Cluster Analysis - Team Project | Fall 2024

Group: Shwetha Vedavinayagam, Teshani Jayasinghe, Komalpreet, Rachel Mendonsa

- Partional Clustering - Red Wine Quality Dataset
    - Introduction
    - Dataset Description
    - Objective
    - Assessing Clustering Tendency
        - Hopkins Statistic
    - Exploratory Data Analysis
        - Visualizing Distance Matrix
    - Determing the optimal number of clusters
        - 1) Elbow method
        - 2) Silhoutte method
        - 3) Gap Statistic
        - 4) Nbclust function
    - K Means clustering
        - Cluster validation
        - Internal Validation
        - External Validation
    - K medoids
        - Internal Validation Measures
        - External Validation
- Hierarchical Clustering - Swiss dataset
    - Introduction
    - Dataset Description
    - Objective
    - Insights from the Swiss Dataset
        - Exploratory Data Analysis
    - Assessing Clustering Tendency
        - Hopkins Statistic
        - Dissimilarity (DM) matrix
    - Optimal Clusters
    - Hierarchical Clustering
        - Comparing linkage methods using the cluster plot
        - Visualizing the cluster tree - Dendrograms
        - Agnes - Agglomerative Nesting Clustering
        - Divisive Analysis Clustering
        - Comparing the linkage methods by verifying the cluster tree
        - Average linkage
        - Comparison of the linkage methods using ***their dendrograms***
        - Comparing and Visualizing Dendrograms
        - Correlation matrix between a list of dendrogams
    - Cluster Validation Statistics

        - Cluster Validation
        - Internal Validation
- Research Component
    - Anomaly Detection
        - Anomaly Detection using DBSCAN
        - Anomaly Detection Using K-Means
    - Clustering in Cybersecurity
        - Anomaly Detection Using DBSCAN in Cybersecurity
        - Anomaly Detection Using K-Means in Cybersecurity
    - Example code - Anomaly Detection in transctions dataset using DBSCAN

Partional Clustering - Red Wine Quality Dataset

Introduction

The red wine quality dataset can be found (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009 (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)). This data set is related to red variants of the Portuguese “Vinho Verde” wine.It includes 11 physicochemical properties of 1599 observations and one target variable- Quality, stating a score for each observation.

Dataset Description

All the 12 variable of the data set are numerical.

- **fixed acidity** : non-volatile acids in the wine, primarily tartaric acid
- **volatile acidity**: Amount of acetic acid in the wine
- **citric acid**: Amount of citric acid in the wine
- **residual sugar** :Sugar remaining after fermentation
- **chlorides**:Salt content in the wine
- **free sulfur dioxide** : Amount of sulfur dioxide that is available to act as a preservative and antioxidant in the wine
- **total sulfur dioxide** :Overall sulfur dioxide content in the wine
- **density** :Mass per unit volume of the wine
- **pH** : Acidity of the wine
- **sulphates** : Concentration of sulfates in the wine
- **alcohol** : Ethanol content in the wine
- **Quality** : Rating of the wine’s overall quality (0-10)

Objective

The primary purpose of conducting clustering analyses using k-means and Partitioning Around Medoids (PAM) on the red wine dataset is to categorize wines into groups with similar physicochemical properties.

```
library(tidyverse)
library(dplyr)
library(tidyr)
library(ggplot2)
library(corrplot)
library(skimr)
library(knitr)
library(stringr)
library(viridis)
library(factoextra)
library(hrbrthemes)
library(tibble)
library(forcats)
library(mclust)
library(fpc)
library(LPCM)
library(cluster)
library(NbClust)
library(funModeling)
library(clValid)
library(dbscan)
library(plotly)
knitr::opts_chunk$set(
  warning = FALSE,  # Suppress warnings
  message = FALSE   # Suppress messages
)
```

```
Rwine<-read.csv("winequality-red.csv",sep = ";")
```

```
str(Rwine)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073
0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

```
head(Rwine)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
## 6       5
```

```
colnames(Rwine)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

Removing the Quality column, since it is used to verify the clusters.

```
new_data <- Rwine[, -ncol(Rwine)]

# View the new data set
head(new_data)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
```

```
nrow(Rwine)
```

```
## [1] 1599
```

```
library(hopkins)
set.seed(123)
```

Assessing Clustering Tendency

Hopkins Statistic

We conducted the Hopkins Statistic test iteratively 10 times, using the mean of the Hopkins value and using 0.5 as the threshold to reject the alternative hypothesis. That is, if H < 0.5, then it is unlikely that D has statistically significant clusters.

```
hopkins_valuesp <- numeric()
for(i in 1:10) {
  hopkins_valuesp[i]<-hopkins(new_data, m=(nrow(new_data)-1))
}
mean(hopkins_valuesp)
```

```
## [1] 0.9999987
```

```
hopkins_value<-hopkins(new_data,m=(nrow(new_data)-1))
hopkins_value
```

```
## [1] 0.9999989
```

Since Hopkins value is greater than 0.7 we can concluse that are data set is clusterable.

Checking for missing values

```
missing_values <- sapply(new_data, function(x) sum(is.na(x)))

print(missing_values)
```

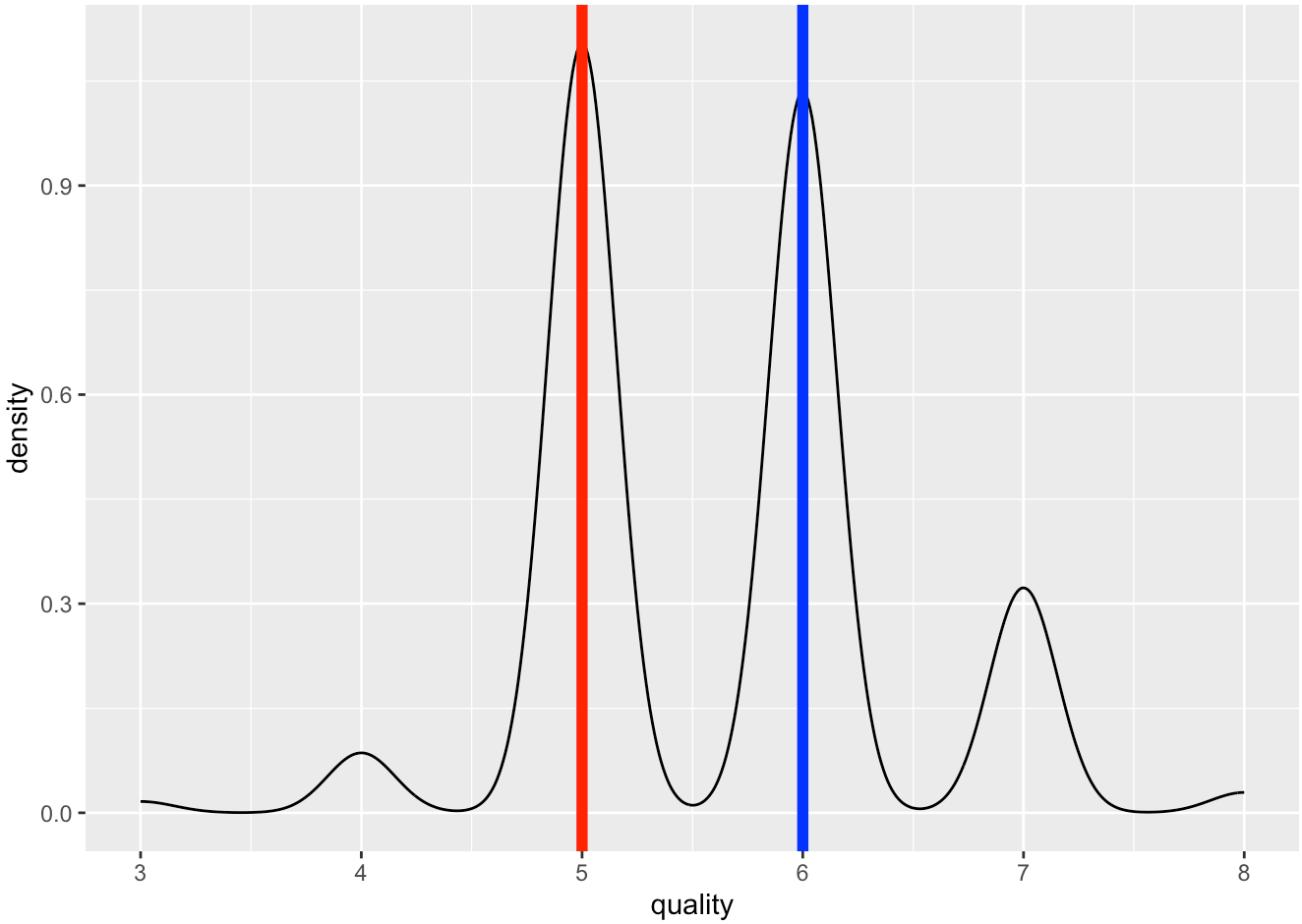
```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                    0
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol
##                    0                    0
```

There are no missing values in the data set.

Exploratory Data Analysis

Density Plot:

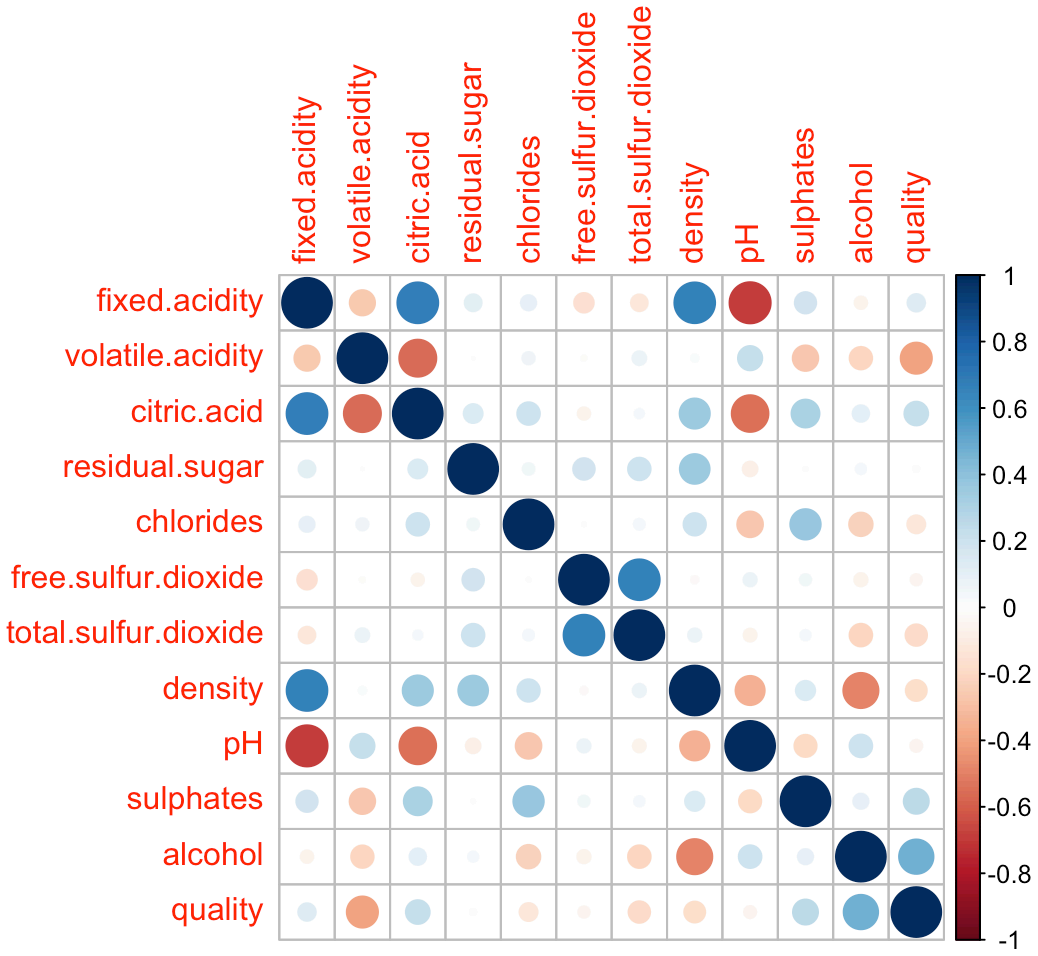
```
options(scipen = 999)

density_wine <- ggplot(Rwine, aes(x = quality)) + geom_density()

density_wine +
  geom_vline(xintercept = 5, col = "red", size = 2) +
  geom_vline(xintercept = 6, col = "blue", size = 2)
```



A simple density plot on the dataset shows that our dataset has 2 bumps (considering threshold density as 0.5 for significance), suggesting that there are more number of data for quality 5 and 6.

Correlation Plot:

```
corrplot(cor(Rwine))
```



Alcohol vs. Quality: Shows a strong positive correlation, indicating that higher alcohol content tends to be associated with better wine quality.

Density vs. Alcohol: Displays a negative correlation, suggesting wines with higher alcohol tend to have lower density.

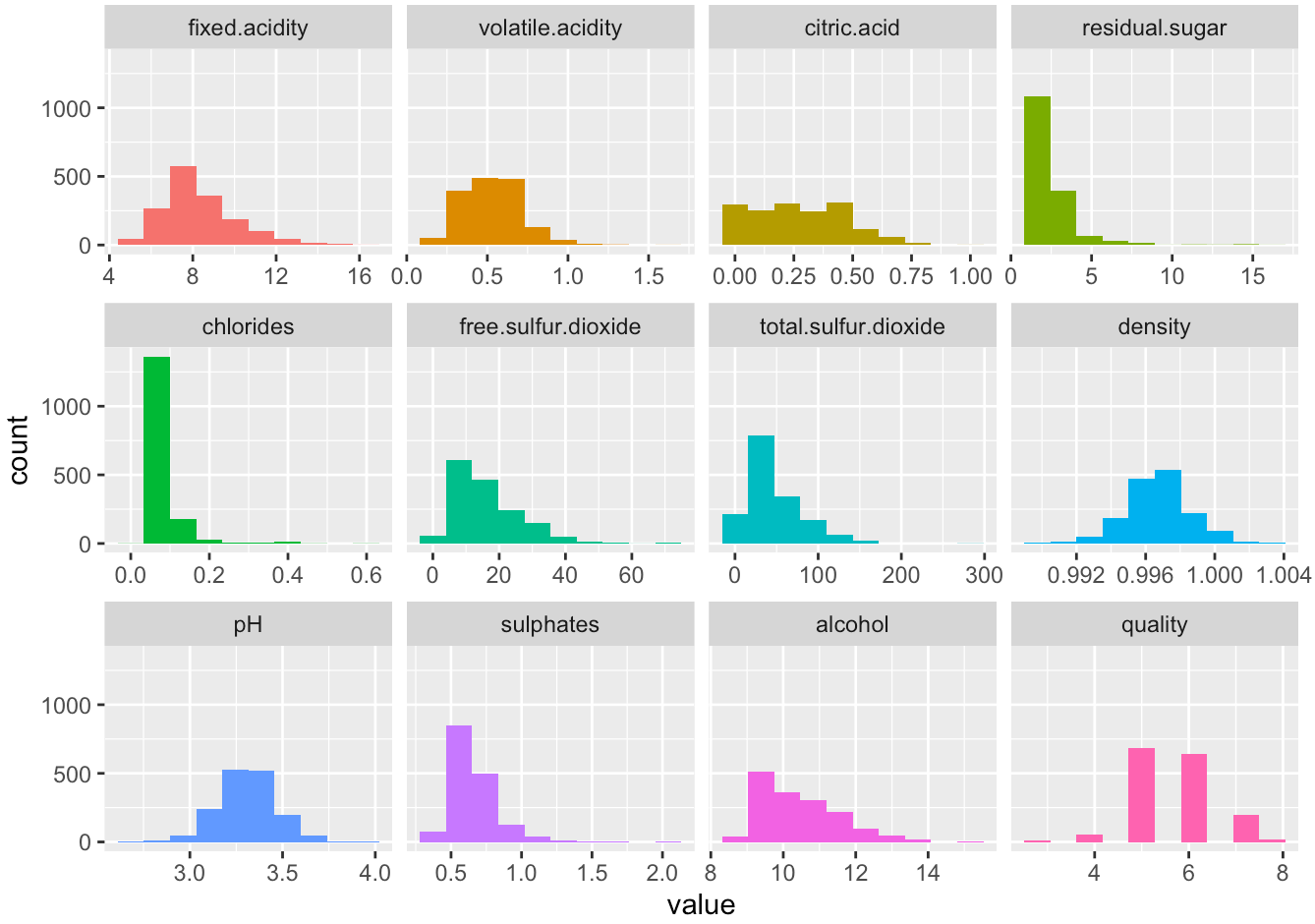
Citric Acid vs. Fixed Acidity: Strong positive correlation

pH vs. Fixed Acidity: Negative correlation,

Free Sulfur Dioxide vs. Total Sulfur Dioxide: High positive correlation, as expected since these variables are closely related.

Histogram Analysis:

```
plot_num(Rwine)
```

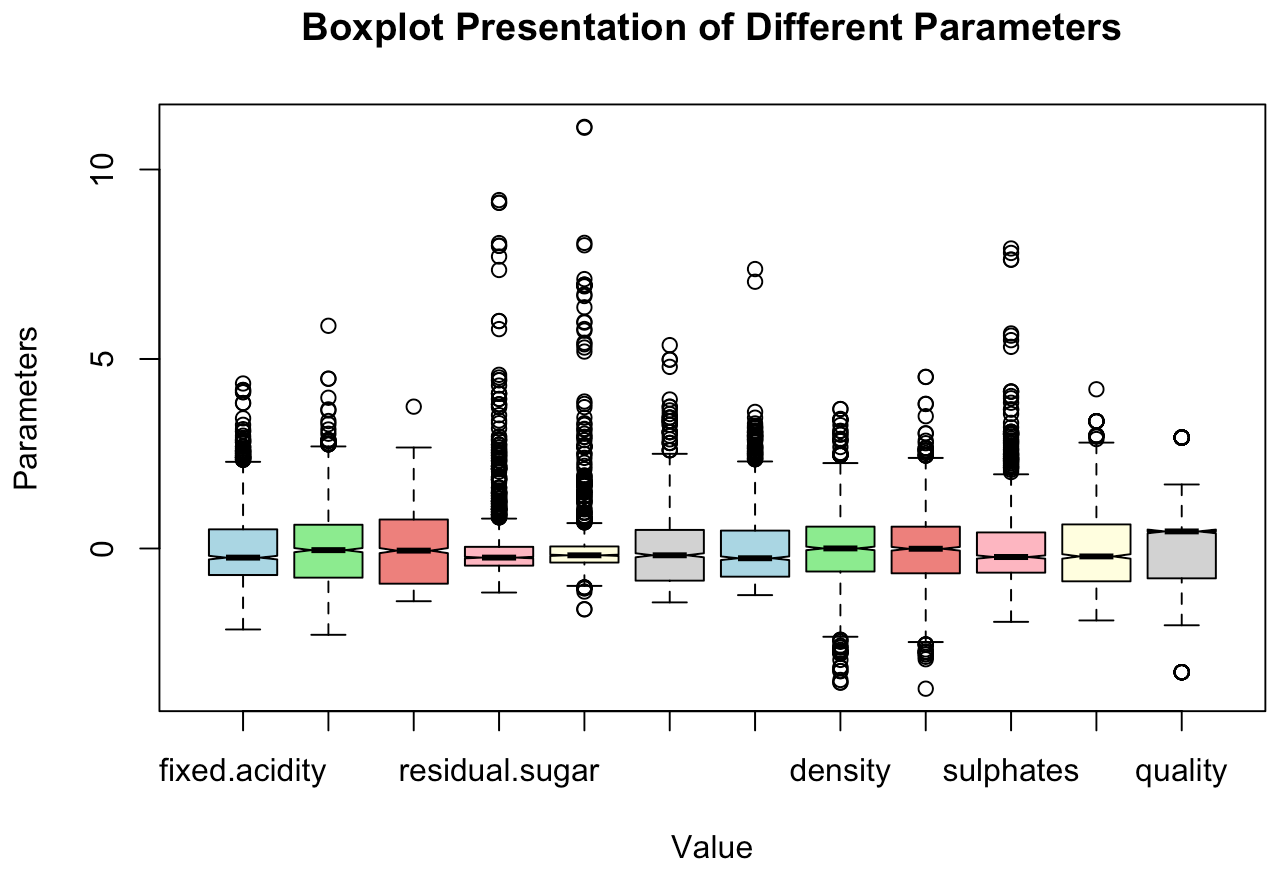


From the histograms, we observe that:

1. The pH value seems to dispaly a normal distribution with major samples exhibiting values between 3.0 and 3.5.
2. The free sulfur dioxide seems to be between the 1-72 count with peaking around 10 mark.
3. The total sulfur dioxide seems to a have a spread between 0 and 175 and exhibiting peak around 50.
4. The alcohol content seems to vary from 8 to 14 with major peaks around 10 with a lower count between 13 and 14.
5. The fixed acidity, volatile acidity and density can almost be considered to be normally distributed.
6. Majorly, the variables distributions are right-tailed.

Boxplot Analysis:

```
box_colors <- c("lightblue", "lightgreen", "lightcoral", "lightpink", "lightyellow", "li
ghtgray")


boxplot(scale(Rwine),
        xlab = "Value",
        ylab = "Parameters",
        main = "Boxplot Presentation of Different Parameters",
        col = box_colors,
        border = "black",
        notch = TRUE,
        outline = TRUE,
        )
```



We have scaled all the values for boxplot analysis to bring them at similar scales and avoid overrepresntation of any independant parameter. A simple analysis of the boxplot shows that there are major outliers in residual sugar, chlorides and sulphates and minor outliers in citric acid.

Mean value for each Chemical character

```
# Group by quality and calculate the mean for each variable
grouped_by_quality <- Rwine %>%
  group_by(quality) %>%
  summarise(across(everything(), mean))

print(grouped_by_quality)
```

```
## # A tibble: 6 × 12
##   quality fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
##     <int>         <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1       3          8.36            0.884       0.171           2.64    0.122
## 2       4          7.78            0.694       0.174           2.69    0.0907
## 3       5          8.17            0.577       0.244           2.53    0.0927
## 4       6          8.35            0.497       0.274           2.48    0.0850
## 5       7          8.87            0.404       0.375           2.72    0.0766
## 6       8          8.57            0.423       0.391           2.58    0.0684
## # ℹ 6 more variables: free.sulfur.dioxide <dbl>, total.sulfur.dioxide <dbl>,
## #   density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>
```

The mean values are almost similar for quality level 7,8 for all the variable except alcohol.

A clear difference is shown in citric acid levels in lower quality wines and higher quality wines.

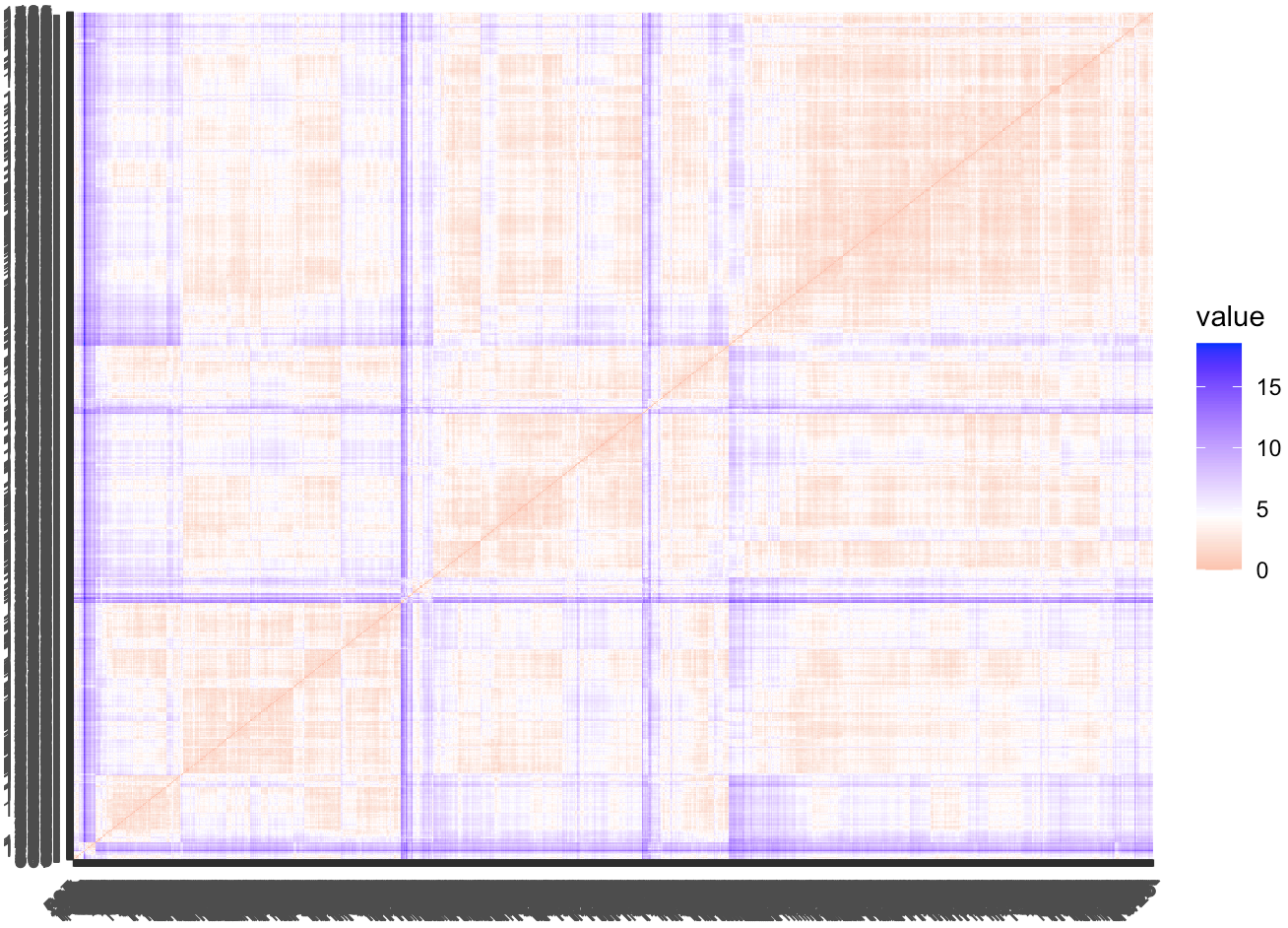
A clear difference is also seen in the alochol level of lower quality wines and higher quality wines. However the quality 4 has a higher alcohol level than quality 5. Since the values are almost similar with clear distinctivity, this might lead to not showing clear clusters when K means algorthim is applied.

Scaling the data

```
df.scaled<-scale(new_data)
```

Visualizing Distance Matrix

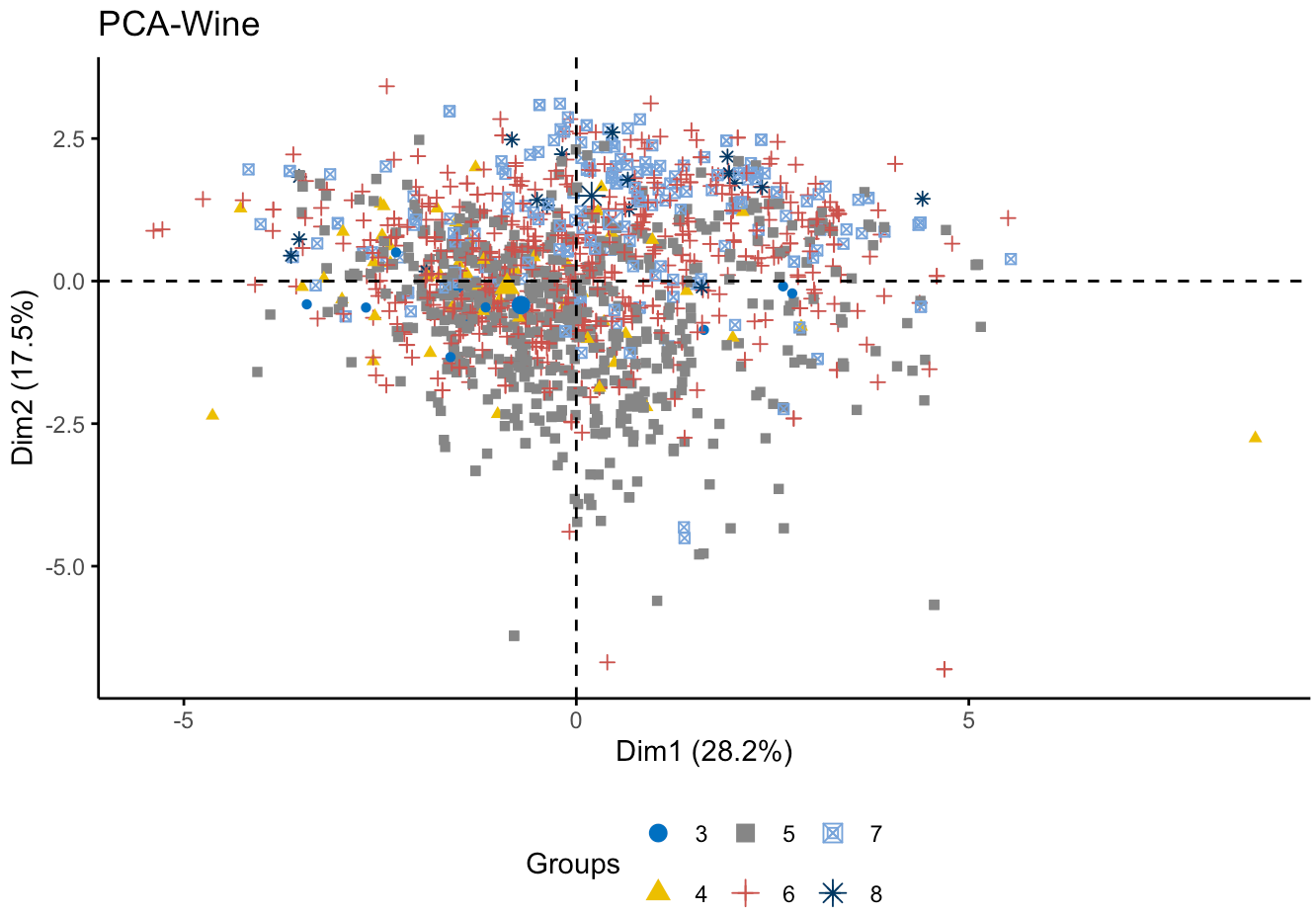
```
library(stats)
library(factoextra)

dist.eucl<-dist(df.scaled,method="euclidean")
fviz_dist(dist.eucl)
```



PCA -

Visualizing the data to assess whether the contain any meaningful clusters.We perform PCA to reduce the dimensionality to plot.

```
library("factoextra")

fviz_pca_ind(prcomp(df.scaled),title="PCA-Wine",habillage = Rwine$quality,palette = "jc
o",geom="point",ggtheme=theme_classic(),legend="bottom")
```

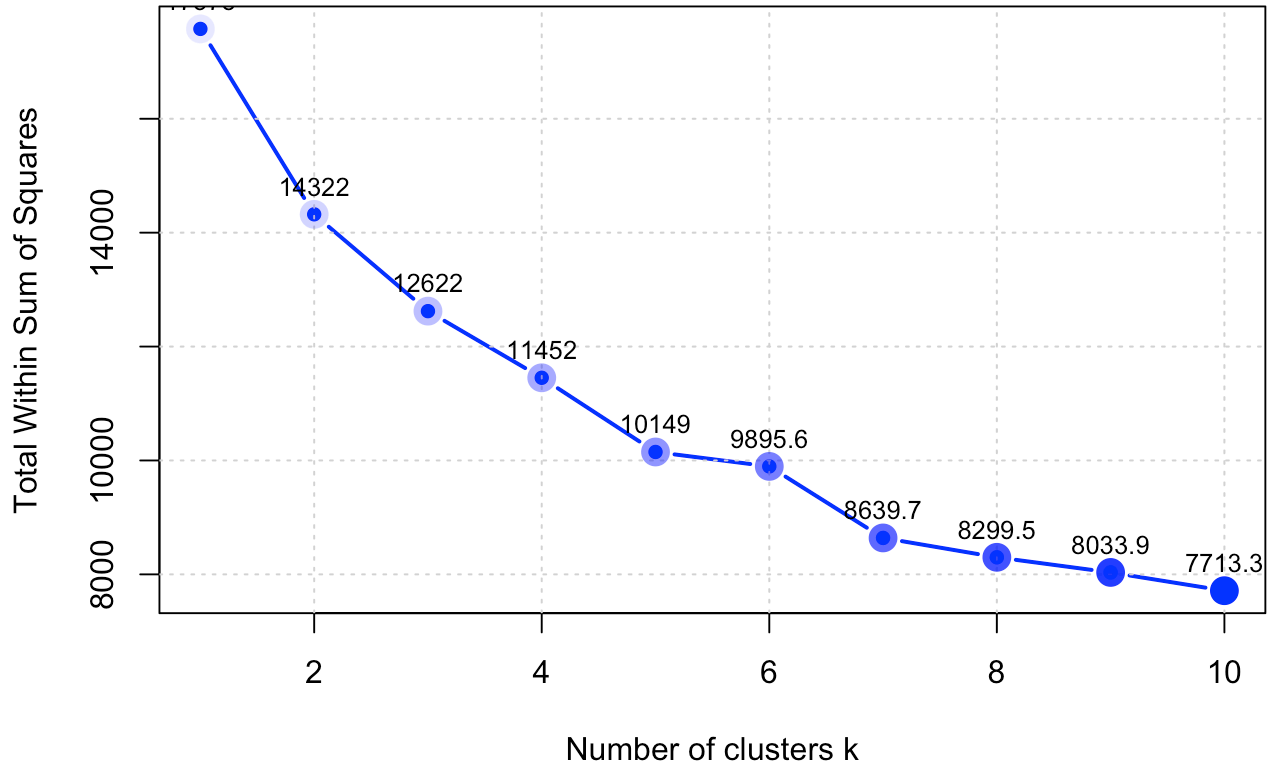


This plot visualizes the results of a Principal Component Analysis (PCA) applied to a dataset.The colors and shapes represent different levels of wine quality. In this plot, it seems difficult to identify clear clusters, suggesting the wine samples may not separate distinctly based on these components.

