Cluster Analysis

library(RCurl)

# Collected data from github repository  
x <- getURL("https://raw.githubusercontent.com/adeelahmedcsku/student-dataset/main/20-01-2020-data%20developed%20records%20students-changes%20in%20grading%20system%20-%20data%20developed%20records%20students.csv")  
studentsData <- read.csv(text = x)

studentsData$Gender =as.numeric(factor(studentsData$Gender))  
str(studentsData$Gender)

## num [1:939] 2 2 2 2 2 2 2 1 1 1 ...

Factoring the program attribute:

studentsData$Program =as.numeric(factor(studentsData$Program))  
str(studentsData$Program)

## num [1:939] 2 2 2 2 2 2 2 2 2 2 ...

Factoring the Is.Research.Oriented attribute:

studentsData$Is.Research.Oriented = as.numeric(factor(studentsData$Is.Research.Oriented))  
str(studentsData$Is.Research.Oriented)

## num [1:939] 2 2 1 1 1 1 1 2 2 2 ...

Factoring the Student.Type attribute:

studentsData$Student.Type = as.numeric(factor(studentsData$Student.Type))  
str(studentsData$Student.Type)

## num [1:939] 2 1 2 2 1 1 2 2 2 1 ...

Factoring the Financial.Condition attribute:

studentsData$Financial.Condition = as.numeric(factor(studentsData$Financial.Condition))  
str(studentsData$Financial.Condition)

## num [1:939] 3 2 3 1 3 3 2 1 2 1 ...

Factoring the Class attribute:

studentsData$Class = as.numeric(factor(studentsData$Class))  
str(studentsData$Class)

## num [1:939] 2 1 3 1 2 2 5 1 6 3 ...

# Cluster Analysis

Clustering is one of the important data mining methods for discovering knowledge in multidimensional data. The goal of clustering is to identify pattern or groups of similar objects within a data set of interest. In the literature, it is referred as “pattern recognition” or “unsupervised machine learning” - “unsupervised” because we are not guided by a priori ideas of which variables or samples belong in which clusters. “Learning” because the machine algorithm “learns” how to cluster.

The Clustering involves several steps. The first step is to calculate the dissimilarity matrix amoung the cluster units.

The value of distance measures is intimately related to the scale on which measurements are made. Therefore, variables are often scaled (i.e. standardized) before measuring the inter-observation dissimilarities.

studentsData.scaled <- scale(studentsData)

The Euclidean distance of the dataset is show below.

dist.eucl <- dist(studentsData.scaled, method = "euclidean")  
dist.eucl.matrix<-round(as.matrix(dist.eucl)[1:13, 1:13], 2)  
rownames(dist.eucl.matrix) <- c("G","B.Y","Pro","S.M","C.P","R.O","S.T","F.C","A.M","F.E.M","Total","CGPA","Class")  
colnames(dist.eucl.matrix) <- c("G","B.Y","Pro","S.M","C.P","R.O","S.T","F.C","A.M","F.E.M","Total","CGPA","Class")  
dist.eucl.matrix

## G B.Y Pro S.M C.P R.O S.T F.C A.M F.E.M Total CGPA Class  
## G 0.00 3.50 4.66 3.80 3.28 3.42 5.66 4.69 7.40 5.69 4.75 2.57 5.01  
## B.Y 3.50 0.00 5.24 4.06 3.14 3.26 5.11 4.57 7.39 4.18 2.52 3.43 4.98  
## Pro 4.66 5.24 0.00 3.84 4.08 4.53 3.10 4.46 4.70 4.87 5.68 4.45 3.02  
## S.M 3.80 4.06 3.84 0.00 3.96 3.96 4.48 3.89 6.57 4.93 4.74 3.23 5.14  
## C.P 3.28 3.14 4.08 3.96 0.00 2.50 4.69 4.82 7.23 4.76 4.43 3.53 4.28  
## R.O 3.42 3.26 4.53 3.96 2.50 0.00 5.67 5.78 8.08 5.77 4.51 4.73 5.17  
## S.T 5.66 5.11 3.10 4.48 4.69 5.67 0.00 4.58 4.16 4.08 5.21 4.63 4.34  
## F.C 4.69 4.57 4.46 3.89 4.82 5.78 4.58 0.00 5.06 2.71 4.55 3.32 4.38  
## A.M 7.40 7.39 4.70 6.57 7.23 8.08 4.16 5.06 0.00 4.64 6.68 6.25 4.53  
## F.E.M 5.69 4.18 4.87 4.93 4.76 5.77 4.08 2.71 4.64 0.00 3.79 4.46 3.99  
## Total 4.75 2.52 5.68 4.74 4.43 4.51 5.21 4.55 6.68 3.79 0.00 4.31 5.54  
## CGPA 2.57 3.43 4.45 3.23 3.53 4.73 4.63 3.32 6.25 4.46 4.31 0.00 4.53  
## Class 5.01 4.98 3.02 5.14 4.28 5.17 4.34 4.38 4.53 3.99 5.54 4.53 0.00

The most dissimilarity we can see in the table is between Attendance Marks(A.M) and Research Oriented (R.O) which is 8.08. while the most similar is class participation marks and research oriented attribute which 2.50.

We can also try to compute the Manhattan distance:

dist.man <- dist(studentsData.scaled, method = "manhattan")  
dist.man.matrix<-round(as.matrix(dist.man)[1:13, 1:13], 2)  
rownames(dist.man.matrix) <-c("G","B.Y","Pro","S.M","C.P","R.O","S.T","F.C","A.M","F.E.M","Total","CGPA","Class")  
colnames(dist.man.matrix)<-c("G","B.Y","Pro","S.M","C.P","R.O","S.T","F.C","A.M","F.E.M","Total","CGPA","Class")  
dist.man.matrix

## G B.Y Pro S.M C.P R.O S.T F.C A.M F.E.M Total CGPA  
## G 0.00 9.49 11.99 9.39 7.10 7.28 15.32 12.01 20.56 15.45 13.88 6.97  
## B.Y 9.49 0.00 14.95 9.84 7.56 8.27 14.86 12.17 21.85 11.86 5.17 6.06  
## Pro 11.99 14.95 0.00 8.82 9.97 12.32 7.63 11.87 13.45 11.76 16.69 12.01  
## S.M 9.39 9.84 8.82 0.00 9.70 9.72 11.76 9.07 19.72 13.33 13.18 7.15  
## C.P 7.10 7.56 9.97 9.70 0.00 5.38 13.30 13.98 22.54 13.43 12.72 9.04  
## R.O 7.28 8.27 12.32 9.72 5.38 0.00 15.86 17.22 25.77 16.86 12.66 13.25  
## S.T 15.32 14.86 7.63 11.76 13.30 15.86 0.00 12.95 10.99 11.21 15.36 12.30  
## F.C 12.01 12.17 11.87 9.07 13.98 17.22 12.95 0.00 12.23 6.52 10.64 8.23  
## A.M 20.56 21.85 13.45 19.72 22.54 25.77 10.99 12.23 0.00 12.92 18.34 16.67  
## F.E.M 15.45 11.86 11.76 13.33 13.43 16.86 11.21 6.52 12.92 0.00 9.34 12.68  
## Total 13.88 5.17 16.69 13.18 12.72 12.66 15.36 10.64 18.34 9.34 0.00 9.43  
## CGPA 6.97 6.06 12.01 7.15 9.04 13.25 12.30 8.23 16.67 12.68 9.43 0.00  
## Class 12.73 12.56 5.61 14.43 10.70 13.94 12.16 10.95 11.83 7.90 14.30 11.87  
## Class  
## G 12.73  
## B.Y 12.56  
## Pro 5.61  
## S.M 14.43  
## C.P 10.70  
## R.O 13.94  
## S.T 12.16  
## F.C 10.95  
## A.M 11.83  
## F.E.M 7.90  
## Total 14.30  
## CGPA 11.87  
## Class 0.00

The maximum distance is between Research Oriented and Attendance marks is maximum and minimum distance is between class participation and research oriented which is same as euclidean distance. So there is no difference in these two distance matrices we can say that there is no outlier in our dataset.

