max(rock$area),length=length(rock$area)),Area.mix.WEI[["prob"]][2]*dWEI(seq(mi
n(rock$area), max(rock$area),length=length(rock$area)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
>
lines(seq(min(rock$area),max(rock$area),length=length(rock$area)),Area.mix.WEI[["prob"]][1]*dWEI(seq(mi
n(rock$area),max(rock$area),length=length(rock$area)), mu = mu.hat1, sigma = sigma.hat1) +
Area.mix.WEI[["prob"]][2]*dWEI(seq(min(rock$area),max(rock$area),length=length(rock$area)), mu =
mu.hat2, sigma = sigma.hat2),lty = 1, lwd = 3, col = 1)
```



By employing a mixture of two Weibull distributions, we observe that the AIC value has improved compared to those obtained with a single Weibull distribution. **The AIC value has decreased from 898.3867 to 896.1547, while the BIC value has increased from 902.1291 to 905.5107**. **According to AIC, two Weibull distributions mixture is a better fit for the data, but BIC suggest that it makes our model more complex**. A lower AIC is preferable.

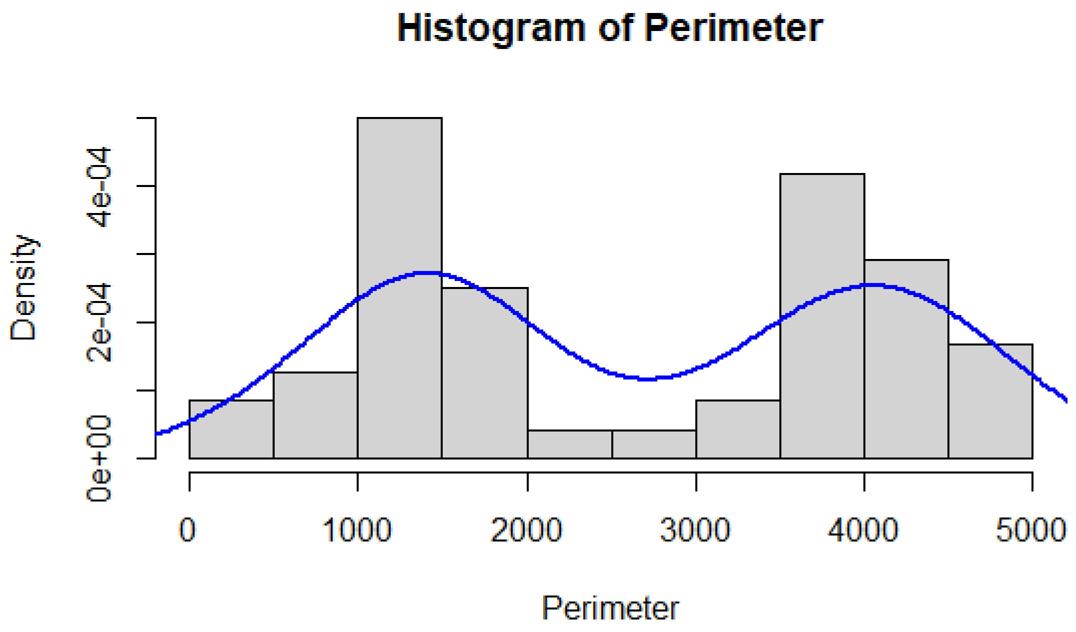
2. Perimeter

```
> summary(rock$peri)
   Min.      1st Qu.        Median      Mean        3rd Qu.       Max.
  308.6      1414.9         2536.2      2682.2      3989.5        4864.2

> sd(rock$peri)
[1] 1431.661

> var(rock$peri)
[1] 2049654

> boxplot(rock$peri, main = "Boxplot of Perimeter")
```

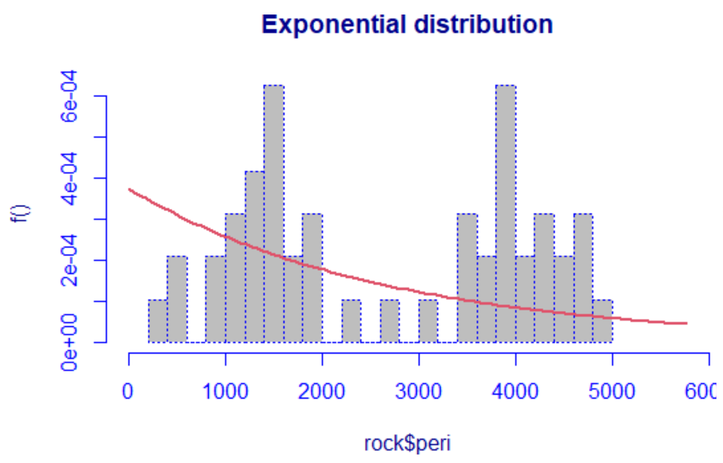


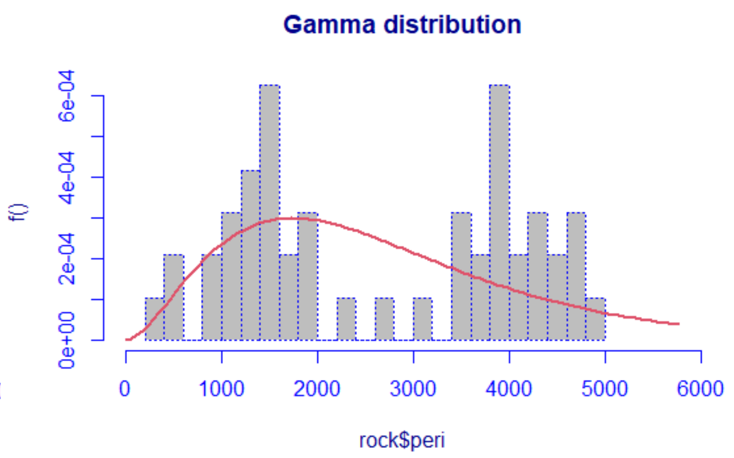
This boxplot shows that the values of the perimeter variable is spread between the range of 308.6 to 4864.2. The median of the Perimeter is 2536.2 which means the 50% of data is below this value and 50% is above the median value. There is no outliers in this variable values.

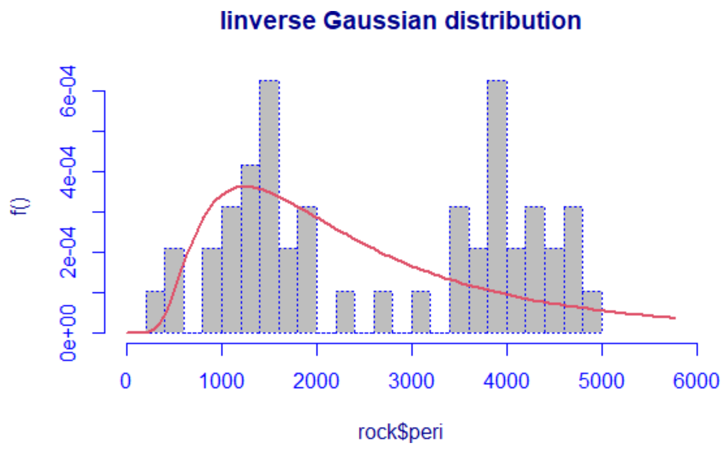
```
> hist(rock$peri, freq=F, main="Histogram of Perimeter", xlab="Perimeter")
> lines(density(rock$peri), col="blue", lwd=2)
```

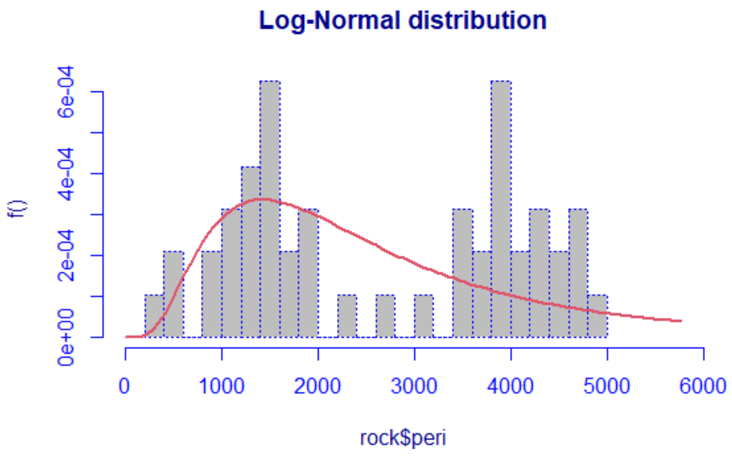


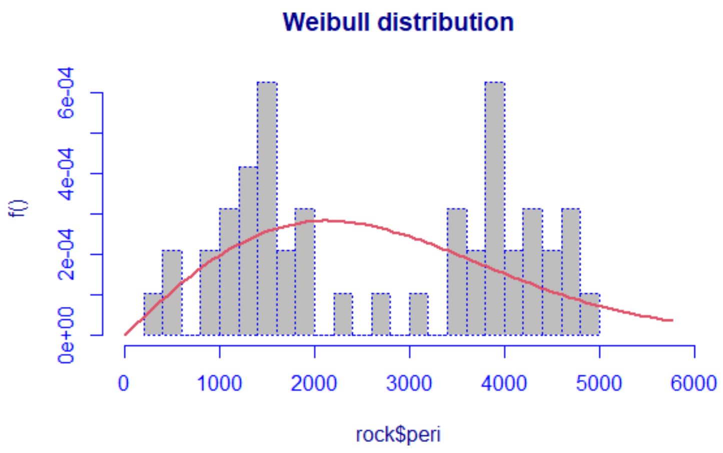
```
> skewness(rock$peri)
[1] 0.02423405

> fit.EXP <- histDist(rock$peri, family=EXP, nbins = 24, main="Exponential distribution")
> fit.GA <- histDist(rock$peri, family=GA, nbins = 24, main="Gamma distribution")
> fit.IG <- histDist(rock$peri, family=IG, nbins = 24, main="Iinverse Gaussian distribution")
> fit.LOGNO <- histDist(rock$peri, family=LOGNO, nbins = 24, main="Log-Normal distribution")
> fit.WEI <- histDist(rock$peri, family=WEI, nbins = 24, main="Weibull distribution")
```











```
> data.frame(row.names = c("Exponential", "Gamma", "Inverse Gaussian", "Log-Normal",
"Weibull"),AIC=c(AIC(fit.EXP), AIC(fit.GA), AIC(fit.IG), AIC(fit.LOGNO),
AIC(fit.WEI)),SBC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc))
```

	AIC	SBC
Exponential	855.8621	857.7333
Gamma	835.6588	839.4012
Inverse Gaussian	843.7996	847.5420
Log-Normal	841.5695	845.3119
Weibull	**832.8034**	**836.5458**

The **Akaike information** criterion and the **Bayesian information criterion** are techniques utilized for assessing and contrasting statistical models. Their objective is to determine which model best fits our distribution. Consequently, we can conclude that the **Weibull model** is the optimal model for our data as it has the lowest value for AIC and BIC.

Additionally, we can utilize the Likelihood-Ratio Test to compare two nested models.
NOW, THE LIKELIHOOD RATIO TEST. I THINK THIS IS A HYPOTHESIS TESTING METHOD. IT COMPARES THE FIT OF TWO MODELS, ONE NESTED WITHIN THE OTHER. THE NULL HYPOTHESIS IS USUALLY A RESTRICTED VERSION OF THE ALTERNATIVE HYPOTHESIS.

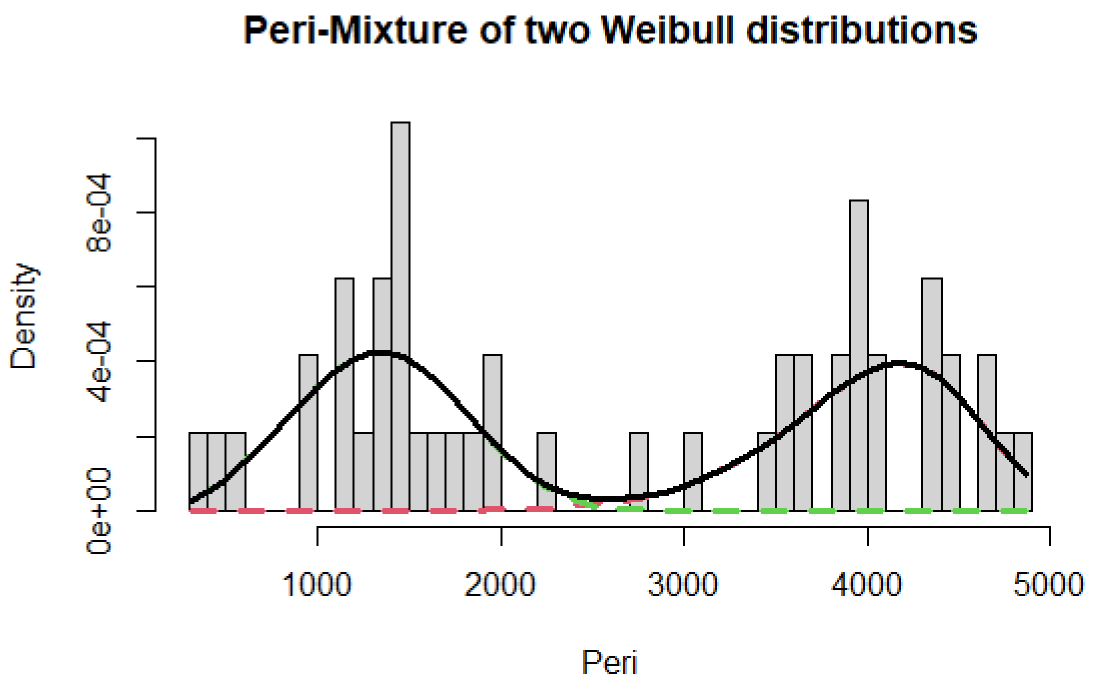
```
> LR.test(fit.IG,fit.LOGNO)
 Likelihood Ratio Test for nested GAMLSS models.
 (No check whether the models are nested is performed).

 Null model: deviance= 839.7996 with  2 deg. of freedom
 Alternative model: deviance= 837.5695 with  2 deg. of freedom

 LRT = 2.230019 with 0 deg. of freedom and p-value= 0
```

In this analysis, we are assessing two different distributions, the Inverse Gaussian distribution, which represents the null hypothesis, and the Log-Normal distribution which represents the alternative hypothesis. The obtained p-value equals 0, which is lower than the pre-determined significance level of 0.05. Therefore, we can reject the null hypothesis (Inverse Gaussian), which suggested that the Inverse Gaussian distribution was the inappropriate model for our data. **We accept the alternative hypothesis that the Log-Normal distribution is a better fit.** This decision is also supported by the AIC and BIC values, which are lower for the Log-Normal distribution compared to the Inverse Gaussian distribution.

Here we are trying to have mixture of two Weibull Distributions:

```
> Peri.mix.WEI <- gamlssMXfits(data=rock, n = 5, rock$peri~1, family = WEI, K = 2)
model= 1
model= 2
model= 3
model= 4
model= 5

> mu.hat1 <- exp(Peri.mix.WEI[["models"]][[1]][["mu.coefficients"]])

> sigma.hat1 <- exp(Peri.mix.WEI[["models"]][[1]][["sigma.coefficients"]])

> mu.hat2 <- exp(Peri.mix.WEI[["models"]][[2]][["mu.coefficients"]])

> sigma.hat2 <- exp(Peri.mix.WEI[["models"]][[2]][["sigma.coefficients"]])

> hist(rock$peri, breaks = 50,freq = FALSE, xlab = "Peri" ,main="Peri-Mixture of two Weibull distributions")
```

```
>lines(seq(min(rock$peri),max(rock$peri),length=length(rock$peri)),Peri.mix.WEI[["prob"]][1]*dWEI(seq(min(rock$peri), max(rock$peri),length=length(rock$peri)), mu = mu.hat1, sigma = sigma.hat1),lty=2,lwd=3,col=2)

>lines(seq(min(rock$peri),max(rock$peri),length=length(rock$peri)),Peri.mix.WEI[["prob"]][2]*dWEI(seq(min(rock$peri), max(rock$peri),length=length(rock$peri)), mu = mu.hat2, sigma = sigma.hat2),lty=2,lwd=3,col=3)