Determing the optimal number of clusters

1) Elbow method

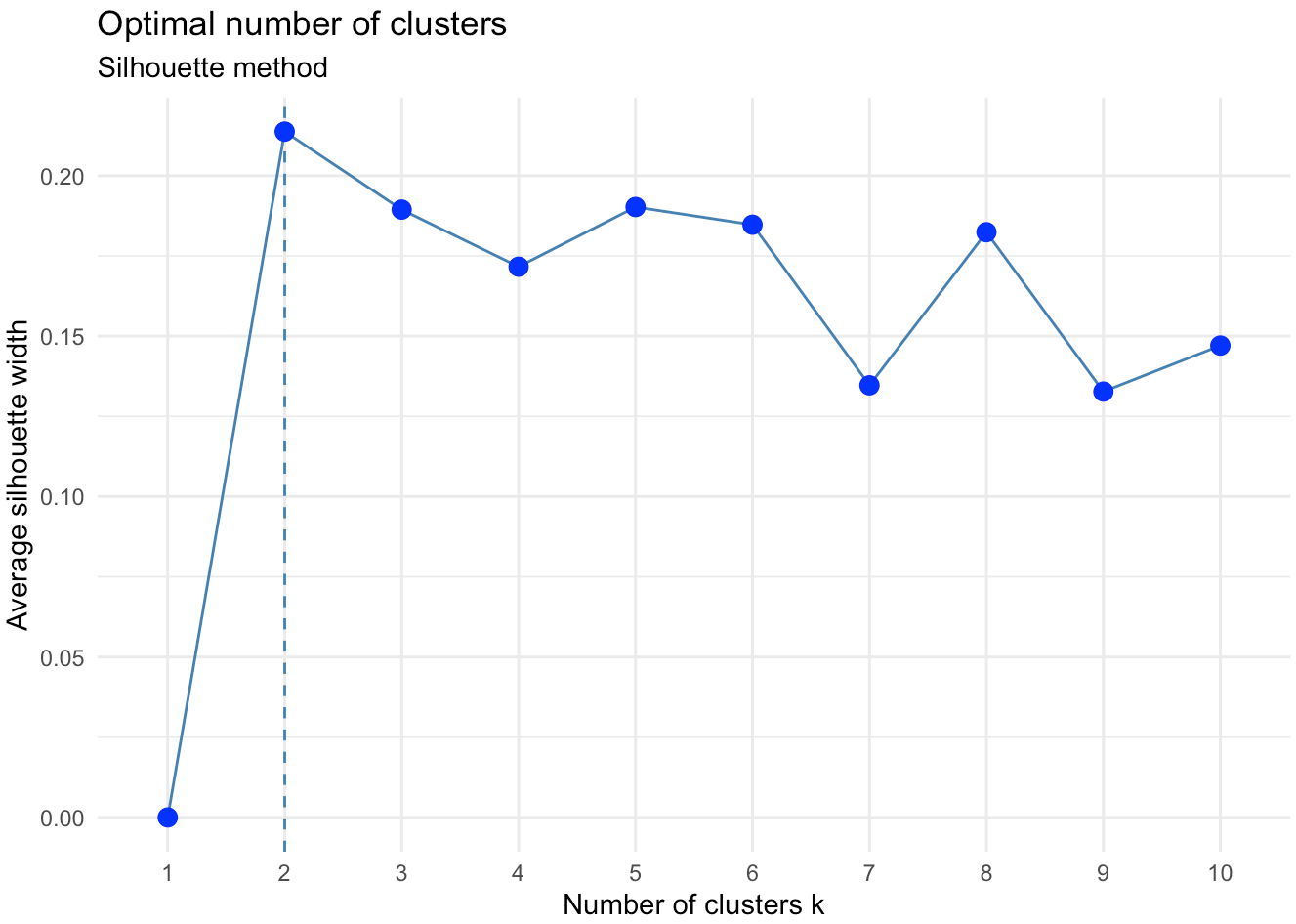
```
wss <- numeric(10)
for (i in 1:10) {
  wss[i] <- sum(kmeans(df.scaled, centers = i)$tot.withinss)
}

plot(1:10, wss, type = "b", pch = 16, col = "blue", lwd = 2,
     xlab = "Number of clusters k", ylab = "Total Within Sum of Squares")

grid()

for (i in 1:10) {
  text(i, wss[i], labels = round(wss[i], 1), pos = 3, cex = 0.8, col = "black")
  points(i, wss[i], col = rgb(0, 0, 1, alpha = i/10), pch = 16, cex = 2)
}
```



The Elbow appears to occur around 3 or 4 clusters, as the rate of decrease in WSS slows down after that point.

2) Silhoutte method

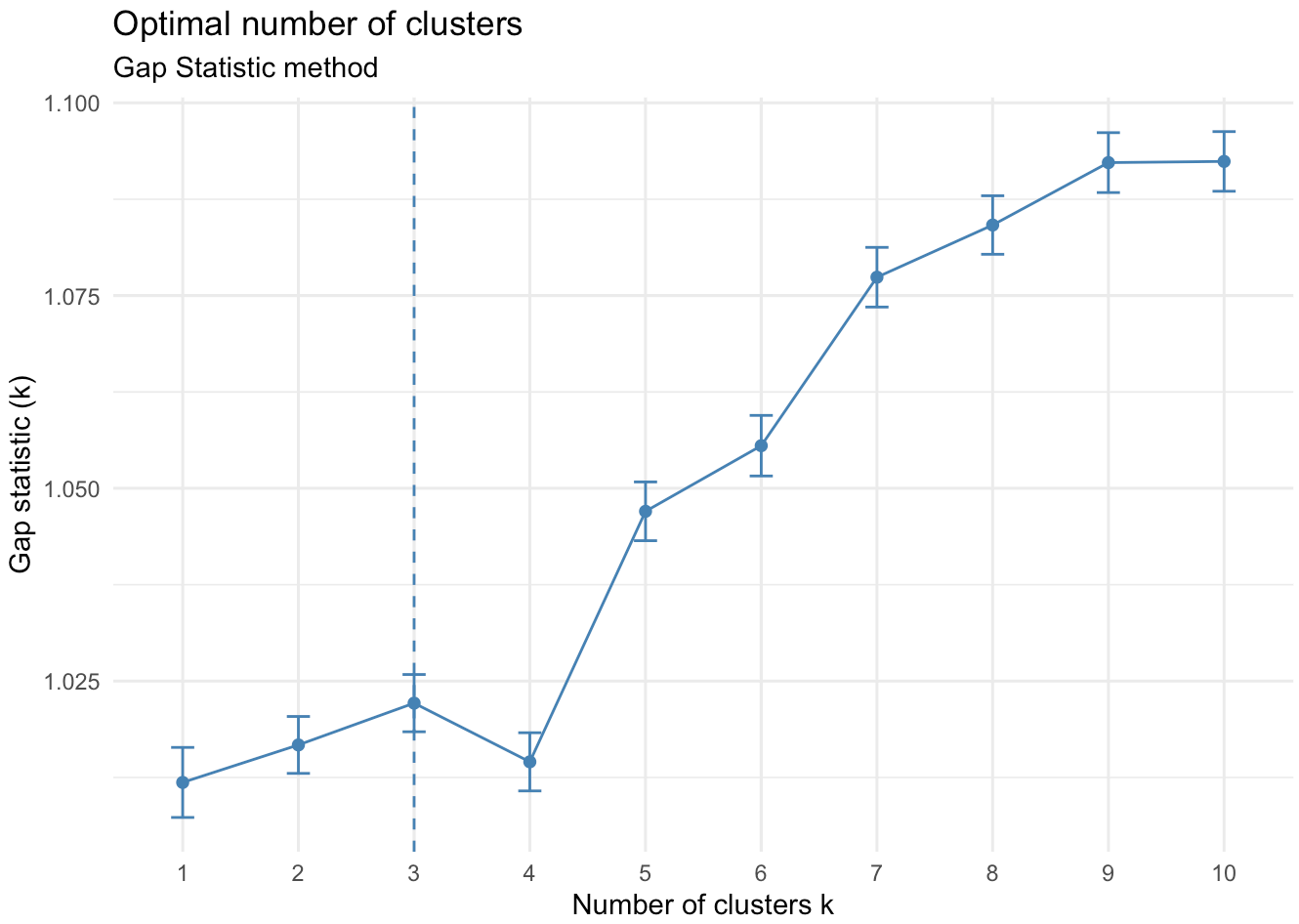
```
fviz_nbclust(df.scaled, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette method") +
  theme_minimal() +
    geom_line(size = 3, color = "blue") +
  geom_point(size = 3, color = "blue")
```



The silhouette method suggest 2 clusters as optimal.

3) Gap Statistic

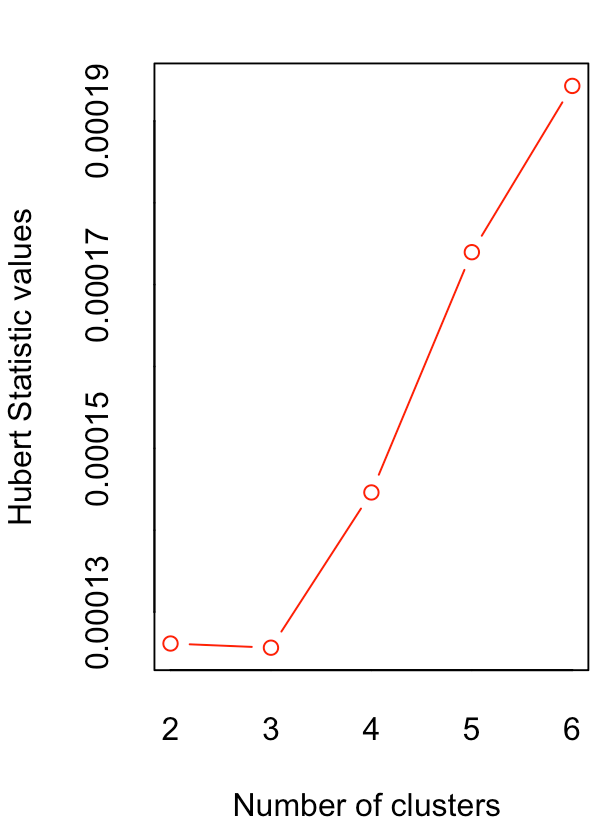
```
set.seed(123)
fviz_nbclust(df.scaled, kmeans, nstart = 25, method = "gap_stat", nboot = 50) +
  labs(subtitle = "Gap Statistic method") +
  theme_minimal() +                # Optional: Cleaner theme
  geom_line(size = 1.5, color = "blue")  # Thicker lines, color set to blue
```

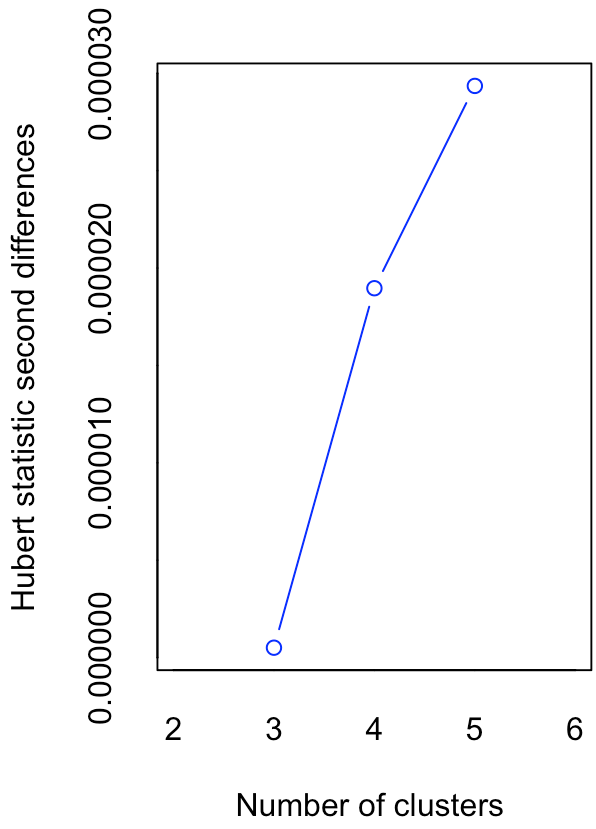


The Gap Statistic method suggest 3 clusters as optimal.

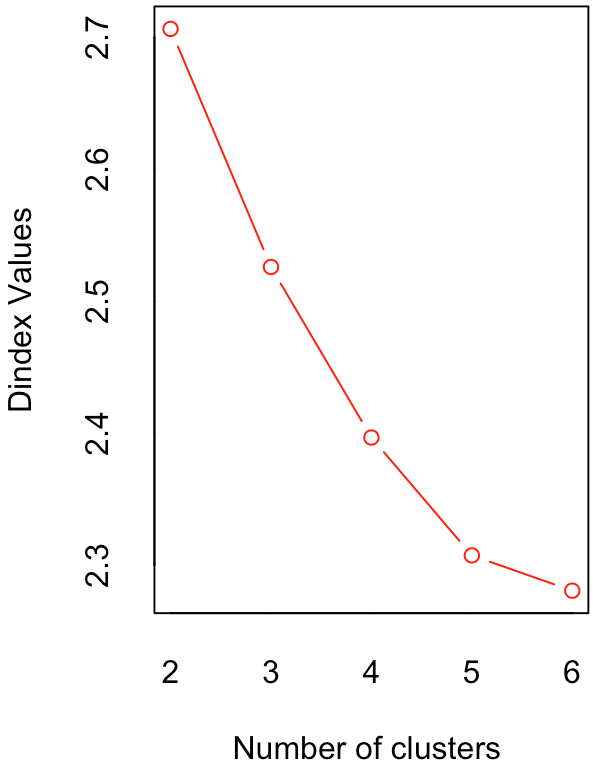
4) Nbclust function

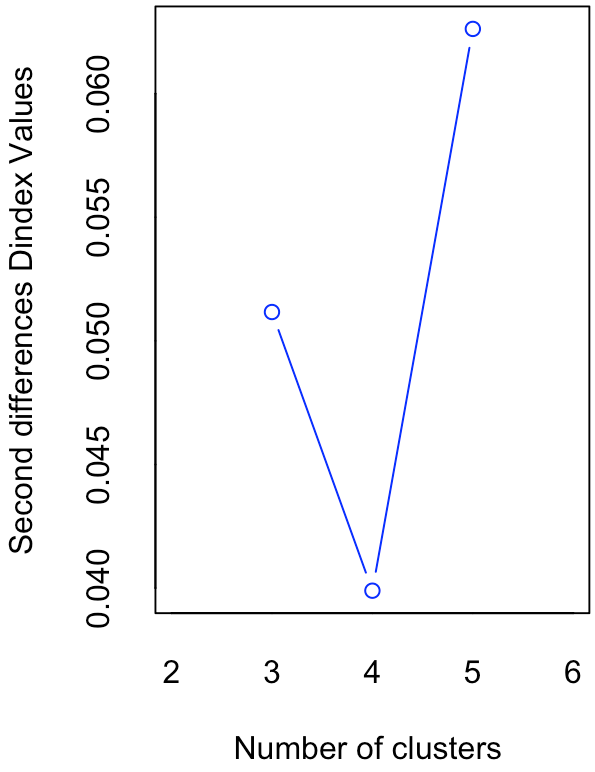
```
library("NbClust")
library("factoextra")

nb<-NbClust(df.scaled,distance="euclidean",min.nc=2,max.nc=6,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 correspo
nds 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 p
eak in Dindex
##                 second differences plot) that corresponds to a significant increase o
f the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 7 proposed 2 as the best number of clusters
## * 5 proposed 3 as the best number of clusters
## * 4 proposed 4 as the best number of clusters
## * 5 proposed 5 as the best number of clusters
## * 2 proposed 6 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

The NbClust function proposes 2 as the best number of clusters.

```
table(Rwine$quality)
```

```
##
##   3   4   5   6   7   8
##  10  53 681 638 199  18
```

There are 6 quality values in the target variable, suggesting there should be 6 clusters.

Hence we performed K means algorithm for cluster number 2,3 and 6 as suggested by the above plots and analysis, to identify the ideal number of clusters.

Explaination for the number of clusters

K Means clustering

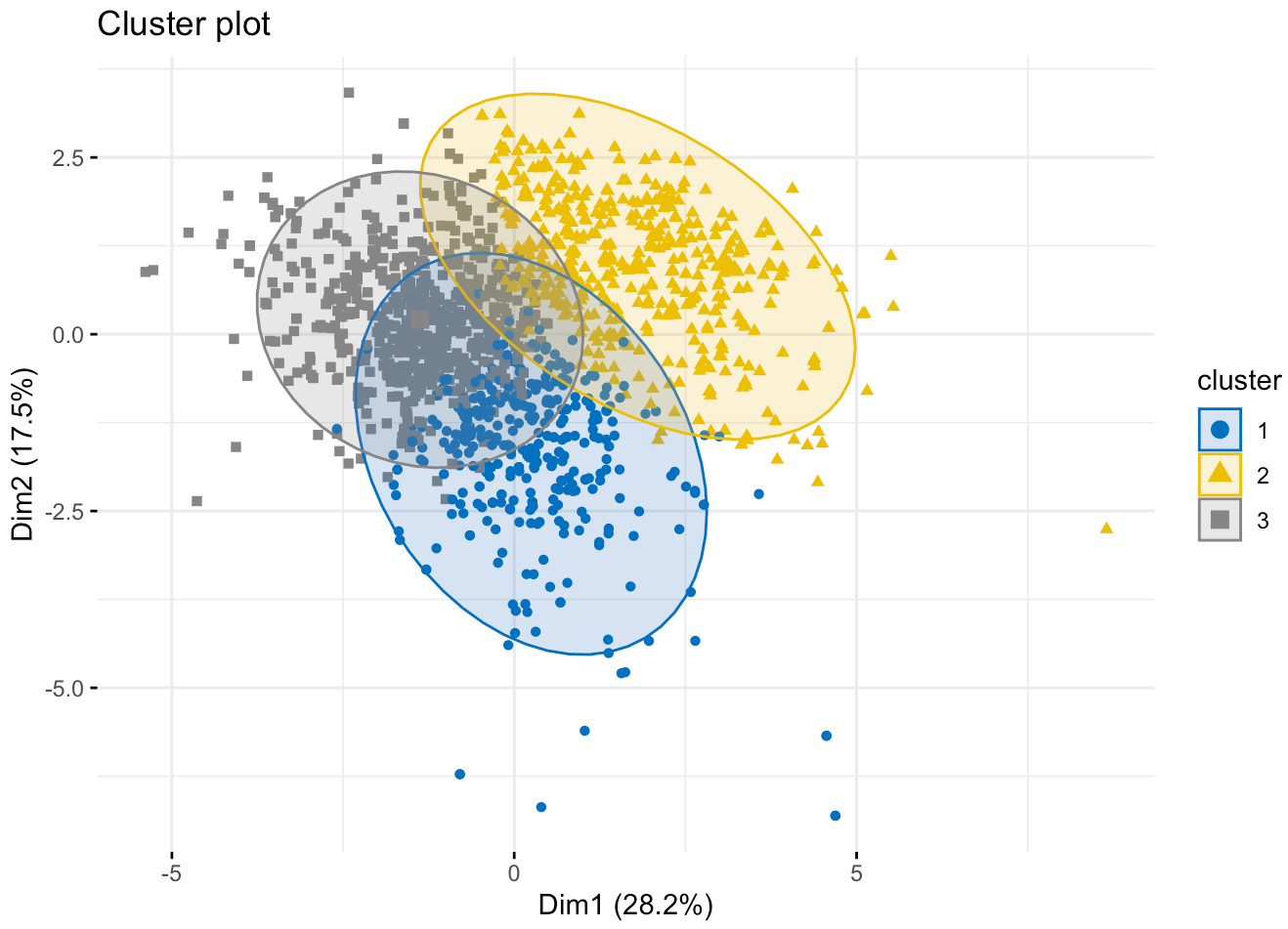
3 Clusters

```
set.seed(123)
library("factoextra")

km.res3<-eclust(df.scaled,"kmeans",k=3,nstart=25,graph = FALSE)

km.res3
```

```
## K-means clustering with 3 clusters of sizes 373, 502, 724
##
## Cluster means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar    chlorides
## 1   -0.09011718       0.03972118   0.1001378     0.40040745 -0.005526519
## 2    1.00367463      -0.68547433   1.0204527     0.03104004  0.276076371
## 3   -0.64949027       0.45482336  -0.7591418    -0.22780950 -0.188575893
##   free.sulfur.dioxide total.sulfur.dioxide    density         pH  sulphates
## 1           1.0718969            1.3258820  0.2846387 -0.1798214 -0.1885221
## 2          -0.4767114           -0.4815366  0.4383036 -0.7518363  0.5544470
## 3          -0.2216967           -0.3492025 -0.4505506  0.6139437 -0.2873116
##       alcohol
## 1 -0.51318854
## 2  0.28250279
## 3  0.06851232
##
## Clustering vector:
##    [1] 3 1 3 2 3 3 3 3 3 1 3 1 3 2 1 1 1 2 3 2 1 1 2 1 3 3 3 2 3 3 3 3 1 1 3 3 3
##   [38] 2 3 1 1 1 2 2 3 3 1 2 3 1 2 3 3 1 1 1 2 1 3 3 1 1 3 3 3 3 3 3 2 3 3 1 1 3
##   [75] 1 2 2 3 3 1 3 2 1 2 3 3 2 3 1 3 1 2 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 1 1 3
##  [112] 1 1 2 3 2 3 3 3 1 3 3 3 3 1 1 3 3 3 3 1 3 3 3 3 3 3 3 1 1 3 3 3 3 3 1 3 1
##  [149] 3 3 2 2 1 1 1 1 1 1 3 3 3 3 3 1 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3
##  [186] 1 1 3 1 1 1 3 1 3 3 1 3 2 3 3 2 1 3 3 3 2 2 1 1 2 2 3 2 1 3 1 3 3 3 1 1 1
##  [223] 3 3 3 1 2 3 1 3 3 3 1 3 3 3 3 3 3 3 2 2 1 2 2 3 3 1 3 3 2 3 2 1 3 1 2 3 2
##  [260] 2 3 3 3 1 2 2 3 2 3 2 1 2 2 3 1 1 3 2 2 2 2 2 3 2 1 1 2 1 3 1 3 2 2 3 2 2
##  [297] 1 3 3 3 3 2 3 3 1 2 3 2 2 3 2 1 1 1 1 3 1 1 1 1 1 1 3 1 1 1 2 2 2 2 2 2 1
##  [334] 3 3 2 2 1 2 2 2 2 2 2 2 2 1 3 2 2 3 2 3 3 2 1 3 2 2 2 2 1 1 2 2 2 2 2 2 2
##  [371] 3 2 2 1 2 2 2 2 2 2 1 2 2 2 2 1 3 1 3 1 2 3 2 2 1 2 2 1 2 2 3 1 3 2 2 3 2 2
##  [408] 2 2 2 1 1 3 2 1 1 2 1 2 3 1 3 3 2 3 3 3 3 2 2 2 1 2 2 2 2 1 2 2 3 2 2 2 2
##  [445] 3 3 2 2 3 2 2 2 3 2 3 2 2 1 2 2 2 3 2 1 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 3 2
##  [482] 2 2 2 2 2 2 2 2 2 1 2 2 1 1 2 3 1 2 1 3 2 2 2 2 2 2 2 1 2 2 1 2 2 2 1 2 2
##  [519] 2 1 2 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 2 2 1 2 3 2 2 1 3 2 2 2 3 2 2 3 2
##  [556] 2 2 2 2 2 2 1 1 1 2 2 2 2 2 3 2 3 2 2 2 2 2 1 1 2 2 2 2 2 1 3 2 1 3 2 1 1
##  [593] 1 2 3 1 2 2 3 2 3 2 3 2 1 3 2 1 2 3 1 2 3 2 2 1 1 2 2 2 1 1 2 3 1 1 3 3 3
##  [630] 1 3 2 3 1 1 3 1 1 3 2 2 1 2 1 2 3 3 3 3 1 2 1 2 2 2 1 2 2 3 3 3 3 3 2 2 1
##  [667] 2 2 2 2 1 3 1 3 2 2 2 3 1 2 2 3 1 3 1 3 3 1 3 2 3 1 2 1 1 3 3 3 1 2 1 3 3
##  [704] 1 3 3 3 3 3 2 1 1 3 3 1 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 2 3 3 1 3 3 3 3 1 3
##  [741] 3 1 3 1 1 3 1 1 3 3 3 3 1 3 2 3 3 3 3 1 1 3 3 3 3 3 1 1 1 3 1 1 1 2 2 3 3
##  [778] 3 2 1 3 3 1 3 3 2 2 1 1 1 1 3 3 2 2 1 2 2 2 1 3 3 3 3 2 2 2 3 3 3 2 2 3
##  [815] 2 2 2 2 3 1 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 3 1 1 2 3 2 3 1 1 2 3 3 3 3 3 2
##  [852] 2 1 1 1 3 1 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 3 3 1 3 3 3 2 3 1 3 3 2
##  [889] 3 1 1 3 2 3 3 3 2 3 2 3 2 3 3 3 3 1 3 3 3 3 2 2 2 2 3 2 3 1 1 3 2 1 3 1 2
##  [926] 1 1 1 2 2 3 3 1 3 3 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 3 3 2 3
##  [963] 3 2 2 2 2 1 2 3 2 2 2 3 2 1 1 1 3 2 2 3 1 2 2 3 2 1 3 2 3 1 3 1 1 3 3 3 3
## [1000] 3 3 2 2 3 1 3 2 2 2 2 2 2 3 3 3 2 2 1 1 3 2 2 3 2 3 3 3 3 1 3 3 3 3 3 3 2
## [1037] 3 3 2 3 3 3 3 2 3 3 3 3 2 2 3 2 3 2 1 1 2 1 2 2 2 2 2 2 3 3 3 2 2 1 2 1 1
## [1074] 3 1 1 2 2 2 1 2 1 1 1 1 1 2 2 2 2 1 2 3 2 3 2 3 2 2 2 2 3 3 3 3 3 2 2 3 2
## [1111] 3 3 2 2 3 3 3 3 3 3 3 3 3 2 3 2 3 3 1 1 3 1 3 3 2 3 2 2 1 1 1 1 3 3 1 1 3
## [1148] 2 3 2 2 3 3 2 3 3 1 3 2 2 2 2 2 3 3 2 2 2 3 3 2 3 2 1 1 3 3 3 3 2 2 2 1 3
## [1185] 1 3 3 3 1 3 2 3 3 3 3 3 1 3 1 1 3 3 2 1 3 3 3 2 3 3 3 3 3 2 2 2 1 1 2 2 2
## [1222] 2 1 2 2 1 1 2 3 1 2 1 1 2 3 1 3 3 3 3 1 1 2 1 1 3 3 3 3 3 3 1 3 3 3 3 1 3
## [1259] 3 3 2 3 1 3 3 3 3 2 3 3 3 3 3 1 3 1 2 3 1 2 3 3 3 1 3 2 3 3 1 1 3 3 3 3 3
```

```
## [1296] 1 1 3 3 3 3 3 2 1 1 1 1 3 1 1 1 3 3 3 1 1 3 2 1 2 1 3 2 2 3 3 3 3 3 1 1 1
## [1333] 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 1 3 3 3 3 3 3 3 1 2 2 3 2 3 3 3 3 1 1
## [1370] 3 2 2 2 1 3 1 3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 3 3 3 3 1 3 3 1 3 3 1 1 2 2 3 2
## [1407] 2 3 2 3 3 3 2 2 2 3 2 3 3 1 3 1 3 3 2 2 2 3 3 1 3 1 3 3 1 1 1 3 3 1 3 1 3
## [1444] 3 1 1 3 3 1 3 3 2 3 1 2 3 1 1 2 2 1 3 3 3 3 3 1 3 1 3 2 3 2 3 1 3 1 3 3 2
## [1481] 3 2 2 2 3 3 3 3 3 3 3 3 3 3 1 3 3 1 3 3 3 3 1 3 3 2 3 3 2 2 2 3 3 3 3 3 3 3
## [1518] 3 2 3 3 3 3 1 3 3 3 3 1 1 3 3 3 1 3 3 3 3 3 1 3 3 3 2 2 3 3 3 2 3 3 3 3 3
## [1555] 3 3 3 3 1 1 1 1 3 3 3 3 2 3 3 3 2 3 1 3 1 3 2 3 3 3 3 3 3 1 2 2 1 3 1 1 3
## [1592] 3 3 3 3 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1] 3386.103 5040.596 4195.309
##  (between_SS / total_SS =  28.2 %)
##
## Available components:
##
##  [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
##  [6] "betweenss"    "size"         "iter"         "ifault"       "silinfo"
## [11] "nbclust"      "data"
```