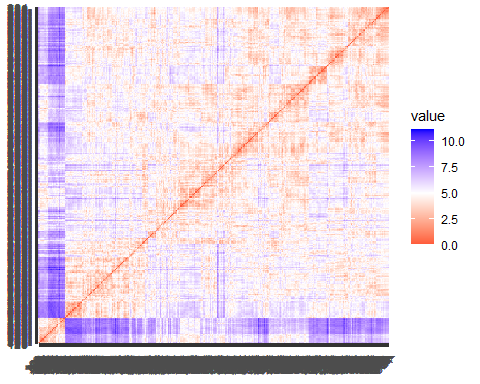
We can do the same observations by looking at the graphical visualization of the distance matrix:

library(factoextra)

## Loading required package: ggplot2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

fviz\_dist(dist.eucl)

 The red represents high similarity and blue represents low dissimilarity.

# Hierarchical clustering method

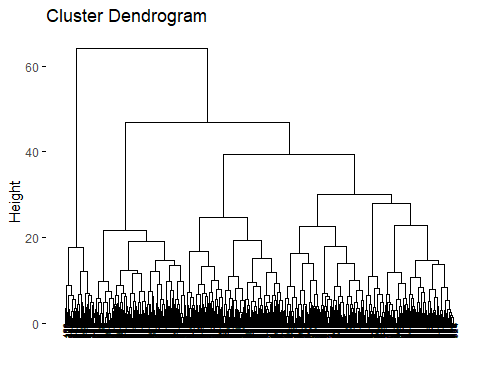
Hierarchical clustering is a method of CA which seeks to build a hierarchy of clusters. Hierarchical clustering methods do not require the number of clusters K as an input (i.e., to be fixed in advance).

Strategies for hierarchical clustering generally fall into two types: 1. Agglomerative hierarchical clustering 2. Divisive hierarchical clustering

#Agglomerative hierarchical clustering

## Using Euclidean Distance and Ward’s Linkage Methods:

res.hc <- hclust(d = dist.eucl, method = "ward.D2")  
fviz\_dend(res.hc, cex = 0.5)

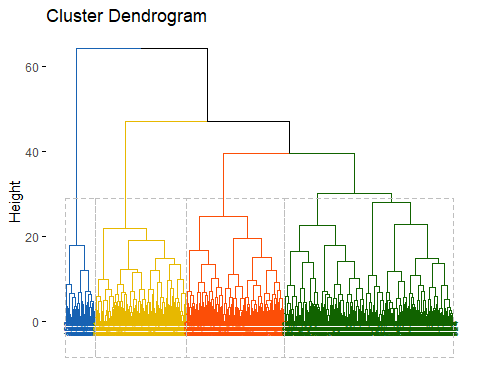
 The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the cophenetic distance between the two objects.

# Compute cophentic distance  
res.coph <- cophenetic(res.hc)

# Correlation between cophenetic distance and  
# the original distance  
cor(dist.eucl, res.coph)

## [1] 0.6551251

grp <- cutree(res.hc, k = 4)  
 fviz\_dend(res.hc, k = 4, cex = 0.5, k\_colors = c("#1762b3", "#E7B800", "#FC4E07","#116300"), color\_labels\_by\_k = TRUE,rect = TRUE)

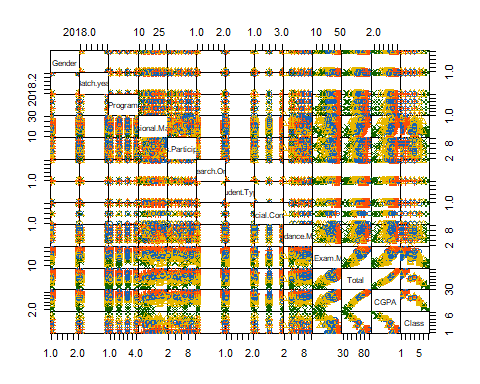
 The last cluster is bigger than the first two clusters

table(grp)

## grp  
## 1 2 3 4   
## 410 236 221 72

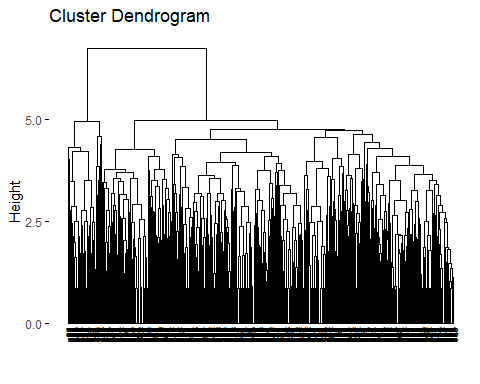
We can visualize the clustering results in the original space via the matrix of pairwise scatterplots:

pairs(studentsData, gap=0, pch=grp, col=c("#1762b3", "#E7B800", "#FC4E07","#116300")[grp])



## Using Euclidean Distance and Average Linkage Methods:

res.hc <- hclust(d = dist.eucl, method = "average")  
fviz\_dend(res.hc, cex = 0.5)



The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the cophenetic distance between the two objects.

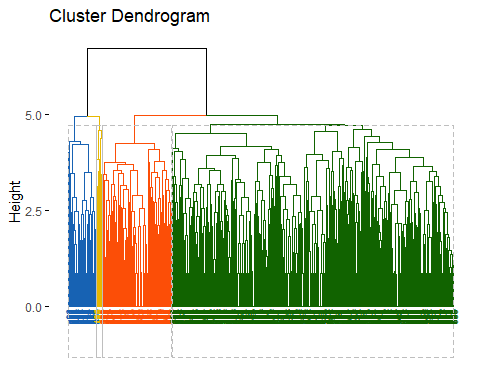
# Compute cophentic distance  
res.coph <- cophenetic(res.hc)

# Correlation between cophenetic distance and  
# the original distance  
cor(dist.eucl, res.coph)

## [1] 0.7236952

The correlation value is 0.723 which is very good with compare to the previous WARD.D2 method.

grp <- cutree(res.hc, k = 4)  
 fviz\_dend(res.hc, k = 4, cex = 0.5, k\_colors = c("#1762b3", "#E7B800", "#FC4E07","#116300"), color\_labels\_by\_k = TRUE,rect = TRUE)