>lines(seq(min(rock$peri),max(rock$peri),length=length(rock$peri)),Peri.mix.WEI[["prob"]][1]*dWEI(seq(min(rock$peri),max(rock$peri),length=length(rock$peri)), mu = mu.hat1, sigma = sigma.hat1) + Peri.mix.WEI[["prob"]][2]*dWEI(seq(min(rock$peri),max(rock$peri),length=length(rock$peri)), mu = mu.hat2, sigma = sigma.hat2),lty = 1, lwd = 3, col = 1)
```



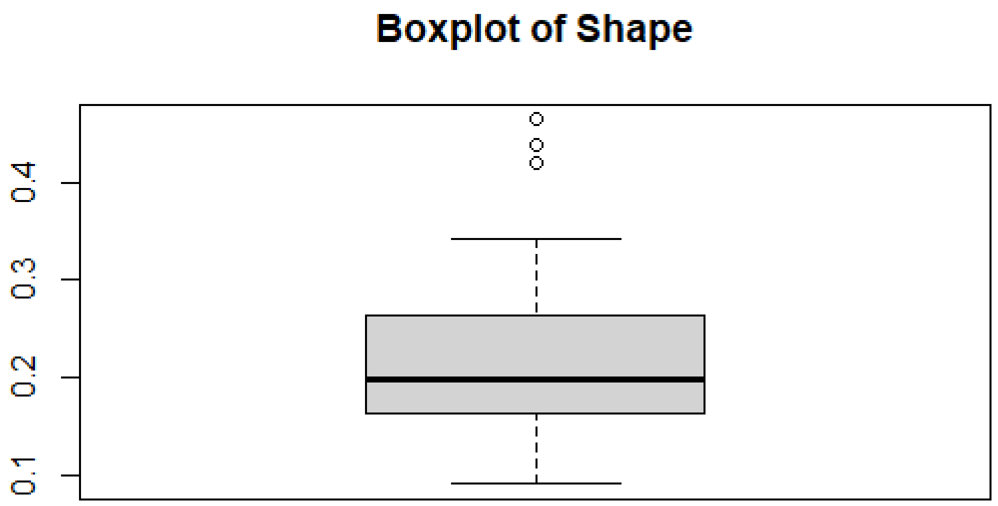
```
> Peri.mix.WEI$aic
[1] 805.4731
> Peri.mix.WEI$sbc
[1] 814.8291
```

By employing a mixture of two Weibull distributions, we observe that the AIC and BIC values have improved compared to those obtained with a single Weibull distribution. **The AIC value has decreased from 832.8034 to 805.4731, while the BIC value has decreased from 836.5458 to 814.8291. This describes that the mixture of two Weibull distributions is a better fit.**

3. Shape

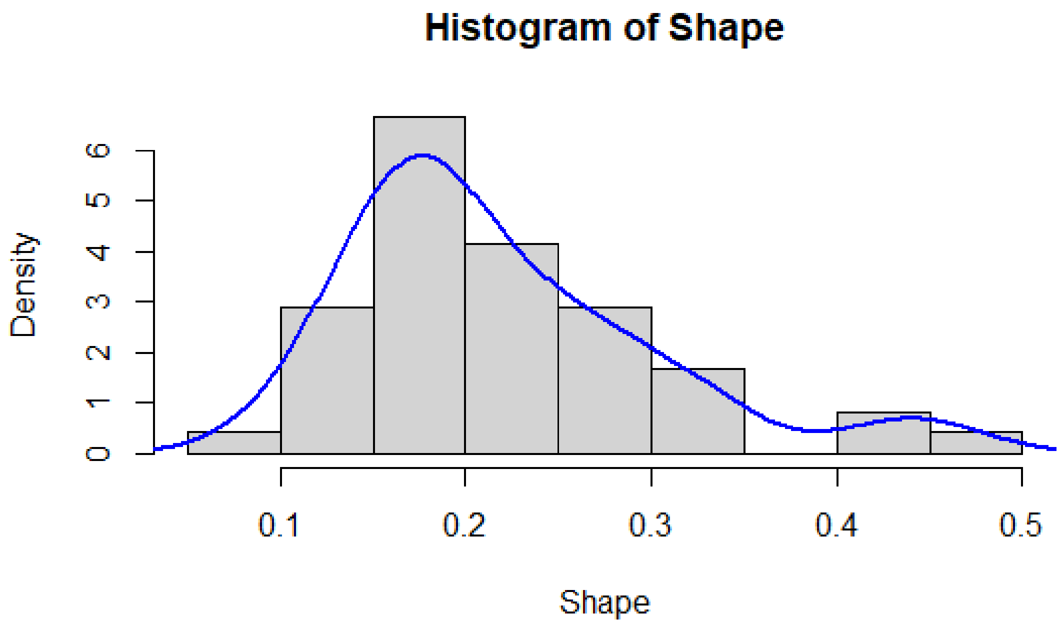
```
> summary(rock$shape)
   Min.     1st Qu.     Median     Mean      3rd Qu.    Max.
0.09033    0.16226     0.19886    0.21811   0.26267    0.46413
```

```
> sd(rock$shape)
[1] 0.08349645
> var(rock$shape)
[1] 0.006971657

> boxplot(rock$shape, main = "Boxplot of Shape")

> skewness(rock$shape)
[1] 1.132977
```



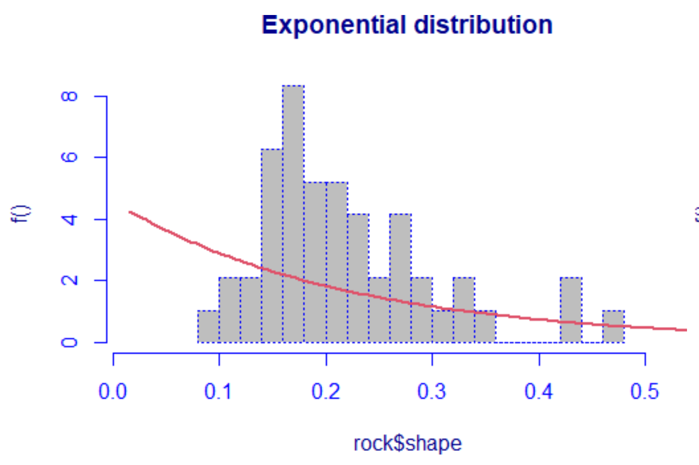
This boxplot represents that the 50% of data lies between the 1st Quartile (0.16226) and the 3rd Quartile (0.26267) for the variable shape. This variable contains the outliers. It is positively skewed which can be visualized using the histogram.

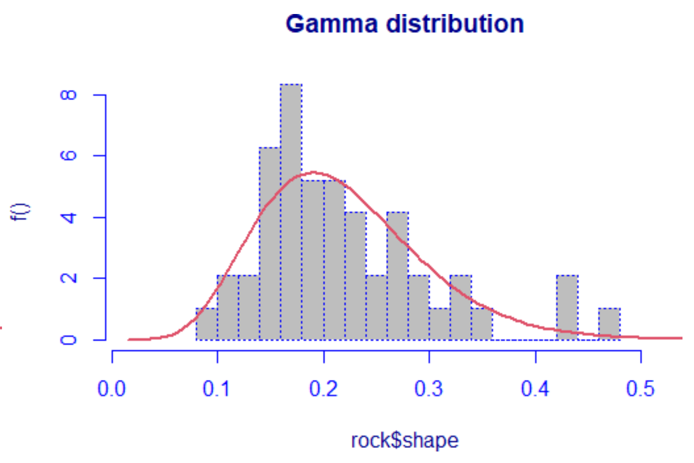
```
> hist(rock$shape, freq=F, main="Histogram of Shape", xlab="Shape")
> lines(density(rock$shape), col="blue", lwd=2)
```

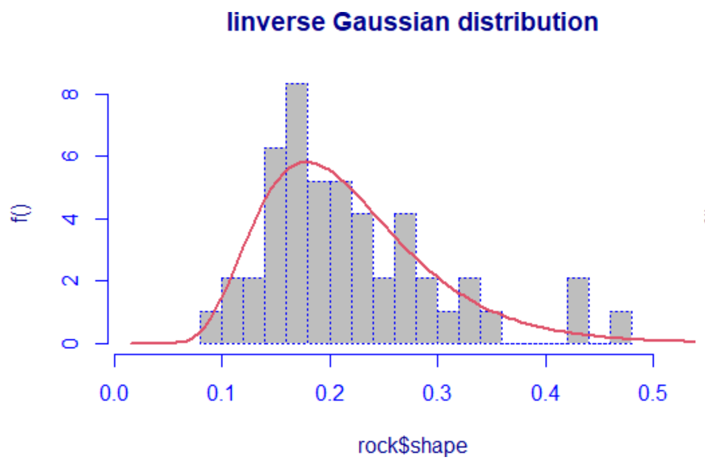


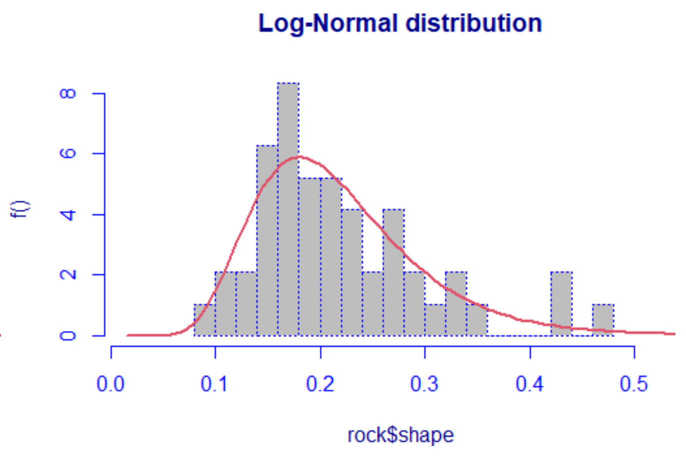
The histogram shows a positively skewed distribution, with a peak around 0.3 and a long tail extending towards higher values of the "Shape" variable.

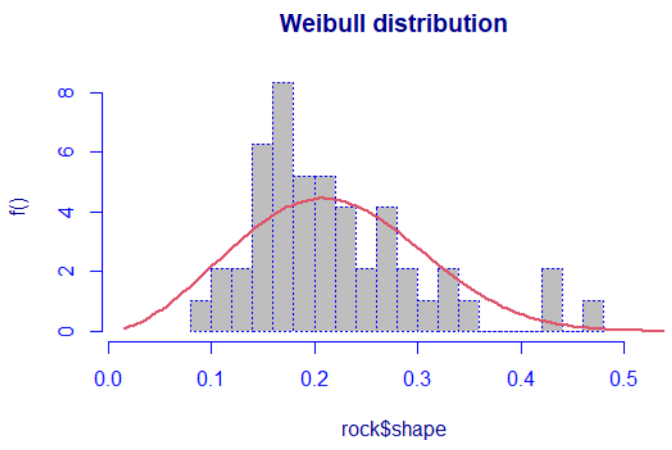
```
> fit.EXP <- histDist(rock$shape, family=EXP, nbins = 24, main="Exponential distribution")
> fit.GA <- histDist(rock$shape, family=GA, nbins = 24, main="Gamma distribution")
> fit.IG <- histDist(rock$shape, family=IG, nbins = 24, main="Iinverse Gaussian distribution")
> fit.LOGNO <- histDist(rock$shape, family=LOGNO, nbins = 24, main="Log-Normal distribution")
> fit.WEI <- histDist(rock$shape, family=WEI, nbins = 24, main="Weibull distribution")
```







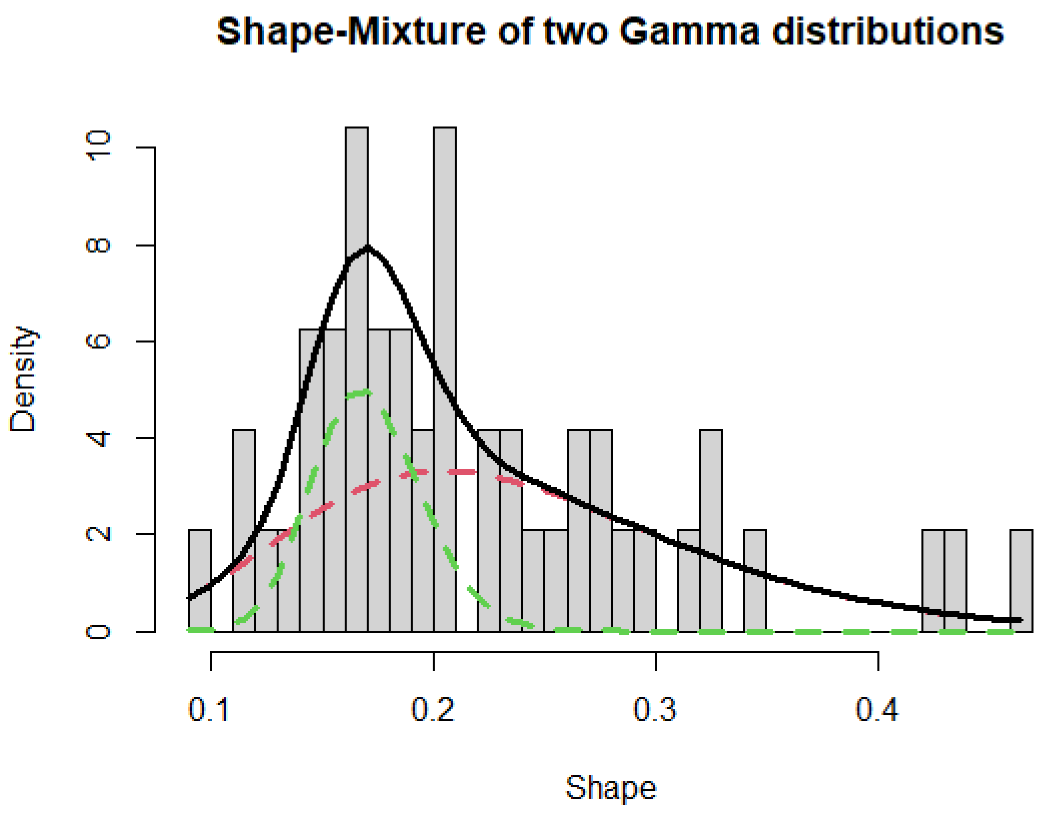




```
> data.frame(row.names = c("Exponential", "Gamma", "Inverse Gaussian", "Log-Normal",
"Weibull"),AIC=c(AIC(fit.EXP), AIC(fit.GA), AIC(fit.IG), AIC(fit.LOGNO),
AIC(fit.WEI)),SBC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc))
```

	AIC	SBC
Exponential	-48.18436	-46.31316
Gamma	-109.44080	-105.69840
Inverse Gaussian	**-112.01738**	**-108.27498**
Log-Normal	-111.87592	-108.13351
Weibull	-101.48441	-97.74201

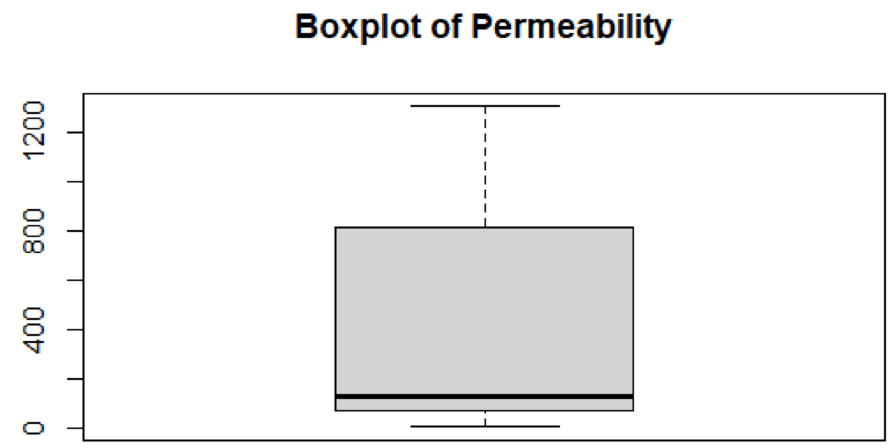
```
Shape.mix.GA <- gamlssMXfits(data=rock, n = 5, rock$shape~1, family = GA, K = 2)
Shape.mix.GA$aic
Shape.mix.GA$sbc
mu.hat1 <- exp(Shape.mix.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(Shape.mix.GA[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(Shape.mix.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(Shape.mix.GA[["models"]][[2]][["sigma.coefficients"]])
hist(rock$shape, breaks = 50,freq = FALSE, xlab = "Shape" ,main="Shape-Mixture of two Gamma
distributions")
lines(seq(min(rock$shape),max(rock$shape),length=length(rock$shape)),Shape.mix.GA[["prob"]][1]*dGA(seq
(min(rock$shape), max(rock$shape),length=length(rock$shape)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(rock$shape),max(rock$shape),length=length(rock$shape)),Shape.mix.GA[["prob"]][2]*dGA(seq
(min(rock$shape), max(rock$shape),length=length(rock$shape)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(rock$shape),max(rock$shape),length=length(rock$shape)),Shape.mix.GA[["prob"]][1]*dGA(seq
(min(rock$shape),max(rock$shape),length=length(rock$shape)), mu = mu.hat1, sigma = sigma.hat1) +
Shape.mix.GA[["prob"]][2]*dGA(seq(min(rock$shape),max(rock$shape),length=length(rock$shape)), mu =
mu.hat2, sigma = sigma.hat2),lty = 1, lwd = 3, col = 1)
```



The plot represents shape variable, ranging from approximately 0 to 0.4 x-axis. Y-axis represents the density on probability density function. Black curve represents the overall mixture density of the two Gama distributions. Green and red curves represent the individual Gama Distribution that make up the mixture.

4. Permeability

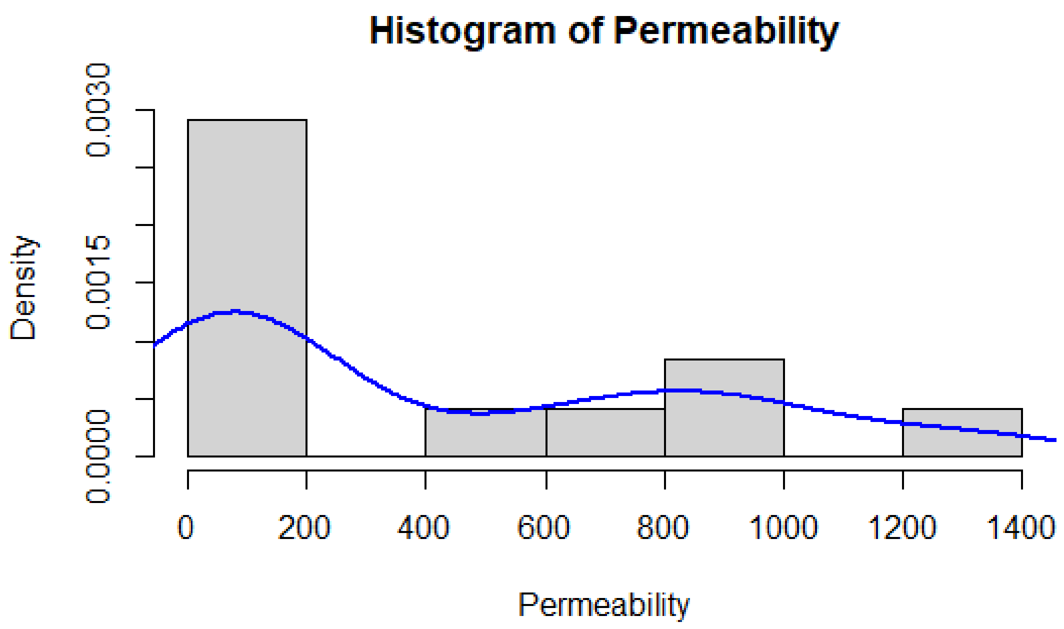
```
> summary(rock$perm)
   Min.        1st Qu.         Median        Mean          3rd Qu.        Max.
   6.30        76.45           130.50        415.45        777.50         1300.00