```
library("factoextra")

fviz_cluster(km.res3,geom="point",ellipse.type = "norm",
             palette="jco",ggtheme = theme_minimal())
```



There is some overlap between the clusters, especially between Cluster 2 (yellow) and Cluster 3 (gray). This suggests that these clusters share similar characteristics in the reduced dimensional space. Cluster 1 (blue) appears more distinct, indicating better separation from the other clusters. Overlapping points between clusters imply that these clusters are not perfectly distinct in this reduced dimensional space. There may be similarities in the original dataset among these clusters.

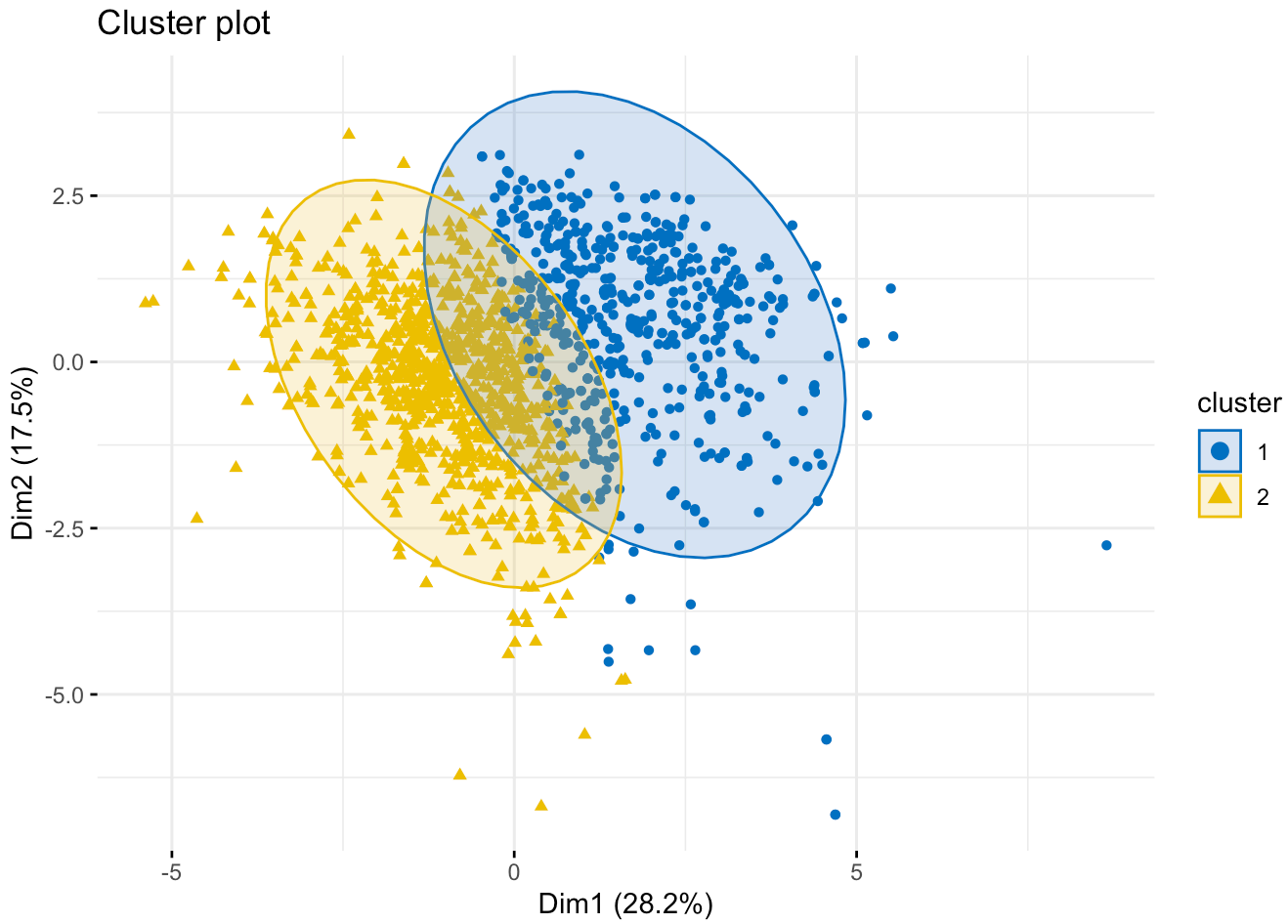
2 Clusters

```
library("factoextra")

km.res2<-eclust(df.scaled,"kmeans",k=2,nstart=25,graph = FALSE)

km.res2
```

```
## K-means clustering with 2 clusters of sizes 590, 1009
##
## Cluster means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1     0.9207271       -0.6361774   0.9851367     0.16097077  0.2641746
## 2    -0.5383835        0.3719967  -0.5760462    -0.09412562 -0.1544728
##   free.sulfur.dioxide total.sulfur.dioxide    density         pH  sulphates
## 1          -0.2356324           -0.2152177  0.4994664 -0.7077417  0.5314734
## 2           0.1377831            0.1258458 -0.2920566  0.4138430 -0.3107723
##      alcohol
## 1  0.1730540
## 2 -0.1011911
##
## Clustering vector:
##    [1] 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 1 1 2 1 1 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2
##   [38] 1 2 1 1 2 1 1 2 2 2 1 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
##   [75] 1 1 1 2 2 2 2 1 2 1 2 2 1 2 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2
##  [112] 2 2 1 2 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 2 2 2 2 2 2 1
##  [149] 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2
##  [186] 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 1 2 2 2 1 1 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2
##  [223] 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 1 2 2 2 2 2 1 2 1 2 2 2 1 2 1
##  [260] 1 2 2 2 2 2 1 1 2 1 2 1 2 1 1 2 2 2 2 2 1 1 1 1 1 2 1 2 2 1 2 2 1 2 1 1 2 1 1
##  [297] 1 2 2 2 2 2 1 2 2 1 1 2 1 1 2 1 2 2 2 2 2 2 2 2 1 2 1 2 2 1 1 1 1 1 1 1 1 1 2
##  [334] 2 2 1 1 2 1 1 1 1 1 1 1 2 2 1 1 2 1 2 2 1 2 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1
##  [371] 2 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 1 1 1 1 1 2 1 1 2 2 2 1 1 2 1 1
##  [408] 1 1 1 2 2 2 1 2 1 1 2 1 2 1 2 2 1 2 2 2 2 1 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1
##  [445] 2 2 1 1 2 1 1 1 2 1 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1
##  [482] 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [519] 1 2 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2 1
##  [556] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 2
##  [593] 1 1 2 1 1 1 2 1 2 1 2 1 2 2 1 1 1 2 1 1 2 1 1 2 2 1 1 1 2 2 1 2 2 2 2 2 2
##  [630] 2 2 1 2 2 2 2 2 2 2 2 1 1 2 1 2 1 2 2 2 2 2 1 2 1 1 1 1 1 1 2 2 2 2 2 1 1 2
##  [667] 1 1 1 1 2 2 2 2 1 1 1 2 2 1 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 1 1 2 2
##  [704] 1 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
##  [741] 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2
##  [778] 2 1 2 2 2 2 2 2 2 1 1 1 1 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 1 1 1 2 2 2 1 1 2
##  [815] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 1 2 1 2 1 2 1 2 2 2 2 2 1
##  [852] 1 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 2 1 2 2 2 2 1
##  [889] 2 1 2 2 1 2 2 2 2 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 1 1 1 1 2 1 2 2 2 2 1 2 2 2 1
##  [926] 2 1 2 1 1 2 2 2 2 2 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 1 2
##  [963] 2 1 1 1 1 2 1 2 1 1 1 1 1 2 2 2 2 2 1 1 2 2 1 1 2 1 2 2 1 2 2 2 2 1 2 2 2 2
## [1000] 2 2 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 1 1 2 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 1
## [1037] 2 2 1 2 2 2 2 2 1 2 2 2 2 1 1 2 1 2 1 2 2 1 2 1 1 1 1 1 1 2 2 2 1 1 2 1 2 2
## [1074] 2 2 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 2 2 2 2 2 1 1 2 1
## [1111] 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2 1 2 2 1 1 2 2 2 2 1 2 1 1 2 2 2 1 2 2 2 1 2
## [1148] 1 2 1 1 2 2 1 2 2 1 2 1 1 1 1 1 2 2 1 1 1 2 2 1 2 1 2 2 2 2 2 2 1 1 1 1 2
## [1185] 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 1 2 2 2 2 2 1 1 1 2 2 1 1 1
## [1222] 1 2 1 1 1 2 1 2 2 1 2 2 1 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1259] 2 2 1 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 1 2 2 1 1 2 2 2 2 2
## [1296] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 1 1 2 2 2 2 2 2 2 2
## [1333] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 1 2
## [1370] 2 1 1 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 2 2 2 2 2 2 1 1 2 1
```

```
## [1407] 1 2 1 2 2 2 1 1 1 2 1 2 2 2 2 2 2 2 1 1 1 2 2 1 2 2 2 2 1 1 1 2 2 2 2 2 2
## [1444] 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2 2 1
## [1481] 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 1 1 2 2 2 2 2 2 2
## [1518] 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 1 2 2 2 2 2
## [1555] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 1 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2
## [1592] 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 6810.586 7510.572
##  (between_SS / total_SS =  18.5 %)
##
## Available components:
##
##  [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
##  [6] "betweenss"    "size"         "iter"         "ifault"       "silinfo"
## [11] "nbclust"      "data"
```

```
library("factoextra")

fviz_cluster(km.res2,geom="point",ellipse.type = "norm",
             palette="jco",ggtheme = theme_minimal())
```



The ellipses for the two clusters overlap significantly, indicating that there is no clear separation between the two clusters in this 2D representation.This may be because, the clusters are not well-separated in the reduced dimensional space, the data have overlapping features.

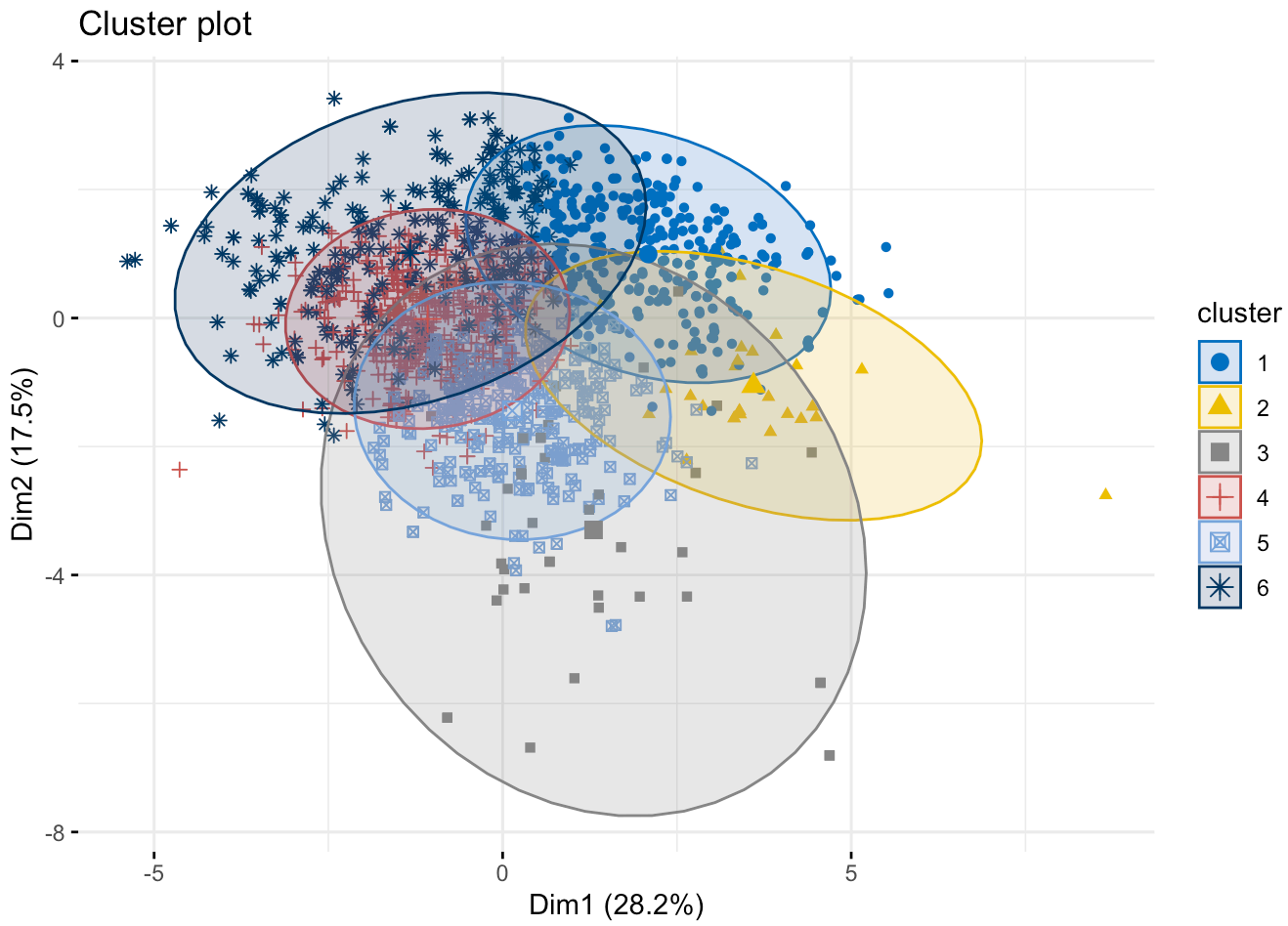
6 clusters

```
library("factoextra")

km.res6<-eclust(df.scaled,"kmeans",k=6,nstart=25,graph = FALSE)

km.res6
```

```
## K-means clustering with 6 clusters of sizes 364, 28, 46, 527, 332, 302
##
## Cluster means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar    chlorides
## 1    1.33655215     -0.670499956  1.12166908     0.07448123 -0.007110832
## 2    0.09538646      0.002199115  1.18118314    -0.38975023  5.782950580
## 3   -0.18857827     -0.051566914  0.40008716     4.24343185  0.206253249
## 4   -0.47936911      0.691294020 -0.83074207    -0.19030201 -0.061397492
## 5   -0.05625796      0.042955144  0.06797321    -0.10026451 -0.031799691
## 6   -0.69270212     -0.437750230 -0.14745270    -0.25767897 -0.416913732
##   free.sulfur.dioxide total.sulfur.dioxide     density         pH   sulphates
## 1         -0.57189053           -0.5463782  0.76367795 -0.8471178  0.34527771
## 2         -0.04950011            0.5101700  0.18001552 -1.7352487  3.66226647
## 3          1.58937177            1.7416784  1.03507819 -0.1944875  0.08530535
## 4         -0.43239594           -0.4170279 -0.05800168  0.4232134 -0.39476514
## 5          0.98409163            1.1853676  0.22967394 -0.1459251 -0.19538265
## 6          0.12449435           -0.2294327 -1.24608499  0.6334369  0.13496581
##      alcohol
## 1  0.1740147
## 2 -0.8694593
## 3 -0.3153203
## 4 -0.4183154
## 5 -0.5878917
## 6  1.2951673
##
## Clustering vector:
##    [1] 4 4 4 1 4 4 4 4 4 3 4 3 4 1 5 5 5 2 4 2 5 5 4 4 4 4 4 4 4 4 4 4 5 3 4 4 4
##   [38] 4 4 3 3 5 2 4 4 6 5 1 4 5 4 4 4 5 5 4 1 3 4 4 5 5 4 4 4 4 4 4 5 4 4 5 5 4
##   [75] 5 1 1 4 4 5 4 2 5 2 6 4 2 4 5 4 5 2 2 4 4 6 4 4 4 4 4 4 4 4 4 4 2 4 5 5 4
##  [112] 5 5 1 4 1 4 4 4 5 4 4 4 4 5 5 4 4 4 4 5 6 6 4 4 4 4 4 5 5 4 4 6 4 6 5 4 5
##  [149] 4 4 6 2 5 5 3 3 3 3 4 4 4 4 4 3 3 5 5 4 4 2 4 4 4 6 4 4 4 4 4 4 4 2 4 4 4
##  [186] 5 5 4 5 5 5 4 5 4 4 5 4 1 6 4 1 5 4 4 4 1 1 5 5 1 1 4 1 5 4 3 4 4 4 5 5 5
##  [223] 4 4 5 5 2 4 5 4 6 4 5 4 4 4 4 4 4 4 2 1 5 1 1 4 4 4 4 4 1 4 1 5 4 5 1 4 2
##  [260] 1 5 4 4 5 1 1 4 6 4 1 5 1 1 4 3 5 4 1 1 1 1 2 4 1 5 5 1 5 4 5 4 2 1 4 1 1
##  [297] 5 4 4 4 4 1 4 4 5 1 4 1 1 4 1 5 5 5 5 6 5 5 5 5 5 5 4 5 3 3 1 1 1 1 1 1 5
##  [334] 4 4 1 6 5 1 1 1 1 1 1 1 5 6 1 1 4 1 4 4 1 6 6 1 1 1 1 5 5 1 1 1 1 1 1 1 1
##  [371] 5 1 1 5 1 1 1 1 1 5 1 1 1 1 5 4 5 4 5 1 6 1 1 5 1 1 3 1 1 4 3 6 1 1 4 1 1
##  [408] 1 1 1 5 5 4 1 5 3 1 5 1 4 5 6 4 1 4 6 6 4 4 1 1 5 1 1 1 1 5 1 1 4 1 1 1 1
##  [445] 6 4 1 1 4 1 1 2 4 1 6 1 4 5 1 1 1 4 1 5 1 1 1 6 1 5 1 1 1 1 1 1 1 1 1 4 3
##  [482] 1 1 1 1 1 1 1 1 1 5 6 6 5 3 1 4 5 1 5 4 3 3 1 1 1 1 1 5 1 1 5 1 1 1 3 1 1
##  [519] 1 5 1 5 5 5 5 5 5 5 5 1 1 1 1 6 1 1 1 4 1 1 5 1 4 1 1 5 4 1 1 1 4 1 1 6 1
##  [556] 1 1 1 1 1 1 5 5 5 1 1 4 4 1 6 1 6 1 1 1 1 4 5 5 1 1 1 1 1 5 4 1 5 6 1 5 5
##  [593] 5 1 4 3 1 1 4 1 4 1 4 1 5 4 1 5 1 6 5 1 4 1 2 5 5 1 1 1 5 5 1 6 4 4 4 4 4
##  [630] 5 4 1 4 4 5 4 5 5 4 1 1 5 1 5 1 4 4 4 6 3 1 5 1 1 1 5 1 1 4 4 4 5 4 1 1 5
##  [667] 1 1 1 1 5 4 5 4 1 1 1 4 5 1 1 4 5 4 5 4 4 4 4 1 4 4 2 5 5 6 4 4 5 1 5 4 4
##  [704] 5 4 4 4 4 6 1 5 5 4 4 5 4 4 4 4 4 4 5 4 5 4 4 4 4 4 6 2 4 4 5 4 4 4 4 5 4
##  [741] 4 5 4 1 5 4 5 5 4 4 4 4 5 4 2 4 4 4 4 5 5 4 4 4 4 4 5 5 5 4 5 5 5 1 1 4 4
##  [778] 4 4 5 4 4 5 4 4 1 1 5 5 5 5 5 4 4 1 1 5 6 1 1 5 4 6 4 4 6 6 6 4 4 4 1 1 6
##  [815] 1 1 1 1 4 4 4 6 4 4 4 4 6 4 6 6 4 6 5 5 4 4 6 6 1 4 1 4 5 5 1 4 4 4 4 4 1
##  [852] 1 5 5 5 4 5 6 1 6 4 6 4 4 4 4 6 6 6 4 6 6 5 1 1 1 4 6 5 4 4 6 6 4 5 4 4 1
##  [889] 6 3 5 4 1 4 4 4 6 4 6 4 6 4 4 4 4 5 6 6 5 6 1 3 1 1 6 6 4 3 5 6 1 5 6 3 6
##  [926] 5 5 5 6 6 4 4 5 4 4 6 6 1 6 6 6 1 1 1 6 1 1 6 6 6 6 6 6 1 6 1 1 1 6 4 6 4
```

```
##  [963] 4 1 6 6 1 5 6 4 1 1 1 6 1 5 5 5 6 1 1 4 6 1 1 4 1 5 4 1 4 5 5 5 5 4 6 6 4
## [1000] 6 6 5 6 6 5 6 6 6 1 5 6 1 4 4 4 1 6 6 6 4 1 1 4 6 4 4 6 6 5 4 4 4 4 4 4 1
## [1037] 6 4 6 6 4 4 6 3 6 6 4 6 1 1 6 2 6 6 5 5 1 5 1 1 1 6 6 1 4 4 6 1 1 5 1 3 5
## [1074] 4 3 5 1 1 1 3 1 3 5 6 5 5 1 6 1 1 5 6 6 6 4 1 4 4 6 4 6 6 6 6 6 6 6 6 4 1
## [1111] 4 6 6 1 6 4 4 4 6 6 6 6 6 1 4 6 6 6 5 5 4 5 6 4 6 6 1 1 5 5 5 5 6 6 5 5 4
## [1148] 1 6 1 6 6 4 1 6 4 5 6 6 1 1 1 6 4 4 2 1 6 6 6 1 6 1 5 5 6 4 6 6 6 6 1 5 4
## [1185] 5 6 4 6 5 4 1 4 6 4 4 4 5 4 6 5 4 6 6 3 6 6 6 5 6 6 4 5 4 1 1 6 5 6 1 6 1
## [1222] 1 5 1 1 5 5 4 6 5 6 5 5 1 6 3 4 6 4 6 5 5 6 5 3 4 4 4 6 4 4 5 4 4 4 6 5 4
## [1259] 4 4 2 4 5 4 6 4 4 1 5 6 6 6 6 5 6 5 6 4 5 1 5 5 4 5 6 1 6 6 5 5 4 5 6 4 5
## [1296] 5 5 6 6 4 6 4 1 6 5 5 5 4 5 5 5 6 4 5 5 5 6 1 5 2 5 6 6 6 6 6 6 4 5 5 5
## [1333] 4 4 4 6 4 4 4 4 4 4 4 4 1 4 6 4 4 4 5 6 4 4 4 4 4 6 3 1 1 4 1 4 6 4 4 5 5
## [1370] 4 2 1 2 5 4 5 4 6 4 4 4 4 5 5 5 5 4 4 4 5 6 4 4 5 5 4 4 5 4 4 5 5 6 1 4 6
## [1407] 1 6 6 6 4 6 1 5 1 4 1 6 4 5 4 5 6 4 1 1 6 6 4 6 6 5 6 4 3 3 5 4 4 5 6 5 4
## [1444] 6 5 5 4 4 5 6 6 1 6 5 1 4 6 5 6 6 5 4 4 4 5 5 5 4 5 4 4 6 6 6 3 6 3 6 4 1
## [1481] 4 1 4 6 4 4 4 4 6 6 6 6 6 5 6 6 5 4 4 4 4 5 4 6 6 4 4 6 1 6 5 4 4 6 4 4 6
## [1518] 6 4 4 6 4 6 5 6 4 4 4 5 5 6 4 4 5 6 4 6 4 6 5 6 6 4 1 1 4 4 6 1 6 4 4 6 4
## [1555] 4 4 4 4 3 5 5 5 4 4 4 6 6 4 4 6 6 6 5 6 3 6 1 6 4 6 6 6 6 5 6 6 6 6 6 3 6
## [1592] 6 6 4 6 6 6 4 6
##
## Within cluster sum of squares by cluster:
## [1] 2195.2138  428.0534  755.3546 2234.3143 1790.5319 1954.7558
##  (between_SS / total_SS =  46.8 %)
##
## Available components:
##
##  [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
##  [6] "betweenss"    "size"         "iter"         "ifault"       "silinfo"
## [11] "nbclust"      "data"
```

```
library("factoextra")

fviz_cluster(km.res6,geom="point",ellipse.type = "norm",
             palette="jco",ggtheme = theme_minimal())
```



Cluster validation

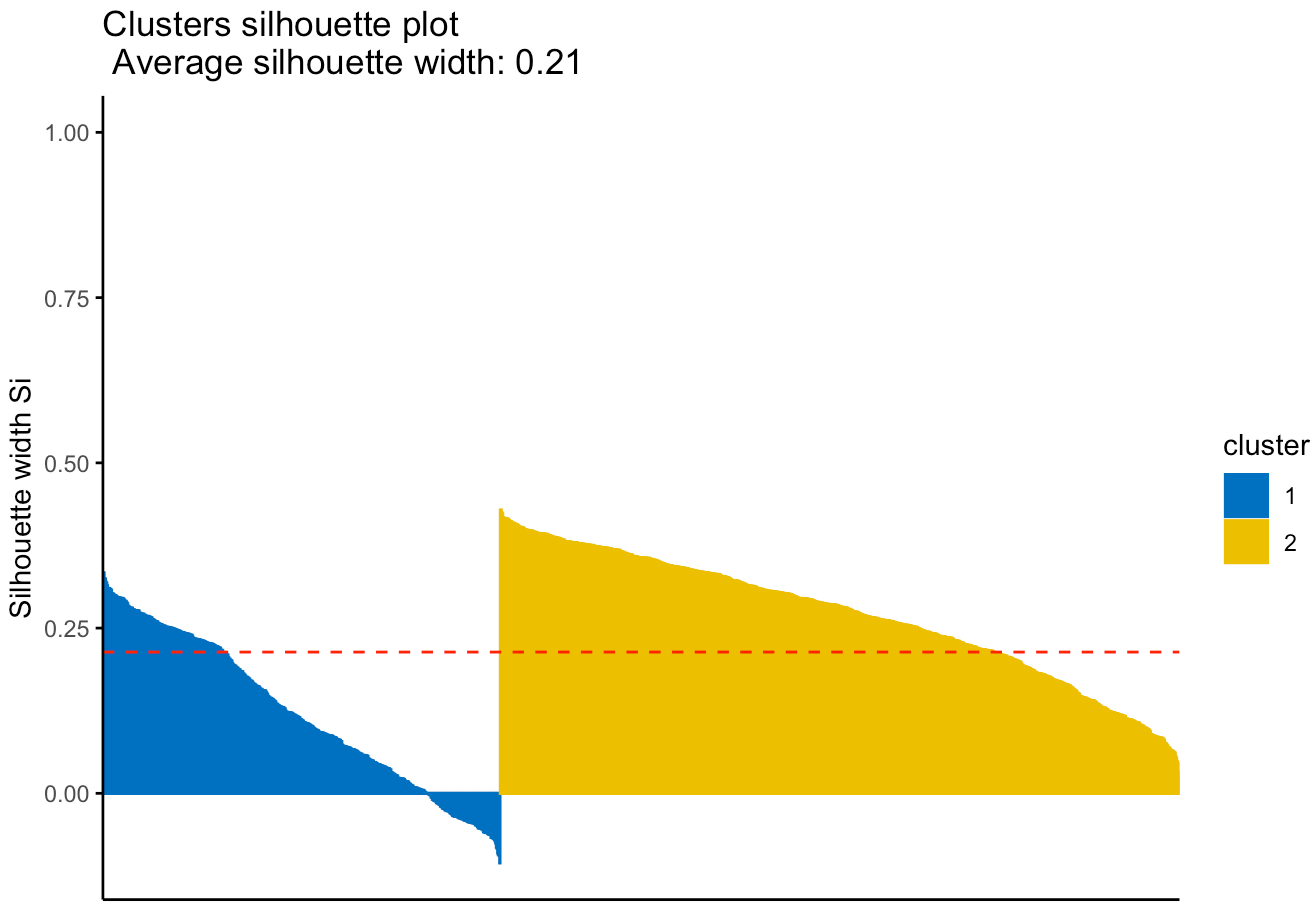
Internal Validation

1) 2 Clusters

Silhoutte Coefficient

```
library(factoextra)
library(cluster)

# Visualize the silhouette plot
fviz_silhouette(km.res2, palette = "jco", ggtheme = theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  590          0.12
## 2       2 1009          0.27
```



The silhouette plot above shows negative values suggesting there are points that are misclassified in cluster 1, as they are closer to cluster 2.

```
silinfo2<-km.res2$silinfo

#Average silhouette width of each cluster (2 clusters)
silinfo2$clus.avg.widths
```

```
## [1] 0.1200212 0.2685250
```

```
#Finding the points with negative width

sil2<-km.res2$silinfo$widths[,1:3]

neg.sil<-which(sil2[,'sil_width']<0)
sil2[neg.sil, ,drop=FALSE]
```

```
##      cluster neighbor      sil_width
## 669        1        2 -0.00009837933
## 1168       1        2 -0.00041065264
## 324        1        2 -0.00055971594
## 1243       1        2 -0.00143609392
## 1575       1        2 -0.00211636788
## 1571       1        2 -0.00234391271
## 901        1        2 -0.00571372132
## 1430       1        2 -0.00622920109
## 1263       1        2 -0.00920488624
## 1157       1        2 -0.00946631648
## 1080       1        2 -0.01063808947
## 525        1        2 -0.01081321772
## 1082       1        2 -0.01137171490
## 1203       1        2 -0.01213462378
## 1088       1        2 -0.01423852298
## 853        1        2 -0.01452214846
## 1084       1        2 -0.01512023762
## 319        1        2 -0.01619104605
## 321        1        2 -0.01619104605
## 969        1        2 -0.01721133578
## 966        1        2 -0.01962267202
## 563        1        2 -0.01971678955
## 1406       1        2 -0.02062188554
## 476        1        2 -0.02263169033
## 479        1        2 -0.02263169033
## 817        1        2 -0.02316877462
## 656        1        2 -0.02410340695
## 981        1        2 -0.02543366954
## 984        1        2 -0.02543366954
## 148        1        2 -0.02625887880
## 416        1        2 -0.02648356201
## 591        1        2 -0.02657546325
## 593        1        2 -0.02657546325
## 614        1        2 -0.02795962031
## 526        1        2 -0.02953324336
## 529        1        2 -0.03012444401
## 464        1        2 -0.03032421030
## 523        1        2 -0.03226392552
## 1484       1        2 -0.03368740597
## 596        1        2 -0.03392537779
## 1092       1        2 -0.03455689114
## 925        1        2 -0.03484811076
## 929        1        2 -0.03484811076
## 858        1        2 -0.03505355783
## 530        1        2 -0.03507979869
## 537        1        2 -0.03507979869
## 1459       1        2 -0.03657304646
## 788        1        2 -0.03706639742
## 789        1        2 -0.03706639742
## 579        1        2 -0.03759959006
## 927        1        2 -0.03801562621
```

```
## 1368       1        2 -0.03839440737
## 711        1        2 -0.03879579189
## 961        1        2 -0.03894977233
## 965        1        2 -0.03894977233
## 1231       1        2 -0.04039064067
## 564        1        2 -0.04100159523
## 362        1        2 -0.04107568564
## 578        1        2 -0.04117899235
## 1076       1        2 -0.04170842186
## 1219       1        2 -0.04264774485
## 1289       1        2 -0.04321025666
## 1290       1        2 -0.04321025666
## 936        1        2 -0.04372034643
## 937        1        2 -0.04372034643
## 394        1        2 -0.04376205167
## 1113       1        2 -0.04427510087
## 23         1        2 -0.04484944741
## 28         1        2 -0.04484944741
## 44         1        2 -0.04490387478
## 524        1        2 -0.04490902957
## 1159       1        2 -0.04612801108
## 21         1        2 -0.04793567527
## 380        1        2 -0.04804501526
## 562        1        2 -0.04852775498
## 305        1        2 -0.05024092070
## 40         1        2 -0.05240774762
## 41         1        2 -0.05240774762
## 701        1        2 -0.05276826886
## 1146       1        2 -0.05295294888
## 1142       1        2 -0.05319101390
## 1427       1        2 -0.05400134306
## 623        1        2 -0.05404799474
## 457        1        2 -0.05523920130
## 151        1        2 -0.05830342934
## 1361       1        2 -0.05871312440
## 974        1        2 -0.05878263266
## 421        1        2 -0.05881331878
## 1483       1        2 -0.05891042464
## 1199       1        2 -0.05984734063
## 1586       1        2 -0.06068231970
## 1098       1        2 -0.06180257386
## 1100       1        2 -0.06180257386
## 429        1        2 -0.06193830145
## 1519       1        2 -0.06260456445
## 567        1        2 -0.06665324935
## 568        1        2 -0.06665324935
## 1226       1        2 -0.06722772517
## 608        1        2 -0.06804211029
## 748        1        2 -0.06821813533
## 577        1        2 -0.07001089829
## 38         1        2 -0.07188274988
## 297        1        2 -0.07531166640
```

```
## 1321       1        2 -0.08161287951
## 51         1        2 -0.08343880633
## 1228       1        2 -0.09054643880
## 704        1        2 -0.09313637532
## 779        1        2 -0.09341001632
## 541        1        2 -0.10549515063
```

The table above give a summary of the negative silhouette points. There are 109 data points that misclassified.

Dunn Index

```
km_stats2<-cluster.stats(dist(df.scaled),km.res2$cluster)

km_stats2$dunn
```

```
## [1] 0.03093463
```

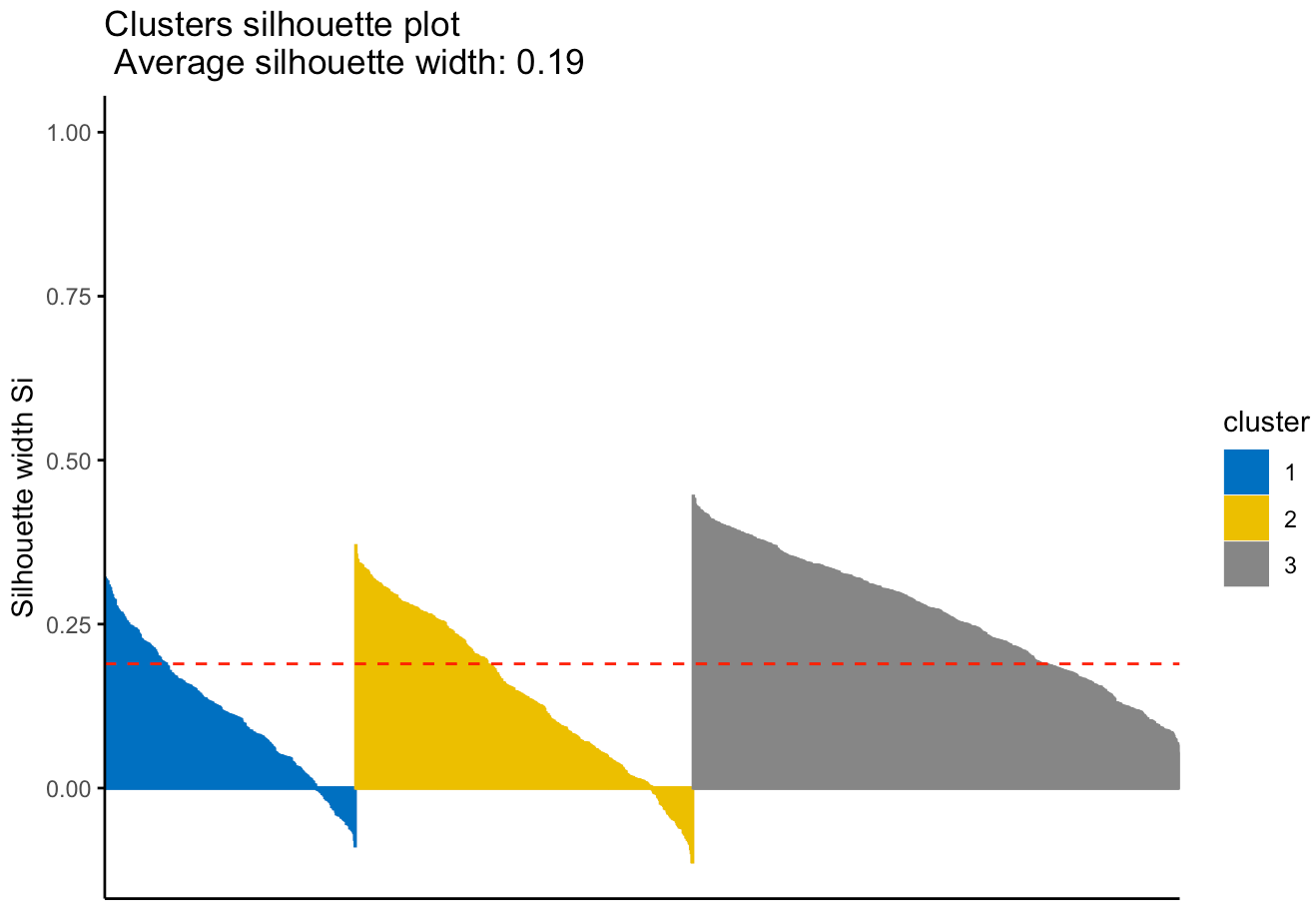
Connectivity Score

```
connectivity_score <- connectivity(distance = dist(df.scaled), clusters = km.res2$cluste
r)

# Print the connectivity score
print(connectivity_score)
```

```
## [1] 301.6595
```

```
library(factoextra)
library(cluster)

# Visualize the silhouette plot
fviz_silhouette(km.res3, palette = "jco", ggtheme = theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  373          0.11
## 2       2  502          0.14
## 3       3  724          0.26
```



A higher silhouette width indicates better-defined clusters. Although the difference is not large, the silhouette width suggests that 2 clusters might be slightly better in terms of cluster quality.