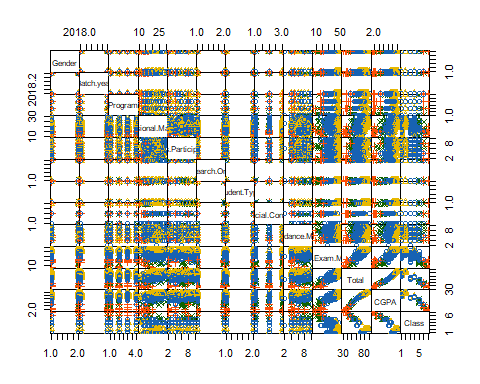
 The last cluster is bigger than the first two clusters

table(grp)

## grp  
## 1 2 3 4   
## 687 169 67 16

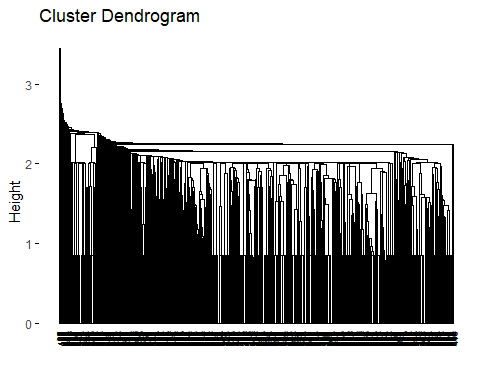
We can visualize the clustering results in the original space via the matrix of pairwise scatterplots:

pairs(studentsData, gap=0, pch=grp, col=c("#1762b3", "#E7B800", "#FC4E07","#116300")[grp])



## Using Euclidean Distance and Single Linkage Methods:

res.hc <- hclust(d = dist.eucl, method = "single")  
fviz\_dend(res.hc, cex = 0.5)



The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the cophenetic distance between the two objects.

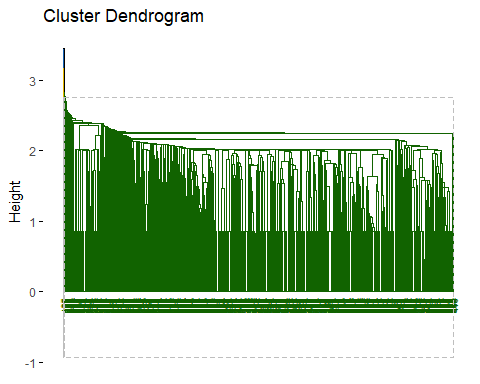
# Compute cophentic distance  
res.coph <- cophenetic(res.hc)

# Correlation between cophenetic distance and  
# the original distance  
cor(dist.eucl, res.coph)

## [1] 0.540343

The correlation value is 0.54 which is very bad with compare to the previous linkage method method.This means that this clustering method does not preserve the true original distances between units.

grp <- cutree(res.hc, k = 4)  
 fviz\_dend(res.hc, k = 4, cex = 0.5, k\_colors = c("#1762b3", "#E7B800", "#FC4E07","#116300"), color\_labels\_by\_k = TRUE,rect = TRUE)

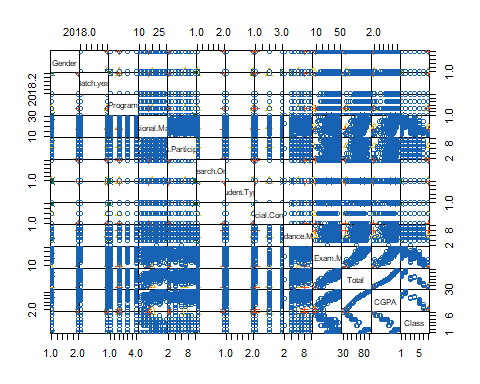
 As we can see, almost all units are grouped in a single cluster and we can check this also through the R code which gives us the clusters size:

table(grp)

## grp  
## 1 2 3 4   
## 936 1 1 1

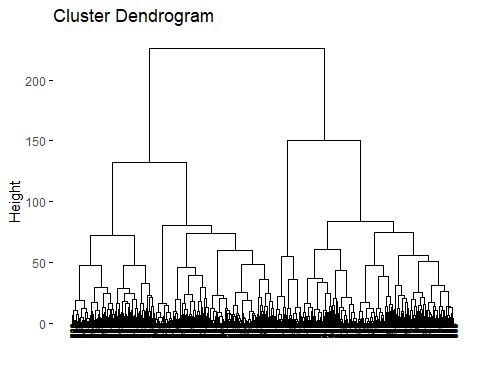
We can visualize the clustering results in the original space via the matrix of pairwise scatterplots:

pairs(studentsData, gap=0, pch=grp, col=c("#1762b3", "#E7B800", "#FC4E07","#116300")[grp])



## Using Manhattan Distance and WARD’S Linkage Methods:

res.hc <- hclust(d = dist.man, method = "ward.D2")  
fviz\_dend(res.hc, cex = 0.5)



The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the cophenetic distance between the two objects.

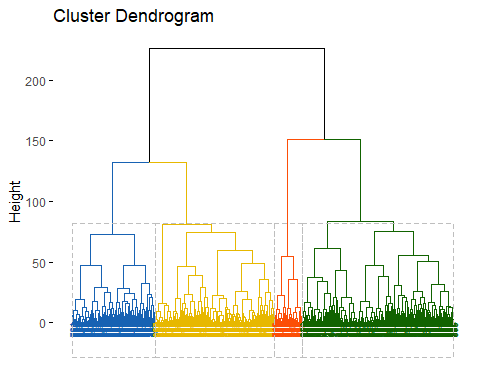
# Compute cophentic distance  
res.coph <- cophenetic(res.hc)

# Correlation between cophenetic distance and  
# the original distance  
cor(dist.eucl, res.coph)

## [1] 0.4771551

The correlation value is 0.47 which is very low it means this clustering method doesnot preserve the orginal distances.

grp <- cutree(res.hc, k = 4)  
 fviz\_dend(res.hc, k = 4, cex = 0.5, k\_colors = c("#1762b3", "#E7B800", "#FC4E07","#116300"), color\_labels\_by\_k = TRUE,rect = TRUE)

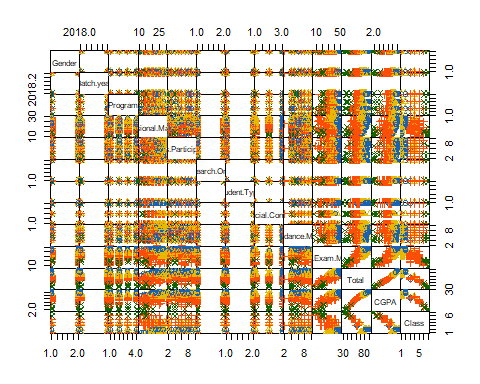
 As we can see, almost all units are grouped in a single cluster and we can check this also through the R code which gives us the clusters size:

table(grp)

## grp  
## 1 2 3 4   
## 204 294 373 68

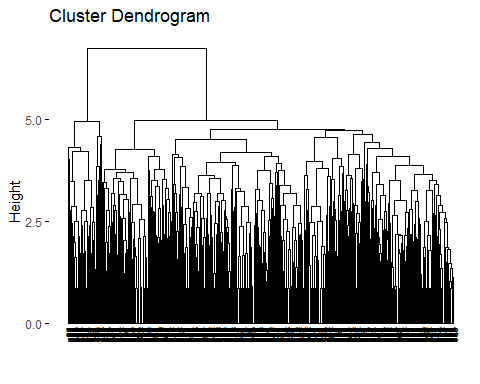
We can visualize the clustering results in the original space via the matrix of pairwise scatterplots:

pairs(studentsData, gap=0, pch=grp, col=c("#1762b3", "#E7B800", "#FC4E07","#116300")[grp])



## Using Manhattan Distance and Average Linkage Methods:

res.hc <- hclust(d = dist.eucl, method = "average")  
fviz\_dend(res.hc, cex = 0.5)



The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the cophenetic distance between the two objects.