> sd(rock$perm)
[1] 437.8182

> var(rock$perm)
[1] 191684.8

> boxplot(rock$perm, main = "Boxplot of Permeability")
```



This boxplot shows that the values of the Permeability variable is spread between the ranges of 6.30 to 1300.00. The median of the Perimeter is 130.50 which means the 50% of data is below this value and 50% is above the median value. There is no outliers in this variable values.

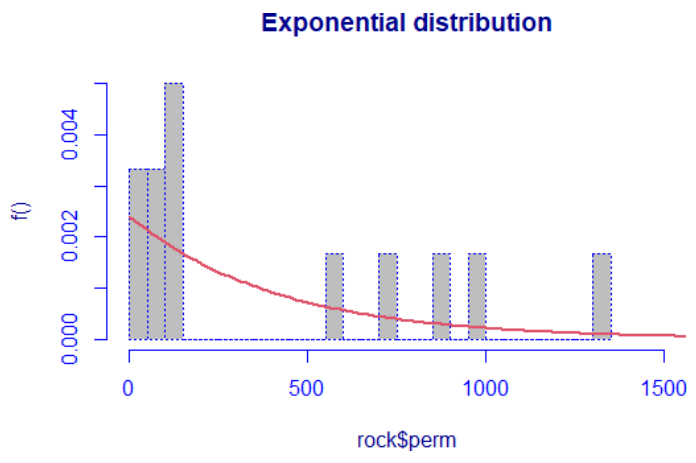
```
> skewness(rock$perm)
[1] 0.6934635
```

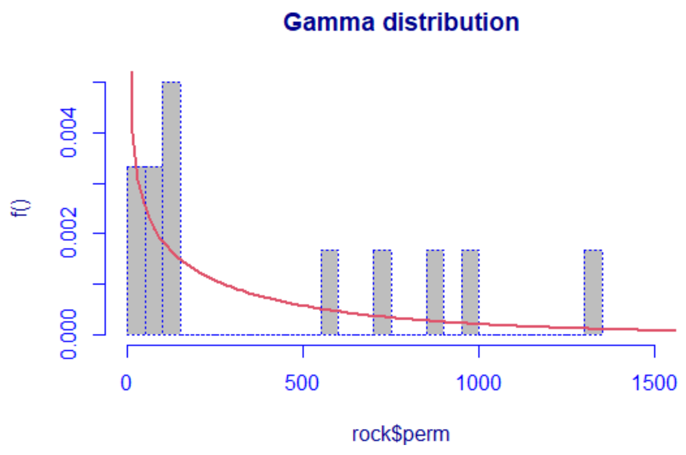
```
> hist(rock$perm, freq=F, main="Histogram of Permeability", xlab="Permeability")
> lines(density(rock$perm), col="blue", lwd=2)
```

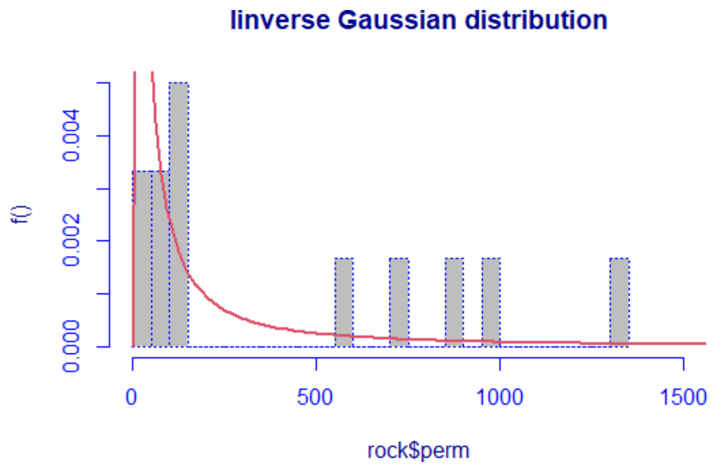


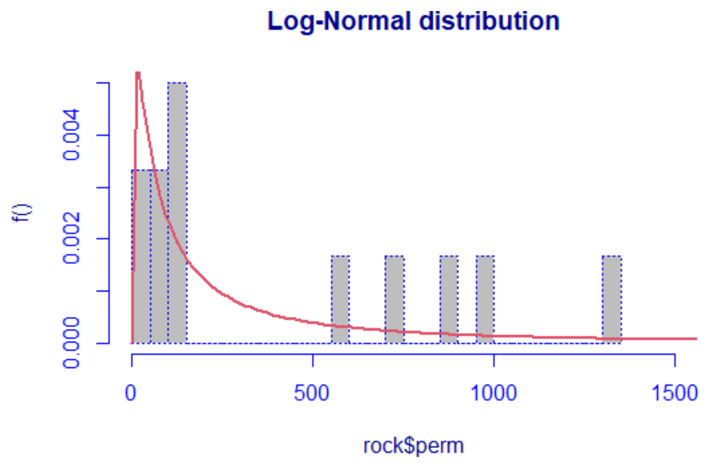
The histogram bars represent the frequency distribution of permeability values, showcasing how data points are distributed across different ranges. The blue line indicating the estimated probability density function of the permeability values. This curve helps visualize the overall distribution shape and how closely it align with the histogram. X-axis ranges from 0 to 1400 and Y-axis indicates density (how concentrated the data is) range. Frequency shows the actual count of data points in each bin.

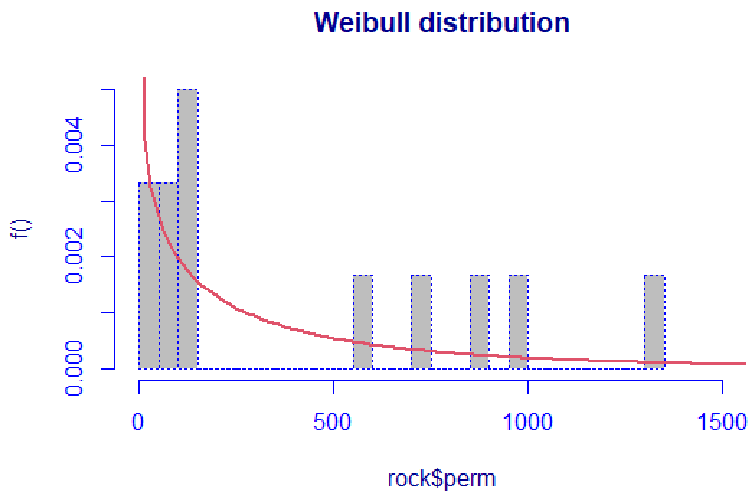
```
> fit.EXP <- histDist(rock$perm, family=EXP, nbins = 24, main="Exponential distribution")
> fit.GA <- histDist(rock$perm, family=GA, nbins = 24, main="Gamma distribution")
> fit.IG <- histDist(rock$perm, family=IG, nbins = 24, main="Iinverse Gaussian distribution")
> fit.LOGNO <- histDist(rock$perm, family=LOGNO, nbins = 24, main="Log-Normal distribution")
> fit.WEI <- histDist(rock$perm, family=WEI, nbins = 24, main="Weibull distribution")
```











```
> data.frame(row.names = c("Exponential", "Gamma", "Inverse Gaussian", "Log-Normal",
"Weibull"),AIC=c(AIC(fit.EXP), AIC(fit.GA), AIC(fit.IG), AIC(fit.LOGNO),
AIC(fit.WEI)),SBC=c(fit.EXP$sbc, fit.GA$sbc, fit.IG$sbc, fit.LOGNO$sbc, fit.WEI$sbc))

                            AIC              SBC
```

Exponential	676.8188	678.6900
Gamma	**672.3676**	**676.1100**
Inverse Gaussian	689.6813	693.4237
Log-Normal	677.2122	680.9546
Weibull	672.8630	676.6054

The Akaike information criterion and the Bayesian information criterion are techniques utilized for assessing and contrasting statistical models. Their objective is to determine which model best fits our distribution. Consequently, we can conclude that the Gamma model is the optimal model for our data as it has the lowest value for AIC and BIC.

Additionally, we can utilize the Likelihood-Ratio Test to compare two nested models.

```
> LR.test(fit.LOGNO,fit.WEI)
 Likelihood Ratio Test for nested GAMLSS models.
 (No check whether the models are nested is performed).

 Null model: deviance= 673.2122 with  2 deg. of freedom
 Altenative model: deviance= 668.863 with  2 deg. of freedom
  df(difference in the number of parameter between the models)

 LRT = 4.349204 with 0 deg. of freedom and p-value= 0
```

In this analysis, we are assessing two different distributions, **the Log Normal distribution, which represents the null hypothesis, and the Weibull distribution which represents the alternative hypothesis**. The obtained p-value equals 0, which is lower than the pre-determined significance level of 0.05. Therefore, we can reject the null hypothesis (Log Normal), which suggested that the Log Normal distribution was the appropriate model for our data. **We accept the alternative hypothesis that the Weibull distribution is a better fit**. This decision is also supported by the AIC and BIC values, which are lower for the Weibull distribution compared to the Log- Normal distribution.

Principal Component Analysis

PCA is a popular method for dimensionality reduction and data visualization. It works by finding the directions in which the data varies the most, and projecting the data onto those directions. This allows us to represent high-dimensional data in a lower-dimensional space while preserving most of the variation.

Purpose: PCA aims to reduce the number of features while retaining as much information as possible

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of data while preserving as much variability as possible. Here's a concise, step-by-step guide to performing PCA:

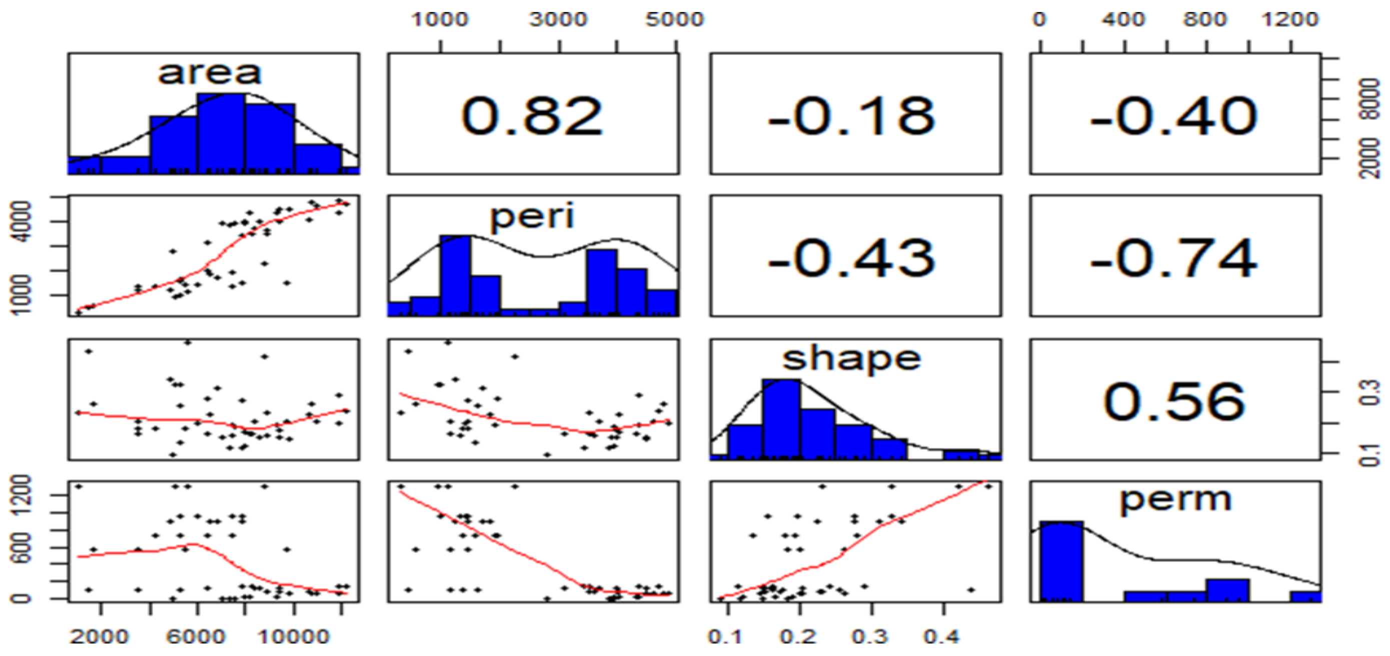
1. **Standardize the Data** (if variables have different units).
2. **Compute the Covariance/Correlation Matrix** (to quantify relationships between variables).

3. **Eigen Decomposition**:
4. Calculate **eigenvectors** (which form the *rotation matrix*) and **eigenvalues** (which indicate variance explained).
5. The eigenvectors define the directions (axes) of maximum variance (i.e., the "rotation" of the coordinate system).
6. **Sort Components** by eigenvalues (descending order).
7. **Project Data** onto the selected principal components (PC scores = data × rotation matrix).

We can perform PCA on the rock dataset by treating each variable as a separate feature and each sample as a separate observation.

```
> (cor(rock))
            area         peri           shape          perm
area        1.0000000    0.8225064      -0.1821611     -0.3966370
peri        0.8225064    1.0000000      -0.4331255     -0.7387158
shape       -0.1821611   -0.4331255     1.0000000       0.5567208
perm        -0.3966370   -0.7387158     0.5567208       1.0000000
```

```
> pairs.panels(rock, hist.col="blue", denisty=TRUE, ellipses=FALSE)
```



We can see that there is a very high positive correlation between area and peri, and negative correlation between the other variables.

To make the variables comparable, we need to standardize (i.e. scale) the data. This can be done by using the **prcomp() function** and setting the scale and center parameters to TRUE. By doing so, the variables are transformed to have a standard deviation of 1 and a mean of 0.

So here we are using the prcomp function in R to compute the principal components.

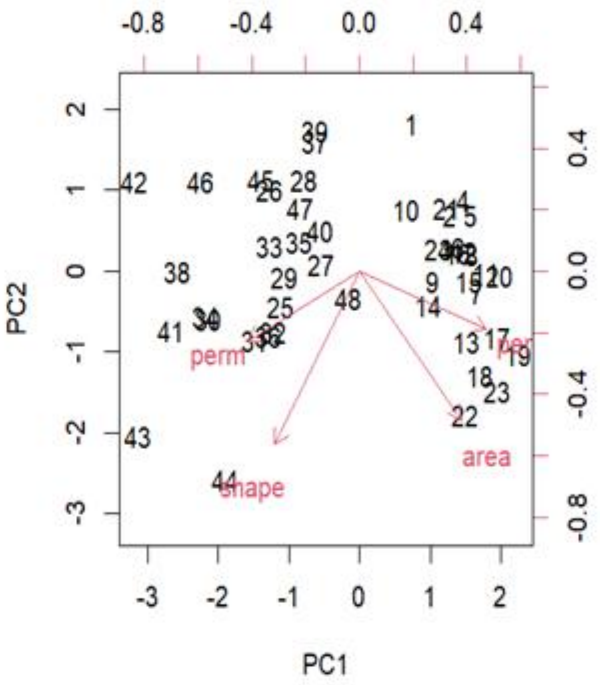
```
> pca <- prcomp(rock, center=TRUE, scale=TRUE)

 > summary (pca)
Importance of components:
                          PC1    PC2     PC3     PC4
   Standard deviation     1.6152 0.9607 0.62933 0.26844
   Proportion of Variance 0.6522 0.2307 0.09901 0.01802
   Cumulative Proportion  0.6522 0.8830 0.98198 1.00000
```

So we have centered and scaled the data before performing PCA. This is important because the variables have different scales, and we want to make sure that they are all treated equally.

```
>   pca$rotation[,1:4]
              PC1          PC2              PC3           PC4
area      0.4744238   -0.6046129      0.37639417     -0.51739034
peri      0.5886286   -0.2366588     -0.06308377      0.77040862
shape    -0.3932268   -0.7054215     -0.58863352      0.03554881
perm     -0.5232697   -0.2842821      0.71264188      0.37082890
```

We can now plot the results of PCA using a biplot, which shows the principal components as arrows and the samples as points. We can also color the points by their values of perm to see if there is any clustering based on this variable.

```
> biplot(pca, scale=0)
```



Let me provide a more detailed analysis of this biplot:
1. Data Distribution Pattern:
- The samples (numbered 1-48) appear to form some distinct groupings:
  - Upper right region: Samples like 1, 10, 21, 24 cluster together
  - Upper left region: Samples 42, 46, 39 form another group
  - Lower region: Samples like 43, 44 are more isolated
- This clustering suggests there are natural groupings in your data based on shared characteristics

2. Variable Relationships (Red Arrows):
- Area:
  - Points toward the positive PC1 direction (right side)
  - Samples on the right side (like 1, 21, 24) likely have larger areas
  - Samples on the left side (like 42, 43) likely have smaller areas

- Shape:
  - Points strongly downward (negative PC2)
  - Samples at the bottom of the plot (like 44) are likely more distinct in shape
  - Samples at the top (like 39, 1) probably have contrasting shape characteristics

In PCA, CVE (Cumulative Variance Explained), Scree Plots, and the Kaiser rule are tools used to decide how many principal components (PCs) to retain.

1. Cumulative Variance Explained:

Cumulative Variance Explained in PCA quantifies the total proportion of variance in the dataset captured by the first kk principal components. It helps determine how many components to retain for dimensionality reduction while preserving a desired level of information.
Cumulative Variance: The sum of variances explained by the first k*k* components. For example, if the first 3 PCs explain 80% of the variance, they collectively retain 80% of the original information.

Here's how to calculate, interpret, and visualize it in R using the rock dataset:

```
# Eigenvalues = (standard deviation)^2
eigenvalues <- rock_pca$sdev^2

# Proportion of variance explained by each PC
var_proportion <- eigenvalues / sum(eigenvalues)

# Cumulative variance explained
cum_var <- cumsum(var_proportion)

summary(rock_pca) # Includes cumulative variance
```

```
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.6152 0.9607 0.62933 0.26844
Proportion of Variance 0.6522 0.2307 0.09901 0.01802
Cumulative Proportion  0.6522 0.8830 0.98198 1.00000
```

- Cumulative Proportion shows that:
- PC1 explains 65.22% variance,
- PC1 + PC2 explain 23.07%,
- PC1 + PC2 + PC3 explain 88.38%,
- All 4 PCs explain 89.38%.

According to the CPVE we have to retain as many PCs as needed to explain at least the 80% of the total variance, hence we have to retain the first 3 PCs.

1. Kaiser's rule:

The Kaiser rule is a method to decide how many principal components (PCs) to retain in PCA. It states:
Retain components with eigenvalues ≥ 1.
Rationale: Eigenvalues represent the variance explained by each PC. When data is standardized (mean = 0, variance = 1), an eigenvalue ≥ 1 means the PC explains more variance than a single original variable.
Limitations: Works best for standardized data (correlation matrix). May over/under-retain components in some cases.