```
silinfo3<-km.res3$silinfo

#Average silhouette width of each cluster (3 clusters)
silinfo3$clus.avg.widths
```

```
## [1] 0.1124378 0.1424774 0.2616282
```

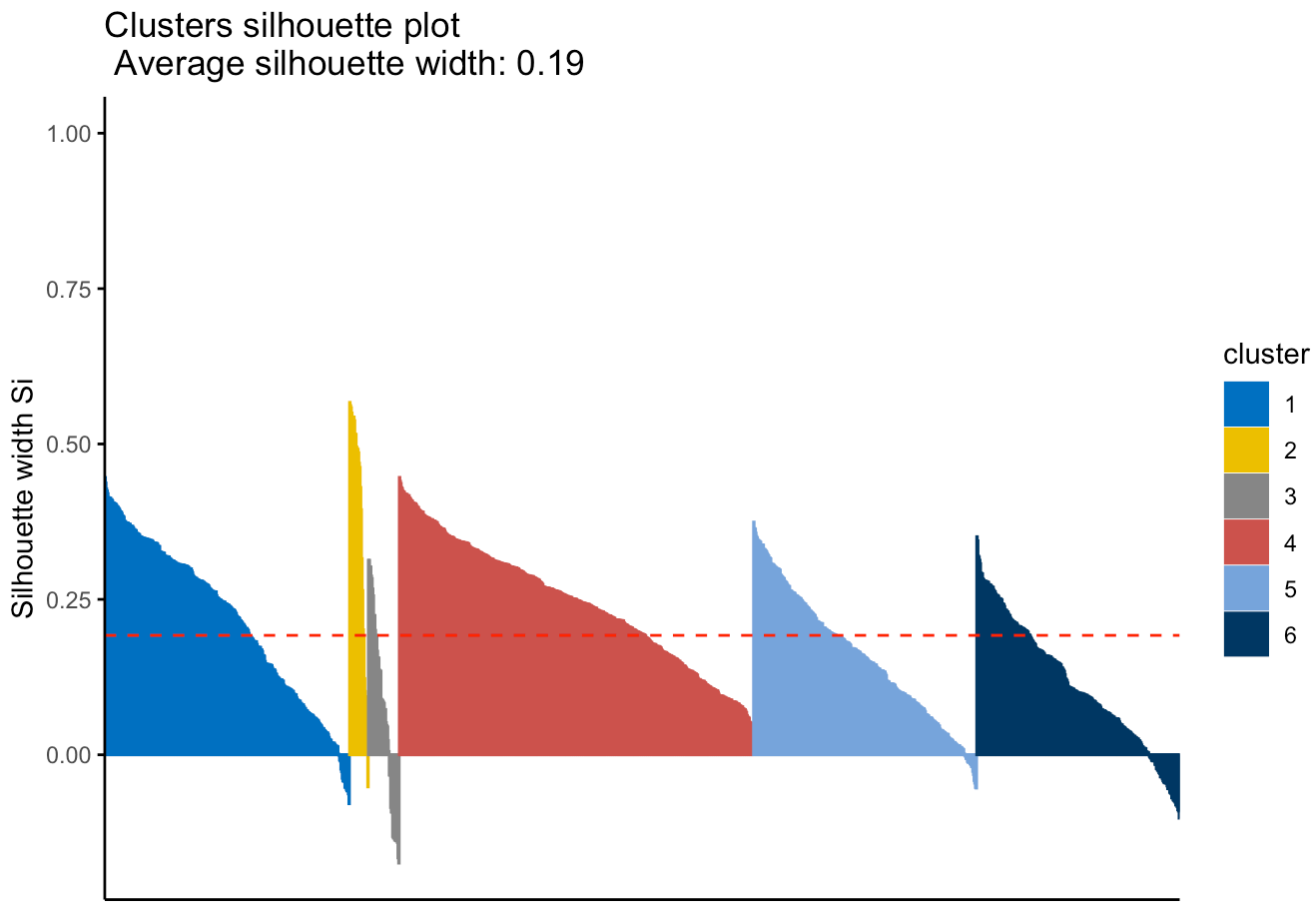
Dunn Index

```
km_stats3<-cluster.stats(dist(df.scaled),km.res3$cluster)

km_stats3$dunn
```

```
## [1] 0.02120514
```

```
library(factoextra)
library(cluster)

fviz_silhouette(km.res6, palette = "jco", ggtheme = theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  364          0.22
## 2       2   28          0.38
## 3       3   46          0.07
## 4       4  527          0.24
## 5       5  332          0.16
## 6       6  302          0.11
```




```
km_stats6<-cluster.stats(dist(df.scaled),km.res6$cluster)

km_stats6$dunn
```

```
## [1] 0.04451991
```

```
connectivity_score6 <- connectivity(distance = dist(df.scaled), clusters = km.res6$clust
er)

# Print the connectivity score
print(connectivity_score6)
```

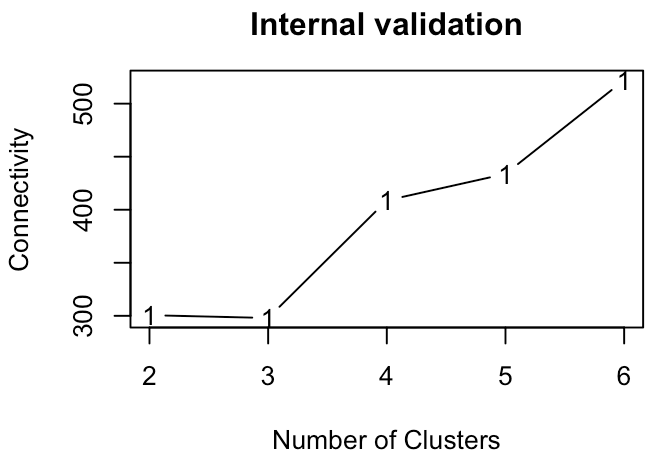
```
## [1] 522.175
```

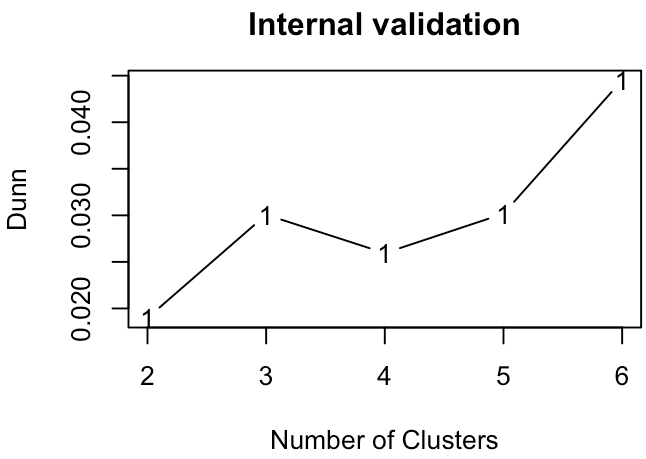
Internal Validation Summary

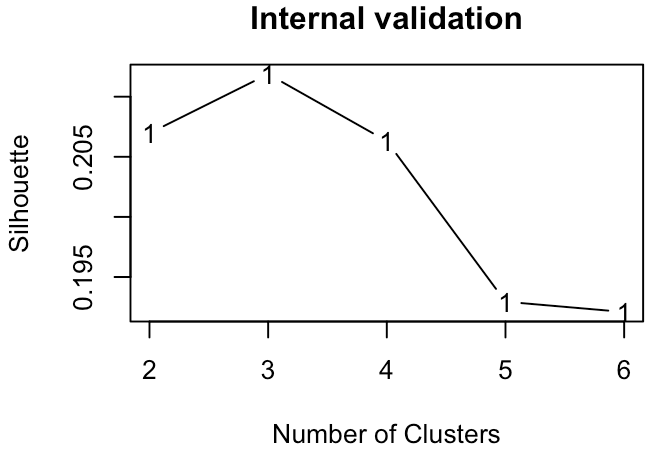
```
library(clValid)

clmethodsK<-c("kmeans")

internK<-clValid(df.scaled,nClust = 2:6,clMethods = clmethodsK,validation = "internal",m
etric = "euclidean",maxitems = 1599)

summary(internK)
```

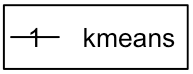
```
##
## Clustering Methods:
##  kmeans
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                              2        3        4        5        6
##
## kmeans Connectivity  300.6099 297.9528 408.7813 433.7774 522.1750
##        Dunn            0.0190   0.0301   0.0259   0.0302   0.0445
##        Silhouette      0.2070   0.2119   0.2063   0.1929   0.1921
##
## Optimal Scores:
##
##              Score    Method Clusters
## Connectivity 297.9528 kmeans 3
## Dunn           0.0445 kmeans 6
## Silhouette     0.2119 kmeans 3
```

```
op<-par(no.readonly=TRUE)
par(mfrow=c(2,2),mar=c(4,4,3,1))
plot(internK, legend=FALSE)
plot(nClusters(internK), measures(internK, "Dunn")[,,1], type="n", axes=F, xlab="", ylab
="")
legend("center", clusterMethods(internK), col=1:9, lty=1:9, pch=paste(1:9))
```









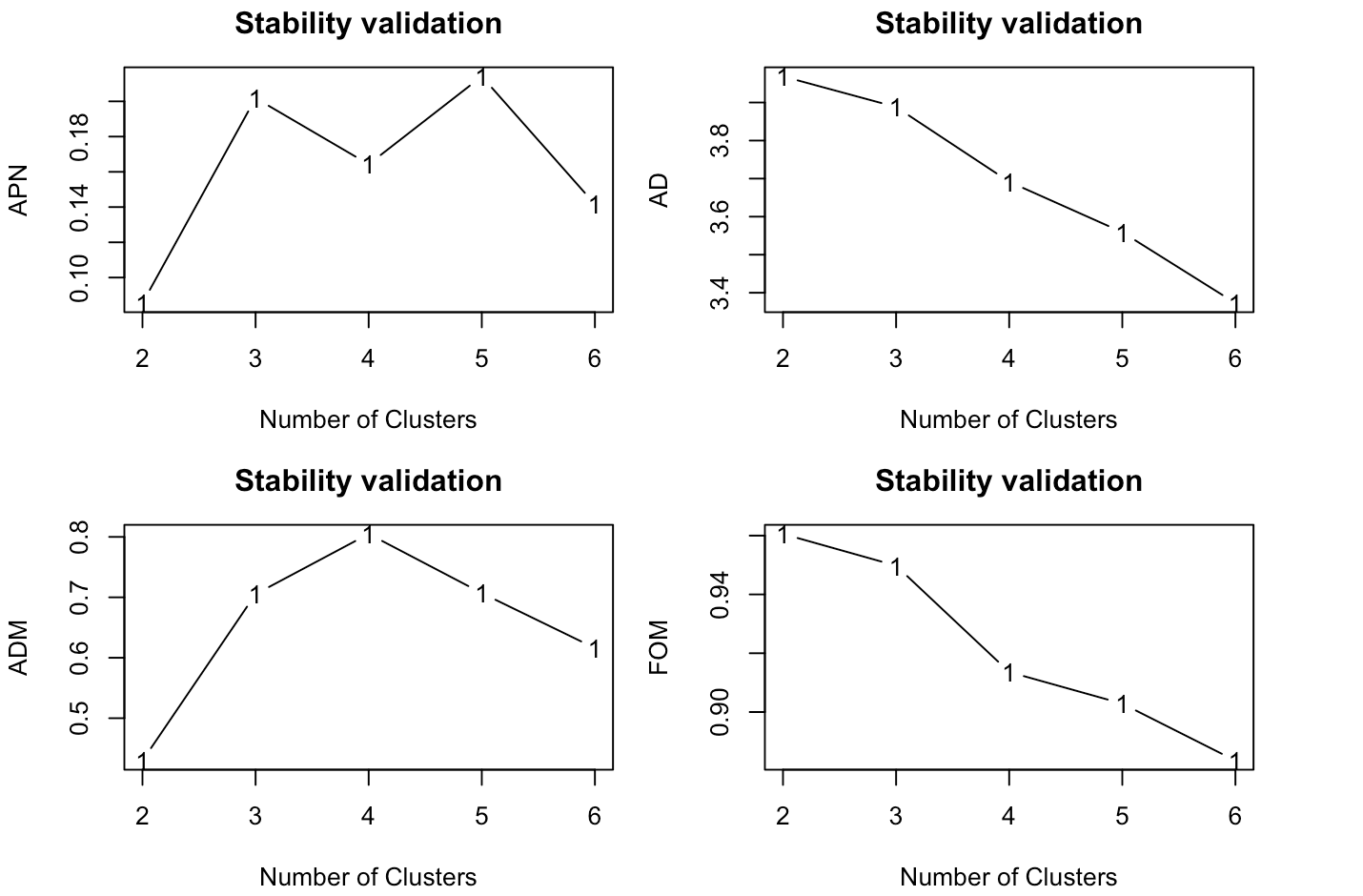
The Internal Validation summary says that connectivity is lowest for 3 clusters, Dunn Index is highest for 6 clusters.

Stability

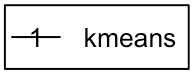
```
library(clValid)
stab<-clValid(df.scaled,nClust = 2:6,clMethods = clmethodsK,validation = "stability",max
items = 1599)
summary(stab)
```

```
##
## Clustering Methods:
##  kmeans
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                  2      3      4      5      6
##
## kmeans APN  0.0855 0.2018 0.1643 0.2142 0.1416
##        AD   3.9687 3.8881 3.6917 3.5591 3.3727
##        ADM  0.4298 0.7053 0.8049 0.7070 0.6160
##        FOM  0.9606 0.9498 0.9136 0.9028 0.8835
##
## Optimal Scores:
##
##     Score  Method Clusters
## APN 0.0855 kmeans 2
## AD  3.3727 kmeans 6
## ADM 0.4298 kmeans 2
## FOM 0.8835 kmeans 6
```

```
par(mfrow=c(2,2),mar=c(4,4,3,1))
plot(stab, measures=c("APN","AD","ADM","FOM"),legend=FALSE)
```




```
plot(nClusters(stab), measures(stab, "APN")[,,1], type="n", axes=F, xlab="", ylab="")
legend("center", clusterMethods(stab), col=1:9, lty=1:9, pch=paste(1:9))
```



The Stability scores above says APN and ADM suggest 2 clusters as best and AD and FOM suggest 6 clusters.

External Validation

1) Ajusted Rand Index (ARI)

```
true_labels <- Rwine$quality
```

```
library("fpc")

quality <- as.numeric(Rwine$quality)
clust_stats3 <- cluster.stats(d = dist(df.scaled),
                              quality, km.res3$cluster)
clust_stats3$corrected.rand
```

```
## [1] 0.02967802
```

```
library("fpc")

clust_stats2 <- cluster.stats(d = dist(df.scaled),
                              quality, km.res2$cluster)
clust_stats2$corrected.rand
```

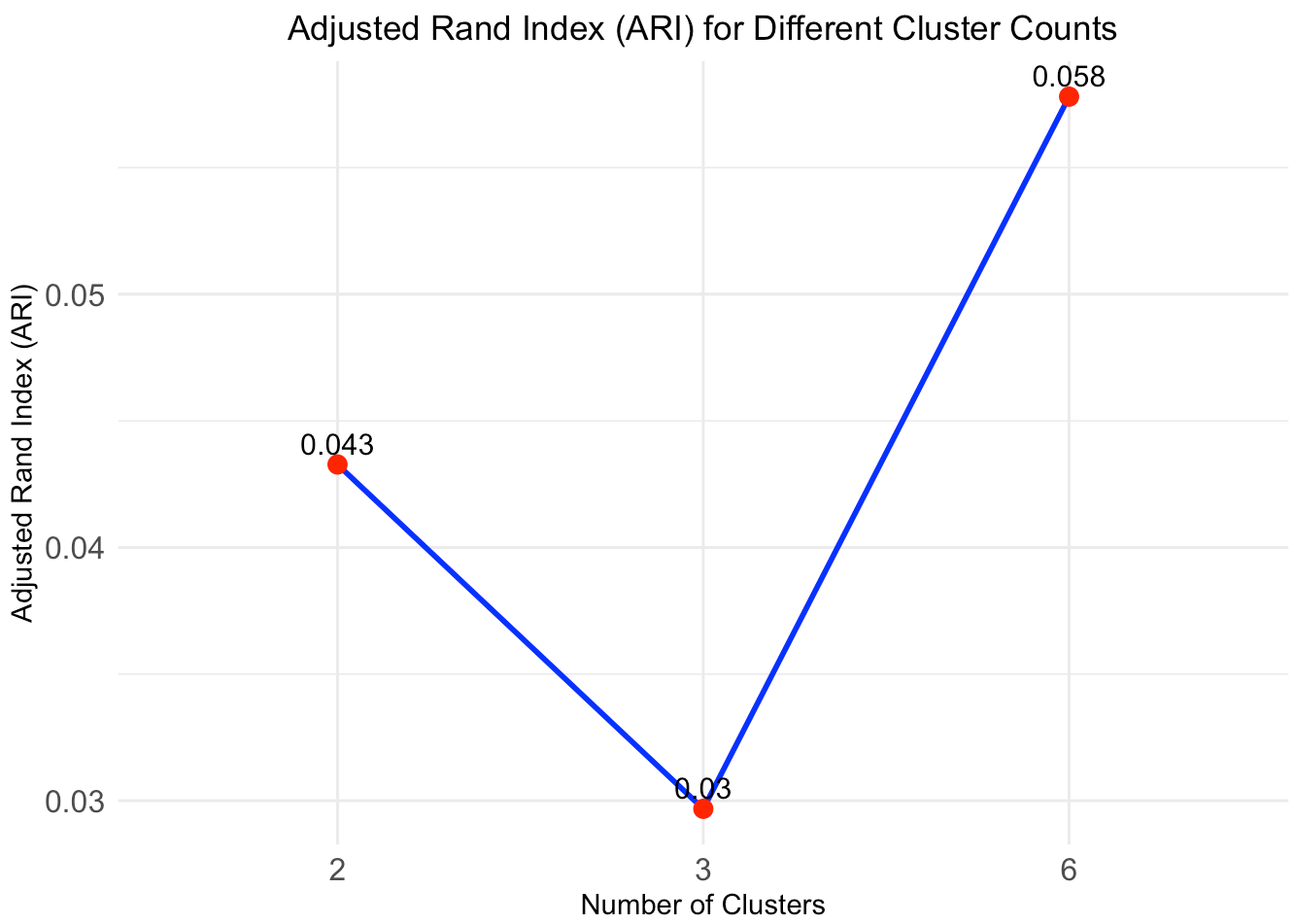
```
## [1] 0.04327633
```

```
library("fpc")

clust_stats6 <- cluster.stats(d = dist(df.scaled),
                              quality, km.res6$cluster)
clust_stats6$corrected.rand
```

```
## [1] 0.0578017
```

ARI Plot

```
ari_values <- c(clust_stats2$corrected.rand, clust_stats3$corrected.rand, clust_stats6$c
orrected.rand)

ari_results <- data.frame(
  Clusters = c(2, 3, 6),
  ARI = ari_values
)

library(ggplot2)

ggplot(ari_results, aes(x = factor(Clusters), y = ARI, group = 1)) +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "red", size = 3) +
  geom_text(aes(label = round(ARI, 3)),
            vjust = -0.5, size = 4, color = "black") +
  theme_minimal() +
  labs(title = "Adjusted Rand Index (ARI) for Different Cluster Counts",
       x = "Number of Clusters",
       y = "Adjusted Rand Index (ARI)") +
  theme(axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        plot.title = element_text(hjust = 0.5))
```



The plot above for ARI suggest 6 clusters as it similar to number of quality values in our ground truth.

2) Meila's Index (Using the formula)

```
meilas_index <- function(true_labels, cluster_labels) {
  N <- length(true_labels)
  M <- 0

  # Loop over each pair of points (i, j)
  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      # Indicator functions for Meila's index
      I_c1_diff <- ifelse(true_labels[i] != true_labels[j], 1, 0)  # c1(i) != c1(j)
      I_c2_eq <- ifelse(cluster_labels[i] == cluster_labels[j], 1, 0)  # c2(i) == c2(j)

      # Update M based on indicator functions
      M <- M + I_c1_diff * I_c2_eq
    }
  }

  # Normalize by N * (N - 1)
  mi_score <- M / (N * (N - 1))

  return(mi_score)
}
```

2 clusters

```
cluster_labels2<- km.res2$cluster
meila_index2 <- meilas_index(true_labels, cluster_labels2)

print(meila_index2)
```

```
## [1] 0.1661966
```

3 clusters

```
cluster_labels3<- km.res3$cluster
meila_index3 <- meilas_index(true_labels, cluster_labels3)

print(meila_index3)
```

```
## [1] 0.1115689
```

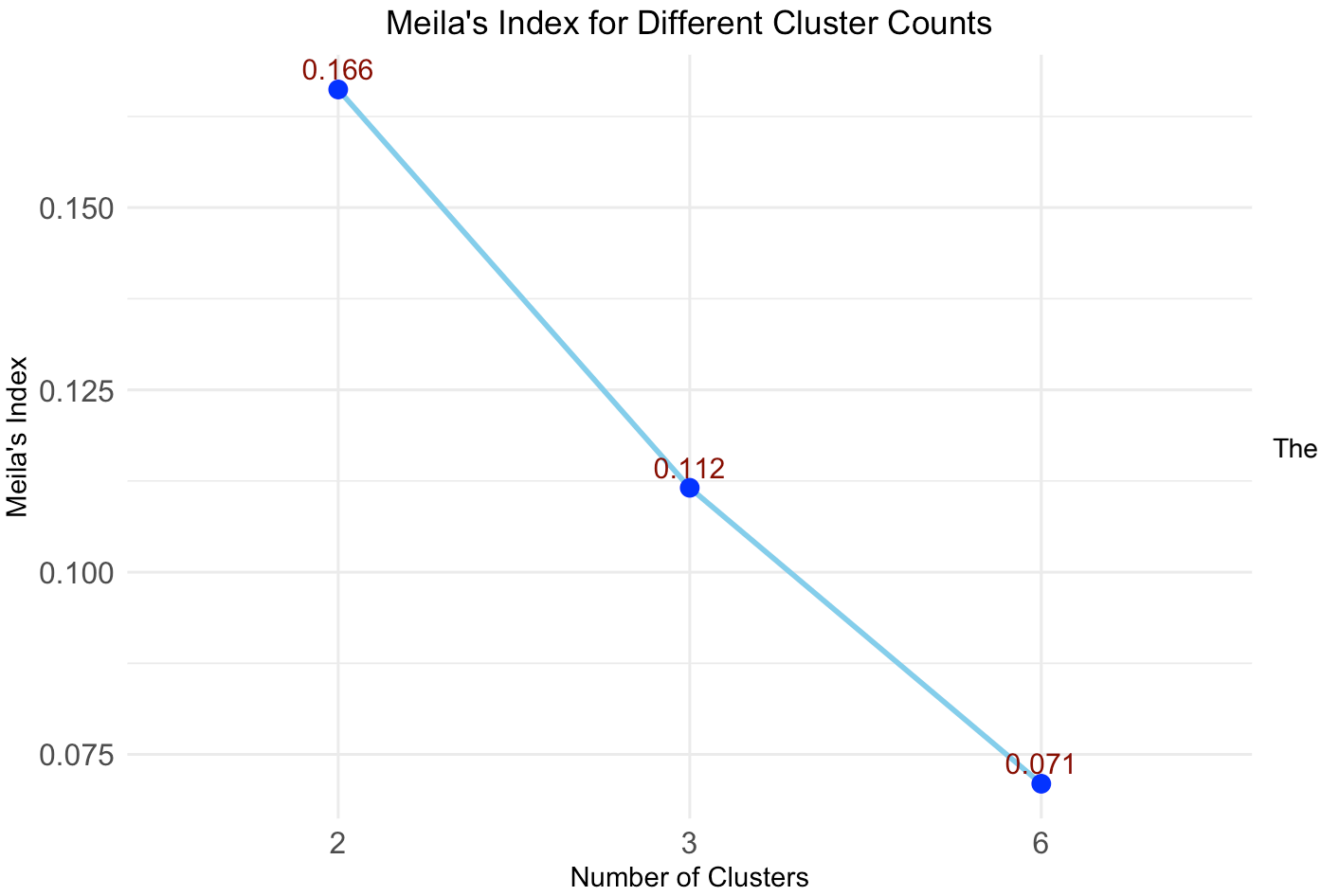
6 clusters

```
cluster_labels6<- km.res6$cluster
meila_index6 <- meilas_index(true_labels, cluster_labels6)

print(meila_index6)
```

```
## [1] 0.07098108
```

Meil's Index Plot

```
mi_results <- data.frame(
  Clusters = c(2, 3, 6),
  MeilaIndex = c(meila_index2, meila_index3, meila_index6)
)

# Plotting the Meila's Index for different cluster counts as a line plot
library(ggplot2)

ggplot(mi_results, aes(x = factor(Clusters), y = MeilaIndex, group = 1)) +
  geom_line(color = "skyblue", size = 1) +
  geom_point(color = "blue", size = 3) +
  geom_text(aes(label = round(MeilaIndex, 3)),
            vjust = -0.5, size = 4, color = "darkred") +
  theme_minimal() +
  labs(title = "Meila's Index for Different Cluster Counts",
       x = "Number of Clusters",
       y = "Meila's Index") +
  theme(axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        plot.title = element_text(hjust = 0.5))
```



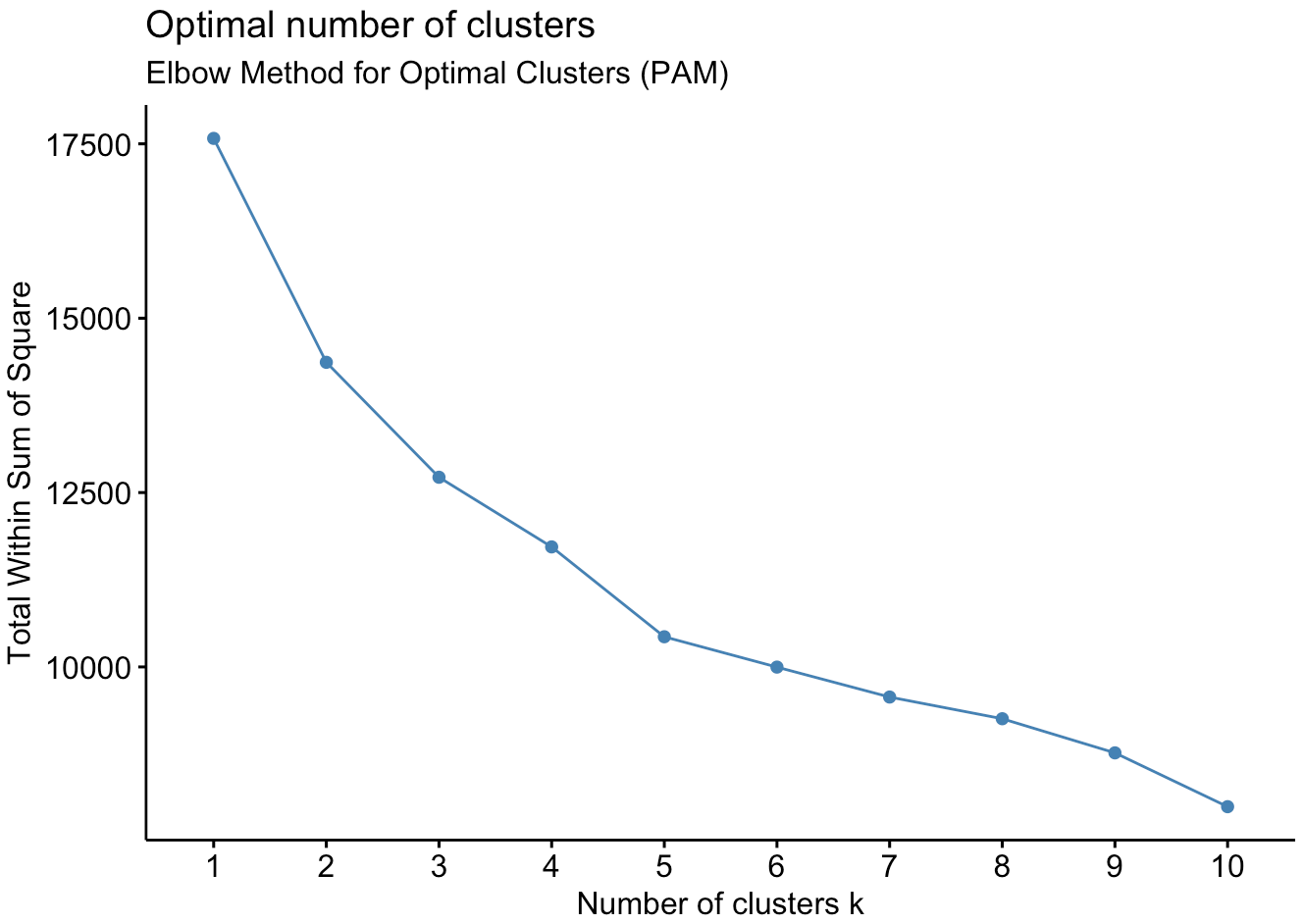
The plot above for Meila's Index suggest 6 clusters as it similar to number of quality values in our ground truth.

K medoids

Estimating the optimal number of clusters using PAM

1. Elbow method