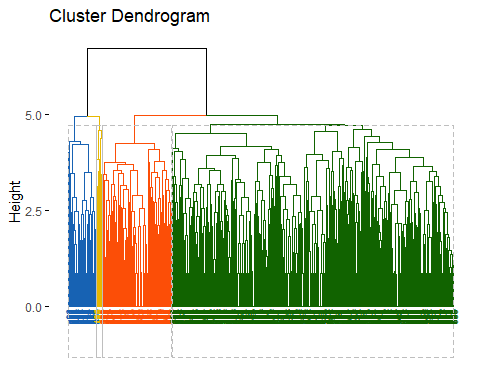
# Compute cophentic distance  
res.coph <- cophenetic(res.hc)

# Correlation between cophenetic distance and  
# the original distance  
cor(dist.eucl, res.coph)

## [1] 0.7236952

The correlation value is 0.723 which is same as the Euclidean distance using the average linkage method and its the best

grp <- cutree(res.hc, k = 4)  
 fviz\_dend(res.hc, k = 4, cex = 0.5, k\_colors = c("#1762b3", "#E7B800", "#FC4E07","#116300"), color\_labels\_by\_k = TRUE,rect = TRUE)

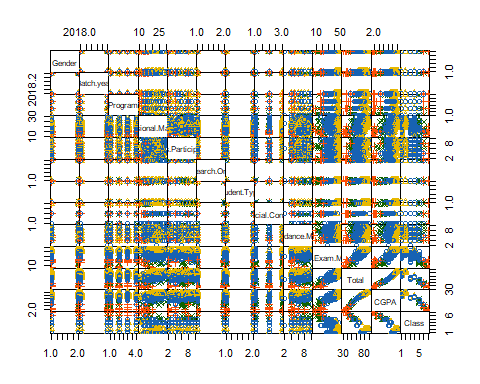
 As we can see, almost all units are grouped in a single cluster and we can check this also through the R code which gives us the clusters size:

table(grp)

## grp  
## 1 2 3 4   
## 687 169 67 16

We can visualize the clustering results in the original space via the matrix of pairwise scatterplots:

pairs(studentsData, gap=0, pch=grp, col=c("#1762b3", "#E7B800", "#FC4E07","#116300")[grp])



The above heirarchical agglomerative clustering analysis of the dataset suggest that the best combination is with the Euclidean distance with linkage method and same with Manhattan distance as well.

# Partitioning clustering

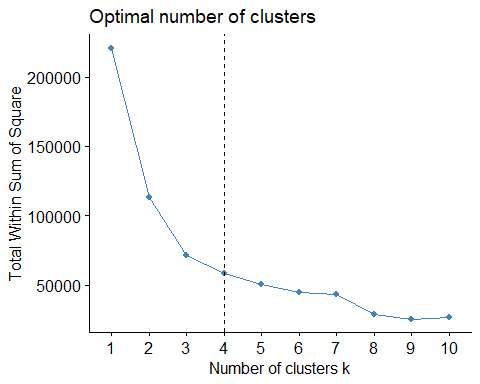
## K-Means Clustering:

K-means clustering is the most commonly used partitioning clustering algorithm. It classifies the n units to the K clusters such that units within the same cluster are as similar as possible (high cluster cohesion), whereas units from different clusters are as dissimilar as possible (high cluster separation).

The basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized.

The K-means clustering requires the users to specify the number of clusters K to be generated.

library(factoextra)  
fviz\_nbclust(studentsData, kmeans, method = "wss") +  
geom\_vline(xintercept = 4, linetype = 2)



The plot above represents the variance within the clusters. It decreases as k increases, but it can be seen a bend (or “elbow”) at k = 4. This bend indicates that additional clusters beyond the fourth have little value. In the next section, we’ll classify the observations into 4 clusters.

The K-means clustering using the cluster size 4 is given below:

kmean.res <- kmeans(studentsData, 4, nstart = 25)

This time we have 4 clusters, the largest being the third, as we can see by the R code:

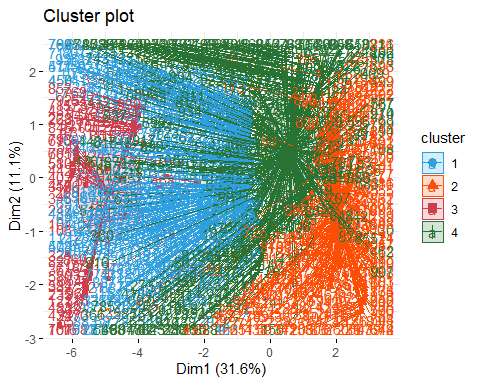
kmean.res$size

## [1] 254 233 80 372

# Print the results  
print(kmean.res)

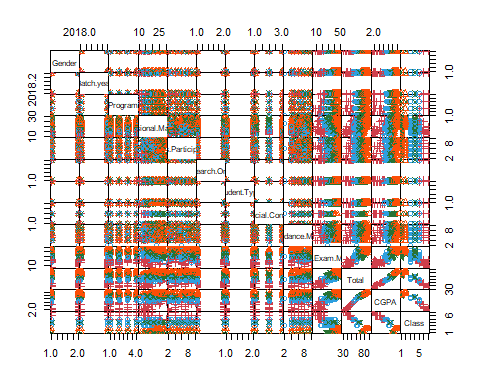
## K-means clustering with 4 clusters of sizes 254, 233, 80, 372  
##   
## Cluster means:  
## Gender Batch.year Program Sessional.Marks Class.Participation  
## 1 1.527559 2018.602 2.397638 17.31102 6.118110  
## 2 1.540773 2018.532 2.377682 23.08584 5.948498  
## 3 1.587500 2018.700 2.300000 13.90000 4.887500  
## 4 1.545699 2018.551 2.408602 21.82527 6.258065  
## Is.Research.Oriented Student.Type Financial.Condition Attendance.Marks  
## 1 1.500000 1.393701 1.909449 6.677165  
## 2 1.347639 1.351931 2.669528 7.180258  
## 3 1.600000 1.900000 2.275000 4.925000  
## 4 1.602151 1.580645 1.672043 7.408602  
## Final.Exam.Marks Total CGPA Class  
## 1 30.88976 60.99606 3.500343 3.826772  
## 2 44.80258 81.01717 4.285116 1.596567  
## 3 18.87500 42.58750 2.277600 6.450000  
## 4 35.44086 70.93280 3.909726 1.930108  
##   
## Clustering vector:  
## [1] 2 2 1 4 2 2 1 4 3 1 4 4 1 4 1 3 1 4 1 1 3 4 4 4 1 4 4 2 2 4 3 3 3 1 2 3 4  
## [38] 1 2 1 4 4 1 4 2 1 4 3 1 4 4 1 4 1 3 1 2 1 2 2 1 4 2 4 1 2 1 4 3 1 4 4 1 4  
## [75] 1 3 2 2 2 1 4 2 2 1 4 3 1 4 4 1 4 1 2 3 1 3 3 1 2 3 4 4 2 3 3 3 3 4 2 1 2  
## [112] 1 4 3 1 4 4 1 4 1 3 2 3 3 3 1 3 4 1 4 1 2 2 2 1 1 1 1 2 4 4 2 1 2 4 4 4 4  
## [149] 4 4 2 1 4 1 4 4 1 2 1 4 3 1 4 4 1 4 1 3 4 2 2 2 2 2 2 4 4 2 2 4 2 1 2 1 4  
## [186] 2 1 4 3 1 4 4 1 4 1 3 2 2 4 2 2 1 4 2 2 1 4 3 1 4 4 1 4 4 4 4 2 1 4 4 2 2  
## [223] 2 1 2 1 4 3 1 4 4 1 4 1 3 1 4 2 2 1 4 3 1 4 4 1 4 4 4 4 2 4 4 2 1 2 4 2 2  
## [260] 2 1 4 3 1 4 4 1 4 1 3 4 1 4 4 1 1 1 1 1 4 4 4 1 2 2 2 4 4 2 2 1 4 4 4 2 1  
## [297] 1 2 2 2 1 4 4 4 2 2 1 4 3 1 4 4 1 4 1 3 4 3 1 4 4 4 4 4 2 4 2 2 1 4 3 1 4  
## [334] 4 1 4 1 3 4 4 2 2 1 4 3 1 4 4 2 1 4 4 1 1 2 2 4 1 1 4 1 1 1 4 1 4 1 4 2 4  
## [371] 1 2 1 4 3 1 4 4 1 4 1 3 1 3 2 4 4 2 4 4 2 2 4 4 1 1 2 4 2 2 1 4 3 1 4 4 4  
## [408] 4 1 2 1 4 2 4 2 2 2 4 2 2 4 2 1 2 1 4 3 1 4 4 1 4 1 3 2 4 4 2 2 1 4 3 1 4  
## [445] 4 2 4 4 4 4 2 4 2 2 1 4 3 1 4 4 2 4 1 2 4 4 2 3 1 4 2 4 2 1 4 3 1 4 4 1 4  
## [482] 1 3 4 4 4 4 1 4 2 2 1 1 2 4 1 1 4 4 4 4 2 4 1 4 2 2 1 4 3 1 4 4 1 4 4 2 1  
## [519] 4 4 2 1 2 2 1 4 3 1 4 4 1 4 1 3 2 1 1 2 4 4 1 4 2 2 1 4 3 1 4 4 1 1 1 2 2  
## [556] 4 4 4 4 2 4 4 4 4 4 4 4 4 1 2 1 4 3 1 4 4 1 4 1 3 1 4 3 1 4 4 1 4 4 4 2 2  
## [593] 2 1 1 1 4 2 2 4 2 2 1 2 1 1 1 2 4 4 2 2 1 4 3 1 4 4 4 1 2 2 1 2 2 4 4 4 2  
## [630] 4 2 4 4 2 2 2 4 1 2 4 2 2 1 2 1 4 3 1 4 4 1 4 1 3 4 2 2 2 2 4 2 2 4 1 2 2  
## [667] 1 4 1 4 2 4 4 4 4 4 2 2 1 4 3 1 4 4 4 2 4 1 4 4 2 2 4 4 4 2 4 1 4 4 2 2 1  
## [704] 4 3 1 4 2 1 4 3 1 4 4 1 4 1 3 2 4 2 1 4 2 2 4 4 4 1 2 4 2 2 4 2 4 4 4 1 2  
## [741] 1 4 4 4 4 4 2 2 1 4 3 1 4 4 2 1 1 1 1 1 4 4 4 4 1 2 1 4 3 1 4 4 1 4 1 3 1  
## [778] 1 1 1 4 2 2 2 4 1 4 4 2 2 1 4 3 1 4 4 4 4 4 2 1 4 2 2 1 4 4 4 4 2 4 4 2 2  
## [815] 2 1 2 1 4 3 1 4 4 1 4 1 3 2 4 4 4 2 4 1 2 2 2 4 1 1 4 2 2 1 4 3 1 4 4 2 1  
## [852] 4 3 1 4 4 1 4 1 3 2 4 4 2 4 2 2 4 2 2 4 2 2 2 4 2 2 2 1 4 2 2 1 1 1 4 1 4  
## [889] 1 1 1 2 4 4 2 2 2 2 4 4 2 2 1 4 3 1 4 4 4 4 4 1 1 4 1 4 4 2 2 1 4 2 1 4 3  
## [926] 1 4 4 1 4 1 3 4 4 2 2 4 2 3  
##   
## Within cluster sum of squares by cluster:  
## [1] 12910.103 13193.377 5826.272 23245.670  
## (between\_SS / total\_SS = 75.0 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

fviz\_cluster(kmean.res, data = studentsData,palette = c("#2E9FDF", "#FC4E07", "#CF3E4B", "#287233"),ellipse.type = "euclid", star.plot = TRUE,repel = TRUE, ggtheme = theme\_minimal() )



We can see that separation between clusters is not so clear, because some units are closer to a different cluster than the one they belong to.

cl2 <- kmean.res$cluster  
pairs(studentsData, gap=0, pch=cl2, col=c("#2E9FDF", "#FC4E07","#CF3E4B", "#287233")[cl2])



From this plot we can understand what variables are useful to find clusters. The variable sessional marks and final exams marks have good seperation between the clusters in the scatered plot.So if we remove some initial variables from the data it will be good to find the more accurate clusters.But, most of the variables in this case are overlapped and it is difficult to find the clusters in it.

## K-Medoids Clustering:

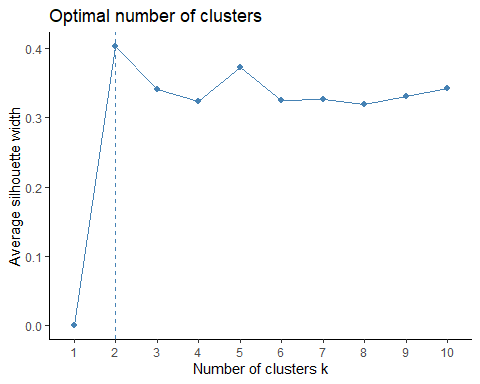
Differently from K-means, in K-medoids clustering each cluster is represented by one of the data points in the cluster. These points are named cluster medoids. The term medoid refers to a unit within a cluster for which average dissimilarity between it and all the other members of the cluster is minimal. It corresponds to the most centrally located point in the cluster. These units (one per cluster) can be considered as a representative example of the members of that cluster.