```
>  eigenvalues <- pca$sdev^2
>  print(eigenvalues)
 [1] 2.60896150 0.92291959 0.39605650 0.07206241
```

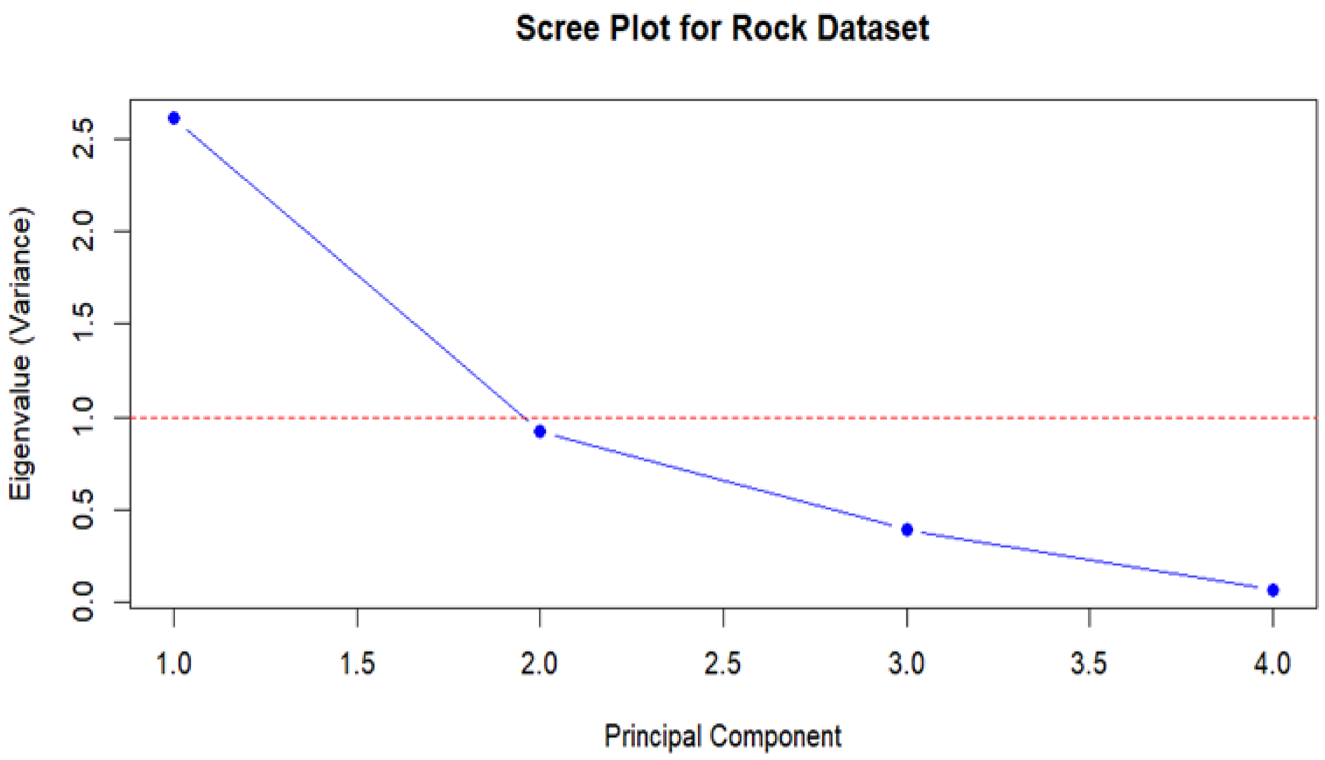
```
>   cat("Kaiser rule retains", kaiser_retain, "components.\n")
     Kaiser rule retains 1 components.
```

Kaiser rule states that retain as many pcs as for which the variance is greater than 1. So this rule suggests that retain only one PC.

2. Scree plot:

A scree plot is a graphical tool used in PCA to visualize the variance explained by each principal component.

```
plot(1:length(eigenvalues),
     eigenvalues,
     type = "b",        # Points connected by lines
     pch = 19,          # Solid circle markers
     col = "blue",
     xlab = "Principal Component",
     ylab = "Eigenvalue (Variance)",
     main = "Scree Plot for Rock Dataset")

# Add reference line for Kaiser criterion (eigenvalue = 1)
abline(h = 1, col = "red", lty = 2)
```



W

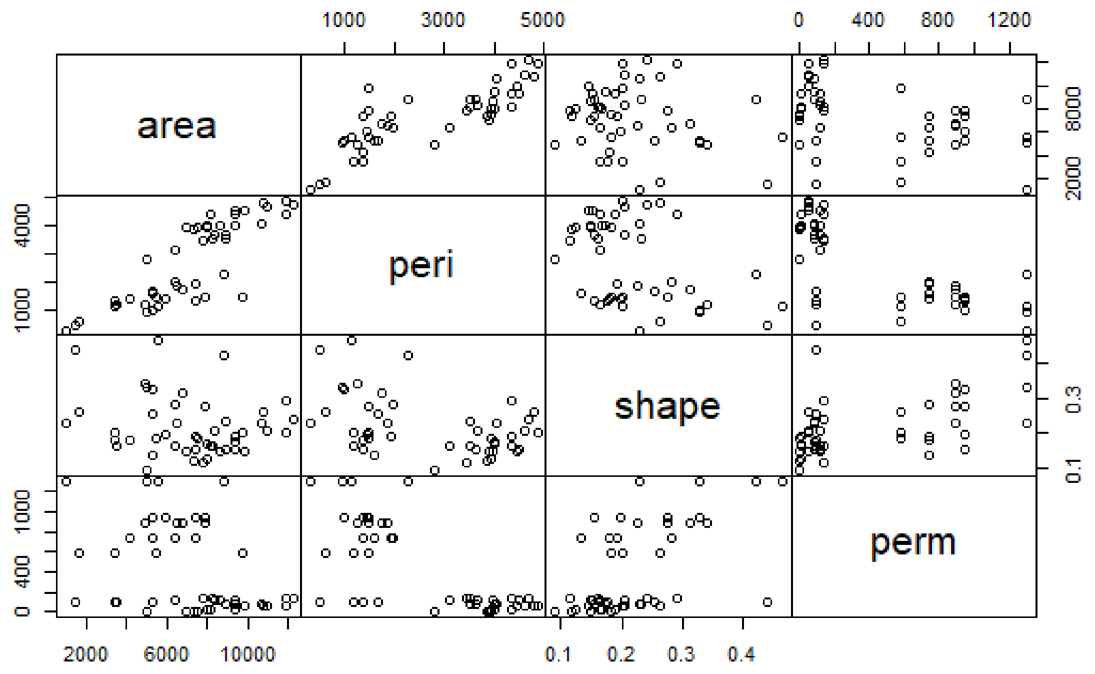
Clustering:-

Clustering is a method for partitioning data into groups or clusters based on their similarity. In our case, we can use clustering to group the rock samples based on the values of the four variables. Before we need to evaluate if the dataset contains meaningful clusters or not, so if there is a cluster tendency.

Assessing Cluster tendency:

We have to apply the process of Assessing Cluster tendency, starting by visualizing the standardized data.

```
# Generate a scatterplot matrix for the 'rock' dataset
pairs(
  rock,        # The dataset to visualize (each pair of variables is plotted)
  gap = 0,     # Set gap between plots to 0 (no spacing between plots)
  pch = 21     # Use plotting character 21 (filled circles) for points

)
```



The pairwise plots compare each variable against every other variable. **The scattered plot above shows that the data has capability of at least two clusters.** But how many clusters should be in our data, it is the subject of next step.

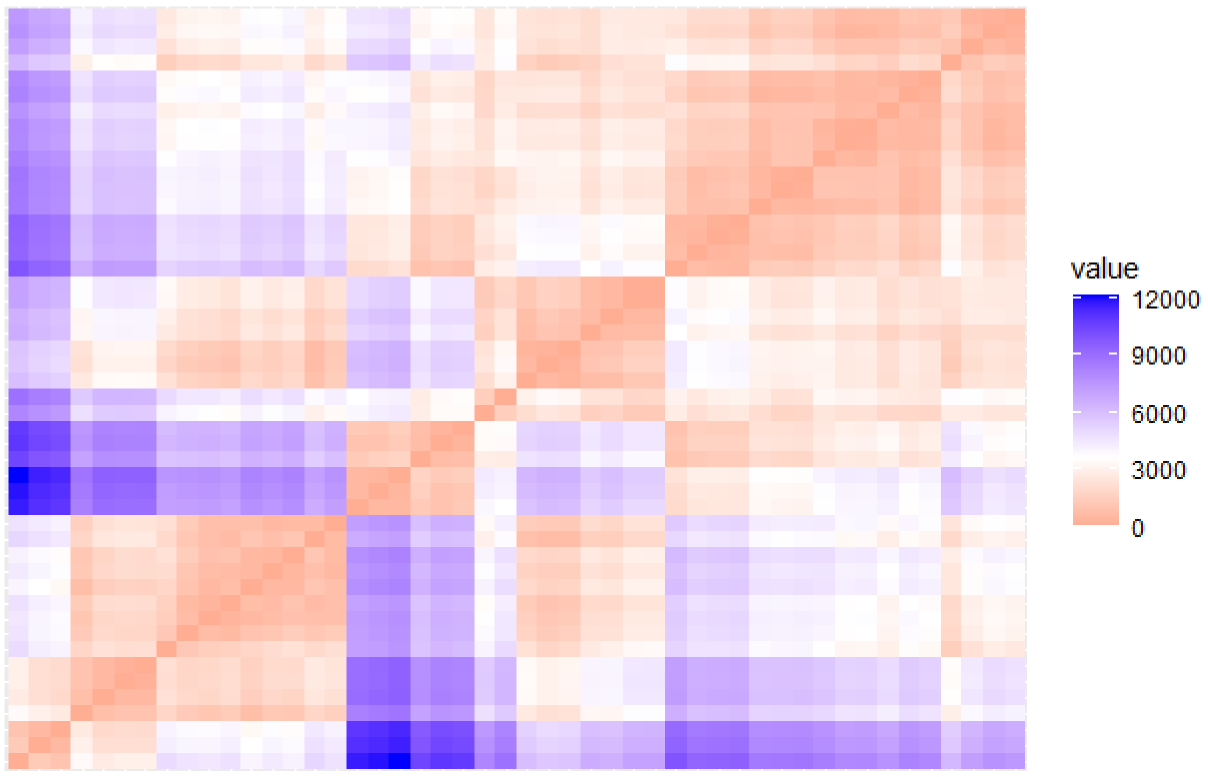
1. The Visual Assessment of cluster Tendency (VAT) algorithm:-

The **Visual Assessment of cluster Tendency (VAT)** is an algorithm that helps validate whether clusters exist in a dataset. Instead of applying a clustering algorithm directly, VAT provides a **visualization** of the data's cluster tendency.

How Does VAT Work?

1. **Computing a Distance Matrix**:
   - VAT first calculates a **distance matrix**, which measures the differences between data points.
   - The matrix provides a pairwise comparison of how similar or different each data point is from every other point.
2. **Rearranging the Data**:
   - The data is rearranged in such a way that similar points are placed close to each other.
   - This reordering helps in visually identifying clusters in the dataset.
3. **Visualization**:
   - The distance matrix is represented as a **heat map**, where colors indicate the level of similarity or dissimilarity.
   - The **color scale** helps in interpreting the data:
     - **Red** represents high similarity.
     - **Blue** represents high dissimilarity.

```
> fviz_dist(dist(rock) # Compute and visualize of the 'rock' dataset , show_labels=F   # Hide row/column labels)
```



So the visual representation is showing that our data has enough capability to have clusters.

2. The Hopkins Statistic

The **Hopkins statistic** helps determine whether a dataset is suitable for clustering. A value close to **0.5** suggests randomness, while a value **near 0** means no meaningful clusters, and a value **near 1** indicates strong cluster tendency.

```
# Install and load the hopkins package

install.packages("hopkins")

library(hopkins)

# Set seed for reproducibility

set.seed(123)

# Calculate the Hopkins statistic for the rock dataset

H <- hopkins(rock)

# Print the Hopkins statistic

print(H)
```

```
[1] 0.9963762
```

Hopkins statistics for rock dataset is close to one which is indication of highly cluster data but the value of 0.99 is also enough for us to do clustering in this dataset.

Hierarchical clustering: Dendrograms

Hierarchical clustering is an unsupervised machine learning technique used to group similar data points. It builds a tree-like structure (dendrogram) to show how data points are clustered.

Hierarchical clustering groups data into a tree-like structure (dendrogram) using two approaches:

- **Agglomerative (Bottom-Up)**: Starts with individual points and merges clusters.
- **Divisive (Top-Down)**: Starts with all points in one cluster and splits them.

Linkage Methods (How clusters are merged)

1. **Single Linkage** → uses the shortest distance between clusters. (Good for chain-like clusters, sensitive to noise)
2. **Complete Linkage** → uses the longest distance.
3. **Average Linkage** → uses the average distance.
4. **Centroid Linkage** → uses the distance between cluster centroids.
5. **Ward's Linkage** → Minimizes variance within clusters.

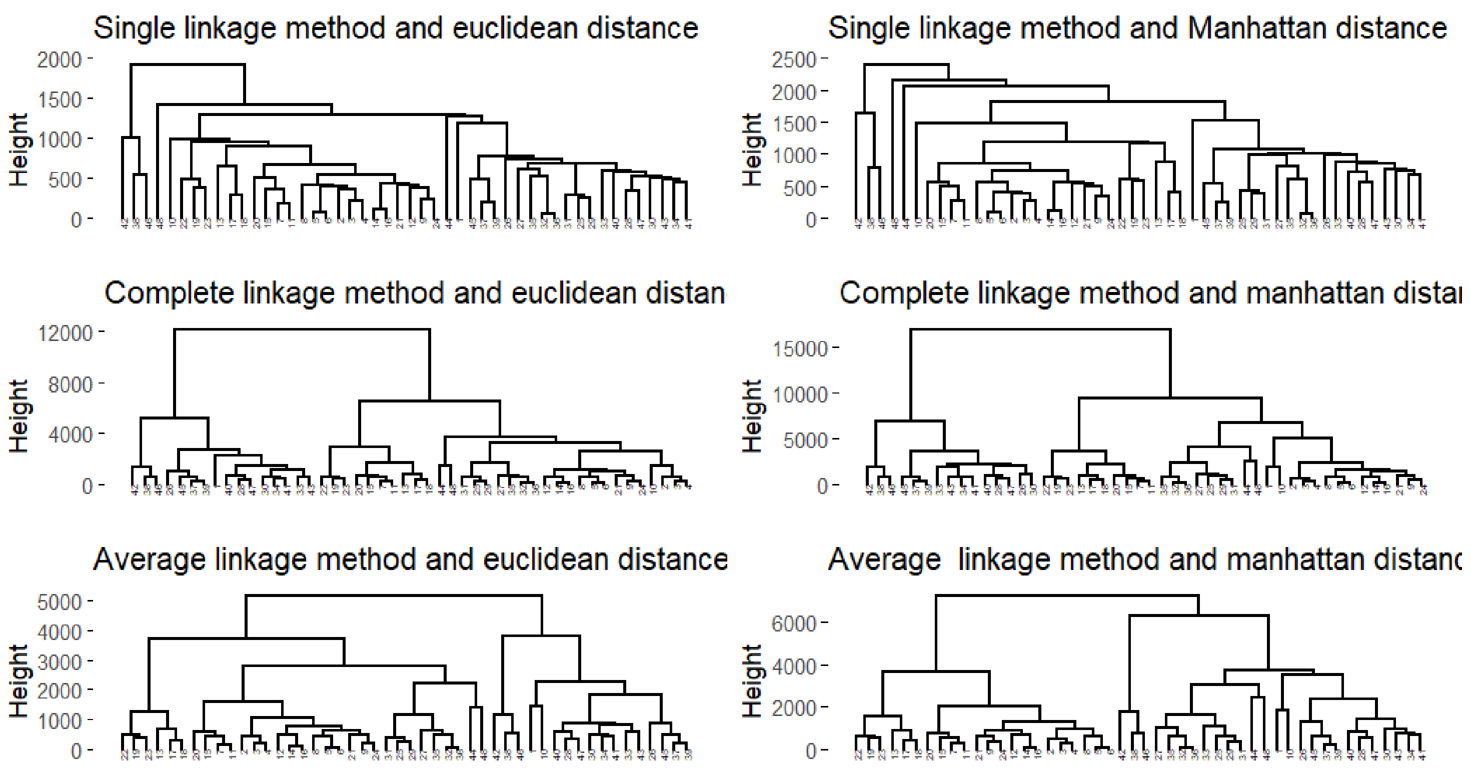
Hierarchical Clustering Algorithm (Agglomerative)

1. Start with each point as a separate cluster.
2. Compute distances between all clusters.
3. Merge the two closest clusters (based on linkage method).
4. Update the distance matrix.
5. Repeat until only one cluster remains.
6. Plot a dendrogram to visualize the hierarchy.

Additionally, we used the centroid, ward, and ward.d2 linkage methods, which merge clusters based on the average points. To evaluate the accuracy of the distances in the tree, we computed the cophenetic distance and the correlation between the cophenetic distances and the original distances. A high correlation coefficient indicates a clustering solution that accurately reflects the original data.