```
library(factoextra)
library(cluster)

fviz_nbclust(df.scaled, pam, method = "wss") +
  labs(subtitle = "Elbow Method for Optimal Clusters (PAM)")
```

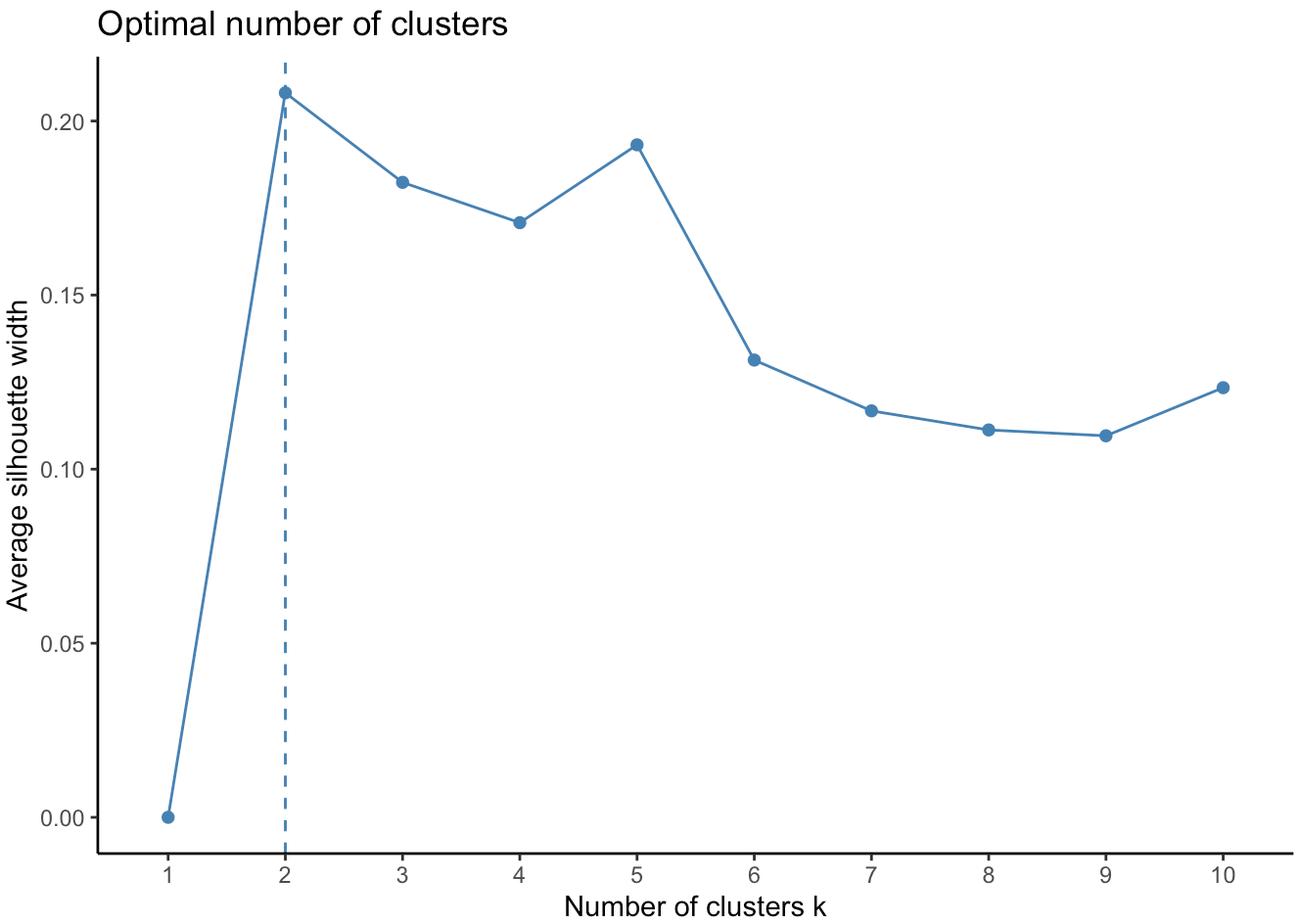


Interpretation:

From the plot we see that the elbow is created at 4 so the optimal number of clusters is 4.

2. Silhouette method

```
library(cluster)
library(factoextra)
fviz_nbclust(df.scaled, pam, method = "silhouette")+
  theme_classic()
```

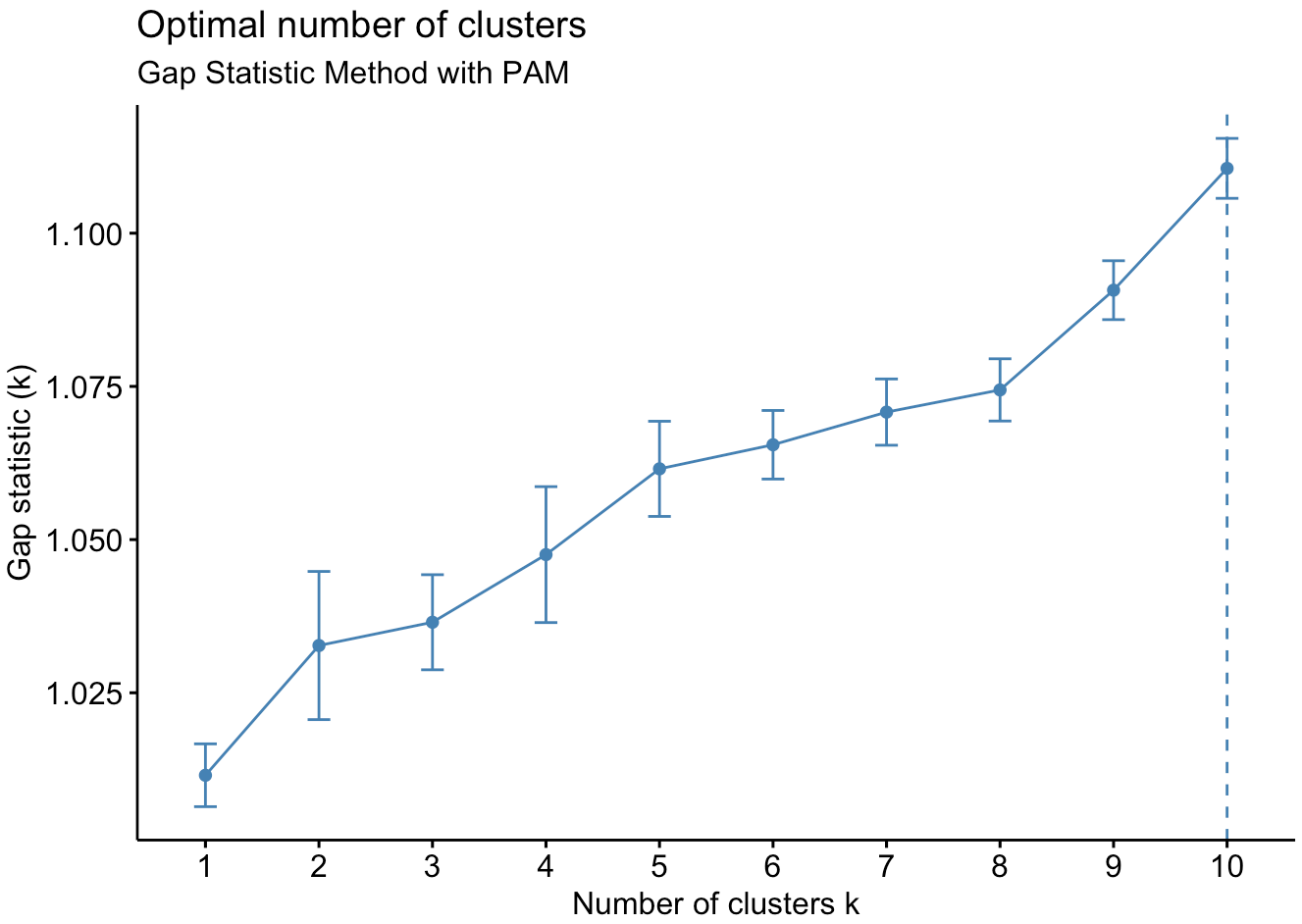


Interpretation:

For the silhouette method the optimal number of clusters suggested is 2.

3. Gap Statistic

```
set.seed(123)
fviz_nbclust(df.scaled, pam, method = "gap_stat", nboot = 50) +
  labs(subtitle = "Gap Statistic Method with PAM")
```



Interpretation:

From gap statistic we get k=10 optimal clusters.

Conclusion

From all the different methods that we saw to apply PAM on our dataset, we can conclude that we do not get accurate results for optimal number of clusters.

Each method gives a different answer. We still tried the optimal k value suggested by Silhouette method (k=2) and Elbow method (k=4). The Gap statistic method suggests k=10 optimal clusters which seems unnecessary and hence we did not apply that.

Instead we applied k=6 clusters (based on ground truth variable) to compare how it will be different from the optimal clusters suggested by above methods.

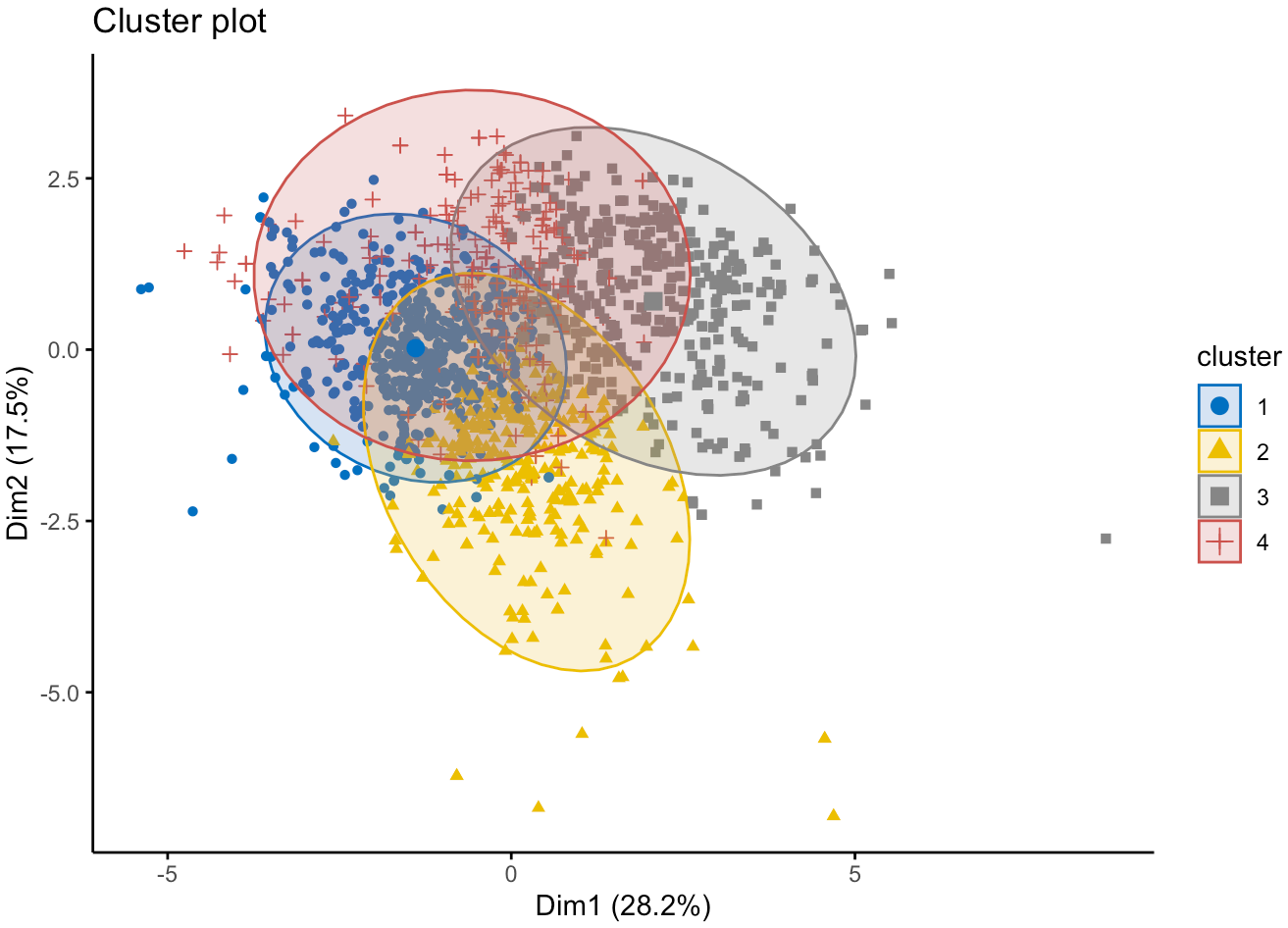
Applying PAM for k=4

```
library(factoextra)
pam.res1 = eclust(df.scaled, "pam", k=4, graph=FALSE)
pam.res1
```

```
## Medoids:
##        ID fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## [1,] 1267    -0.6430645        0.2355610  -1.1343651    -0.16937425 -0.1373951
## [2,] 1200    -0.2410190        0.2914083  -0.2103458    -0.16937425 -0.2436305
## [3,] 1160     1.0799878       -0.6579957   0.8163422    -0.24029985  0.4787701
## [4,] 1418    -0.5856294       -1.0489267   0.3029982    -0.02752304 -0.4985954
##      free.sulfur.dioxide total.sulfur.dioxide     density         pH
## [1,]          0.01195758           -0.3182152 -0.58637168  0.4461972
## [2,]          0.68116360            1.4449533  0.06004281 -0.6549356
## [3,]         -0.46604672           -0.2878157  0.28257894 -0.9787982
## [4,]          0.48996188           -0.2878157 -0.81950477  0.2518796
##        sulphates    alcohol
## [1,] -0.34304689 -0.1154048
## [2,] -0.46103614 -0.8661079
## [3,]  0.06991545  0.3537847
## [4,]  0.65986166  1.5736773
## Clustering vector:
##    [1] 1 2 1 3 1 1 1 1 1 2 1 2 1 3 2 2 2 3 1 3 2 2 3 2 1 1 1 3 1 1 1 1 2 2 1 1 1
##   [38] 3 1 2 2 2 3 1 1 4 2 3 1 2 3 1 1 2 2 1 3 2 1 1 2 2 1 1 1 1 1 1 3 1 1 2 2 1
##   [75] 3 3 3 1 1 2 1 3 2 3 4 1 3 1 2 1 2 3 3 1 1 4 1 1 1 1 1 1 1 2 1 2 3 1 2 2 1
##  [112] 2 2 3 1 3 1 1 1 2 2 1 1 1 2 2 1 1 1 1 2 4 4 1 1 2 1 1 2 2 2 1 4 1 4 2 1 3
##  [149] 1 1 3 3 2 2 2 2 2 2 1 1 1 1 1 2 2 2 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1
##  [186] 2 2 1 2 2 2 1 2 1 1 2 1 3 4 1 3 2 1 1 1 3 3 2 2 3 4 1 3 2 1 2 1 1 1 2 2 2
##  [223] 1 1 2 2 3 1 2 1 4 1 2 1 1 1 1 1 1 1 3 3 2 3 3 1 1 2 1 1 3 1 3 2 1 2 3 1 3
##  [260] 3 1 1 1 2 3 3 1 4 1 3 1 3 3 1 2 1 1 3 3 3 3 3 1 3 2 2 3 1 1 3 1 3 3 1 3 3
##  [297] 2 1 1 1 1 3 1 1 2 3 1 3 3 1 3 2 2 2 4 4 2 2 3 2 3 2 1 2 3 3 3 3 3 3 3 3 2
##  [334] 1 1 3 4 2 3 3 3 3 3 3 3 1 1 3 3 1 3 1 1 3 4 1 3 3 3 3 2 3 3 3 3 3 3 3 3 4
##  [371] 1 3 4 2 3 3 3 4 3 3 3 3 3 3 2 1 2 1 2 3 1 3 3 2 3 3 2 3 3 2 2 4 3 3 1 3 3
##  [408] 3 3 3 2 2 2 3 2 2 3 2 3 1 4 1 1 3 1 1 1 3 3 3 3 2 3 3 3 3 2 3 3 2 3 3 3 3
##  [445] 4 1 3 3 3 3 3 3 1 3 4 3 3 2 3 3 3 1 3 2 3 3 3 4 3 2 3 3 3 3 3 3 3 3 3 3 3
##  [482] 3 3 3 3 3 3 3 3 3 2 4 4 2 4 3 1 2 3 2 1 3 3 3 3 3 3 3 2 3 3 2 3 3 3 2 3 3
##  [519] 3 4 3 2 2 2 2 2 4 4 2 3 3 3 3 4 3 3 3 1 3 3 2 3 3 3 3 2 1 3 3 3 1 3 3 1 3
##  [556] 3 3 3 3 3 3 2 2 2 3 3 3 3 3 4 3 4 3 3 3 3 3 2 2 3 3 3 3 3 3 1 3 2 4 3 2 2
##  [593] 2 3 1 2 3 3 1 3 1 3 1 3 2 1 4 2 3 4 3 3 1 3 3 2 2 3 3 3 2 2 3 4 1 1 1 1 3
##  [630] 2 3 3 1 2 2 1 2 2 1 4 3 2 3 2 3 1 1 1 4 2 3 2 3 3 3 2 3 3 1 1 1 2 1 3 3 2
##  [667] 3 3 3 3 1 1 2 1 3 3 3 1 2 3 3 1 2 1 2 1 1 3 1 3 1 2 3 2 2 4 1 1 2 3 2 1 1
##  [704] 2 1 1 1 1 1 3 2 2 1 2 2 1 2 1 1 1 1 2 1 2 1 1 1 1 1 1 3 1 1 1 1 1 1 1 2 1
##  [741] 1 2 1 3 3 1 2 2 1 1 1 1 2 1 3 1 1 1 1 2 2 3 1 3 1 1 2 2 1 1 1 2 2 3 3 1 1
##  [778] 1 3 1 1 1 2 1 1 3 3 2 2 2 2 2 1 1 3 3 2 4 3 3 2 1 4 1 1 4 4 4 1 3 1 3 3 1
##  [815] 3 3 3 3 1 2 1 4 1 1 1 1 4 1 4 1 1 1 3 3 1 1 4 4 3 1 3 1 3 2 3 1 1 1 1 1 3
##  [852] 3 2 4 4 1 4 4 3 4 1 1 1 1 1 1 4 4 4 1 1 1 1 3 3 3 1 1 2 1 1 1 3 1 2 1 1 3
##  [889] 1 2 1 1 3 1 1 1 4 1 4 1 4 1 1 1 1 2 1 1 1 4 3 3 3 3 4 4 1 1 4 4 3 4 4 1 4
##  [926] 4 4 2 4 4 1 1 2 1 1 4 4 3 4 1 4 3 3 3 4 3 3 4 4 4 4 4 4 3 4 3 3 3 1 1 4 1
##  [963] 1 3 4 4 3 2 4 1 3 3 3 4 4 2 2 2 4 3 3 1 4 3 3 1 3 2 1 3 1 2 1 2 2 1 1 1 1
## [1000] 1 4 3 4 4 2 4 4 4 4 3 4 3 1 1 1 3 4 4 4 1 3 3 1 4 1 1 4 1 2 1 1 1 1 1 2 3
## [1037] 4 1 4 4 1 1 4 3 4 1 1 1 3 3 1 3 1 4 2 2 3 2 3 3 3 4 3 3 1 1 4 3 3 2 4 2 2
## [1074] 1 2 4 3 3 3 2 3 2 2 4 2 2 4 4 3 3 3 4 1 4 1 3 1 3 3 3 4 4 1 4 4 4 4 4 1 3
## [1111] 1 4 4 3 4 1 1 1 4 1 4 1 1 3 1 4 4 1 2 3 1 2 4 1 4 4 3 3 2 2 2 4 1 4 2 4 1
## [1148] 3 4 3 4 4 3 3 1 3 4 4 4 3 3 3 4 1 1 3 3 4 4 1 3 1 3 2 2 1 1 1 1 4 4 3 4 1
## [1185] 2 4 1 4 2 1 3 1 4 1 2 1 2 1 4 2 1 4 4 2 4 4 4 3 4 4 1 1 1 3 3 4 2 4 3 4 3
## [1222] 3 2 3 3 2 2 3 4 2 4 2 2 3 1 4 1 1 1 1 2 2 4 2 2 1 1 1 4 1 1 2 1 1 1 1 2 1
## [1259] 1 1 3 1 2 1 4 1 1 3 1 4 4 4 1 2 1 2 4 1 2 3 1 1 1 2 1 3 4 1 2 2 1 1 4 1 1
```

```
## [1296] 2 2 4 1 1 1 1 3 4 2 2 2 1 2 2 2 4 1 1 1 2 1 3 2 3 2 1 3 4 1 1 1 1 1 2 2 2
## [1333] 1 2 1 4 1 1 1 1 1 1 1 1 3 1 1 1 1 1 2 1 1 1 1 1 1 1 2 3 3 1 3 2 1 1 1 2 2
## [1370] 1 3 3 3 2 1 2 1 4 1 1 1 1 2 2 2 2 1 1 1 2 1 1 1 2 2 1 1 2 1 1 2 2 4 4 3 4
## [1407] 4 1 4 1 1 1 4 3 3 1 3 4 1 2 1 2 4 1 3 3 4 4 1 4 1 2 1 1 2 2 2 1 1 2 4 2 1
## [1444] 1 2 2 1 1 1 4 4 3 4 2 3 1 2 2 3 4 1 1 1 1 1 1 2 1 2 1 3 1 4 1 2 4 2 4 1 3
## [1481] 1 3 3 4 1 1 1 1 1 1 4 1 1 2 1 1 2 1 1 1 1 2 1 4 3 1 1 3 4 4 1 1 1 1 1 1 4
## [1518] 1 3 1 1 1 4 1 1 1 1 1 3 1 1 1 1 2 1 1 1 1 1 2 1 4 1 3 4 1 1 1 3 4 1 1 1 1
## [1555] 1 1 1 1 2 2 2 2 1 1 1 1 4 1 1 1 4 1 2 4 4 4 4 1 1 1 4 1 1 2 4 4 4 1 4 2 4
## [1592] 1 1 1 1 1 1 1 4
## Objective function:
##    build     swap
## 2.553011 2.491614
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
## [11] "nbclust"
```

```
fviz_cluster(list(data=df.scaled, cluster=pam.res1$clustering),
             ellipse.type = "norm", geom = "point", stand = FALSE,
             palette= "jco", ggtheme = theme_classic())
```



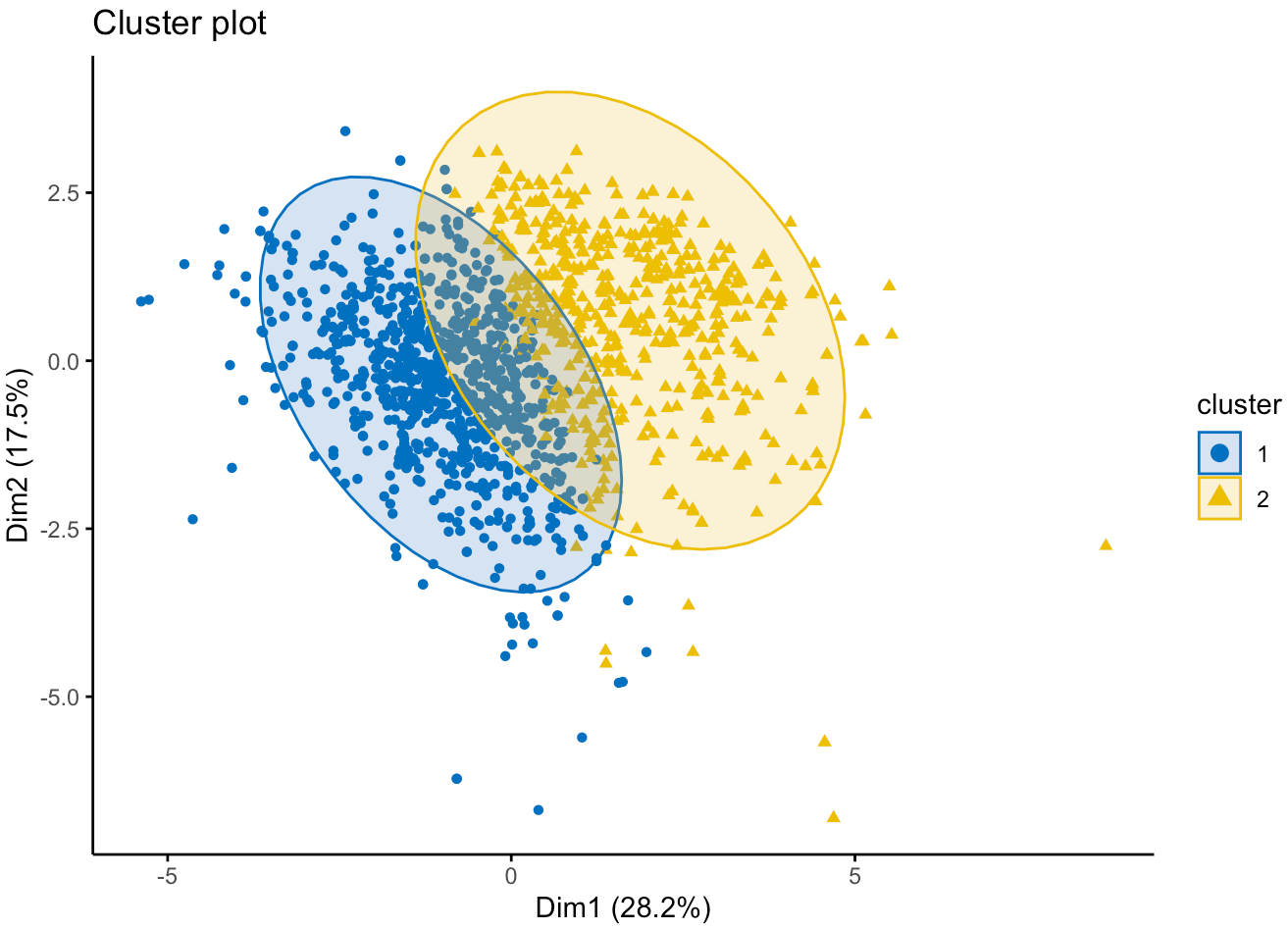
PAM at k=2

```
pam.res2 = eclust(df.scaled, "pam", k=2, graph=FALSE)
pam.res2
```

```
## Medoids:
##        ID fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## [1,]   86    -0.8153698        0.1238664  -0.6210211     -0.2402999 -0.2436305
## [2,] 1160     1.0799878       -0.6579957   0.8163422     -0.2402999  0.4787701
##      free.sulfur.dioxide total.sulfur.dioxide    density         pH   sulphates
## [1,]           0.2987602           -0.1966174 -0.3426416  0.6405147 -0.40204152
## [2,]          -0.4660467           -0.2878157  0.2825789 -0.9787982  0.06991545
##         alcohol
## [1,] -0.3030805
## [2,]  0.3537847
## Clustering vector:
##    [1] 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1
##   [38] 2 1 1 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
##   [75] 2 2 2 1 1 1 1 2 1 2 1 1 2 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1
##  [112] 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2
##  [149] 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
##  [186] 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 1 1 2 2 1 1 2 2 1 2 1 1 1 2 1 1 1 1 1
##  [223] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2 1 1 1 1 1 2 1 2 1 1 1 2 1 2
##  [260] 2 1 1 1 1 2 2 1 2 1 2 1 2 2 1 1 1 1 2 2 2 2 2 1 2 1 1 2 1 1 2 1 2 2 1 2 2
##  [297] 2 1 1 1 1 2 1 1 2 2 1 2 2 1 2 1 1 1 1 1 1 1 2 1 2 2 1 2 2 2 2 2 2 2 2 2 1
##  [334] 1 1 2 2 1 2 2 2 2 2 2 2 1 1 2 2 1 2 1 1 2 1 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2
##  [371] 1 2 2 1 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 1 2 2 2 2 2 1 2 2 1 1 1 2 2 1 2 2
##  [408] 2 2 2 1 1 1 2 1 1 2 1 2 1 2 1 1 2 1 1 1 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2 2 2
##  [445] 1 1 2 2 2 2 2 2 1 2 1 2 2 1 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
##  [482] 2 2 2 2 2 2 2 2 2 1 2 2 1 1 2 1 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [519] 2 1 2 1 1 2 2 2 1 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 1 2
##  [556] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 2 1
##  [593] 2 2 1 1 2 2 1 2 1 2 1 2 1 1 2 2 2 1 2 2 1 2 2 1 1 2 2 2 1 1 2 1 1 1 1 1 2
##  [630] 1 2 2 1 1 1 1 1 1 1 2 2 1 2 1 2 1 1 1 2 1 2 1 2 2 2 2 2 2 1 1 1 1 1 2 2 1
##  [667] 2 2 2 2 1 1 1 1 2 2 2 1 1 2 2 1 1 1 1 1 1 2 1 2 1 1 2 1 1 1 1 1 1 2 2 1 1
##  [704] 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1
##  [741] 1 1 1 2 2 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 1 2 2 1 1
##  [778] 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 2 2 2 1 2 1 2 2 1
##  [815] 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 2 1 1 1 1 1 2
##  [852] 2 2 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 2 1 2 1 1 1 1 2
##  [889] 1 2 1 1 2 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 1 1 1 2 1 1 1 2
##  [926] 1 1 1 2 2 1 1 1 1 1 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 1 2 1
##  [963] 1 2 2 2 2 1 2 1 2 2 2 1 2 1 1 1 1 2 2 1 1 2 2 1 2 1 1 2 1 1 1 1 2 1 1 1 1
## [1000] 1 1 2 2 1 1 1 2 2 2 2 2 2 1 1 1 2 2 2 2 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2
## [1037] 1 1 2 1 1 1 1 2 1 1 1 1 2 2 1 2 1 2 2 2 2 1 2 2 2 2 2 2 2 1 1 1 2 2 1 2 1 1
## [1074] 1 1 1 2 2 2 2 2 2 1 2 1 1 2 2 2 2 2 2 1 2 1 2 1 2 2 2 2 1 1 1 1 1 2 2 1 2
## [1111] 1 1 2 2 1 1 1 1 1 1 2 1 1 2 1 2 1 1 2 2 1 1 1 1 2 1 2 2 1 1 1 2 1 1 1 1 1
## [1148] 2 2 2 2 1 2 2 1 2 2 1 1 2 2 2 2 1 1 2 2 2 1 1 2 1 2 1 1 1 1 1 1 2 2 2 2 1
## [1185] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 2 2 1 1 1 1 2 1 1 1 1 1 2 2 2 1 1 2 2 2
## [1222] 2 1 2 2 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1259] 1 1 2 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [1296] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 2 2 1 1 1 1 1 1 1 1
## [1333] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 2 1
## [1370] 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2
## [1407] 2 1 2 1 1 1 2 2 2 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1
## [1444] 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 2 1 1 2
## [1481] 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 2 1 1 1 1 1 1
```

```
## [1518] 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1 1
## [1555] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [1592] 1 1 1 1 1 1 1 1
## Objective function:
##    build     swap
## 2.932373 2.766446
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
## [11] "nbclust"
```

```
fviz_cluster(list(data=df.scaled, cluster=pam.res2$clustering),
             ellipse.type = "norm", geom = "point", stand = FALSE,
             palette= "jco", ggtheme = theme_classic())
```



PAM at k=6