The most common K-medoids clustering method is the PAM (partitioning around medoids) algorithm.

The K-medoids algorithm requires the user to specify K, the number of clusters to be generated (like in K-means clustering). A useful approach to determine the optimal number of clusters is the silhouette method. The silhouette value, for each unit, is a measure of how similar a unit is to its own cluster (cohesion) compared to other clusters (separation).

The R function fviz\_nbclust() [factoextra package] provides a convenient solution to estimate the optimal number of clusters.

library(cluster)  
library(factoextra)  
fviz\_nbclust(studentsData, pam, method = "silhouette")+  
theme\_classic()

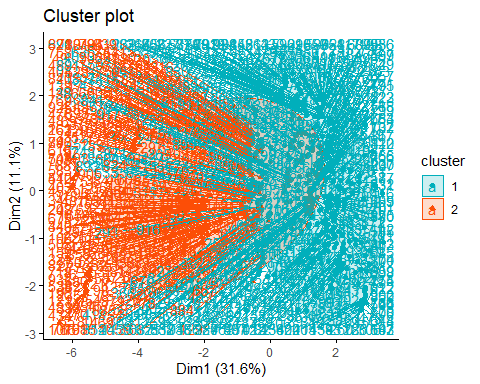
 We can see that the plot suggest us the number of clusters should be 2 from our dataset.

pam.res <- pam(studentsData, 2)  
pam.res$medoids

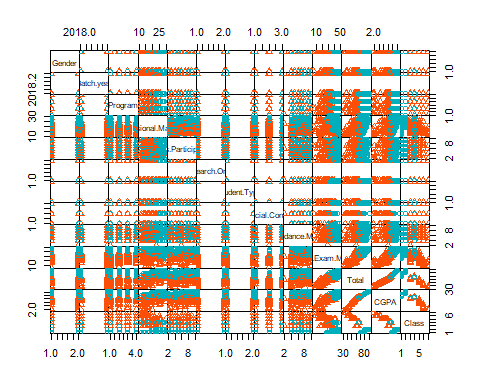
## Gender Batch.year Program Sessional.Marks Class.Participation  
## [1,] 1 2018 3 22 6  
## [2,] 2 2019 2 16 7  
## Is.Research.Oriented Student.Type Financial.Condition Attendance.Marks  
## [1,] 1 1 1 7  
## [2,] 1 1 2 6  
## Final.Exam.Marks Total CGPA Class  
## [1,] 39 74 4.024 1  
## [2,] 28 57 3.283 5

we can visualize the obtained cluster using the below plots.

fviz\_cluster(pam.res,  
palette = c("#00AFBB", "#FC4E07"), # color palette  
ellipse.type = "t", # Concentration ellipse  
repel = TRUE, # Avoid label overplotting (slow)  
ggtheme = theme\_classic()  
)



pam.clus<- pam.res$clustering  
pairs(studentsData, gap=0, pch=pam.clus, col= c("#00AFBB", "#FC4E07")[pam.clus] )

 The obove representation shows that there is very low tendency in the dataset for the clustering and we need to remove many attributes in order to find a suitable clustering in the dataset for the analysis or predictions.

# Cluster Validation:

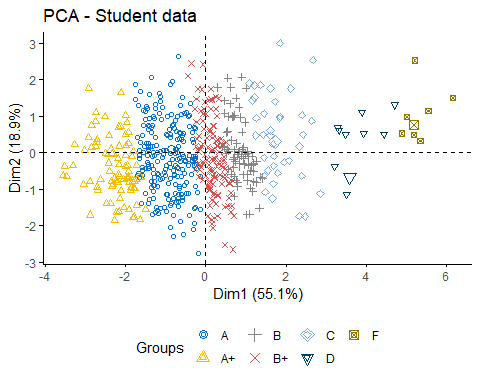
In the cluster analysis with the dataset we applied the clustering techniques on our data set without checking the validation of the clustering. Cluster validation gives us insights about the dataset that is it valid for applying clustering or not? So, in other words it is the measering of the goodness of the cluster results.

Before applying any clustering method on our data, it’s important to evaluate whether the data set contains meaningful clusters (i.e. non-random structures) or not. This process is defined as the assessing of clustering tendency or the feasibility of the cluster analysis.

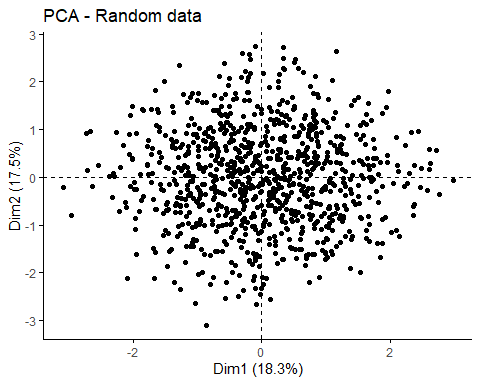
# Collected data from github repository  
x <- getURL("https://raw.githubusercontent.com/adeelahmedcsku/student-dataset/main/20-01-2020-data%20developed%20records%20students-changes%20in%20grading%20system%20-%20data%20developed%20records%20students.csv")  
studentsData <- read.csv(text = x)  
  
studentsData.alter<-data.frame(studentsData[,c(-1,-2,-3,-6,-7,-8,-13)])  
  
# Random data generated from the our data set  
student.alter.random\_df <- apply(studentsData.alter, 2,  
function(x){runif(length(x), min(x), (max(x)))})  
student.alter.random\_df <- as.data.frame(student.alter.random\_df)  
# Standardize the data sets  
studentsData.alter.scaled <- scale(studentsData.alter)  
student.alter.random\_df <- scale(student.alter.random\_df)

As the data contain more than two variables, we need to reduce the dimensionality in order to plot a scatter plot. This can be done using principal component analysis (PCA) algorithm (R function: prcomp()). After performing PCA, we use the function fviz\_pca\_ind() [factoextra R package] to visualize the output.

library("factoextra")  
# Plot faithful data set  
fviz\_pca\_ind(prcomp(studentsData.alter.scaled), title = "PCA - Student data",  
habillage = studentsData$Class, palette = "jco",  
geom = "point", ggtheme = theme\_classic(),  
legend = "bottom")



# Plot the random df  
fviz\_pca\_ind(prcomp(student.alter.random\_df), title = "PCA - Random data",  
geom = "point", ggtheme = theme\_classic())

 It can be seen from the graph that student data contains three real clusters while the random data plot has no meaningful clusters.

# Cluster Tendency:

In the previous sections where we computed the k-means algorithm and the hierarchical clustering impose a classification on the random uniformly distributed data set even if there are no meaningful clusters present in it. This is why, clustering tendency assessment methods should be used to evaluate the validity of clustering analysis. That is, whether a given data set contains meaningful clusters.

So, lets check the cluster tendency in our selected dataset.

We can assess if there are clusters from both a statistical point of view and from a graphical one, by means of, respectively, the Hopkins statistic and the VAT algorithm.

## Hopkins statistic:

Hopkins statistic on the dataset altered by removing the nominal variables.

library(clustertend)  
hopkins(studentsData.alter.scaled, n = nrow(studentsData.alter.scaled)-1)

## $H  
## [1] 0.2001505

Hopkins statistics on the randomly generated data set.