```
> library(gridExtra)

> grid.arrange(fviz_dend(hclust(dist(rock,method="euclidean"),method="single"),cex=0.3, main="Single linkage method and euclidean distance"),fviz_dend(hclust(dist(rock,method="manhattan"),method="single"),cex=0.3,main="Single linkage method and Manhattan distance"),fviz_dend(hclust(dist(rock,method="euclidean"),method="complete"),cex=0.3,main="Complete linkage method and euclidean distance"),
fviz_dend(hclust(dist(rock,method="manhattan"),method="complete"),cex=0.3,main="Complete linkage method and manhattan distance"),fviz_dend(hclust(dist(rock,method="euclidean"),method="average"),cex=0.3,main="Average linkage method and euclidean distance"),fviz_dend(hclust(dist(rock,method="manhattan"),method="average"),cex=0.3,main="Average linkage method and manhattan distance"),ncol=2,nrow=3)
```

```
dist() #computes the distance between data points
hclust() #perfroms hierarchical clustering
fviz_dend() #visualize the dendogram with clusters
```



One way to measure how well the cluster tree generated by the hclust() function reflects our data, is to compute the correlation between the cophenetic distance and the original distances generated by dist() function. Cophenetic dissimilarity or cophenetic distance of two units is a measure of how similar those two units have to be in order to be grouped into the same cluster.

Cophenetic distance is a measure used in hierarchical clustering to quantify the dissimilarity between two data points based on the clustering structure represented by a dendogram.

These calculations measure the **cophenetic correlation coefficient**, which evaluates how well a hierarchical clustering method preserves the original pairwise distances between data points.
so the original distance matrix is based on the actual distances (like Euclidean) between data points, and the cophenetic distance matrix is derived from the dendrogram's structure. The CCC measures the correlation between these two matrices. If CCC is high, the dendrogram's structure matches the original distances well. If it's low, there's a mismatch.

Each value represents the correlation between:

1. **Cophenetic distance** (distance between merged clusters in the dendrogram).
2. **Original distance matrix** (before clustering).

Interpretation of CCC

- **Close to 1:** The hierarchical clustering structure is a good representation of the original data.

- **Close to 0 or Negative:** The hierarchical clustering poorly represents the data.
- Generally, a CCC **above 0.75** is considered good.

```
> cor(cophenetic(hclust(dist(rock),method="single")), dist(rock))
[1] 0.7145951
> cor(cophenetic(hclust(dist(rock),method="complete")), dist(rock))
[1] 0.6753438
> cor(cophenetic(hclust(dist(rock),method="average")), dist(rock))
[1] 0.6853486
> cor(cophenetic(hclust(dist(rock,method = "manhattan"),method="single")), dist(rock, method = "manhattan"))
[1] 0.7095055
> cor(cophenetic(hclust(dist(rock,method = "manhattan"),method="average")), dist(rock,method =
"manhattan"))
[1] 0.7217926
> cor(cophenetic(hclust(dist(rock,method = "manhattan"),method="complete")), dist(rock,method =
"manhattan"))
[1] 0.6751694
```

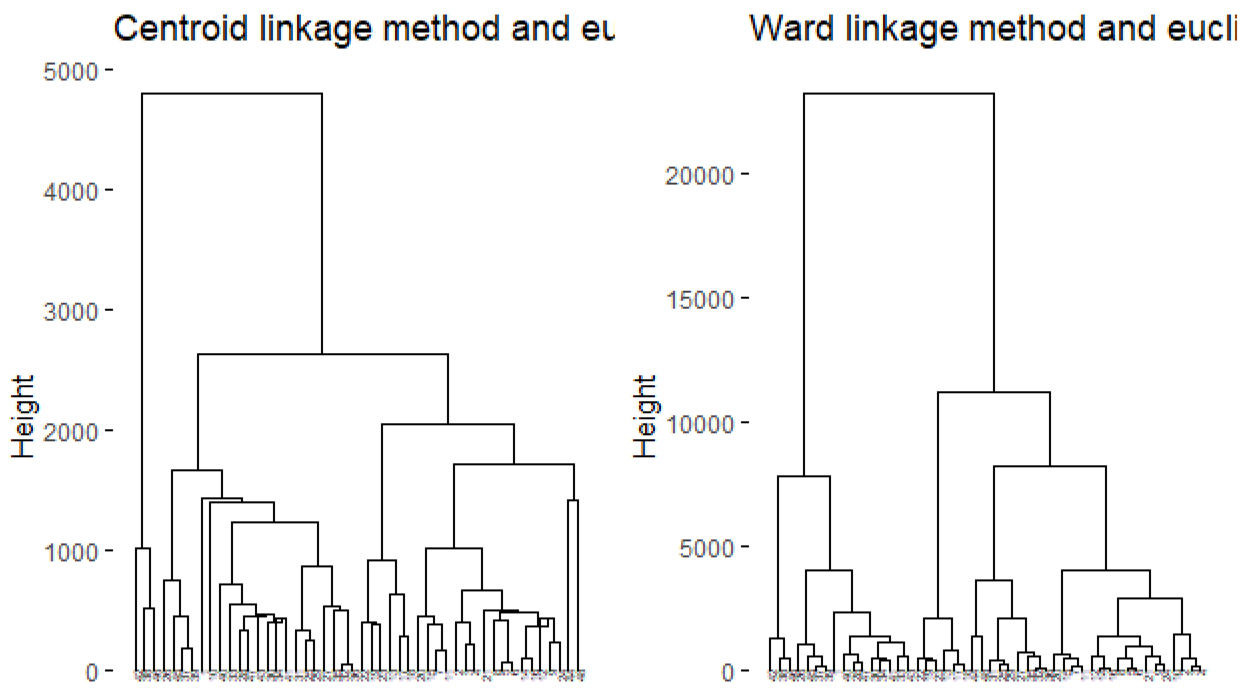
Key Observations:

- **Single linkage (0.7145)** performed best with **Euclidean distance**.
- **Average linkage (0.7218)** was best with **Manhattan distance**.
- **Complete linkage** had the lowest correlation in both cases.

As mentioned earlier, the single, complete, and average linkage methods each have their strengths and weaknesses. The single linkage method can handle groups with non-hyper spherical shapes, but it is very sensitive to outliers. On the other hand, the complete linkage method is the opposite, while the average linkage method provides an intermediate approach between the two.

In contrast, the centroid and ward linkage methods use average points to merge clusters, making them suitable for numeric datasets only. These methods use the minimum, average, and maximum distance between units of clusters to merge them. The ward method minimizes the sum of the squared Euclidean distances of units within a cluster (WSS - Within Deviance) and maximizes BSS - Between Deviance iteratively. Thus, it only accepts an Euclidean distance matrix as input. The centroid method also often employs Euclidean distances.

```
> grid.arrange(fviz_dend(hclust(dist(rock,method="euclidean"),method="centroid"), cex=0.3,main="Centroid
linkage method and euclidean distance"), fviz_dend
(hclust(dist(rock,method="euclidean"),method="ward.D2"),cex=0.3,main="Ward linkage method and euclidean
distance"), ncol=2,nrow=1)
```



```
> cor(cophenetic(hclust(dist(rock),method="centroid")),dist(rock))
[1] 0.7376795
> cor(cophenetic(hclust(dist(rock),method="ward.D2")), dist(rock))
[1] 0.6795253
```

The Optimal Number of Clusters:

Determining the ideal number of clusters, referred to as K, is a crucial task in cluster analysis. However, there is no definitive solution to this problem. To determine K, two main methods are commonly used: Direct methods such as **Elbow** and **Average Silhouette**, and Statistical testing methods like **Gap statistic**.

The **Silhouette Method** helps determine the optimal number of clusters (kkk) in **K-Means Clustering** by measuring how similar each data point is to its own cluster compared to other clusters.

Key Points:

- **Silhouette Score:** Ranges from -1 to 1. A higher score indicates better-defined clusters.
- **Calculation:** For each data point, compute the average distance to all other points in the same cluster (cohesion) and the average distance to all points in the nearest neighboring cluster (separation). The silhouette score is then calculated using these values.
- **Optimal kkk:** The value of kkk that maximizes the average silhouette score across all data points is considered optimal.

Elbow Method:

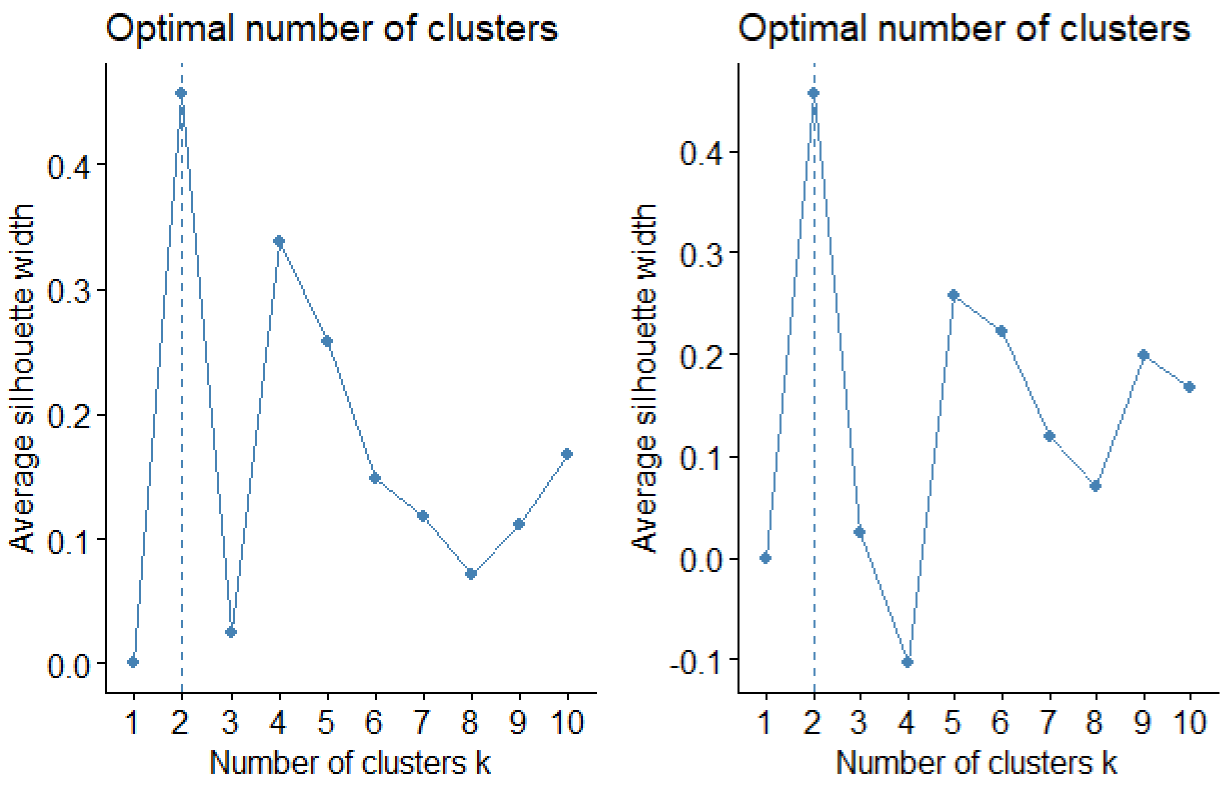
- The Elbow Method helps determine the optimal number of clusters kkk in K-Means Clustering by analyzing the Within-Cluster Sum of Squares (WCSS)**.**
- Measures how well each point fits within its cluster.
- Find the Elbow Point, where WCSS (Within-Cluster Sum of Squares) stops decreasing significantly.

- Measures how close points are to their centroids.
- Lower WCSS means tighter clusters.

The **Gap Statistic** is a statistical approach that formalizes the Elbow method.

Using the Silhouette method, all combinations suggest that the optimal K value is 2, except for the combination of Average linkage method with Euclidean distance which suggests 8 clusters and Centroid Linkage method which suggests 3 clusters.

```
> grid.arrange(fviz_nbclust(rock, hcut, method="silhouette", hc_metric="euclidean",hc_method=
"single"),fviz_nbclust(rock, hcut, method="silhouette", hc_metric="manhattan",hc_method=
"single"),ncol=2,nrow=1)
```

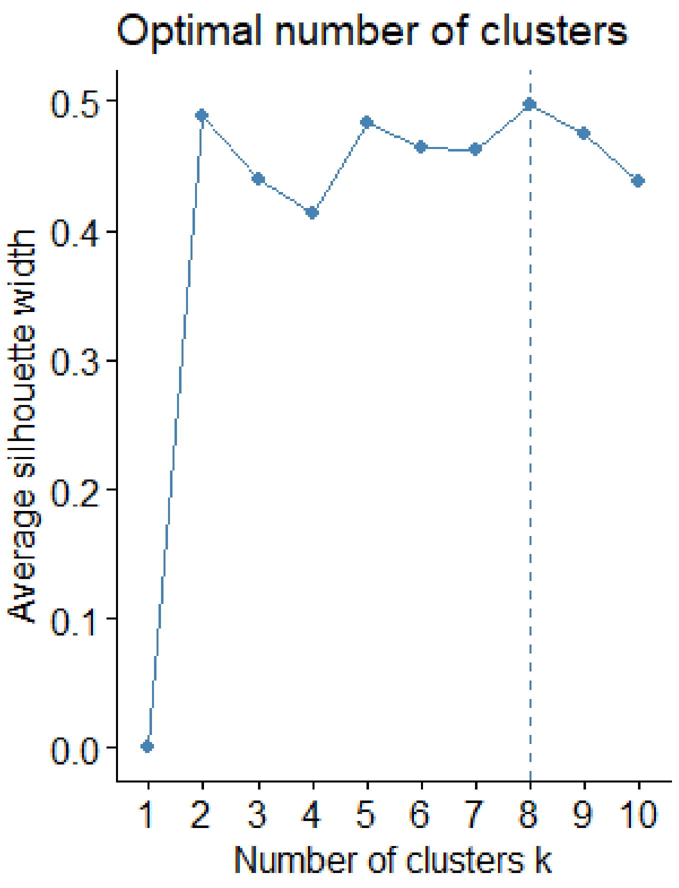


Single linkage method and euclidean distance

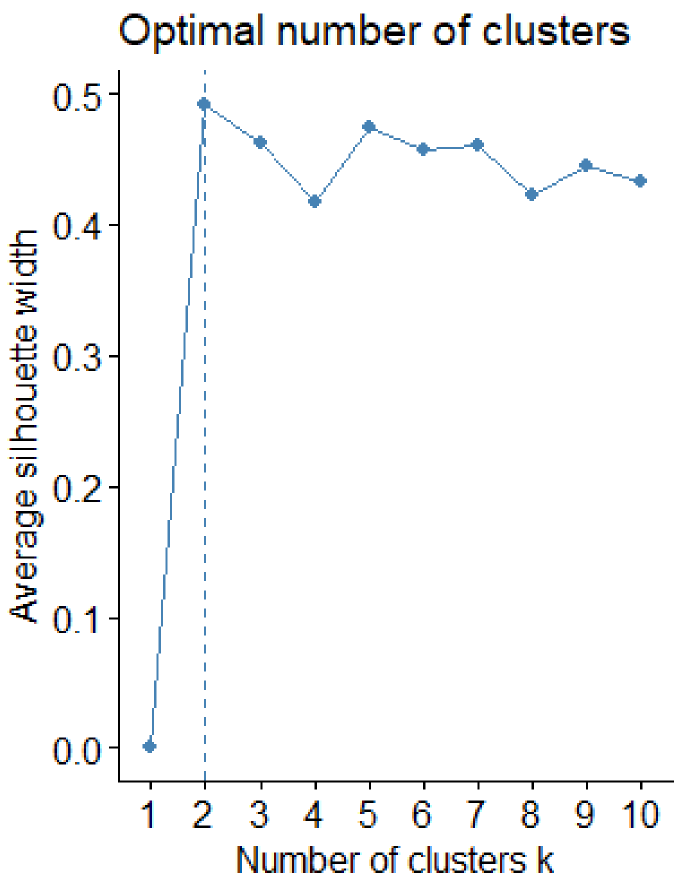
Single linkage method and manhattan distance

```
> grid.arrange(fviz_nbclust(rock, hcut, method="silhouette", hc_metric="euclidean",hc_method= "average"),
fviz_nbclust(rock, hcut, method="silhouette", hc_metric="manhattan",hc_method= "average"),ncol=2,nrow=1)
```

Average linkage method and euclidean distance

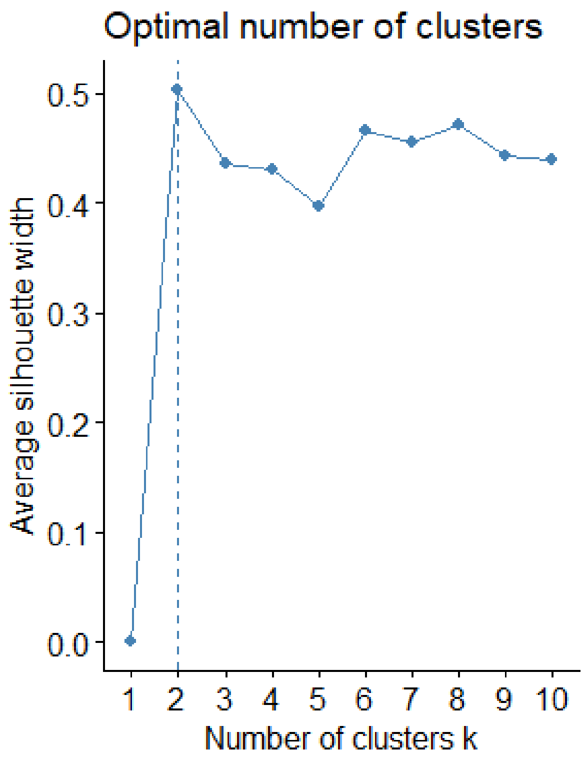


Average linkage method and manhattan distance

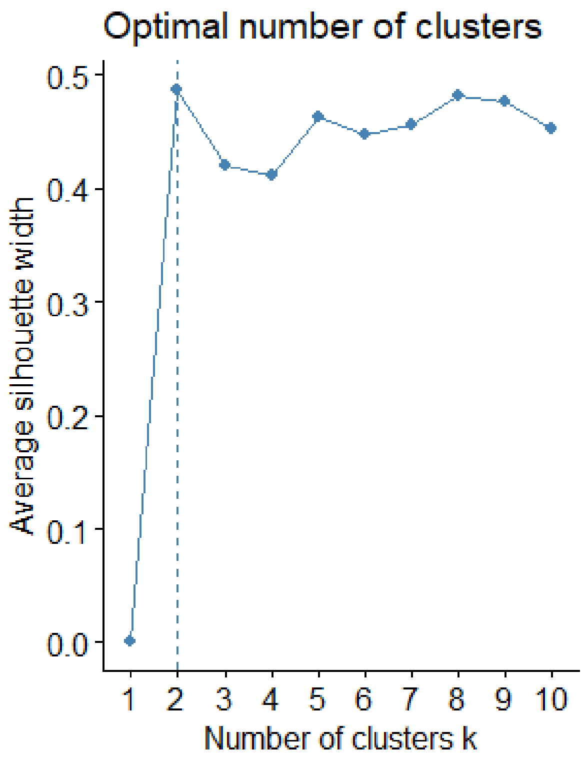