```
pam.res3 = eclust(df.scaled, "pam", k=6, graph=FALSE)
pam.res3
```

```
## Medoids:
##        ID fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## [1,]  274    0.04615639        1.0453468  -0.3643491    -0.09844864 -0.2436305
## [2,] 1200   -0.24101899        0.2914083  -0.2103458    -0.16937425 -0.2436305
## [3,]  407    0.85024746       -0.4904538   1.0216798    -0.02752304 -0.5198424
## [4,] 1267   -0.64306452        0.2355610  -1.1343651    -0.16937425 -0.1373951
## [5,]  693    0.16102655       -0.2112173   1.2270174    -0.38215106  7.1078575
## [6,] 1418   -0.58562945       -1.0489267   0.3029982    -0.02752304 -0.4985954
##      free.sulfur.dioxide total.sulfur.dioxide     density           pH
## [1,]         -0.56164758           -0.2574163  0.31966829 -0.007210449
## [2,]          0.68116360            1.4449533  0.06004281 -0.654935624
## [3,]         -0.65724844           -0.5614109  0.71705425 -0.460618072
## [4,]          0.01195758           -0.3182152 -0.58637168  0.446197173
## [5,]          0.01195758            0.4721707  0.61108466 -1.820840939
## [6,]          0.48996188           -0.2878157 -0.81950477  0.251879621
##        sulphates    alcohol
## [1,] -0.10706841 -0.9599458
## [2,] -0.46103614 -0.8661079
## [3,] -0.04807379  0.3537847
## [4,] -0.34304689 -0.1154048
## [5,]  3.01964650 -1.3352974
## [6,]  0.65986166  1.5736773
## Clustering vector:
##    [1] 1 2 1 3 1 4 4 4 4 2 4 2 4 1 2 2 2 5 1 5 2 2 1 2 4 4 1 1 4 4 4 4 2 2 4 1 4
##   [38] 3 4 2 2 1 5 1 4 6 2 3 4 2 1 4 4 2 2 4 3 2 4 4 1 2 4 4 4 4 4 4 3 1 4 2 2 1
##   [75] 2 3 3 4 4 2 4 5 2 5 6 4 2 1 2 1 2 2 5 1 4 6 4 1 1 1 1 4 1 1 4 1 5 1 2 2 1
##  [112] 2 2 3 1 3 1 1 4 2 1 4 1 1 2 2 1 1 1 4 2 6 6 4 1 1 1 1 2 2 1 1 6 4 6 2 4 2
##  [149] 4 3 3 5 2 2 2 2 2 2 4 4 1 1 4 2 2 2 2 4 4 5 1 1 1 4 4 4 4 4 1 1 1 5 4 4 1
##  [186] 2 2 1 2 2 2 4 2 1 1 2 4 3 6 4 3 2 4 4 4 3 3 2 2 3 3 1 3 2 1 2 4 1 4 2 2 2
##  [223] 4 1 1 2 5 1 2 4 6 4 2 4 1 4 4 4 4 1 5 3 2 3 3 1 4 2 4 1 3 4 3 2 4 2 3 4 5
##  [260] 3 4 1 4 2 3 3 1 6 4 3 4 3 3 1 2 4 4 3 3 3 3 5 1 3 2 2 3 4 4 2 4 5 3 4 3 3
##  [297] 2 1 1 1 4 3 1 4 2 3 1 3 3 3 3 2 2 2 6 6 2 2 3 2 3 2 1 2 3 3 3 3 3 3 3 3 2
##  [334] 4 4 3 6 2 3 3 3 3 3 3 3 4 4 3 3 1 1 1 1 3 6 4 3 3 3 3 2 1 3 3 3 3 3 3 3 6
##  [371] 4 3 6 2 3 3 3 6 3 3 3 3 3 3 1 1 1 1 1 3 4 3 3 2 3 3 2 3 3 1 2 6 3 3 1 3 3
##  [408] 3 3 3 2 2 2 3 2 2 3 2 3 4 6 4 1 3 1 4 4 1 1 3 3 2 3 3 3 3 2 3 3 1 3 3 3 3
##  [445] 6 1 3 3 1 3 3 5 4 3 6 3 3 2 3 3 3 1 3 2 3 3 3 6 3 2 3 3 3 3 3 1 3 3 1 1 3
##  [482] 3 3 3 3 3 3 3 3 3 1 6 6 1 6 3 1 2 3 1 1 3 3 3 3 3 3 3 2 3 3 2 3 3 3 2 3 1
##  [519] 3 6 3 1 2 2 2 2 6 6 2 1 3 3 3 6 3 3 1 1 3 3 1 3 1 3 3 2 1 3 3 3 4 3 3 4 3
##  [556] 3 3 3 3 3 3 2 2 2 3 3 1 1 3 6 3 6 3 3 3 3 3 2 2 3 3 3 3 3 2 1 3 2 6 3 2 2
##  [593] 2 3 1 2 3 3 1 3 1 3 1 3 1 1 3 3 3 6 3 3 4 3 2 2 2 3 3 3 2 2 1 6 1 1 1 1 1
##  [630] 2 1 3 4 1 2 4 2 2 1 6 3 1 3 1 3 1 4 1 3 2 3 2 3 3 3 1 3 3 1 4 1 1 4 3 3 2
##  [667] 1 3 3 3 4 1 2 1 3 3 3 1 2 3 3 4 2 1 2 1 1 1 1 3 1 1 5 2 2 6 4 4 1 3 2 4 4
##  [704] 2 1 1 4 4 4 3 2 2 1 1 2 4 1 4 1 4 1 2 4 2 1 1 1 4 4 4 5 1 1 4 1 1 1 1 2 1
##  [741] 4 2 4 3 2 4 2 2 4 4 1 1 2 1 5 4 4 1 1 2 2 1 1 1 1 1 2 2 4 1 4 2 2 1 1 1 4
##  [778] 4 3 4 4 4 2 4 4 3 3 2 2 2 2 2 4 4 3 1 2 6 3 3 2 1 6 1 1 6 6 6 1 3 4 3 3 4
##  [815] 3 3 3 6 4 1 4 6 4 4 4 4 6 4 6 4 4 4 3 3 1 4 6 6 3 4 3 4 2 2 3 4 4 1 4 4 3
##  [852] 3 2 6 6 4 6 6 3 6 4 4 3 4 4 4 6 6 6 4 4 4 4 3 3 6 4 4 2 4 1 4 6 4 2 1 1 6
##  [889] 4 2 4 4 3 4 4 4 6 4 6 1 6 4 4 4 4 2 4 4 4 6 3 3 3 3 6 6 4 4 6 6 3 6 6 4 6
##  [926] 6 6 1 6 6 4 4 2 4 4 6 6 3 6 4 6 3 3 3 6 3 3 6 6 6 6 6 3 6 3 3 3 4 4 6 1
##  [963] 4 6 6 6 3 2 6 4 3 3 3 3 3 2 2 2 6 3 3 1 6 3 3 4 3 2 4 3 4 2 4 2 2 4 4 4 1
## [1000] 4 6 6 6 6 2 6 6 6 6 3 6 3 1 4 4 3 6 6 6 4 3 3 4 6 4 1 6 4 2 4 4 4 1 4 1 3
## [1037] 6 1 6 6 1 4 6 3 6 4 4 4 3 3 4 5 4 6 1 1 3 2 3 3 3 6 3 3 4 1 6 3 3 2 6 2 2
```

```
## [1074] 1 2 6 3 3 3 2 3 2 2 6 2 2 3 6 3 3 6 6 4 6 4 3 4 1 6 1 6 6 4 6 6 6 6 6 1 3
## [1111] 4 6 6 3 6 4 4 4 6 4 6 4 4 3 4 6 6 4 2 3 1 2 6 4 6 6 3 3 2 2 2 6 4 6 2 6 4
## [1148] 3 6 3 6 6 1 3 4 1 6 6 6 3 3 3 6 1 1 5 3 6 6 4 3 4 3 2 2 4 4 4 4 6 6 3 6 4
## [1185] 2 6 4 6 2 1 3 4 6 4 1 4 2 4 6 2 4 6 6 2 6 6 6 1 6 6 4 4 4 3 3 6 2 6 3 6 3
## [1222] 3 2 3 3 2 2 1 6 2 6 2 2 3 4 6 1 4 1 4 2 2 6 1 2 4 4 4 6 4 4 2 4 4 4 4 2 4
## [1259] 4 4 5 4 2 1 6 4 4 3 4 6 6 6 4 2 4 2 3 4 2 3 4 4 4 2 4 3 6 4 2 2 4 4 6 1 4
## [1296] 2 2 6 4 1 4 4 3 6 2 2 2 4 2 2 2 6 1 4 4 2 4 3 2 5 2 4 3 6 4 4 4 4 4 2 2 2
## [1333] 4 1 4 6 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 2 4 4 4 4 4 4 4 2 3 3 1 3 1 4 1 1 2 2
## [1370] 4 5 6 5 2 4 2 1 6 4 4 4 1 2 2 2 2 4 4 1 2 4 4 4 2 2 1 1 2 4 4 2 2 6 6 3 6
## [1407] 3 4 6 4 4 4 3 2 3 4 3 6 4 2 4 2 6 4 3 3 6 3 4 6 4 2 4 4 2 2 2 4 4 2 6 2 4
## [1444] 4 2 2 4 4 4 6 6 3 6 2 3 4 2 2 3 6 4 4 4 4 4 4 2 4 2 1 1 4 6 4 2 6 2 6 1 3
## [1481] 4 3 3 3 4 4 4 4 4 4 6 4 4 2 4 4 2 4 4 4 4 2 4 6 3 4 4 3 3 6 4 4 4 4 1 1 6
## [1518] 4 3 4 4 4 6 4 4 4 4 4 3 4 4 4 4 2 4 4 4 4 4 2 4 6 4 3 6 4 4 4 3 6 4 4 4 1
## [1555] 4 4 4 4 2 2 2 2 4 4 4 4 6 4 4 4 6 4 2 6 6 6 6 4 4 4 6 4 4 2 6 6 6 4 6 2 6
## [1592] 4 4 4 4 4 4 4 6
## Objective function:
##    build     swap
## 2.382334 2.343865
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
## [11] "nbclust"
```

```
fviz_cluster(list(data=df.scaled, cluster=pam.res3$clustering),
             ellipse.type = "norm", geom = "point", stand = FALSE,
             palette= "jco", ggtheme = theme_classic())
```

Interpretation:

After applying pam on k=4, 2 and 6 we see that neither of the one is able to form separate clusters. The points are overlapping from one cluster to another.

The reason for this could be due to the ground truth variable (Quality). We see that quality variable has 6 groups and so we have created clusters with 6 groups as well. But wine quality can be determined by various factors and quality variable can be a factor not a ground truth due to which we are not able to form seperate clusters.

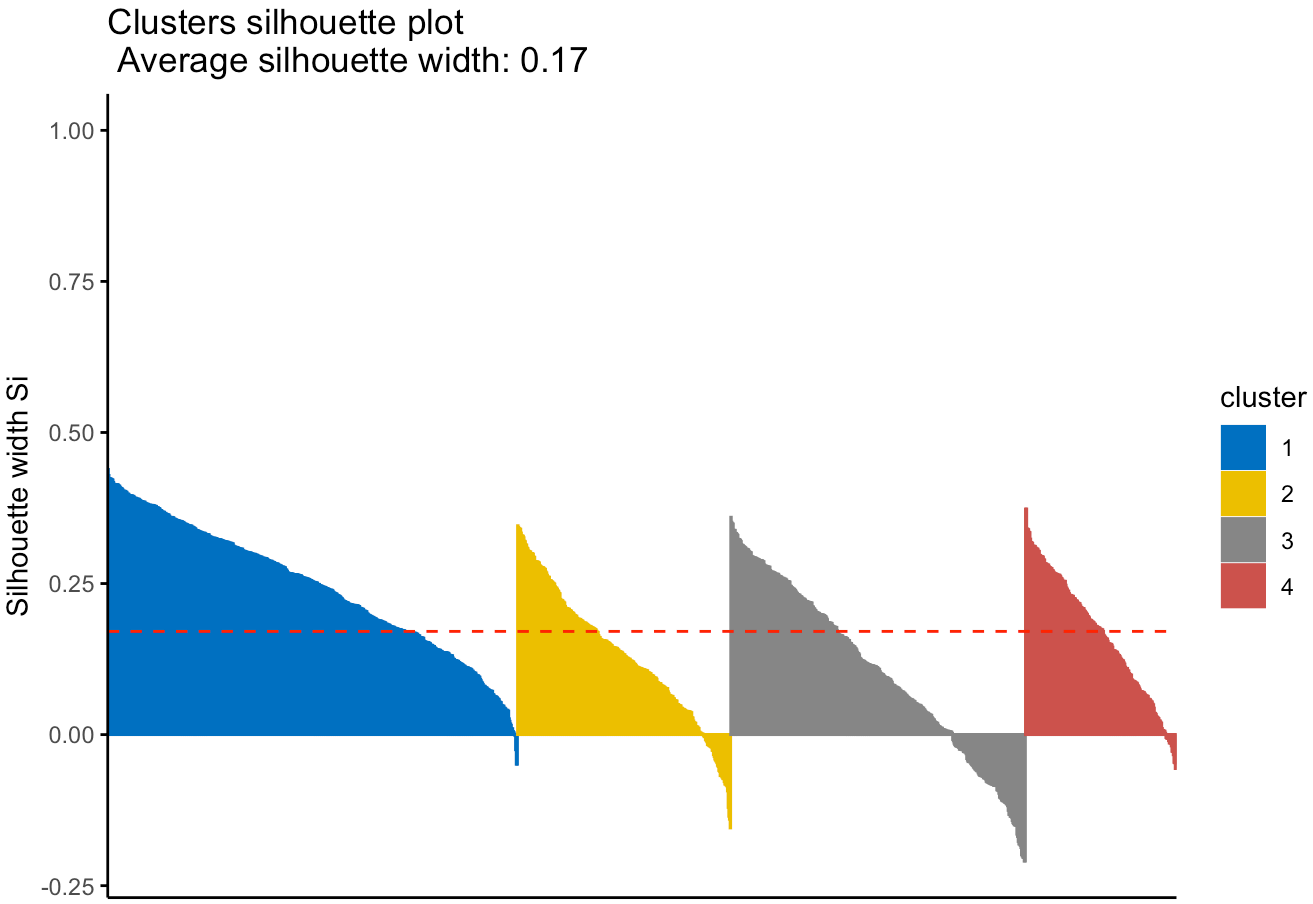
We will still proceed with the further classifications to see how it will interpret our dataset.

Internal Classification

Silhouette plot

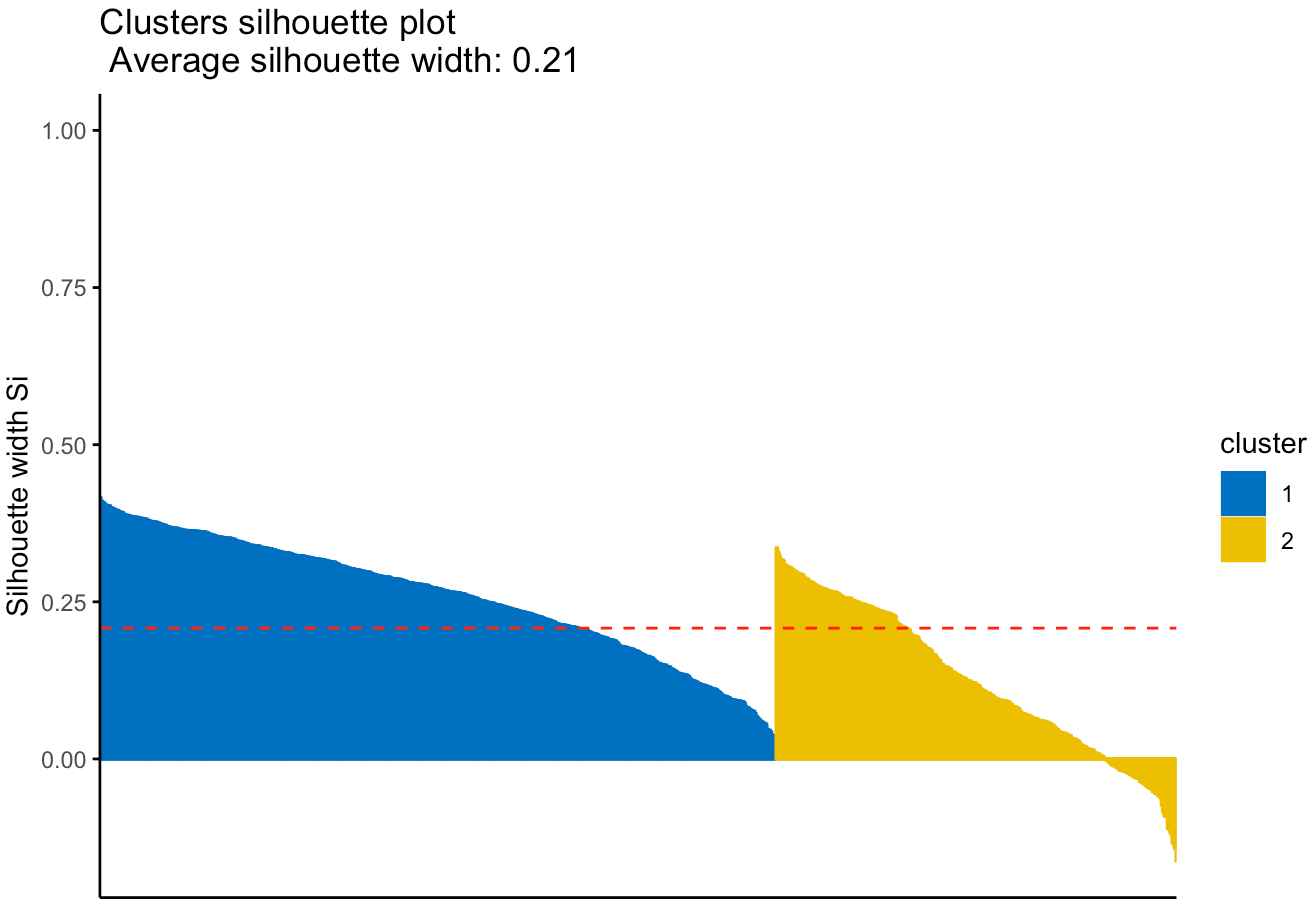
```
fviz_silhouette(pam.res1, palette="jco", ggtheme= theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  613          0.24
## 2       2  319          0.13
## 3       3  441          0.10
## 4       4  226          0.16
```



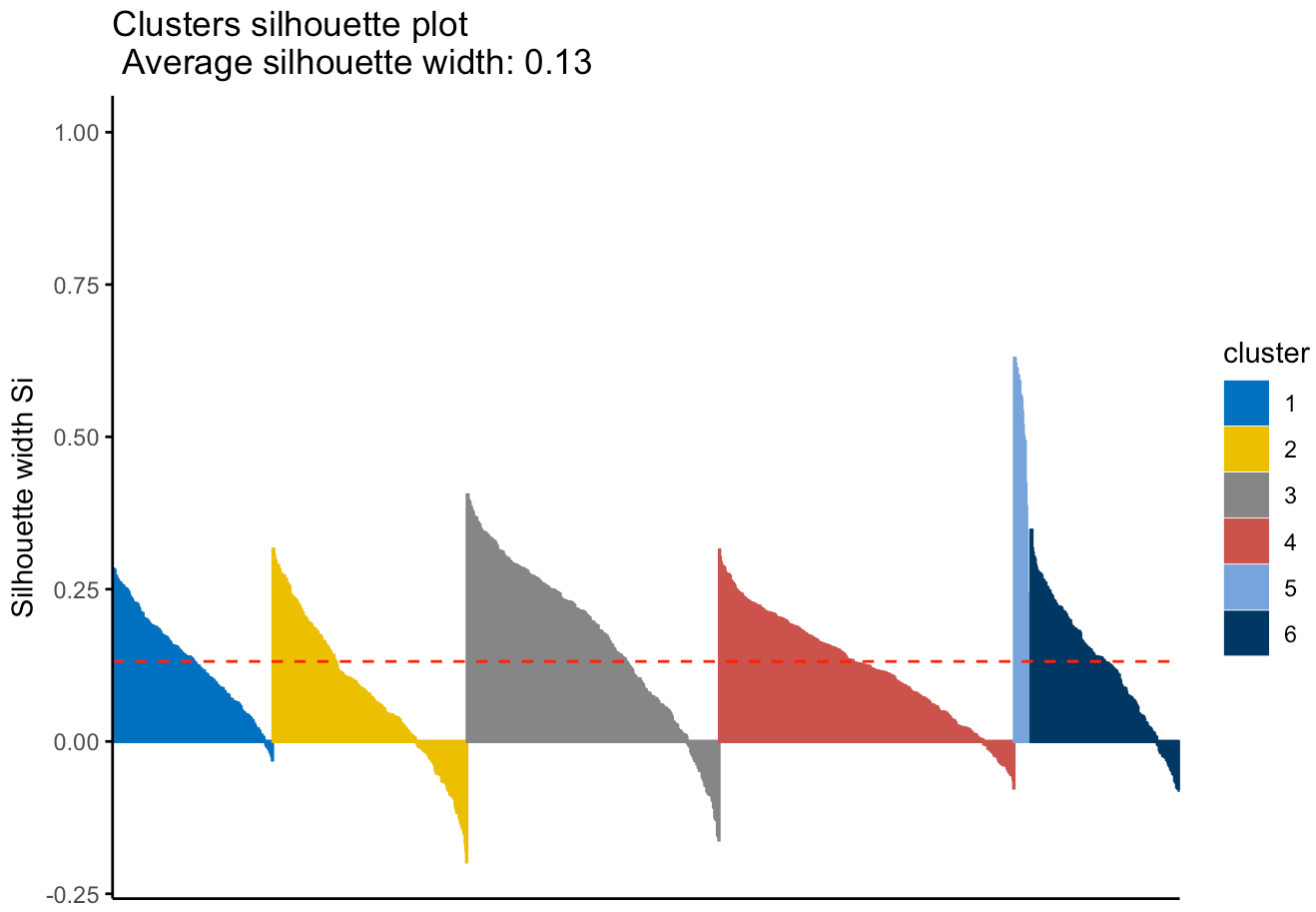

```
fviz_silhouette(pam.res2, palette="jco", ggtheme= theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1 1004          0.26
## 2       2  595          0.12
```




```
fviz_silhouette(pam.res3, palette="jco", ggtheme= theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1  240          0.13
## 2       2  291          0.08
## 3       3  378          0.17
## 4       4  442          0.12
## 5       5   25          0.47
## 6       6  223          0.12
```



interpretation: Average Silhouette width for each cluster is seen in the silhouette plot. k=2 clusters has highest silhouette width of 0.21, where as plot with k=4 clusters has average width of 0.17 and plot with k=6 clusters has lowest average so=ilhouette width of 0.13.

So this suggests we choose k=2 clusters which is having maximum average silhouette width.

Dunn Index using PAM

```
library(fpc)
stats1 = cluster.stats(dist(df.scaled), pam.res1$cluster)
stats1$dunn
```

```
## [1] 0.02178405
```

```
stats2 = cluster.stats(dist(df.scaled), pam.res2$cluster)
stats2$dunn
```

```
## [1] 0.02435284
```

```
stats3 = cluster.stats(dist(df.scaled), pam.res3$cluster)
stats3$dunn
```

```
## [1] 0.01936127
```

Interpretation:

The Dunn Index value for each cluster result is below.

for k=4, Dunn Index = 0.02178405 for k=2, Dunn Index = 0.02435284 for k=6, Dunn Index = 0.01936127

Computing ClValid

Internal Validation Measures

```
library(clValid)
clmethods <- c("pam")
intern_pam = clValid(df.scaled, nClust =2:6, clMethods= clmethods, validation="interna
l",maxitems = 1599)
summary(intern_pam)
```

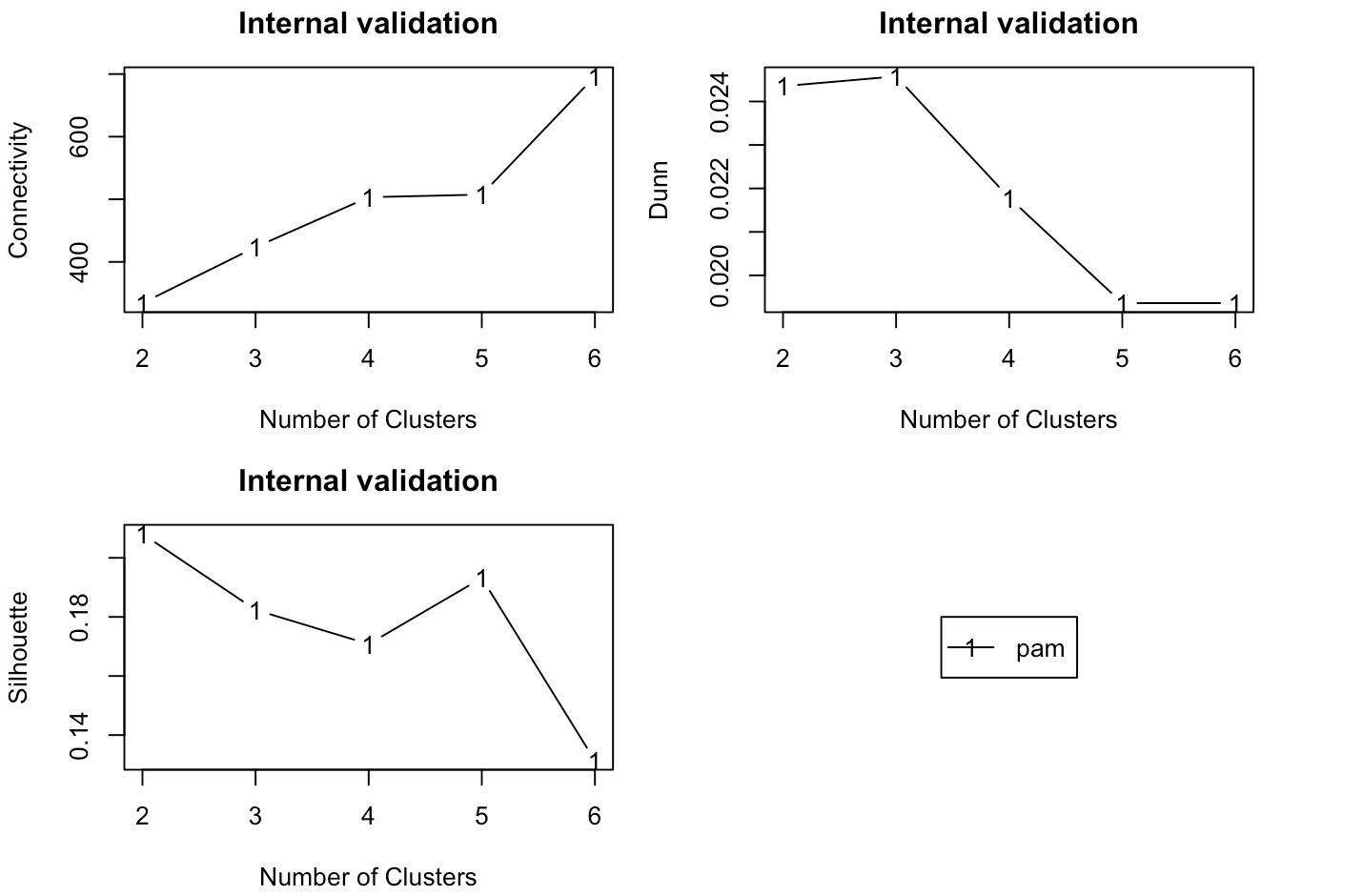
```
##
## Clustering Methods:
##  pam
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                            2        3        4        5        6
##
## pam Connectivity  334.2024 423.6127 503.3202 507.5361 696.2833
##     Dunn            0.0244   0.0246   0.0218   0.0194   0.0194
##     Silhouette      0.2081   0.1824   0.1708   0.1932   0.1313
##
## Optimal Scores:
##
##              Score    Method Clusters
## Connectivity 334.2024 pam    2
## Dunn           0.0246 pam    3
## Silhouette     0.2081 pam    2
```

Interpretation:

As per the internal validation measure it suggests that for connectivity and Silhouette 2 clusters perform better where as for Dunn Index 3 clusters perform better.

Internal Measure plots

```
op = par(no.readonly=TRUE)
par(mfrow=c(2,2),mar=c(4,4,3,1))
plot(intern_pam, legend=FALSE)
plot(nClusters(intern_pam),measures(intern_pam,"Dunn")[,,1],type="n",axes=F,xlab="",ylab
="")
legend("center", clusterMethods(intern_pam),col=1:9, lty=1:9,pch=paste(1:9))
```




```
par(op)
```

Stability measures

```
stab_pam <- clValid(df.scaled, nClust = c(2,4,6), clMethods = clmethods, validation = "s
tability",maxitems = 1599)
optimalScores(stab_pam)
```

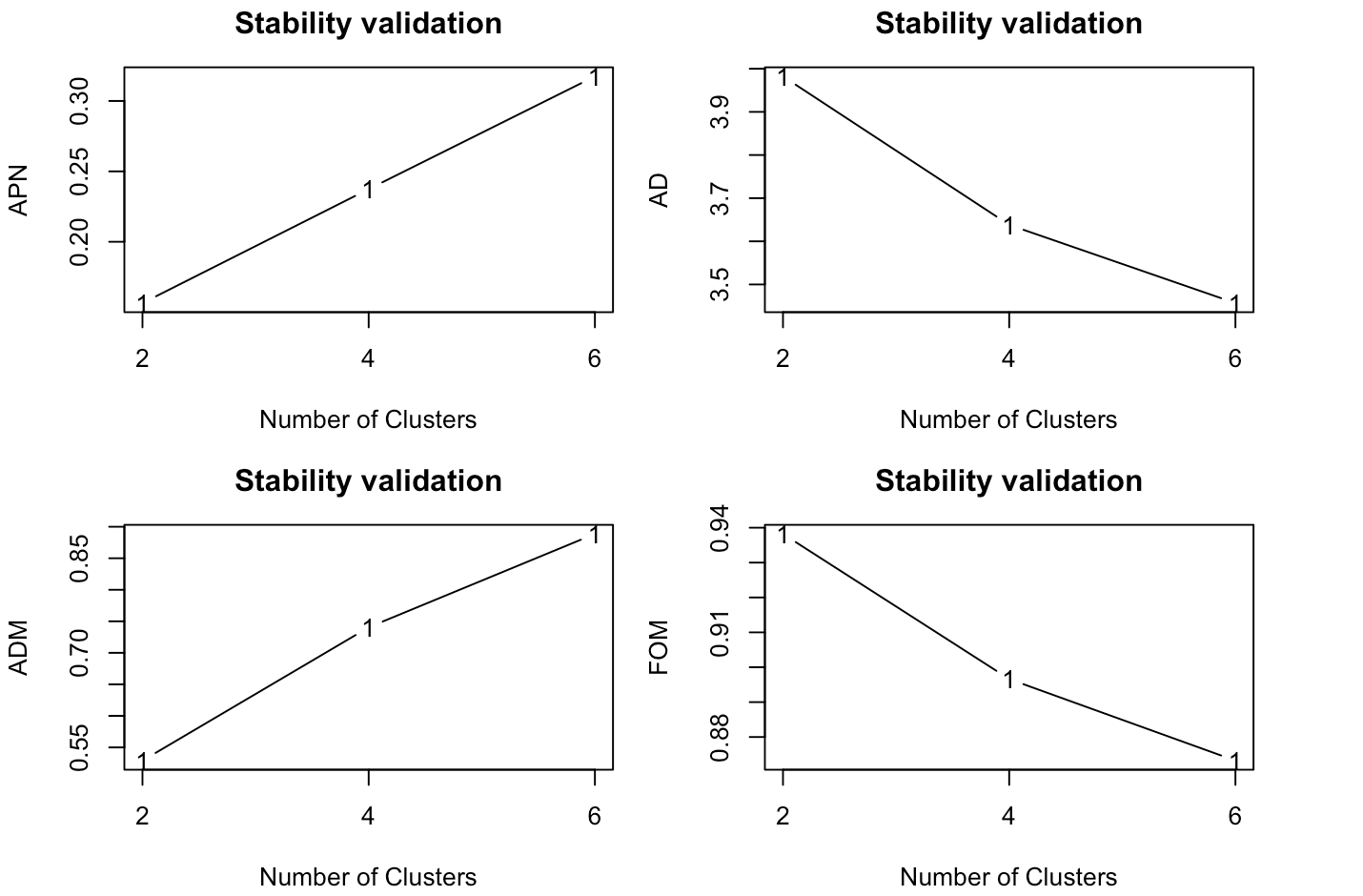
```
##         Score Method Clusters
## APN 0.1564498    pam        2
## AD  3.4567538    pam        6
## ADM 0.5289862    pam        2
## FOM 0.8732232    pam        6
```

Interpretation:

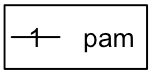
For APN and ADM measures, PAM with 3 clusters gives the best score and for other measures PAM with 6 clusters has the best score.

Stability Measure plots