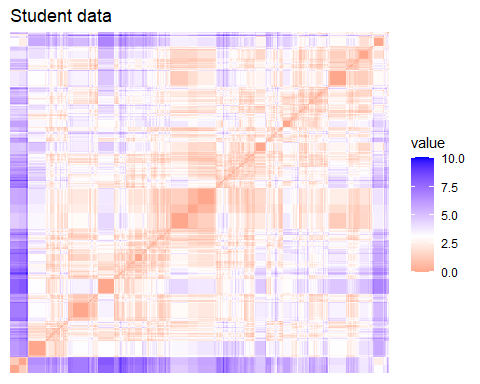
hopkins(student.alter.random\_df, n = nrow(student.alter.random\_df)-1)

## $H  
## [1] 0.5032931

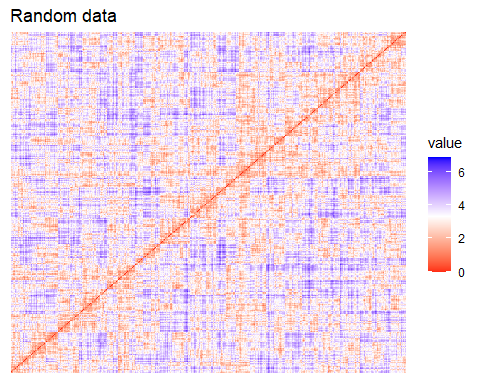
It can be seen that the student data set which is altered and removed the categorical attrbutes is clusterable because the value of H = 0.19 which is closed to zero. While the random sample of the same data set is not clusterable because the value of the H=0.5.

## VAT algorithm:

fviz\_dist(dist(studentsData.alter.scaled), show\_labels = FALSE)+ labs(title = "Student data")



fviz\_dist(dist(student.alter.random\_df), show\_labels = FALSE)+ labs(title = "Random data")

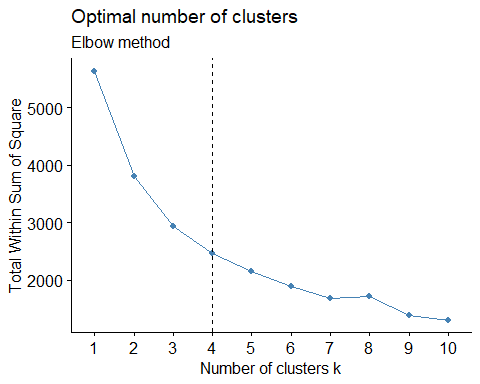
 The dissimilarity matrix image confirms that there is a cluster structure in the standardized Student data set but not in the random one.

# Determining the optimal number of clusters:

There are many ways to calculate the optimal numbers of clusters, like Elbow method,Silhouette method and gap statistics method.

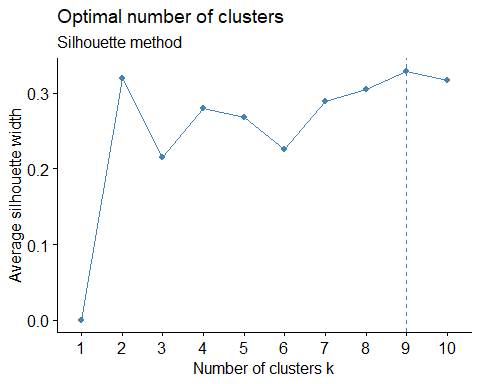
## Elbow method

fviz\_nbclust(studentsData.alter.scaled, kmeans, method = "wss") +  
geom\_vline(xintercept = 4, linetype = 2)+  
labs(subtitle = "Elbow method")



## Silhouette method

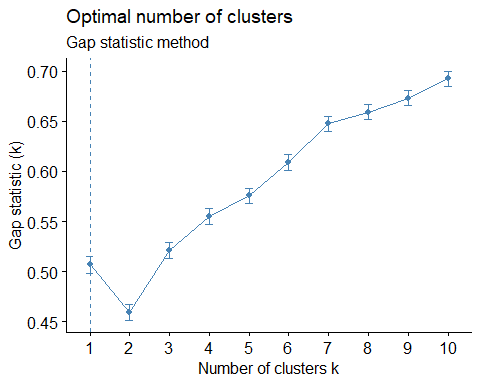
fviz\_nbclust(studentsData.alter.scaled, kmeans, method = "silhouette")+  
labs(subtitle = "Silhouette method")



## Gap statistic

# nboot = 50 to keep the function speedy.  
# recommended value: nboot= 500 for your analysis.  
# Use verbose = FALSE to hide computing progression.  
set.seed(123)  
fviz\_nbclust(studentsData.alter.scaled, kmeans, nstart = 25, method = "gap\_stat", nboot = 500)+  
labs(subtitle = "Gap statistic method")

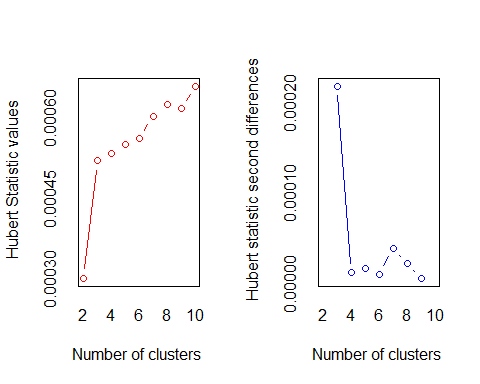
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations



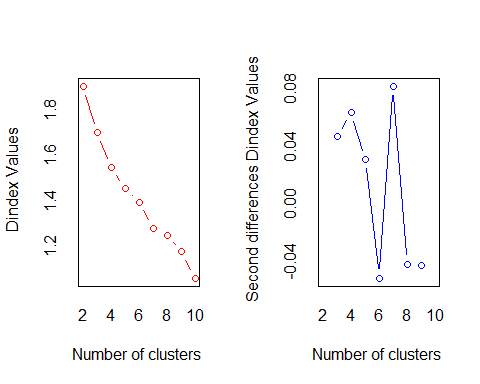
* Elbow method: 4 clusters solution suggested
* Silhouette method: 9 clusters solution suggested
* Gap statistic method: 1 clusters solution suggested

## NbClust Method:

library("NbClust")  
  
 nb <- NbClust(studentsData.alter.scaled, distance = "euclidean", min.nc = 2, max.nc = 10, method = "kmeans")



## \*\*\* : 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   
## \* 4 proposed 3 as the best number of clusters   
## \* 5 proposed 4 as the best number of clusters   
## \* 2 proposed 7 as the best number of clusters   
## \* 1 proposed 9 as the best number of clusters   
## \* 5 proposed 10 as the best number of clusters   
##   
## \*\*\*\*\* Conclusion \*\*\*\*\*   
##   
## \* According to the majority rule, the best number of clusters is 2   
##   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The results of NbClust() can be graphically visualized using fviz nbclust() (in the factoextra package) as follows:

library("factoextra")  
fviz\_nbclust(nb)