```
> grid.arrange(fviz_nbclust(rock, hcut, method="silhouette", hc_metric="euclidean",hc_method= "complete"),
fviz_nbclust(rock, hcut, method="silhouette", hc_metric="manhattan",hc_method=
"complete"),ncol=2,nrow=1)
```

Complete linkage method and euclidean distance

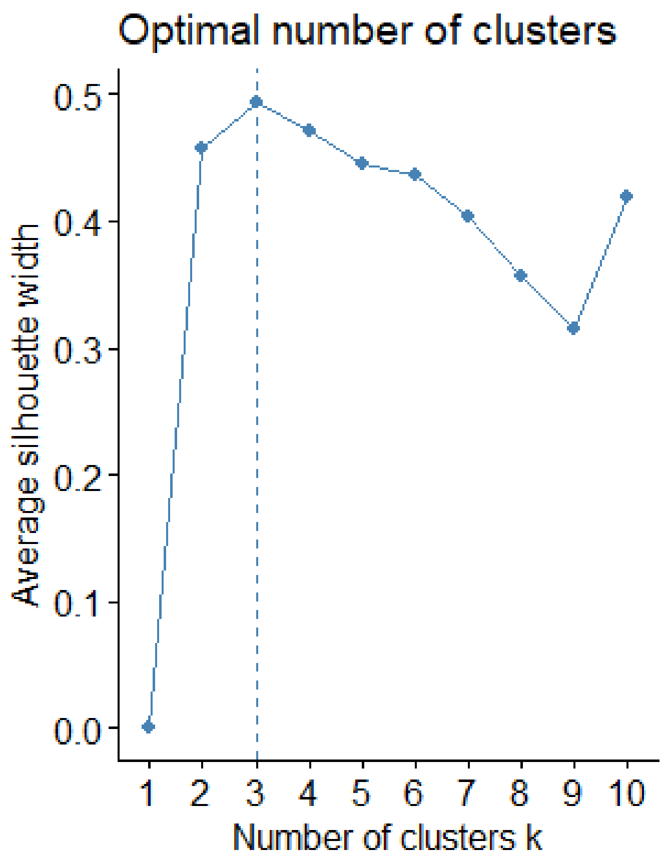


Complete linkage method and manhattan distance

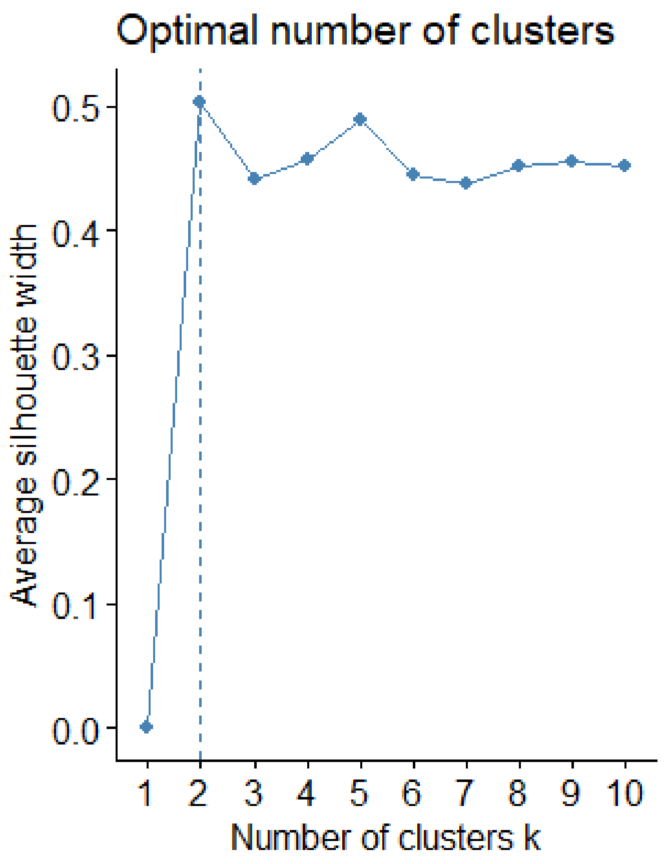


```
> grid.arrange(fviz_nbclust(rock, hcut, method = "silhouette",hc_metric="euclidean", hc_method= "centroid"),
fviz_nbclust(rock, hcut, method = "silhouette",hc_metric="euclidean", hc_method=
"ward.D2"),ncol=2,nrow=1)
```

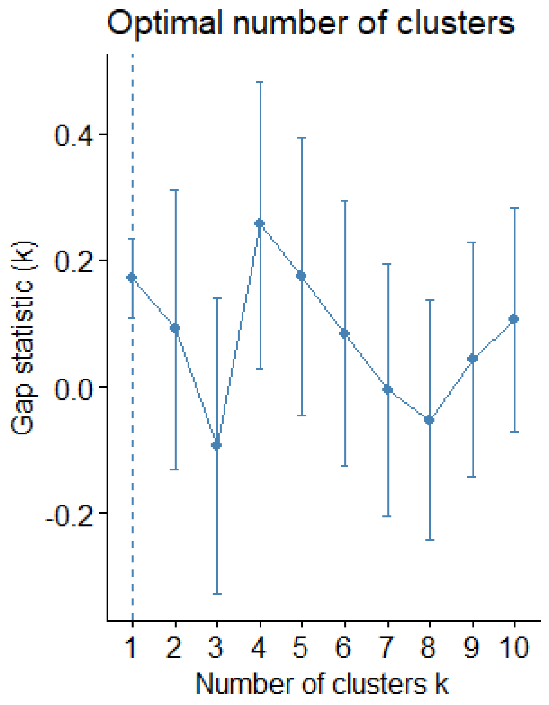
Centroid linkage method

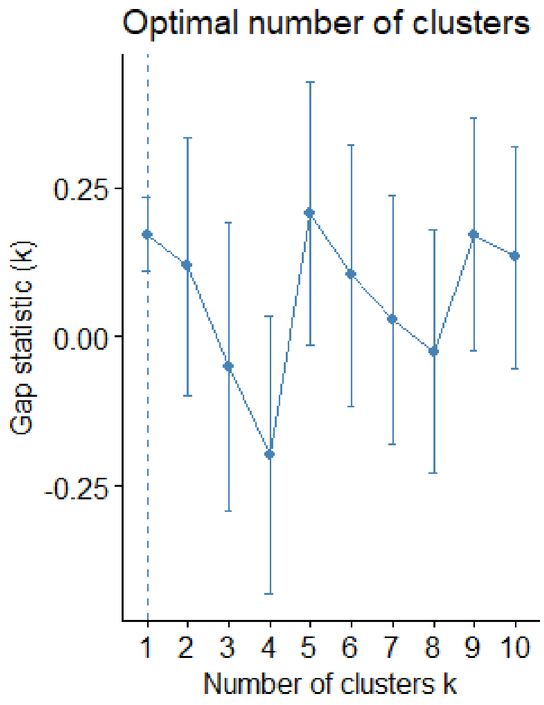


Ward linkage method



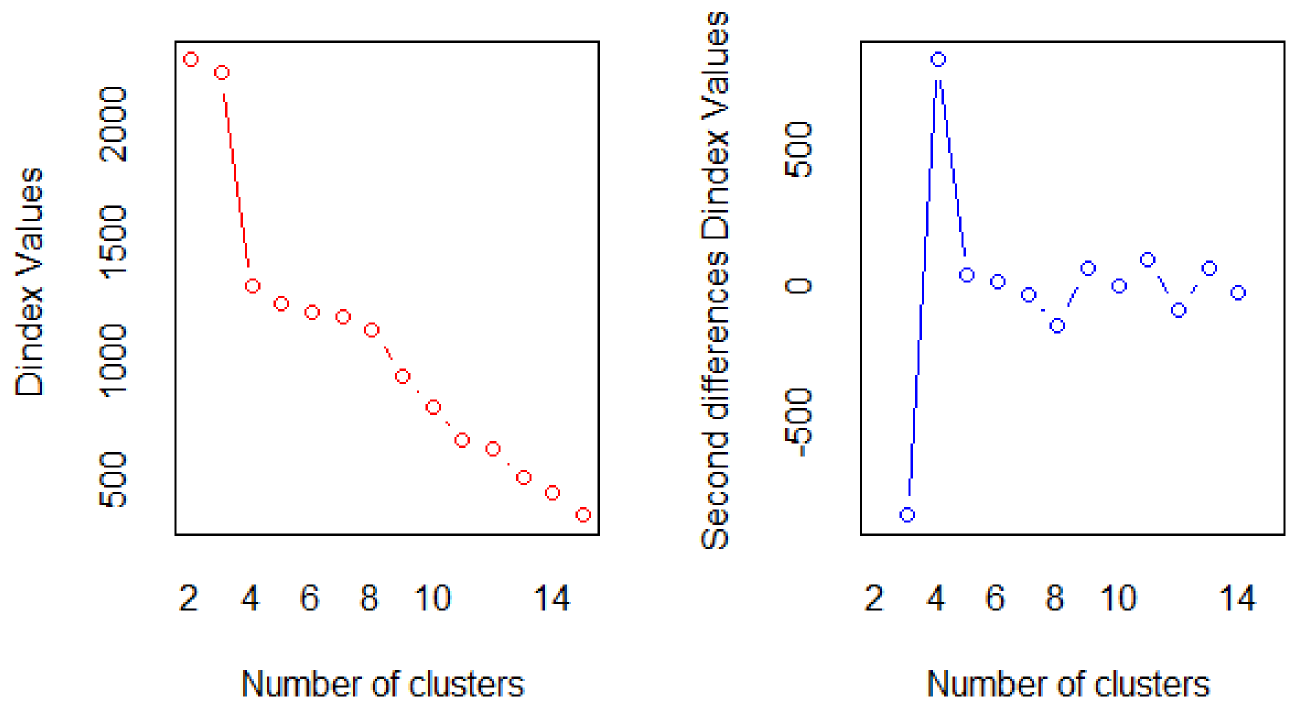
```
> grid.arrange(fviz_nbclust(rock, hcut, method = "gap_stat", nboot = 500,hc_metric="euclidean", hc_method=
"single"), fviz_nbclust(rock, hcut, method = "gap_stat", nboot = 500,hc_metric="manhattan", hc_method=
"single"),ncol=2,nrow=1)
```





```
> library(NbClust)
> grid.arrange(arrangeGrob(fviz_nbclust(NbClust(rock,distance="euclidean" ,method="single")),
top=textGrob("Single linkage method with euclidean distance",gp=gpar(fontsize=10))),
arrangeGrob(fviz_nbclust(NbClust(rock, distance="euclidean",method="average")),top=textGrob("Average
linkage method with euclidean distance",gp=gpar(fontsize=10))), arrangeGrob(fviz_nbclust(NbClust(rock,
distance="euclidean",method="complete")),top=textGrob("Complete linkage method with
euclideandistance",gp=gpar(fontsize=10))), arrangeGrob(fviz_nbclust(NbClust(rock,
distance="manhattan",method="single")),top=textGrob("Single linkage method with manhattan
distance",gp=gpar(fontsize=10))), arrangeGrob(fviz_nbclust(NbClust(rock,
distance="manhattan",method="average")), top=textGrob("Average linkage method with manhattan
distance",gp=gpar(fontsize=10))), arrangeGrob(fviz_nbclust(NbClust(rock,
distance="manhattan",method="complete")), top=textGrob("Complete linkage method with manhattan
distance",gp=gpar(fontsize=10))),ncol = 3, nrow = 2)

*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 6 proposed 2 as the best number of clusters
* 1 proposed 3 as the best number of clusters
* 8 proposed 4 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 1 proposed 13 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 6 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is 4


*******************************************************************
```

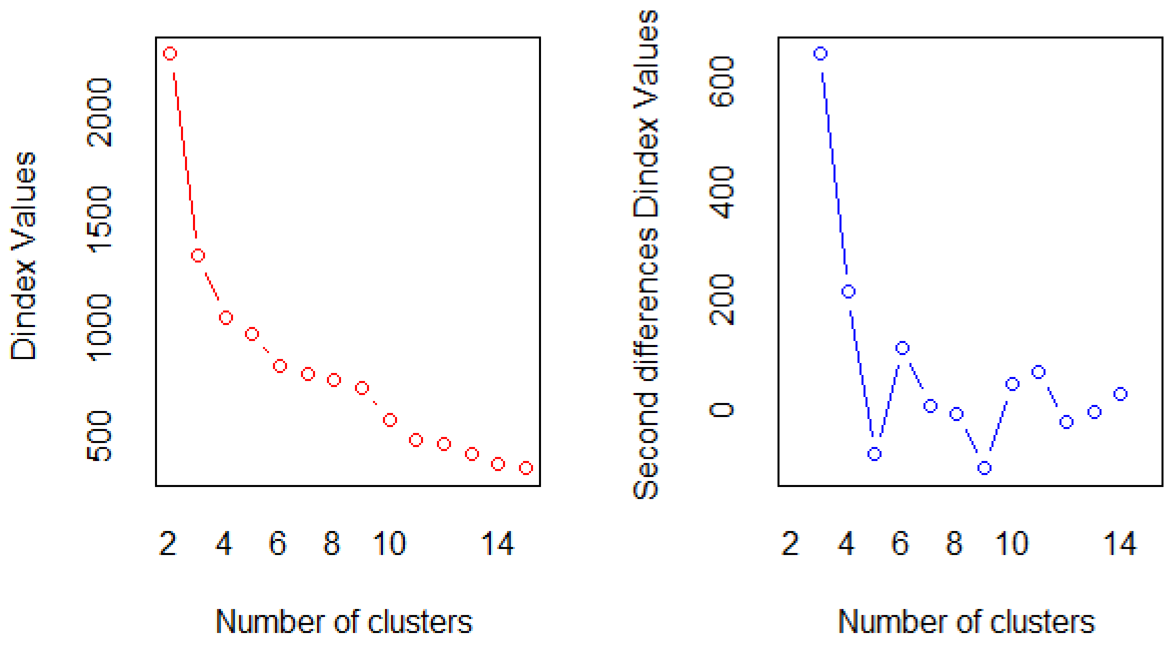
NbClust has performed the comparison between the combination of single, average, and complete linkage methods and the Euclidean and Manhattan distance. So, if we combine them it would be 6 combinations, after performing the Nbclust on these combinations we have 4 best number of clusters. Furthermore, we can perform NbClust on Centroid and Ward methods for better results.