```
par(mfrow=c(2,2),mar=c(4,4,3,1))
plot(stab_pam,measure = c("APN","AD","ADM","FOM"), legend= FALSE)
```




```
plot(nClusters(stab_pam),measures(stab_pam,"APN")[,,1],type="n",axes=F,xlab="",ylab="")
legend("center", clusterMethods(stab_pam),col=1:9, lty=1:9,pch=paste(1:9))
par(op)
```



External Validation

```
table(Rwine$quality, pam.res1$cluster)
```

```
##
##       1   2   3   4
##   3   7   0   3   0
##   4  35   7   8   3
##   5 287 234 142  18
##   6 252  73 201 112
##   7  31   5  80  83
##   8   1   0   7  10
```

From the ground truth variable table for k=4 clusters we see that the maximum groupings is for quality 5 and 6.

Corrected Rand Index

```
library(fpc)
quality <- as.numeric(Rwine$quality)
cs1= cluster.stats(d = dist(df.scaled), quality, pam.res1$cluster)
ari1= cs1$corrected.rand
ari1
```

```
## [1] 0.06454827
```

```
cs2= cluster.stats(d = dist(df.scaled), quality, pam.res2$cluster)
ari2= cs2$corrected.rand
ari2
```

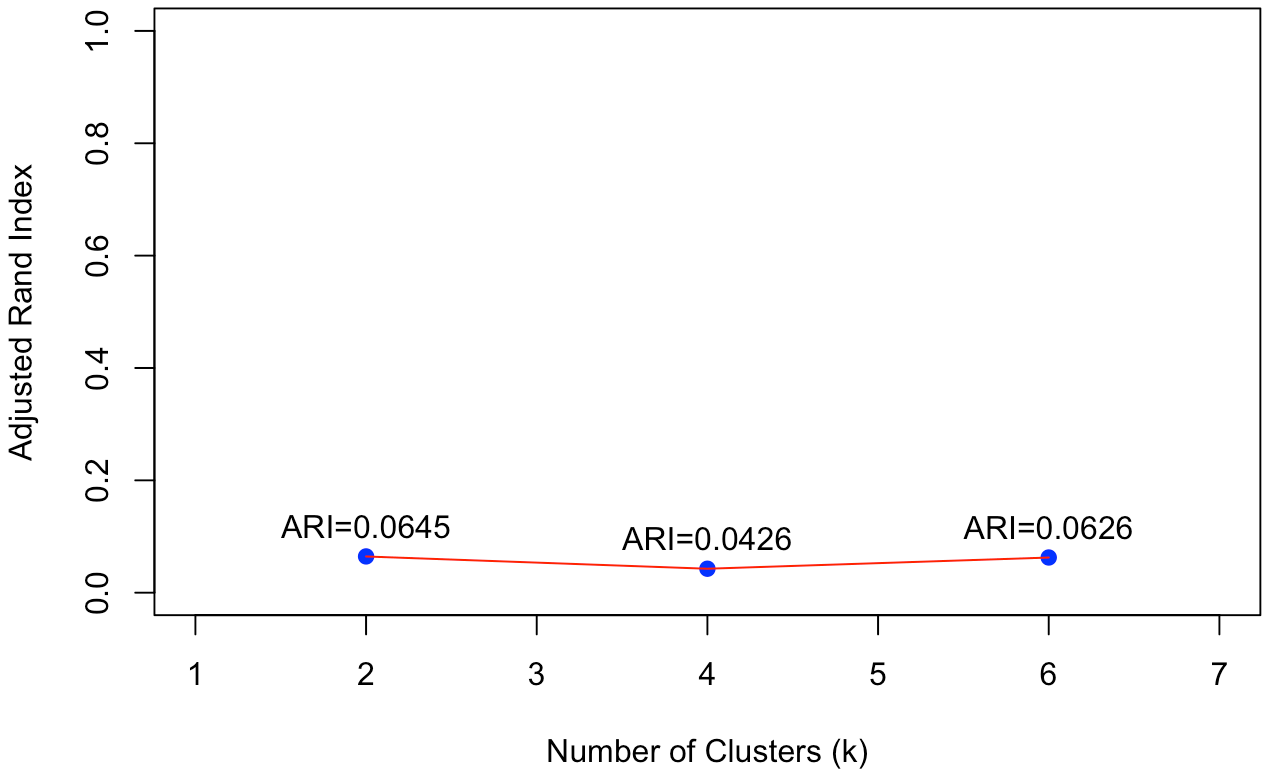
```
## [1] 0.04264211
```

```
cs3= cluster.stats(d = dist(df.scaled), quality, pam.res3$cluster)
ari3= cs3$corrected.rand
ari3
```

```
## [1] 0.06264732
```

Interpretation: The corrected Rand index value shows that agreement between quality type and cluster solution is 0.06454827. It has a range of -0.5 to 1 and value of 0.0645 shows a very low index value.

```
k_values <- c(2, 4, 6)
ari_values <- c(ari1, ari2, ari3)

plot(k_values, ari_values,
     main = "Adjusted Rand Index for different k",
     xlab = "Number of Clusters (k)",
     ylab = "Adjusted Rand Index",
     xlim = c(1, 7), ylim = c(0, 1),
     pch = 19, col = "blue")
lines(k_values, ari_values, col = "red", lwd = 1)
text(k_values, ari_values,
     labels = sprintf("ARI=%.4f", ari_values),
     pos = 3, col = "black")
```



Interpretation: We have 2 ARI values that are similar and highest value suggests that k=6 as ground truth variable has 6 types.

Meila's VI

```
meilas_index <- function(true_labels, cluster_labels) {
  N <- length(true_labels)
  M <- 0
  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      I_c1_diff <- ifelse(true_labels[i] != true_labels[j], 1, 0)
      I_c2_eq <- ifelse(cluster_labels[i] == cluster_labels[j], 1, 0)
      M <- M + I_c1_diff * I_c2_eq
    }
  }
  mi_score <- M / (N * (N - 1))
  return(mi_score)
}
```

```
cluster_label1 <- pam.res1$cluster
true_label <- Rwine$quality
meila_index1 <- meilas_index(true_label, cluster_label1)
print(meila_index1)
```

```
## [1] 0.0837245
```

```
cluster_label2 <- pam.res2$cluster
meila_index2 <- meilas_index(true_label, cluster_label2)
print(meila_index2)
```

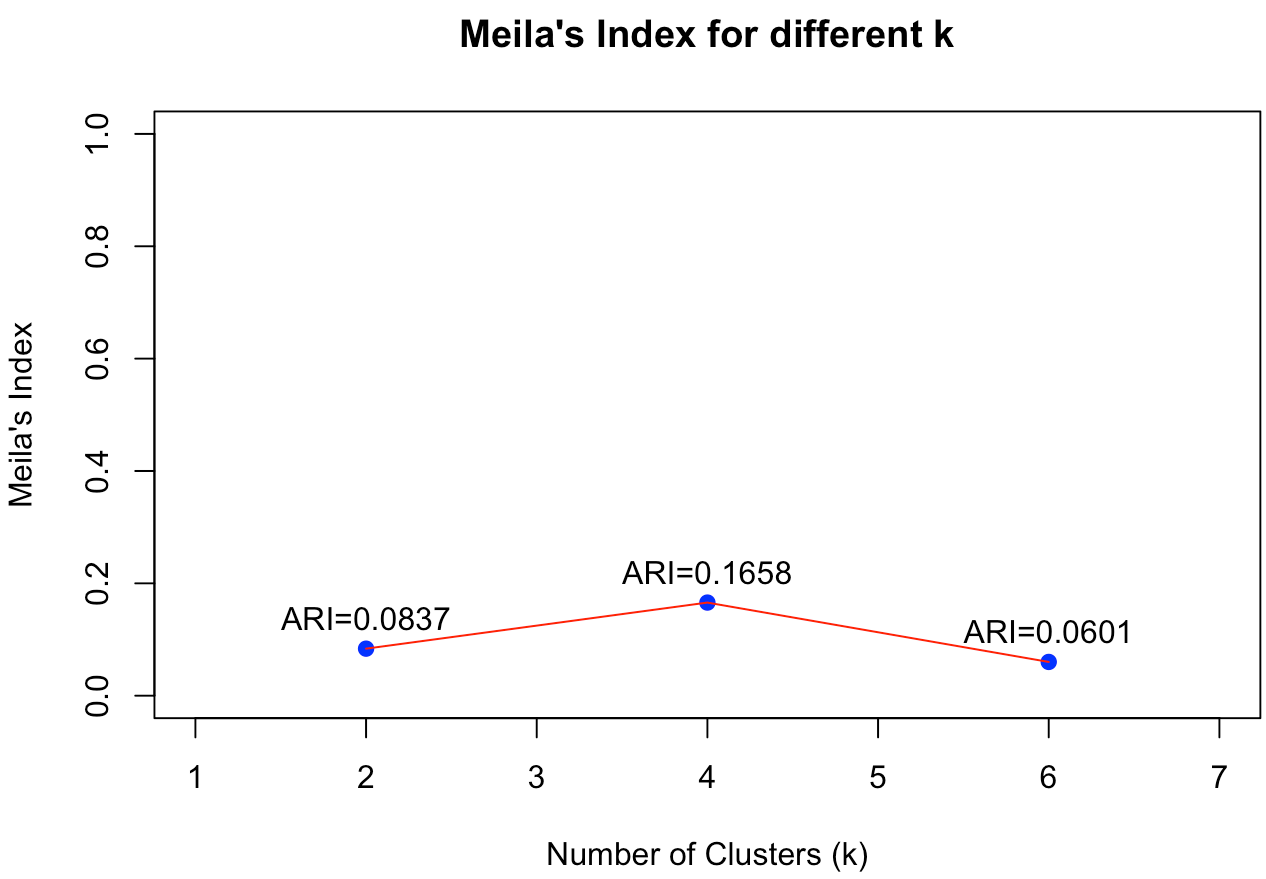
```
## [1] 0.1657615
```

```
cluster_label3 <- pam.res3$cluster
meila_index3 <- meilas_index(true_label, cluster_label3)
print(meila_index3)
```

```
## [1] 0.0600837
```

Interpretation: The agreement between the quality types and cluster solution is 0.0837245 using Meila's VI.

```
k_values <- c(2, 4, 6)
vi_values <- c(meila_index1, meila_index2, meila_index3)

plot(k_values, vi_values,
     main = "Meila's Index for different k",
     xlab = "Number of Clusters (k)",
     ylab = "Meila's Index",
     xlim = c(1, 7), ylim = c(0, 1),
     pch = 19, col = "blue")
lines(k_values, vi_values, col = "red", lwd = 1)
text(k_values, vi_values,
     labels = sprintf("ARI=%.4f", vi_values),
     pos = 3, col = "black")
```



Interpreattion:

The lowest Meila's VI value 0.0601, suggests that k=6. This can be due to ground truth variable which has 6 categories.

Hierarchical Clustering - Swiss dataset

Introduction

This report analyzes the `swiss` dataset, which contains standardized fertility measures and socioeconomic indicators for 47 French-speaking provinces of Switzerland in 1888. The dataset illustrates the demographic transition during a period when fertility rates began to decline from previously high levels.

Dataset Description

Swiss Fertility and Socioeconomic Indicators (1888) Data

The `swiss` dataset contains 47 observations on 6 variables. Here is a summary of the variables:

- **Fertility**: Common standardized fertility measure.
- **Agriculture**: Percentage of males involved in agriculture as an occupation.
- **Examination**: Percentage of draftees receiving the highest mark on the army examination.
- **Education**: Percentage of draftees with education beyond primary school.
- **Catholic**: Percentage of the population identifying as Catholic.

- **Infant.Mortality**: Live births who live less than one year (as a percentage).

Objective

The purpose of clustering is to group provinces into segments with similar socio-economic profiles. This analysis aims to uncover patterns that can inform regional development strategies and help policymakers address regional disparities.

In this study, we will be doing hierarchical clustering on the dataset and compare various linkage methods like Complete, Single, Average, Ward.D, Ward.D2 through various methods.

```
library(ggplot2)
library(factoextra)
library(dplyr)
library(ggrepel)
```

```
data("swiss")
df <- swiss
head(df)
```

```
##              Fertility Agriculture Examination Education Catholic
## Courtelary        80.2        17.0          15        12     9.96
## Delemont          83.1        45.1           6         9    84.84
## Franches-Mnt      92.5        39.7           5         5    93.40
## Moutier           85.8        36.5          12         7    33.77
## Neuveville        76.9        43.5          17        15     5.16
## Porrentruy        76.1        35.3           9         7    90.57
##              Infant.Mortality
## Courtelary               22.2
## Delemont                 22.2
## Franches-Mnt             20.2
## Moutier                  20.3
## Neuveville               20.6
## Porrentruy               26.6
```

```
head(df)
```

```
##              Fertility Agriculture Examination Education Catholic
## Courtelary        80.2        17.0          15        12     9.96
## Delemont          83.1        45.1           6         9    84.84
## Franches-Mnt      92.5        39.7           5         5    93.40
## Moutier           85.8        36.5          12         7    33.77
## Neuveville        76.9        43.5          17        15     5.16
## Porrentruy        76.1        35.3           9         7    90.57
##              Infant.Mortality
## Courtelary               22.2
## Delemont                 22.2
## Franches-Mnt             20.2
## Moutier                  20.3
## Neuveville               20.6
## Porrentruy               26.6
```

```
# structure of dataset
str(df)
```

```
## 'data.frame':    47 obs. of  6 variables:
##  $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 ...
##  $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...
##  $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
##  $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
##  $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
##  $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...
```

```
summary(df)
```

```
##    Fertility      Agriculture     Examination      Education
##  Min.   :35.00   Min.   : 1.20   Min.   : 3.00   Min.   : 1.00
##  1st Qu.:64.70   1st Qu.:35.90   1st Qu.:12.00   1st Qu.: 6.00
##  Median :70.40   Median :54.10   Median :16.00   Median : 8.00
##  Mean   :70.14   Mean   :50.66   Mean   :16.49   Mean   :10.98
##  3rd Qu.:78.45   3rd Qu.:67.65   3rd Qu.:22.00   3rd Qu.:12.00
##  Max.   :92.50   Max.   :89.70   Max.   :37.00   Max.   :53.00
##     Catholic       Infant.Mortality
##  Min.   :  2.150   Min.   :10.80
##  1st Qu.:  5.195   1st Qu.:18.15
##  Median : 15.140   Median :20.00
##  Mean   : 41.144   Mean   :19.94
##  3rd Qu.: 93.125   3rd Qu.:21.70
##  Max.   :100.000   Max.   :26.60
```

Insights from the Swiss Dataset

- **Fertility**: The fertility rates across the provinces range from 35 to 92.5, with the median being 70.4. The distribution is fairly symmetrical, indicating that most provinces have moderate fertility rates, though there are some provinces with significantly higher fertility levels.

- **Agriculture**: The percentage of males involved in agriculture varies widely, from 1.2% to 89.7%. While the median is 54.1%, the high variability (IQR of 31.75%) suggests that some regions rely heavily on agriculture, while others are less dependent.
- **Examination**: Examination scores range from 3% to 37%, with a median of 16%. The distribution of this variable is fairly balanced, indicating moderate variation in how draftees perform on the army exam across different provinces.
- **Education**: Education levels beyond primary school range from 1% to 53%, with a median of 8%. The lower median and relatively narrow IQR indicate that most provinces have lower levels of higher education.
- **Catholic**: The percentage of the population identifying as Catholic spans from 2.15% to 100%, with a median of 15.14%. The skew towards higher percentages suggests that some provinces are predominantly Catholic, while others have minimal Catholic populations.
- **Infant Mortality**: Infant mortality rates range from 10.8% to 26.6%, with a median of 20%. This suggests that infant mortality rates are relatively consistent across provinces, with only moderate variation.

Key Observations:

- Significant variation exists in **Agriculture** and **Fertility**, indicating regional differences in economic reliance and family size.
- **Education** and **Examination** scores have more consistent distributions, pointing to generally lower education levels and moderate army exam performance across provinces.
- **Catholic** percentages suggest a divide between provinces with high and low Catholic populations.
- **Infant Mortality** shows relatively little variation, implying uniformity in health outcomes across the regions.

These insights provide an overview of the socio-economic conditions in Swiss provinces during 1888 and can guide further analysis such as clustering or exploring correlations between variables.

Missing Values Check

```
missing_values <- sapply(df, function(x) sum(is.na(x)))

# Print the number of missing values for each column
print(missing_values)
```

```
##        Fertility      Agriculture      Examination        Education
##                0                0                0                0
##         Catholic Infant.Mortality
##                0                0
```

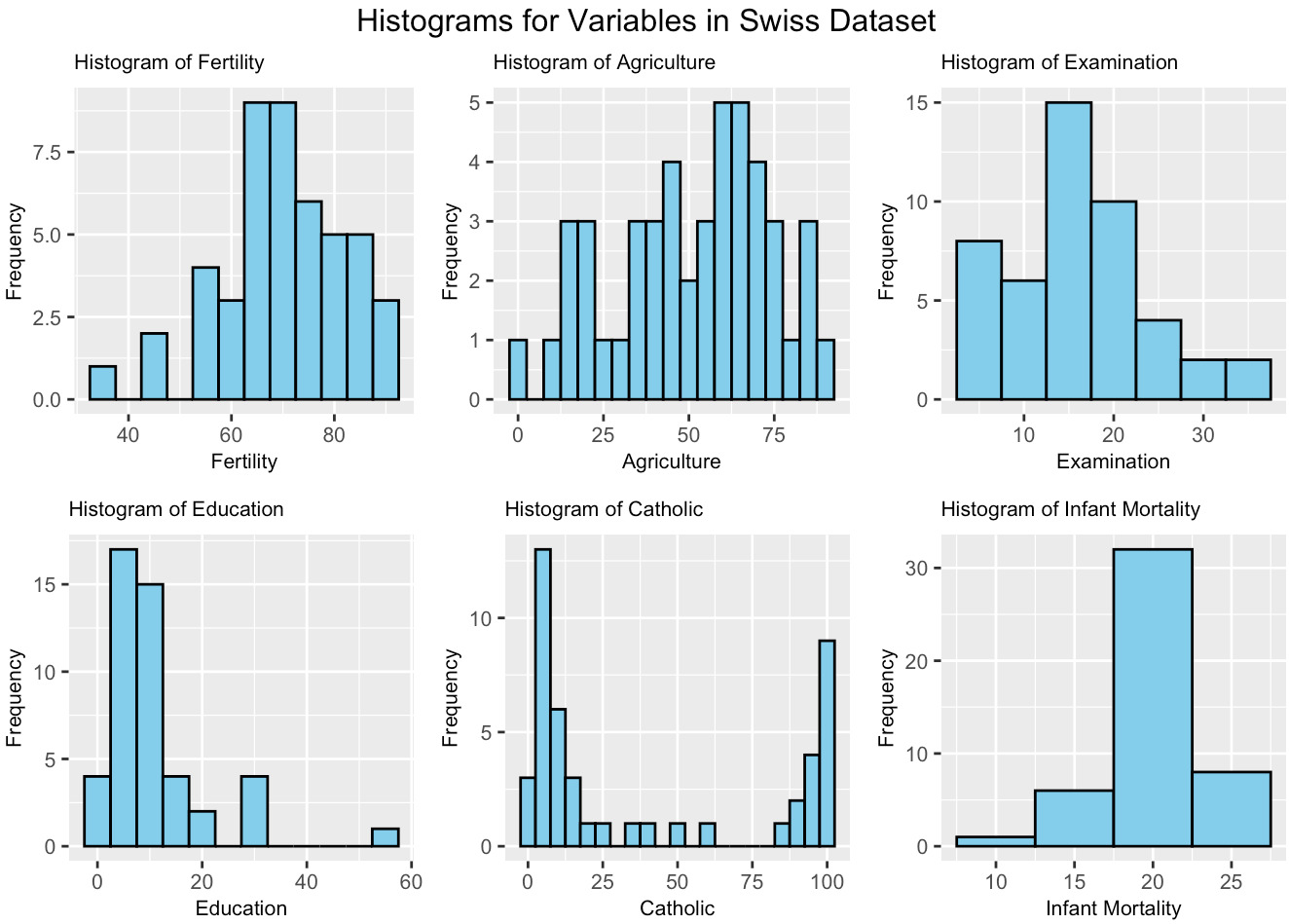
There are no missing values in the dataset.

Exploratory Data Analysis

Histograms

```r
# Load necessary libraries
library(ggplot2)
library(gridExtra)

# Create histograms for each variable in the df dataset with font size set to 8
p1 <- ggplot(df, aes(x = Fertility)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  ggtitle("Histogram of Fertility") +
  xlab("Fertility") +
  ylab("Frequency") +
  theme(text = element_text(size = 8),  # Set font size for all text
        axis.title = element_text(size = 8),
        axis.text = element_text(size = 8),
        plot.title = element_text(size = 8))

p2 <- ggplot(df, aes(x = Agriculture)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  ggtitle("Histogram of Agriculture") +
  xlab("Agriculture") +
  ylab("Frequency") +
  theme(text = element_text(size = 8),
        axis.title = element_text(size = 8),
        axis.text = element_text(size = 8),
        plot.title = element_text(size = 8))

p3 <- ggplot(df, aes(x = Examination)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  ggtitle("Histogram of Examination") +
  xlab("Examination") +
  ylab("Frequency") +
  theme(text = element_text(size = 8),
        axis.title = element_text(size = 8),
        axis.text = element_text(size = 8),
        plot.title = element_text(size = 8))

p4 <- ggplot(df, aes(x = Education)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  ggtitle("Histogram of Education") +
  xlab("Education") +
  ylab("Frequency") +
  theme(text = element_text(size = 8),
        axis.title = element_text(size = 8),
        axis.text = element_text(size = 8),
        plot.title = element_text(size = 8))

p5 <- ggplot(df, aes(x = Catholic)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  ggtitle("Histogram of Catholic") +
  xlab("Catholic") +
  ylab("Frequency") +
  theme(text = element_text(size = 8),
        axis.title = element_text(size = 8),
```

```
        axis.text = element_text(size = 8),
        plot.title = element_text(size = 8))

p6 <- ggplot(df, aes(x = Infant.Mortality)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  ggtitle("Histogram of Infant Mortality") +
  xlab("Infant Mortality") +
  ylab("Frequency") +
  theme(text = element_text(size = 8),
        axis.title = element_text(size = 8),
        axis.text = element_text(size = 8),
        plot.title = element_text(size = 8))

# Combine the plots into a 2x3 grid with font size set to 8
grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 3,
             top = "Histograms for Variables in Swiss Dataset",
             widths = c(2, 2, 2), heights = c(2, 2))  # Adjust the size of the grid
```




```
library(e1071)
skewness_values <- apply(df, 2, skewness)
print(skewness_values)
```

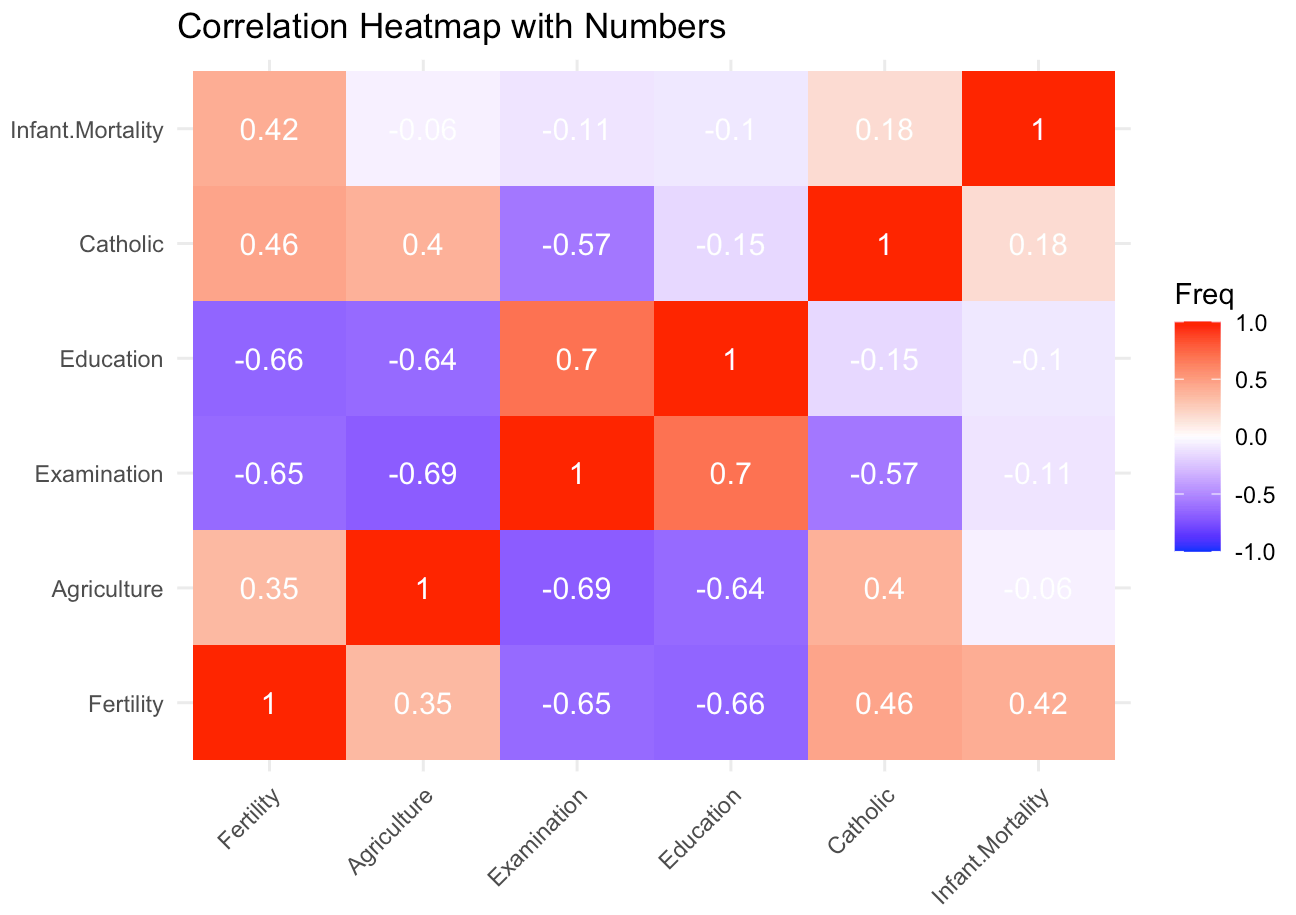
```
##        Fertility      Agriculture      Examination        Education
##       -0.4556871       -0.3203637        0.4463996        2.2684389
##         Catholic Infant.Mortality
##        0.4789257       -0.3314326
```

Interpretation of Histograms:

- **Fertility:** Slightly left-skewed, meaning most provinces have higher fertility rates, with few having lower values.
- **Agriculture:** Slightly left-skewed, indicating moderate agricultural participation in most provinces, with a few having very low involvement.
- **Examination:** Right-skewed, suggesting most provinces have low percentages of top-performing draftees, with a few excelling significantly.
- **Education:** Right-skewed, showing that most provinces have low percentages of draftees with advanced education, while only a few have high percentages.
- **Catholic:** Right-skewed, meaning most provinces have low Catholic populations, with a few having very high percentages.
- **Infant Mortality:** Approximately symmetric, indicating a balanced distribution of infant mortality rates across provinces with no significant outliers

Correlation Matrix

```
# Calculate correlation matrix
cor_matrix <- cor(df[, c("Fertility", "Agriculture", "Examination", "Education", "Cathol
ic", "Infant.Mortality")])
cor_matrix <- as.data.frame(as.table(cor_matrix))

ggplot(cor_matrix, aes(Var1, Var2, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = round(Freq, 2)), color = "white", size = 4) +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limits =
c(-1, 1)) +
  labs(title = "Correlation Heatmap with Numbers", x = "", y = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Insights from the Correlation Matrix

- **Fertility** is negatively correlated with **Examination** and **Education** (-0.65 and -0.66, respectively), indicating that regions with higher fertility rates tend to have lower educational attainment and fewer individuals achieving high marks on the army examination.
- **Agriculture** is positively correlated with **Fertility** (0.35), suggesting that regions with a higher proportion of males involved in agriculture tend to have slightly higher fertility rates.
- **Agriculture** shows a strong negative correlation with **Examination** (-0.69) and **Education** (-0.64), indicating that areas more dependent on agriculture tend to have lower examination scores and lower education levels beyond primary school.
- **Examination** and **Education** are positively correlated (0.70), which is expected since areas with better educational opportunities generally result in higher examination scores.
- **Catholic** is positively correlated with **Fertility** (0.46), implying that regions with a higher percentage of the population identifying as Catholic tend to have higher fertility rates.
- **Catholic** shows a moderate negative correlation with **Examination** (-0.57), suggesting that regions with a higher proportion of Catholics tend to have lower examination scores.
- **Infant Mortality** has a moderate positive correlation with **Fertility** (0.42), indicating that regions with higher fertility rates also tend to have higher infant mortality rates, which could reflect less-developed healthcare systems.
- **Infant Mortality** has a slight negative correlation with **Agriculture** (-0.06), suggesting that the link between infant mortality and agricultural occupation is not strong.

Assessing Clustering Tendency

Hopkins Statistic

We conducted the Hopkins Statistic test iteratively 10 times, using the mean of the Hopkins value and using 0.5 as the threshold to reject the alternative hypothesis. That is, if H < 0.5, then it is unlikely that D has statistically significant clusters.

```
library(hopkins)
hopkins_values <- numeric()
for(i in 1:10) {
  hopkins_values[i]<-hopkins(df, m=(nrow(df)-1))
}
mean(hopkins_values)
```

```
## [1] 0.9332609
```

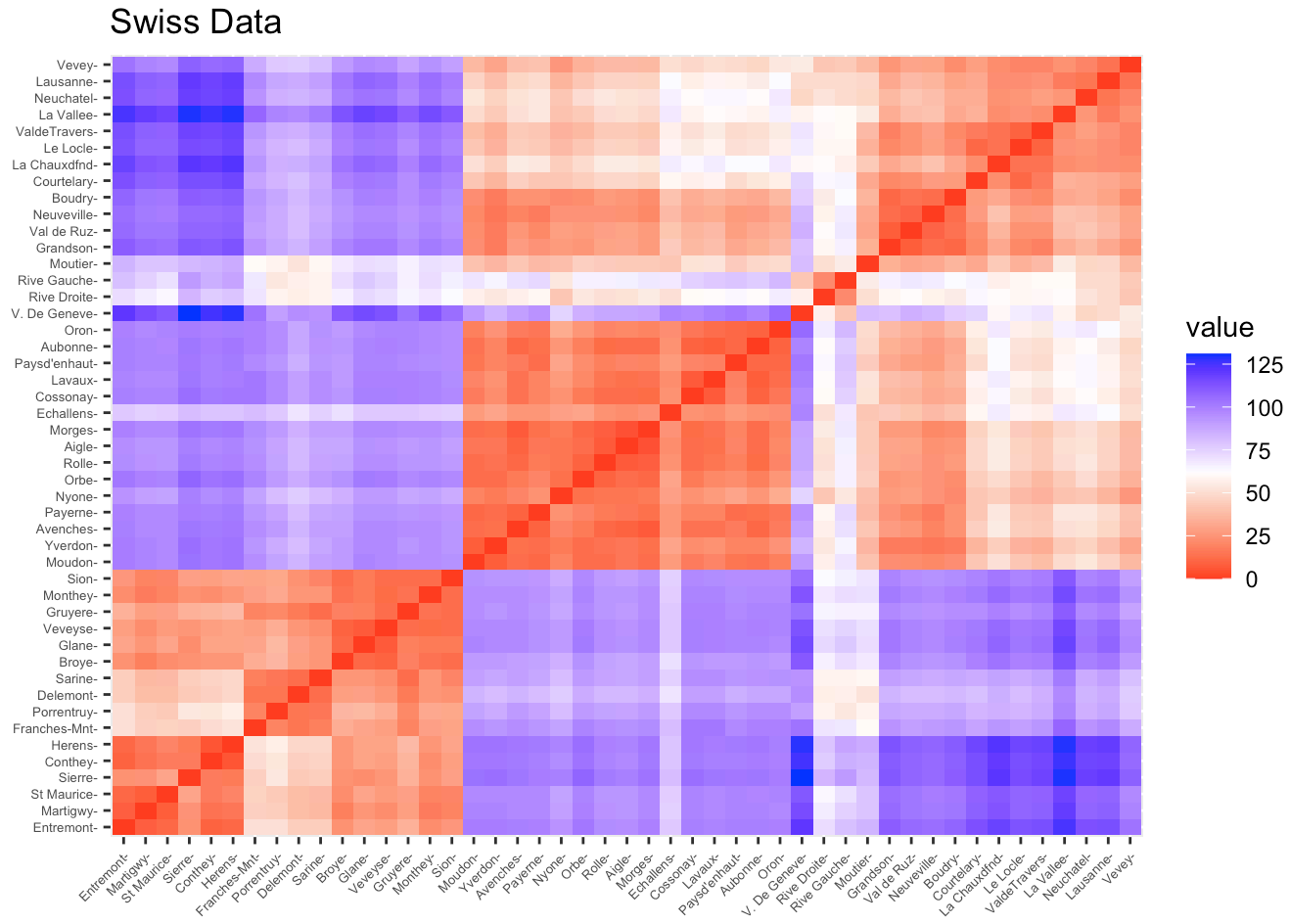
Conclusion

The Hopkins statistic for the **Swiss dataset** is **0.9391782**, which indicates a high likelihood of clustering structure in the data. A Hopkins value closer to 1 suggests that the data is not uniformly distributed and contains meaningful clusters. Therefore, performing clustering analysis on this dataset is appropriate and expected to yield insightful groupings based on the socio-economic indicators.