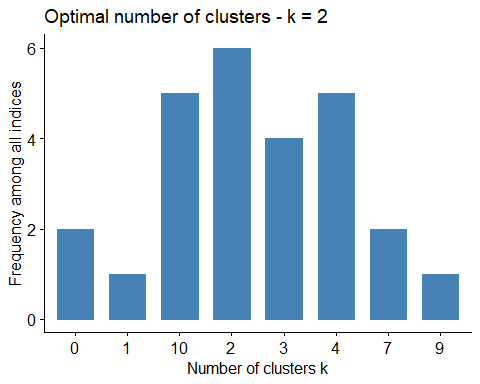
## Warning in if (class(best\_nc) == "numeric") print(best\_nc) else if  
## (class(best\_nc) == : the condition has length > 1 and only the first element  
## will be used

## Warning in if (class(best\_nc) == "matrix") .viz\_NbClust(x, print.summary, : the  
## condition has length > 1 and only the first element will be used

## Warning in if (class(best\_nc) == "numeric") print(best\_nc) else if  
## (class(best\_nc) == : the condition has length > 1 and only the first element  
## will be used

## Warning in if (class(best\_nc) == "matrix") {: the condition has length > 1 and  
## only the first element will be used

## Among all indices:   
## ===================  
## \* 2 proposed 0 as the best number of clusters  
## \* 1 proposed 1 as the best number of clusters  
## \* 6 proposed 2 as the best number of clusters  
## \* 4 proposed 3 as the best number of clusters  
## \* 5 proposed 4 as the best number of clusters  
## \* 2 proposed 7 as the best number of clusters  
## \* 1 proposed 9 as the best number of clusters  
## \* 5 proposed 10 as the best number of clusters  
##   
## Conclusion  
## =========================  
## \* According to the majority rule, the best number of clusters is 2 .



# Clustering Validation Statistics:

The term cluster validation is used to design the procedure of evaluating the goodness of clustering algorithm results. Generally, clustering validation statistics can be categorized into 3 classes:

1. Internal cluster validation: Consider the insights of the cluster.
2. External Cluster Validation: Consider the external provided class labels etc.
3. Relative Cluster Validation: Consider the variation in the parameter values like no of clusters etc.

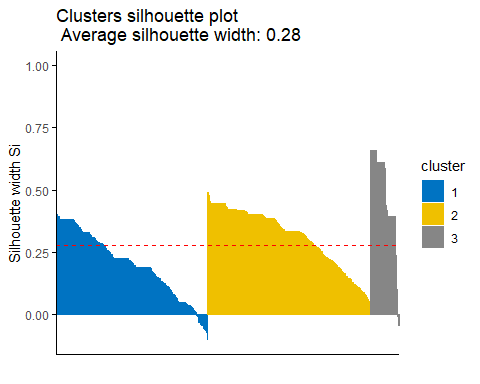
The internal cluster validation allows us to estimate the optimal number of clusters and to select the appropriate clustering algorithm, by means of two indices: the Silhouette Width and the Dunn index.

## Silhouette plot

The silhouette coecient (Si) measures how similar an object i is to the other objects in its own cluster versus those in the neighbor cluster.

km.res <- eclust(studentsData.alter.scaled, "kmeans", k = 3, nstart = 25, graph = FALSE)  
 fviz\_silhouette(km.res, palette = "jco", ggtheme = theme\_classic())

## cluster size ave.sil.width  
## 1 1 414 0.20  
## 2 2 445 0.31  
## 3 3 80 0.48



# Silhouette information  
silinfo <- km.res$silinfo  
 silinfo$avg.width

## [1] 0.2758341

The average silhouette width is 0.27. No cluster participate overall widths. Looking at the plot, we can see that in cluster 1 some of the units have a silhouette width higher than the average one and while some have negative silhouette width. While in cluster 2 there most of the units have a silhouette width lower than the average one; in cluster 3 almost half units have a high silhouette width,and some outlier as well with the negative silhouette width others a low one.

It can be seen that several samples, in cluster 1 and 3, have a negative silhouette coeffcient. This means that they are not in the right cluster. We can find the name of these samples and determine the clusters they are closer (neighbor cluster), as follow:

# Silhouette width of observation  
sil <- km.res$silinfo$widths[, 1:3]  
# Objects with negative silhouette  
neg\_sil\_index <- which(sil[, 'sil\_width'] < 0)  
sil[neg\_sil\_index, , drop = FALSE]

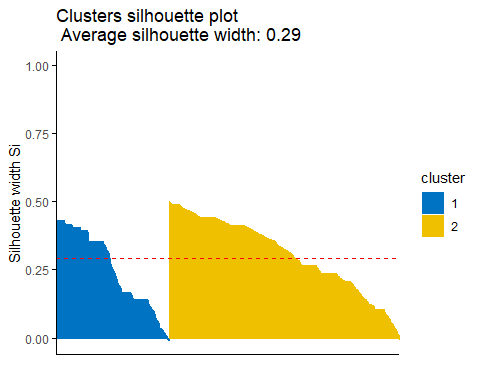
## cluster neighbor sil\_width  
## 564 1 2 -0.003351573  
## 813 1 2 -0.006538247  
## 447 1 2 -0.007013895  
## 914 1 2 -0.007182948  
## 828 1 2 -0.014321073  
## 744 1 2 -0.030442362  
## 214 1 2 -0.031525939  
## 356 1 2 -0.031660254  
## 655 1 2 -0.031739095  
## 180 1 2 -0.032353877  
## 147 1 2 -0.038553938  
## 364 1 2 -0.039353907  
## 763 1 2 -0.039688584  
## 675 1 2 -0.040991680  
## 155 1 2 -0.044752551  
## 755 1 2 -0.044893601  
## 567 1 2 -0.048711872  
## 808 1 2 -0.050543076  
## 148 1 2 -0.051183202  
## 865 1 2 -0.057294588  
## 736 1 2 -0.057977860  
## 598 1 2 -0.059410181  
## 719 1 2 -0.060014101  
## 809 1 2 -0.060734057  
## 293 1 2 -0.064824652  
## 630 1 2 -0.064824652  
## 724 1 2 -0.066606607  
## 829 1 2 -0.074974505  
## 672 1 2 -0.079391650  
## 690 1 2 -0.104082860  
## 152 3 2 -0.009015080  
## 105 3 2 -0.018833522  
## 143 3 2 -0.042865558  
## 110 3 2 -0.044909600

There are some elements of cluster 1 which we should include in the cluster 2 and there are also some elements of the cluster 3 which we need to inculde in the cluster 2.

This is a further confirmation that the optimal number of clusters is 2, in fact, if we run the kmeans with K=2, the plot does not present any unit with a negative silhouette width:

km.res <- eclust(studentsData.alter.scaled, "kmeans", k = 2, nstart = 25, graph = FALSE)  
fviz\_silhouette(km.res, palette = "jco", ggtheme = theme\_classic())

## cluster size ave.sil.width  
## 1 1 308 0.25  
## 2 2 631 0.31



## Dunn Index:

library(fpc)  
# Statistics for k-means clustering  
km\_stats <- cluster.stats(dist(studentsData.alter.scaled), km.res$cluster)  
# Dun index  
km\_stats$dunn

## [1] 0.02317697

The aim of the Dunn index is to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated. A higher Dunn index indicates better clustering, so it should be maximized.