```
> grid.arrange(arrangeGrob(fviz_nbclust(NbClust(rock, distance="euclidean",method="centroid")),
top=textGrob("Centroid linkage method with euclidean distance",gp=gpar(fontsize=10))),
arrangeGrob(fviz_nbclust(NbClust(rock, distance="euclidean",method="ward.D2")), top=textGrob("Ward
linkage method with euclidean distance",gp=gpar(fontsize=10))),ncol = 2, nrow = 1)
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 2 proposed 2 as the best number of clusters
* 9 proposed 3 as the best number of clusters
* 3 proposed 4 as the best number of clusters
* 1 proposed 5 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 2 proposed 11 as the best number of clusters
* 4 proposed 14 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                   ***** Conclusion *****
```

* According to the majority rule, the best number of clusters is 3

****



From a predetermined number of K clusters, we can use partitioning clustering techniques such as k-means and k-medoid. In the k-means algorithm, K points are randomly chosen as centroids from the dataset, and each unit in the dataset is assigned to the nearest centroid based on the Euclidean distance between the observation and the cluster mean to form clusters. Then, the algorithm computes the mean for each cluster and uses it as a new centroid, repeating the process to minimize the increase in the WSS value at each iteration.

However, k-means is not suitable for datasets with categorical variables. The results for the silhouette method and the gap statistics of the k-means algorithm suggest K=2 and K=1, respectively, and the NbClust() indicates a preferred value of K=3.

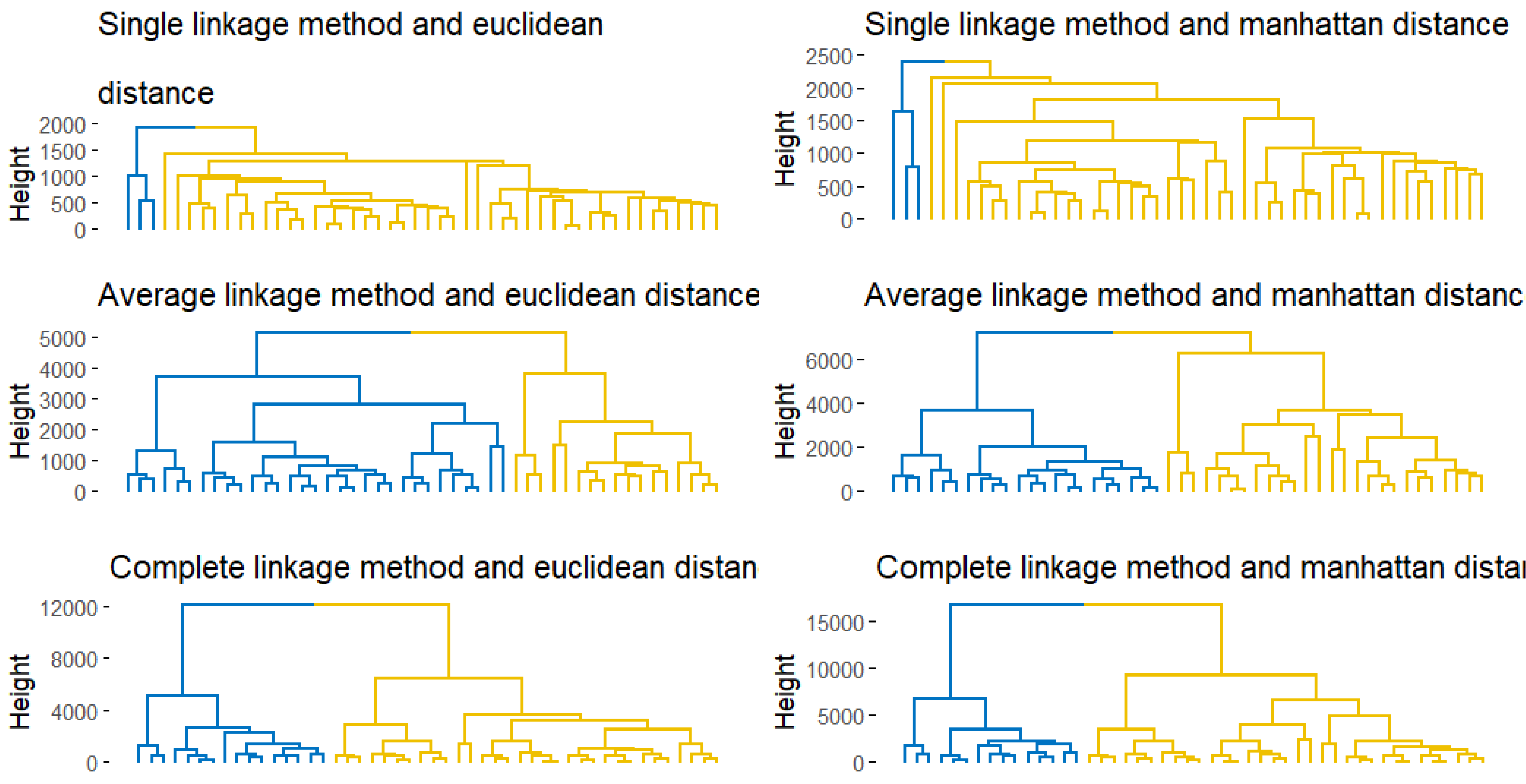
Since most of methods suggests K=2 and for some indexes K=3, now we are going to use the 3 methods (Hierarchical, K-Means, K-Medoids (PAM)), exactly for K=2 and K=3.

Cutting the Dendrograms and Verifying the Cluster Trees

When using hierarchical clustering, selecting a specific number of clusters corresponds to identifying a particular level in the dendrogram where it will be CUT.

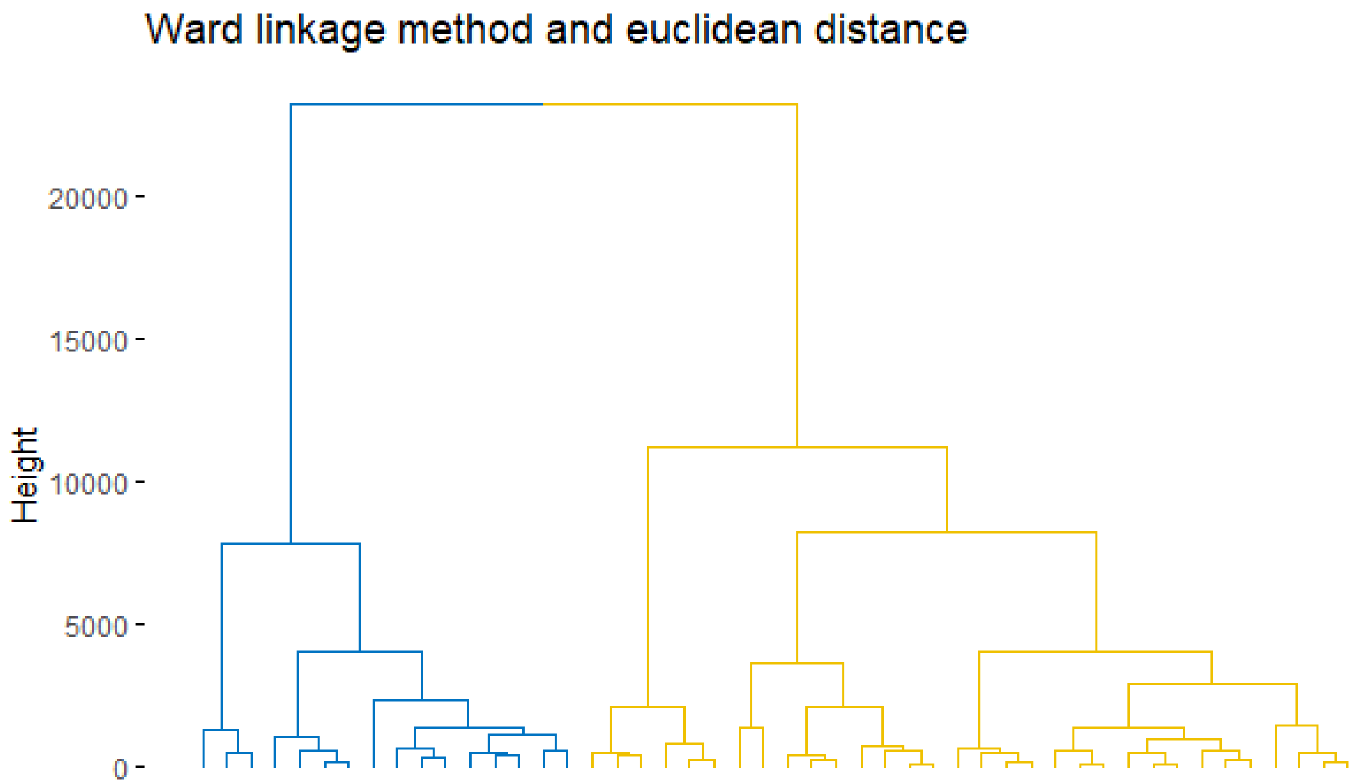
K=2 CUT

```
> grid.arrange(fviz_dend(eclust (rock, "hclust", k = 2,hc_metric = "euclidean",hc_method = "single"),
show_labels = FALSE,palette = "jco", as.ggplot = TRUE, main = "Single linkage method and euclidean
distance"), fviz_dend(eclust (rock, "hclust", k = 2,hc_metric = "manhattan",hc_method = "single"), show_labels
= FALSE,palette = "jco", as.ggplot = TRUE, main = "Single linkage method and manhattan distance"),
fviz_dend(eclust (rock, "hclust", k = 2,hc_metric = "euclidean",hc_method = "average"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Average linkage method and euclidean distance"),
fviz_dend(eclust (rock, "hclust", k = 2,hc_metric = "manhattan",hc_method = "average"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Average linkage method and manhattan distance"),
fviz_dend(eclust (rock, "hclust", k = 2,hc_metric = "euclidean",hc_method = "complete"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Complete linkage method and euclidean distance"),
fviz_dend(eclust (rock, "hclust", k = 2,hc_metric = "manhattan",hc_method = "complete"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Complete linkage method and manhattan
distance"),ncol=2,nrow=3)
```



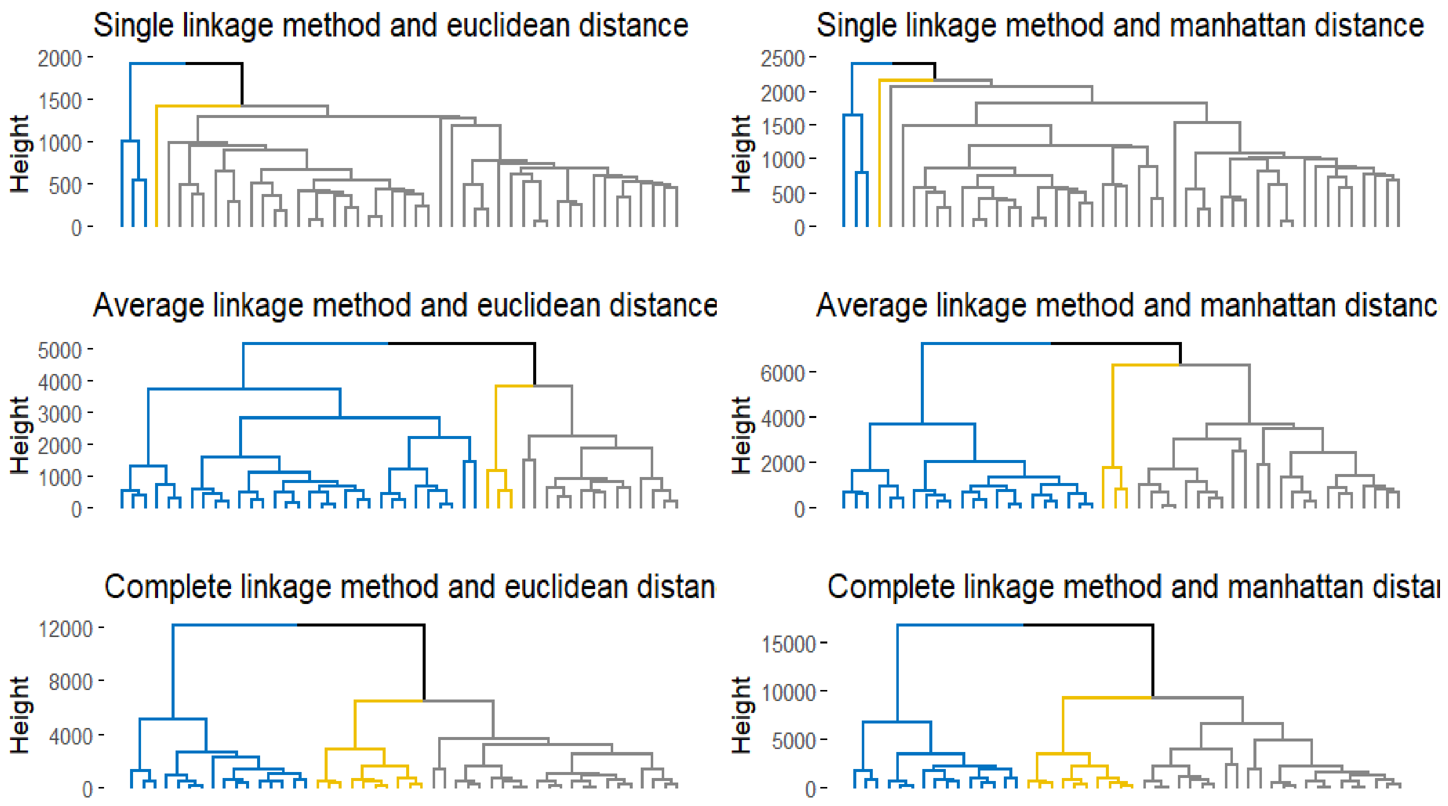
In dendrogram, height represents the distance between the units which is proportional to dissimilarity. As we can see from the visuals that same colored units are very close to each other that's the reason they have formed a branch. These branches then merge together as their Euclidean distance is less with other units, this continues until it becomes a cluster.

```
> fviz_dend(eclust (rock, "hclust", k = 2,hc_metric = "euclidean",hc_method = "ward.D2"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Ward linkage method and euclidean distance")
```

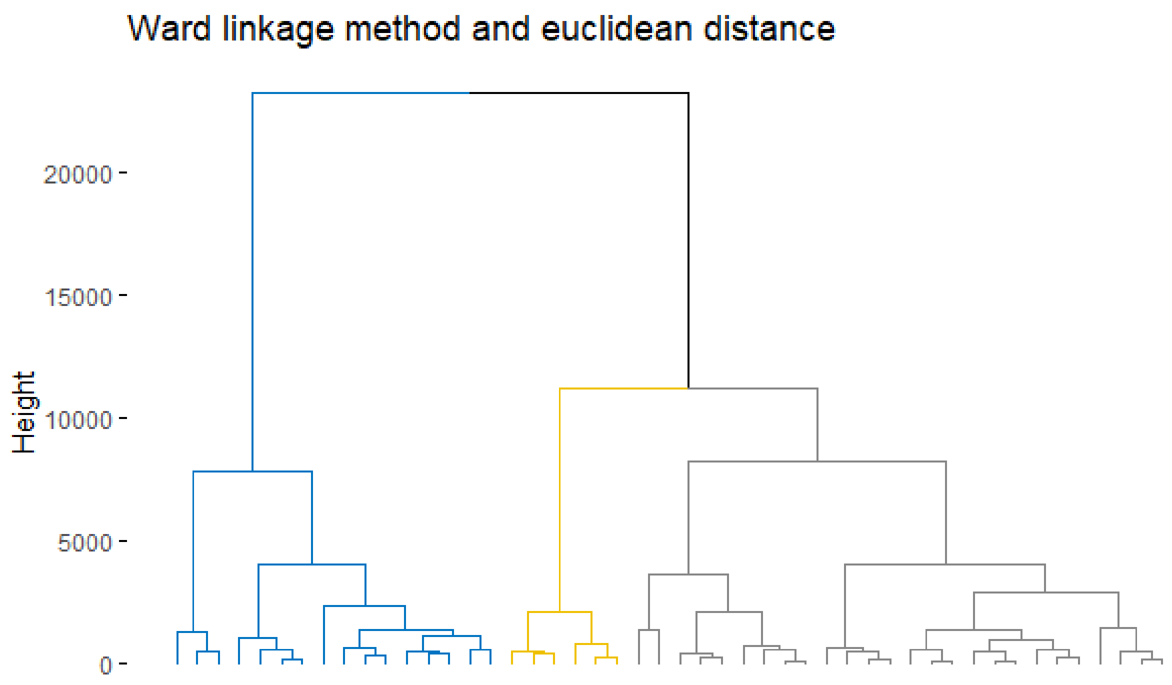


CUT K=3

```
> ## k=3 Partitioning
> #         6 type of linkages
> grid.arrange(fviz_dend(eclust (rock, "hclust", k = 3,hc_metric = "euclidean",hc_method = "single"),
show_labels = FALSE,palette = "jco", as.ggplot = TRUE, main = "Single linkage method and euclidean
distance"), fviz_dend(eclust (rock, "hclust", k = 3,hc_metric = "manhattan",hc_method = "single"), show_labels
= FALSE,palette = "jco", as.ggplot = TRUE, main = "Single linkage method and manhattan distance"),
fviz_dend(eclust (rock, "hclust", k = 3,hc_metric = "euclidean",hc_method = "average"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Average linkage method and euclidean distance"),
fviz_dend(eclust (rock, "hclust", k = 3,hc_metric = "manhattan",hc_method = "average"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Average linkage method and manhattan distance"),
fviz_dend(eclust (rock, "hclust", k = 3,hc_metric = "euclidean",hc_method = "complete"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Complete linkage method and euclidean distance"),
fviz_dend(eclust (rock, "hclust", k = 3,hc_metric = "manhattan",hc_method = "complete"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Complete linkage method and manhattan
distance"),ncol=2,nrow=3)
```



```
> #	Ward Linkage
> fviz_dend(eclust (rock, "hclust", k = 3,hc_metric = "euclidean",hc_method = "ward.D2"), show_labels =
FALSE,palette = "jco", as.ggplot = TRUE, main = "Ward linkage method and euclidean distance")
```



Earlier, we learned that computing the correlation between cophenetic distances and the original distances (known as the dissimilarity matrix) is a method of validating hierarchical clustering dendrograms, which evaluates how accurately the clustering result represents the dataset.

Partitioning Clustering:

Partitioning Clustering is a type of clustering algorithm that divides a dataset into a specified number of non-overlapping groups (or clusters) based on the similarity of the data points.

Key Features:

- **Non-hierarchical**: Unlike hierarchical clustering, partitioning clustering doesn’t create a tree-like structure.
- **Predefined Clusters**: You must specify the number of clusters (e.g., **k** clusters) before running the algorithm.

K-Means Clustering:-

K-Means is a **partition-based clustering** algorithm that divides a dataset into kkk clusters by minimizing the variance within each cluster.

Steps of K-Means Clustering:

1. **Initialization**:

   Choose k initial centroids (which are the mean of data points) randomly from the data points.

2. **Assign Points to Clusters**:

   Assign each data point to the nearest centroid (using Euclidean distance).

3. **Update Centroids**:

   Recalculate centroids as the mean of all points in each cluster.

4. **Repeat**

   Repeat the **assigning** and **updating centroids** steps until (no centroid change).

5. **Output**:

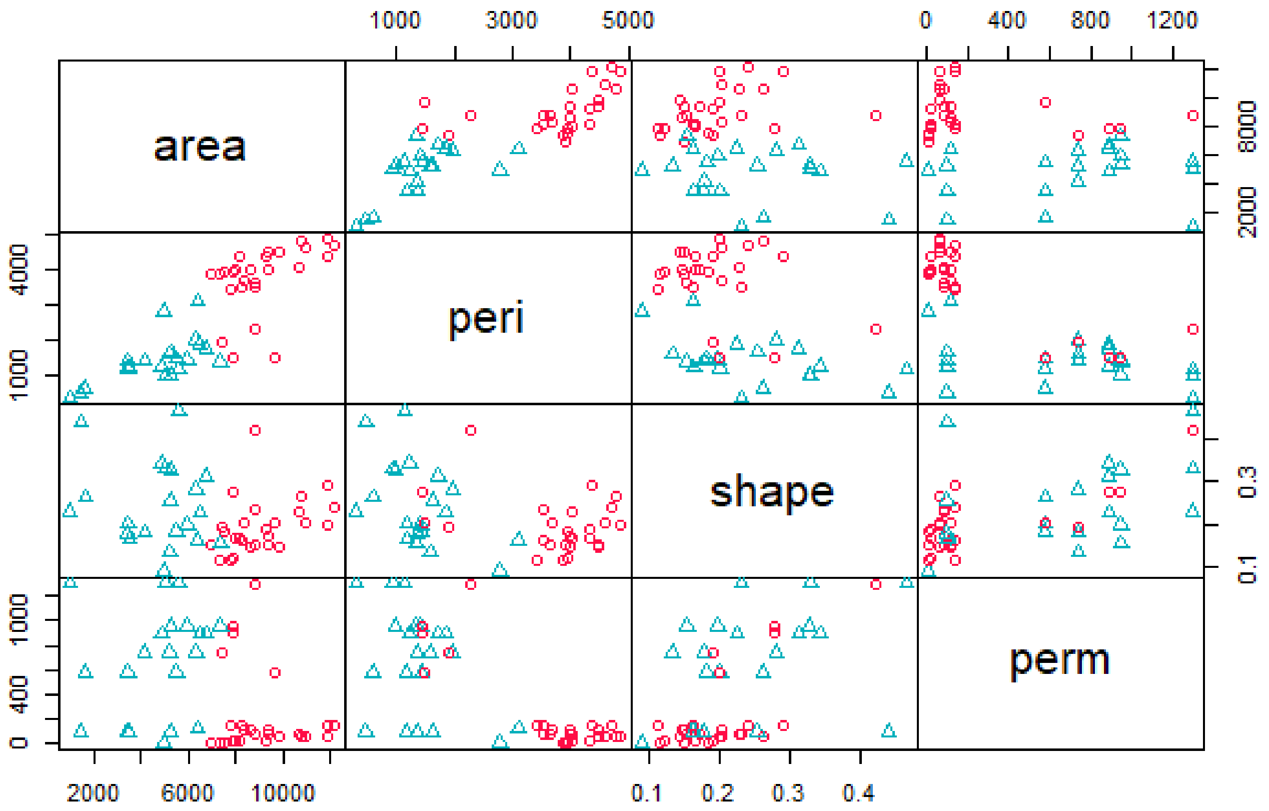
   The algorithm outputs the final clusters and their centroids.

K-means; k=2:

```
> km.res <- kmeans(rock #dataset , 2  #number of clusters, nstart = 25 #The algorithm will try 25 random initializations of the
  centroids.)
> km.res$size # Get the size of each cluster
[1] 27 21
```

Running this k-means algorithm we obtain two clusters: the first is the bigger one, containing 27 units, while the second is the smallest with 21 units. We can visualize the cluster results in the original space:

```
cl <- km.res$cluster
pairs(rock, gap=0, pch=cl, col=c("#ff0d45", "#00AFBB")[cl])
```



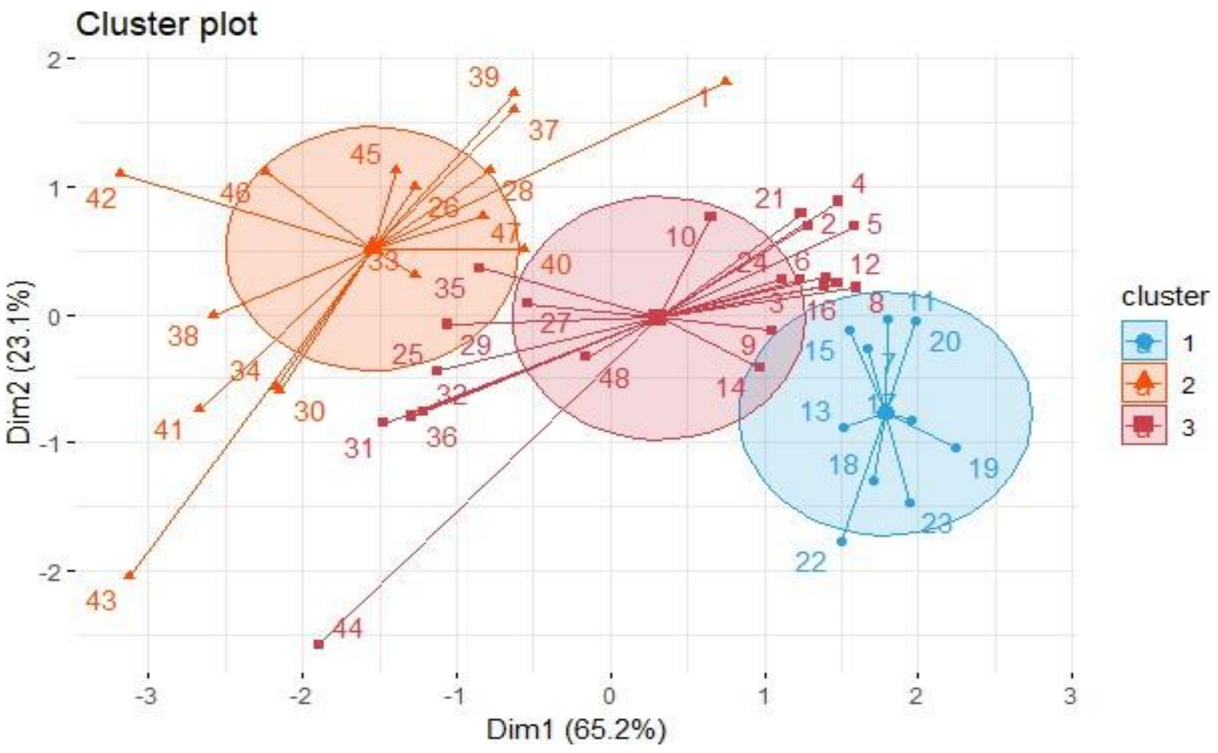
K-means; k=3:

```
> km.res2 <- kmeans(rock, 3, nstart = 25)
> km.res2$size
[1] 10 16 22
```

This time we have 3 clusters, the largest being the third.

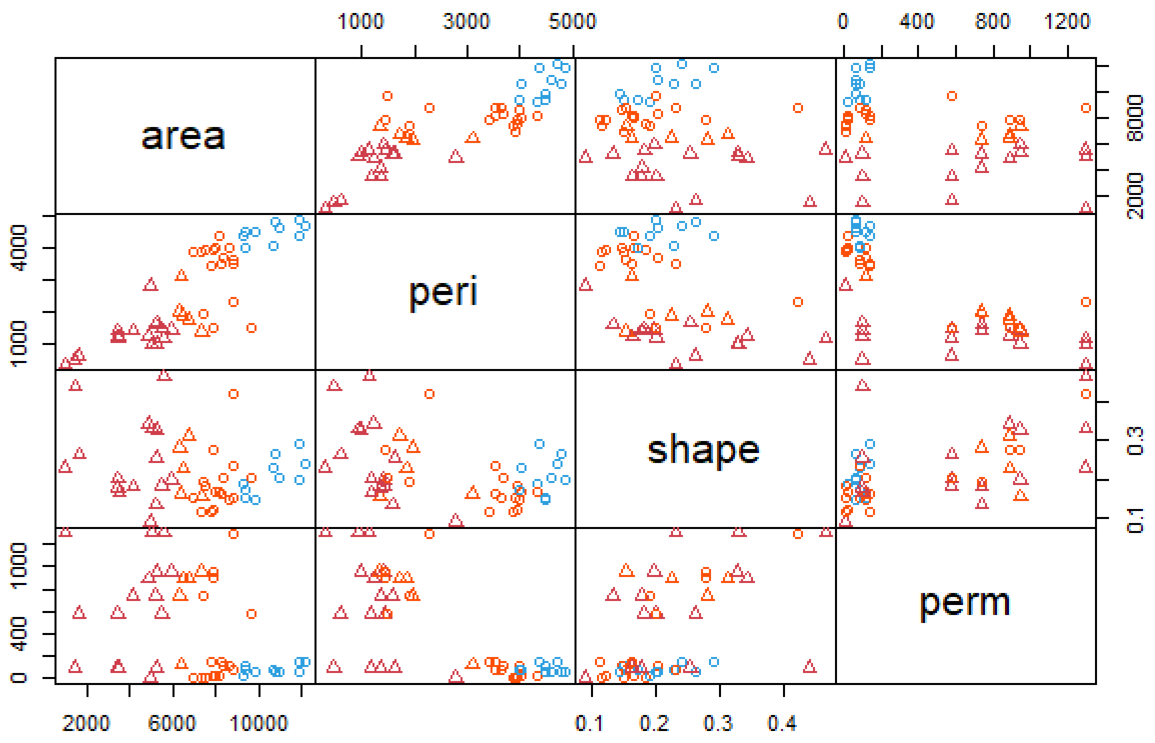
```
> fviz_cluster(km.res2, data = rock, palette = c("#2E9FDF", "#FC4E07", "#CF3E4B"), ellipse.type = "euclid",
star.plot = TRUE, repel = TRUE, ggtheme = theme_minimal() )
```

- **fviz_cluster(km.res2, data = rock, ...)** → Visualizes clusters from the K-means result `km.res2`.
- **palette = c("#2E9FDF", "#FC4E07", "#CF3E4B")** → Specifies colors for clusters.
- **ellipse.type = "euclid"** → Draws **Euclidean distance-based** cluster ellipses.
- **star.plot = TRUE** → Draws lines connecting cluster points to their centroids.
  - **repel = TRUE** → Adjusts labels to avoid overlap.
  - **ggtheme = theme_minimal()** → Uses a minimalistic ggplot



The Dim1 and Dim2, which account for 65.2% and 23.3% of the total variance respectively. Most of the units are correctly classified by K-means but still cluster 1 and 3 are not well separated. There are some units which can be considered as outliers, like unit 43 and 44 which are the parts of cluster 2 and 3 respectively.

```
> cl2 <- km.res2$cluster
> pairs(rock, gap=0, pch=cl, col=c("#2E9FDF", "#FC4E07","#CF3E4B")[cl2])
```



As regard to this plot, the considerations are more or less the same as for the algorithm with 2 clusters. However, in this case, with 3 clusters, it seems a little bit difficult to see separation between clusters, which appear mostly over-lapped each other.
In fact, as we have already seen, in the space spanned by the first 2 PCs several units are closer to other clusters.

K-medoid:-

K-medoids clustering is an unsupervised learning algorithm that partitions a dataset into kkk clusters, each represented by a central data point known as a "medoid (actual data point closest to the center of the cluster)" Unlike k-means, which uses centroids that may not correspond to actual data points, k-medoids selects representative points from the dataset, enhancing robustness to outliers.

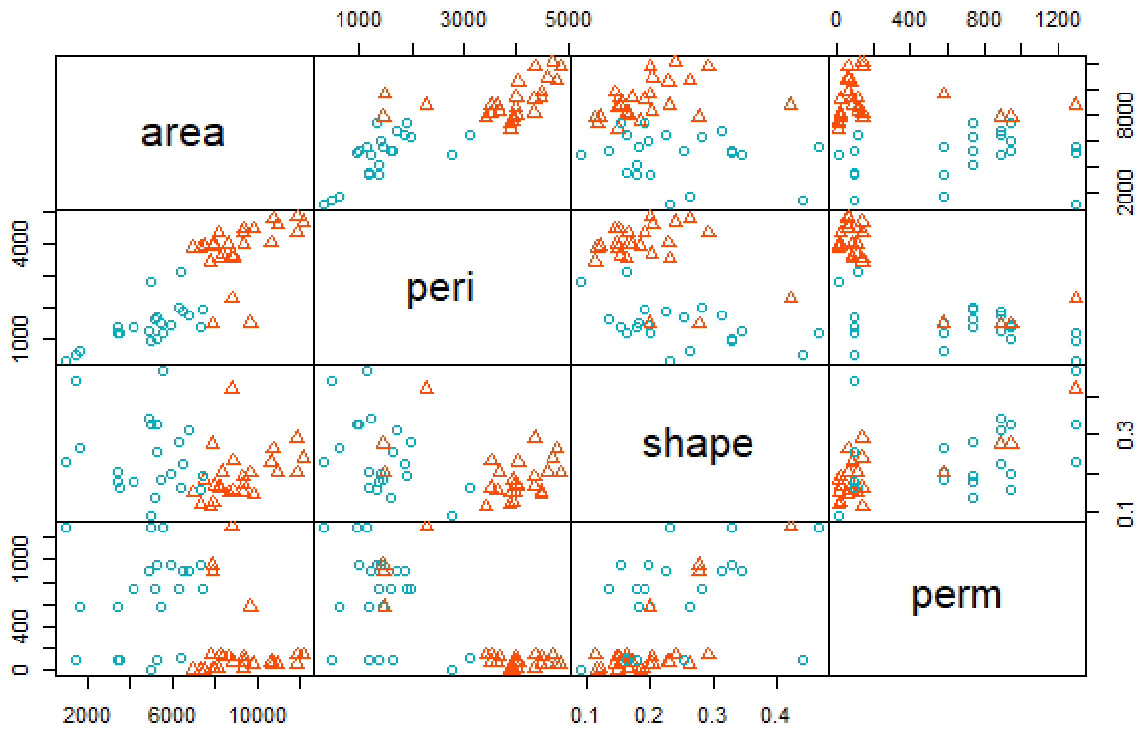
Algorithm Steps:

1. **Initialization:** Select kkk initial medoids randomly from the dataset.
2. **Assignment:** Assign each data point to the nearest medoid based on the chosen distance metric.
3. **Update:** For each cluster, identify the data point that minimizes the total dissimilarity within the cluster and designate it as the new medoid.
4. **Iteration:** Repeat the assignment and update steps until the medoids stabilize or a predefined number of iterations is reached.

```
> library(cluster)
> pam.res <- pam(rock, 2)
> pam.res$medoids
     area      peri       shape      perm
[1,] 5246   1585.42   0.133083   740.0
[2,] 8874   3629.07   0.153481   82.4
```

Here we can see the medoids, namely the representative point, of both clusters. We can also see the values of the variables relative to each medoid.

```
> c2=pam.res$clustering
> pairs(rock, gap=0, pch=c2, col= c("#00AFBB", "#FC4E07")[c2] )
```



Model-Based Clustering:-

Model-based clustering is a way of grouping data points based on statistical models, which is different from traditional methods that use arbitrary rules. We will use a type of statistical model called parsimonious Gaussian mixtures, which is a simple model that uses a low number of parameters to explain the data. This type of model is good for modeling different types of data patterns. However, Gaussian mixture models can be complicated, so we need to simplify them to make them easier to use. We will do this by reducing the number of parameters using a method called eigen-decomposition.

When fitting the model to the data, we need to find a balance between how well the model fits the data and how many parameters it uses. If the increase in fit is small, it's not worth using a more complex model. We want to

use a simple model that explains the data well. To do this, we will use a selection criterion, like the Bayesian Information Criterion (BIC), to choose the best model with the right number of clusters and parameters.

In the mclust package, the "VVV" model refers to a Gaussian mixture model with the following characteristics:

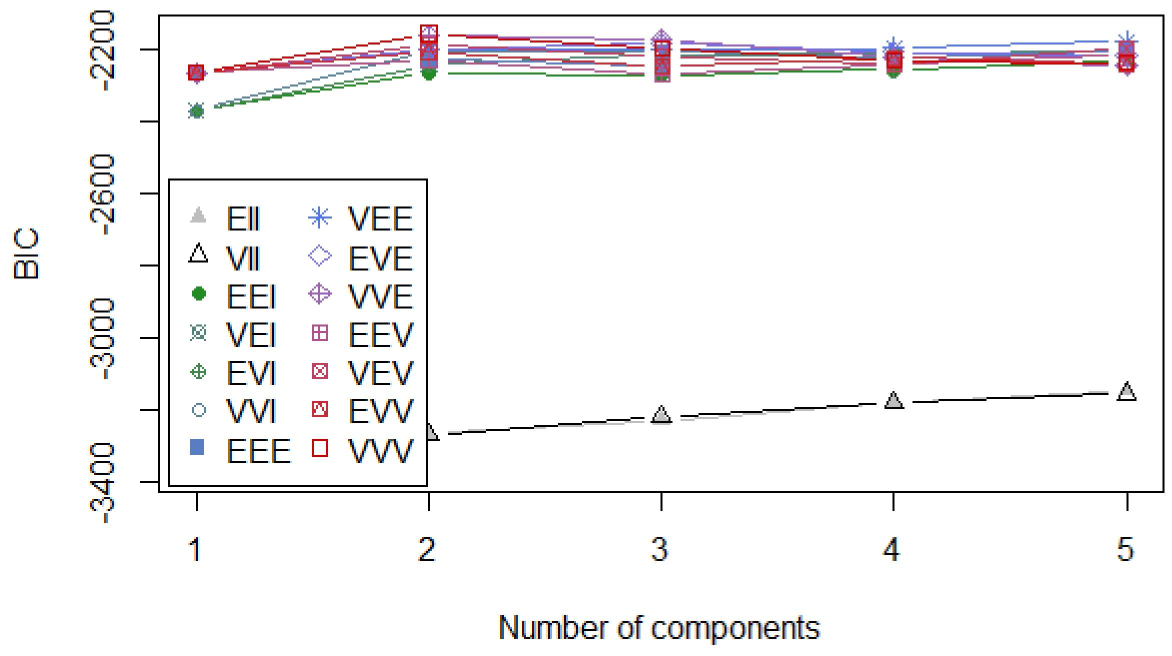
- **Ellipsoidal Shape:** Clusters are modeled as ellipsoids, allowing for varying shapes.
- **Varying Volume:** Each cluster can have a different volume, accommodating clusters of different sizes.
- **Varying Orientation:** Clusters can have different orientations, enabling flexibility in cluster alignment.

- rock: The dataset to be clustered.
- G = 1:5: Specifies the range of possible cluster numbers (from 1 to 5).
- modelNames = NULL: Indicates that all available models should be considered.

```
> # Model Based Clustring
> library(mclust)
> mod <- Mclust(rock, G = 1:5, modelNames = NULL)
fitting ...
  |=============================================================| 100%
> summary(mod$BIC)
Best BIC values:
                    VVV,2          VVE,2           VVE,3
BIC                 -2157.856      -2161.240237    -2176.51261
BIC diff            0.000          -3.383916       -18.65629
```

Based on the Bayesian Information Criterion (BIC), we have identified the top three models. The model that has the lowest BIC value is VVV, which implies that the clusters have variable volume, the same shape, and the same orientation. This model has two clusters. The second-best model is VVE with two clusters, and the last model is VVE, which indicates that the clusters have variable volume, variable shape, and the same orientation. This model also has three clusters.

In your results, the "VVV,2" model has the lowest BIC value of -2157.856, indicating it is the preferred model among the options tested. Therefore, the "VVV,2" model is the best fit for data.



```
> plot(mod, what = "BIC", ylim = range(mod$BIC, na.rm = TRUE), legendArgs = list(x = "bottomleft"))
```

- mod: The Mclust object containing the clustering results.
- what = "BIC": Specifies that the plot should display the BIC values.
- ylim = range(mod$BIC, na.rm = TRUE): Sets the y-axis limits to the range of BIC values, excluding any NA values.
- legendArgs = list(x = "bottomleft"): Positions the legend at the bottom left of the plot.

```
> summary(mod)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------


Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model
with 2 components:

 log-likelihood    n   df   BIC        ICL
      -1022.796   48   29   -2157.856  -2157.856

Clustering table:
 1    2
24   24
```

Here we can see that BIC = -2157.856 is the best value for BIC, which must be maximized (in fact, this is the lowest negative value). Moreover, according to this best model, VVV, both of clusters are characterized by 24 observations each. This classification is based on the maximum a posteriori probability approach, which starts from the soft assignment, according to which each unit has a posterior probability to belong to a different cluster:

```
> head(mod$classification, 24)
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

By using the head () function, you can view the first 24 cluster assignments