Dissimilarity (DM) matrix

Compute the dissimilarity (DM) matrix between the objects in the data set using the Euclidean distance measure.

```
fviz_dist(dist(df), lab_size = 5)+
labs(title = "Swiss Data")
```



Conclusion

The dissimilarity matrix image confirms that there is a cluster structure in the swiss data set.

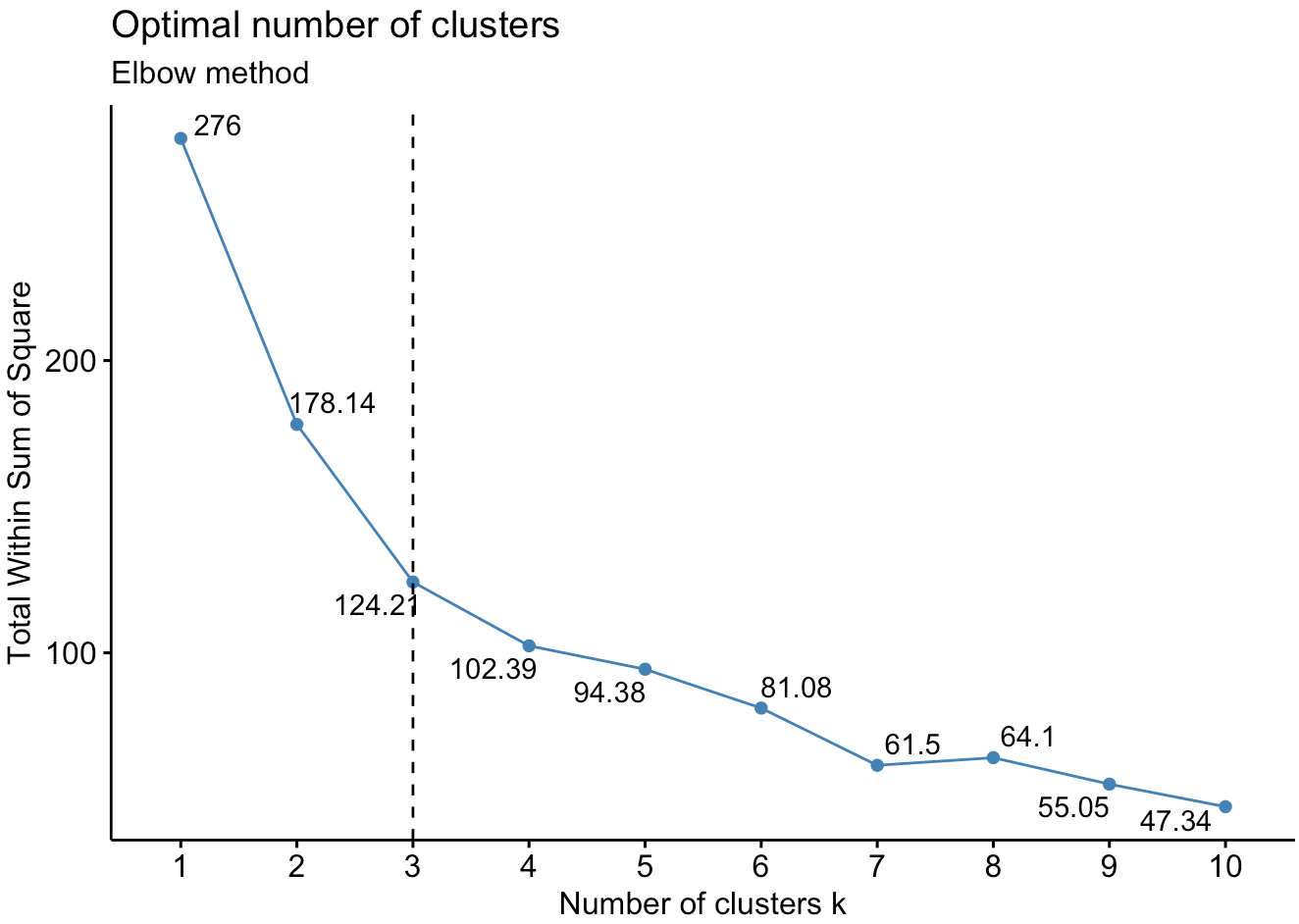
Optimal Clusters

```
df <- scale(df)
```

Elbow method

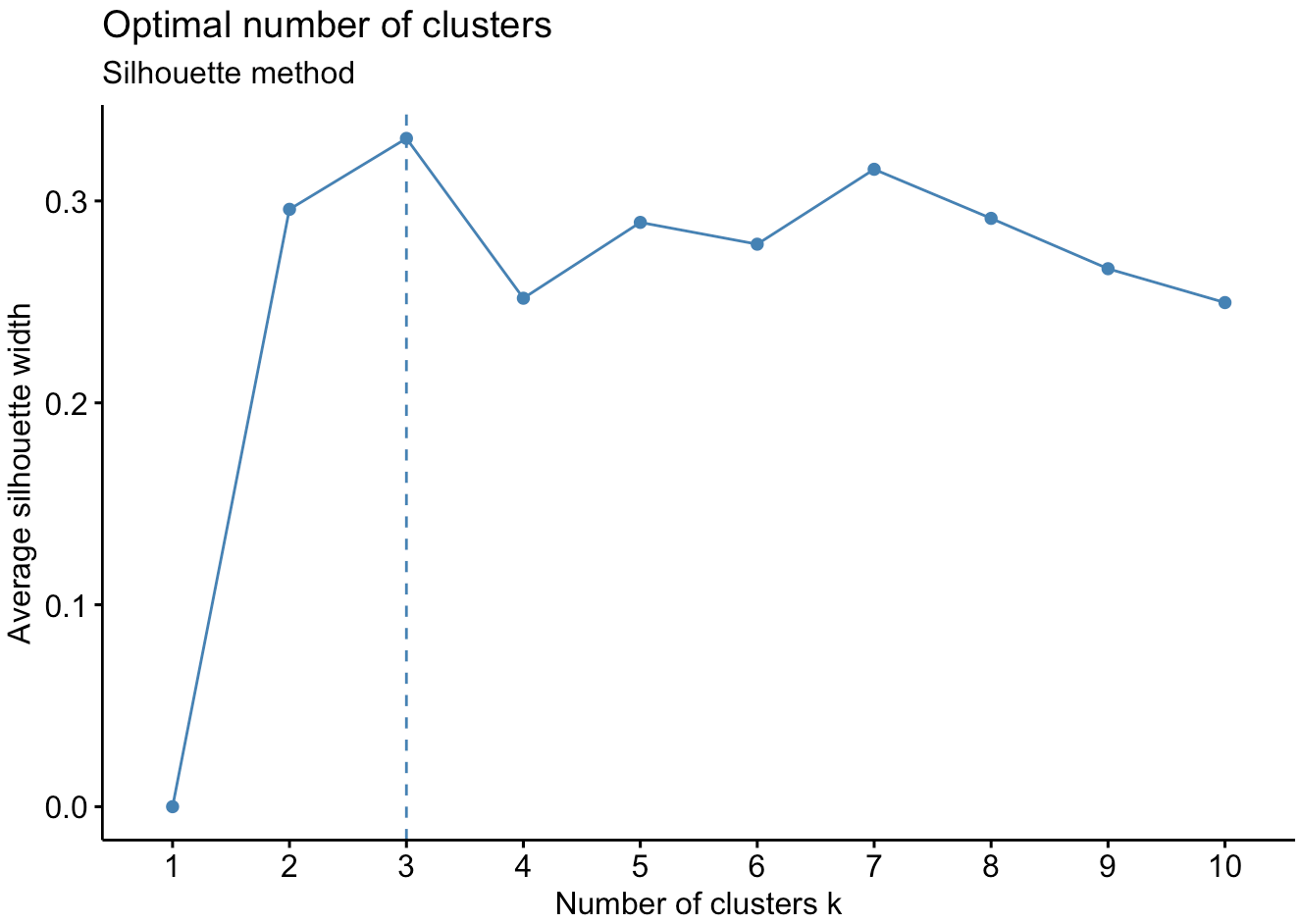
```
library(factoextra)
```

```
fviz_nbclust(df, kmeans, method = "wss") +
geom_vline(xintercept = 3, linetype = 2)+
labs(subtitle = "Elbow method") + geom_text_repel(aes(label = round(..y.., 2)), stat =
"summary", method = "wss")
```



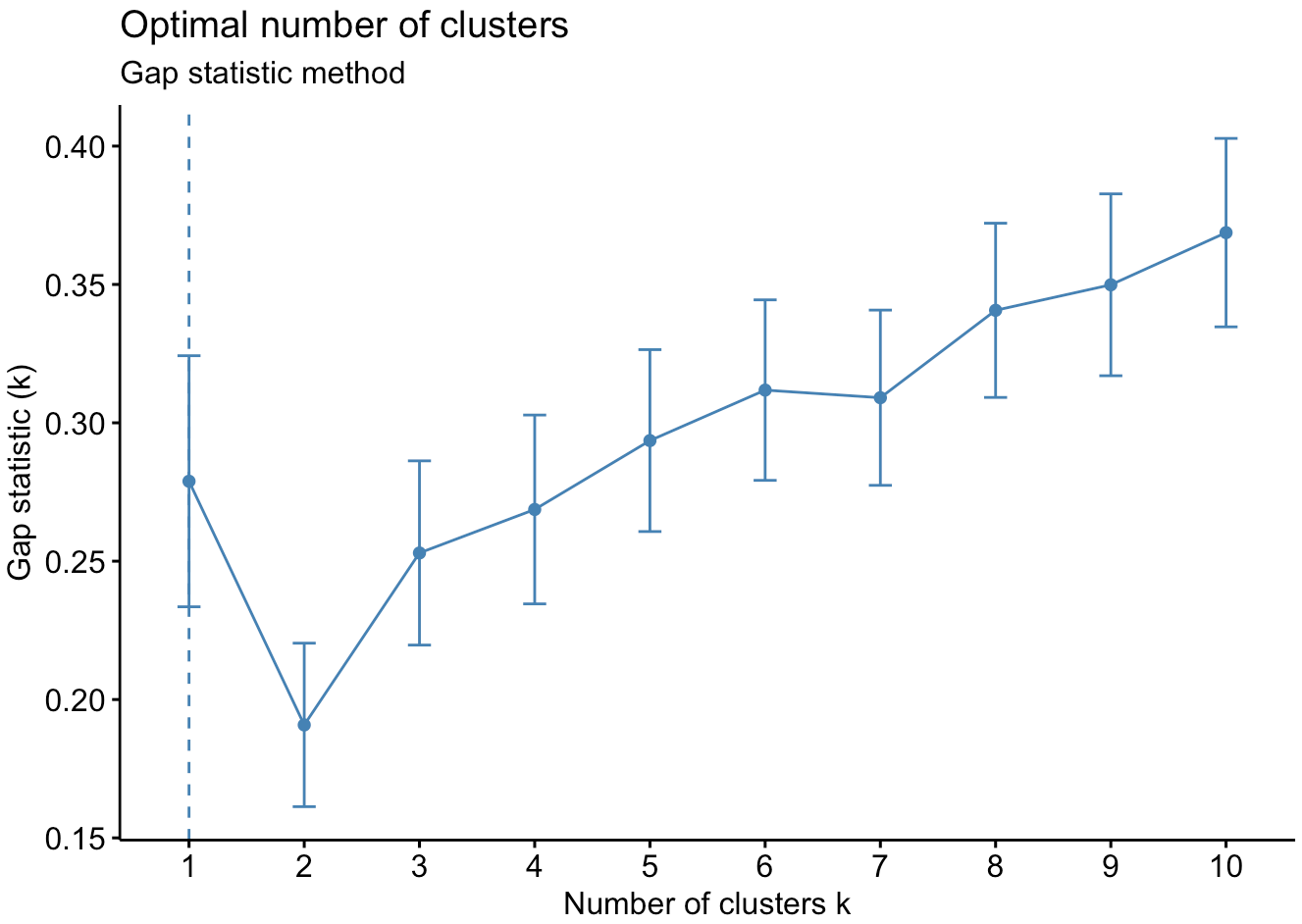
Silhouette method

```
fviz_nbclust(df, kmeans, method = "silhouette")+
labs(subtitle = "Silhouette method")
```



Gap statistic

```
set.seed(123)
fviz_nbclust(df, kmeans, nstart = 25, method = "gap_stat", nboot = 50)+
labs(subtitle = "Gap statistic method")
```



Elbow Method

The Elbow method suggests a **3-cluster solution** based on the point where the rate of decrease in the within-cluster sum of squares (WCSS) slows significantly.

Silhouette Method

The Silhouette method suggests a **3-cluster solution**, as this number maximizes the average silhouette width, indicating well-separated and compact clusters.

Gap Statistic Method

The Gap Statistic method suggests a **1-cluster solution** by comparing the total within-cluster variation to that of a null reference distribution.

Conclusion

Based on these observations, a **3-cluster solution** is recommended as the optimal choice for the data, as it is supported by both the Elbow and Silhouette methods.

Hierarchical Clustering

We are going to do different linkage methods to compare the clustering.

```
##Computing distance matrix
res.dist <- dist(df, method = "euclidean")
```

Comparing linkage methods using the cluster plot

In this section, we will compare the clusters formed by different linkage methods and find which linkage method clusters distinctively.

```
res.hc <- hclust(d = res.dist, method = "ward.D2")
# Cut tree into 3 groups
grp.hc <- cutree(res.hc, k = 3)
```

```
head(grp.hc, n = 4)
```

```
##   Courtelary     Delemont Franches-Mnt      Moutier
##            1            2            2            1
```

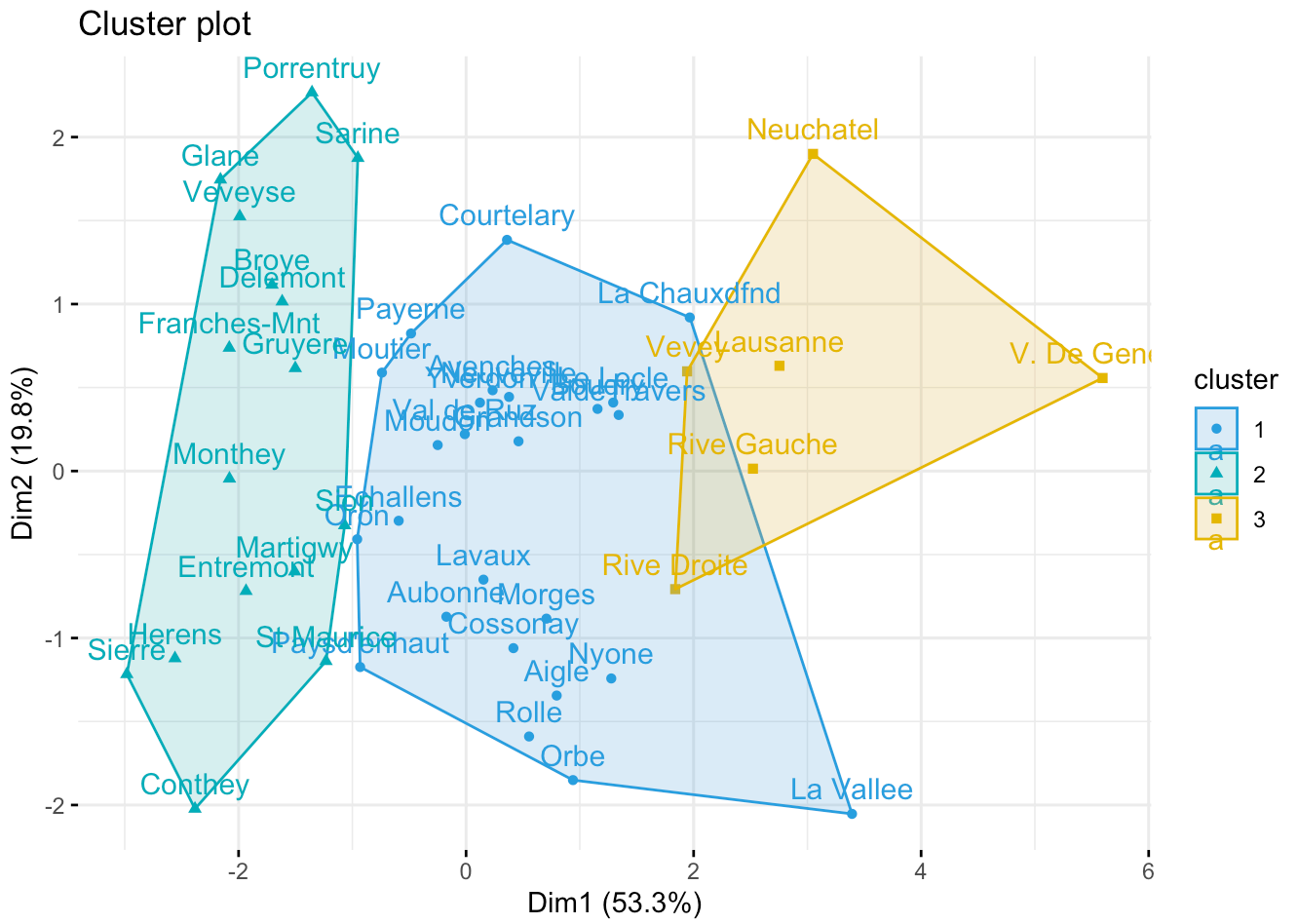
```
# Number of members in each cluster
table(grp.hc)
```

```
## grp.hc
##  1  2  3
## 25 16  6
```

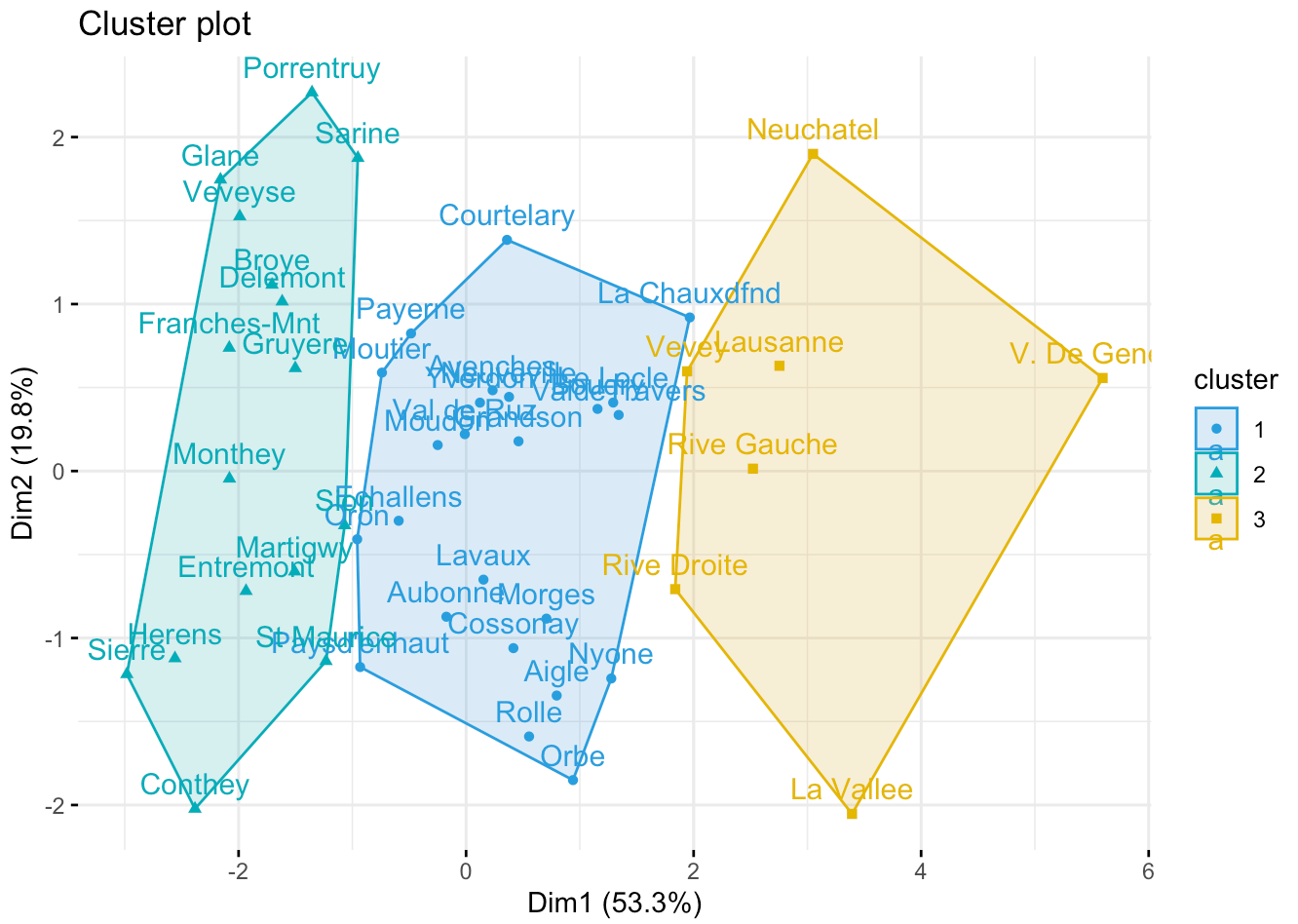
```
# Get the names for the members of cluster 1
rownames(df)[grp.hc == 1]
```

```
##  [1] "Courtelary"   "Moutier"      "Neuveville"   "Aigle"        "Aubonne"
##  [6] "Avenches"     "Cossonay"     "Echallens"    "Grandson"     "La Vallee"
## [11] "Lavaux"       "Morges"       "Moudon"       "Nyone"        "Orbe"
## [16] "Oron"         "Payerne"      "Paysd'enhaut" "Rolle"        "Yverdon"
## [21] "Boudry"       "La Chauxdfnd" "Le Locle"     "Val de Ruz"   "ValdeTravers"
```

```
## ----cluster-plot, fig.width=6, fig.height=6----------------------------
fviz_cluster(list(data = df, cluster = grp.hc),
palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
ellipse.type = "convex", # Concentration ellipse
show.clust.cent = FALSE, ggtheme = theme_minimal())
```



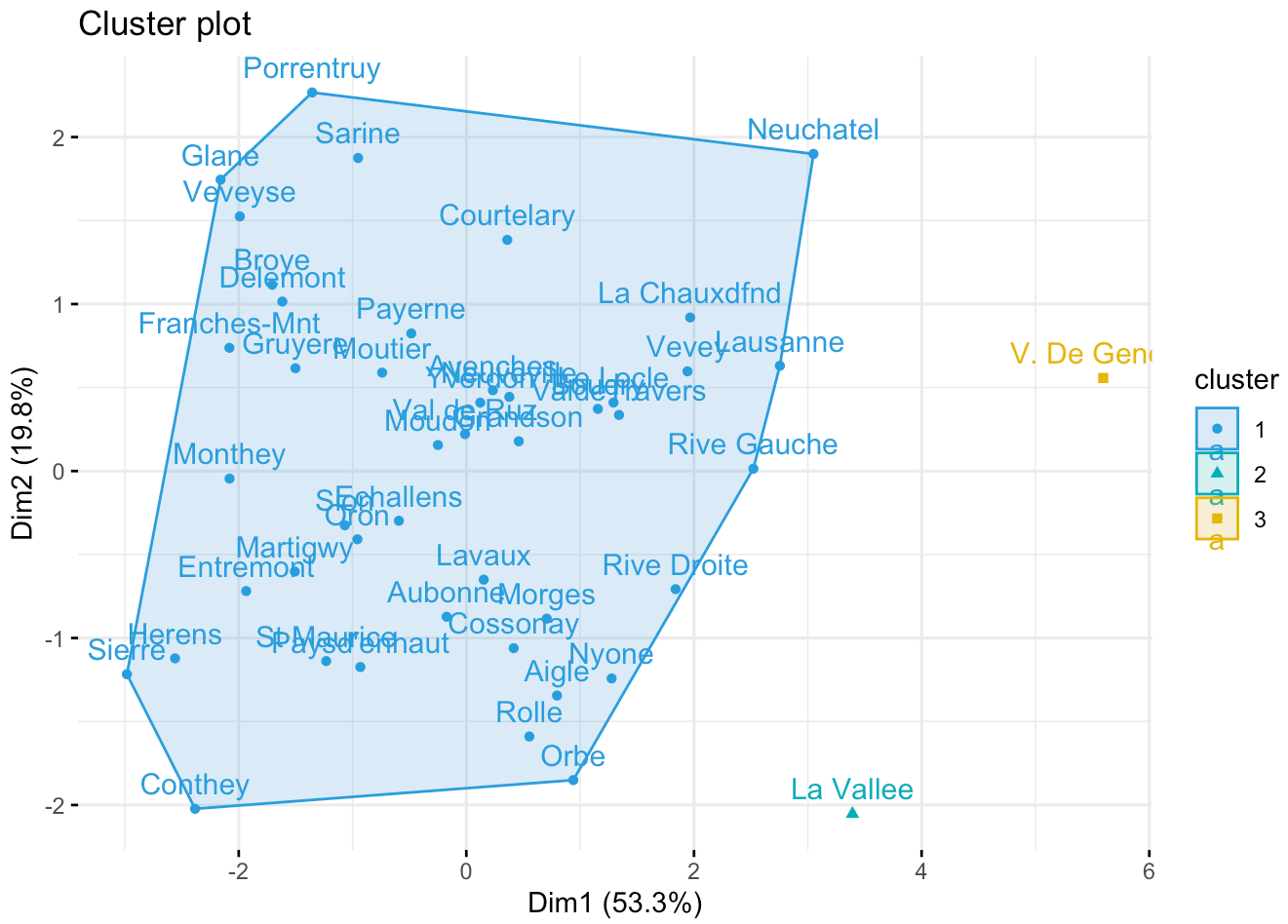

```
res.hc1 <- hclust(d = res.dist, method = "ward.D")
# Cut tree into 3 groups and color by groups
grp.hc1 <- cutree(res.hc1, k = 3)
fviz_cluster(list(data = df, cluster = grp.hc1),
palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
ellipse.type = "convex", # Concentration ellipse
show.clust.cent = FALSE, ggtheme = theme_minimal())
```




```
# Number of members in each cluster
table(grp.hc1)
```

```
## grp.hc1
##  1  2  3
## 24 16  7
```

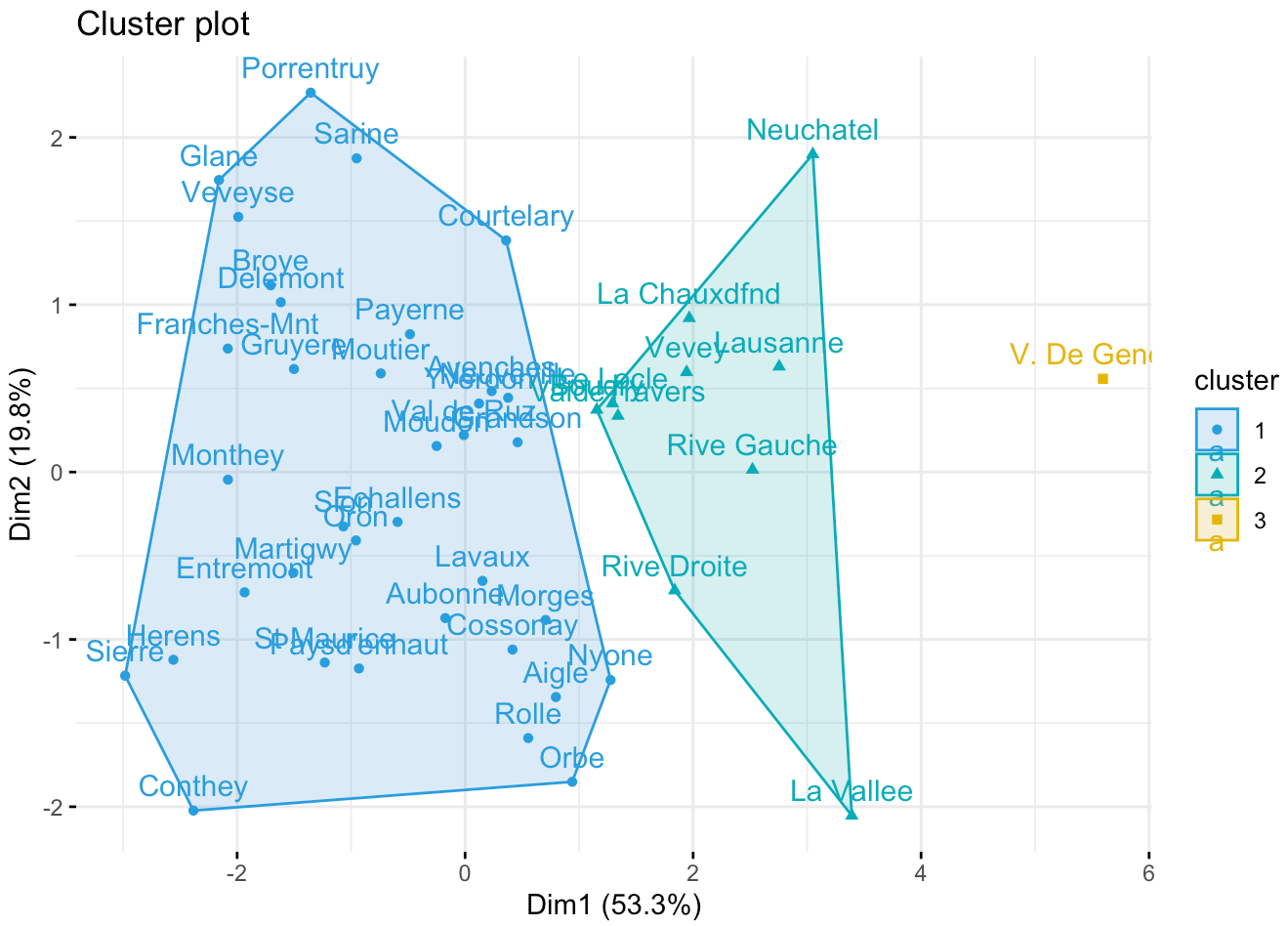
```
res.hc2 <- hclust(d = res.dist, method = "average")
# Cut tree into 3 groups
grp.hc2 <- cutree(res.hc2, k = 3)
fviz_cluster(list(data = df, cluster = grp.hc2),
palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
ellipse.type = "convex", # Concentration ellipse
show.clust.cent = FALSE, ggtheme = theme_minimal())
```




```
# Number of members in each cluster
table(grp.hc2)
```

```
## grp.hc2
##  1  2  3
## 45  1  1
```

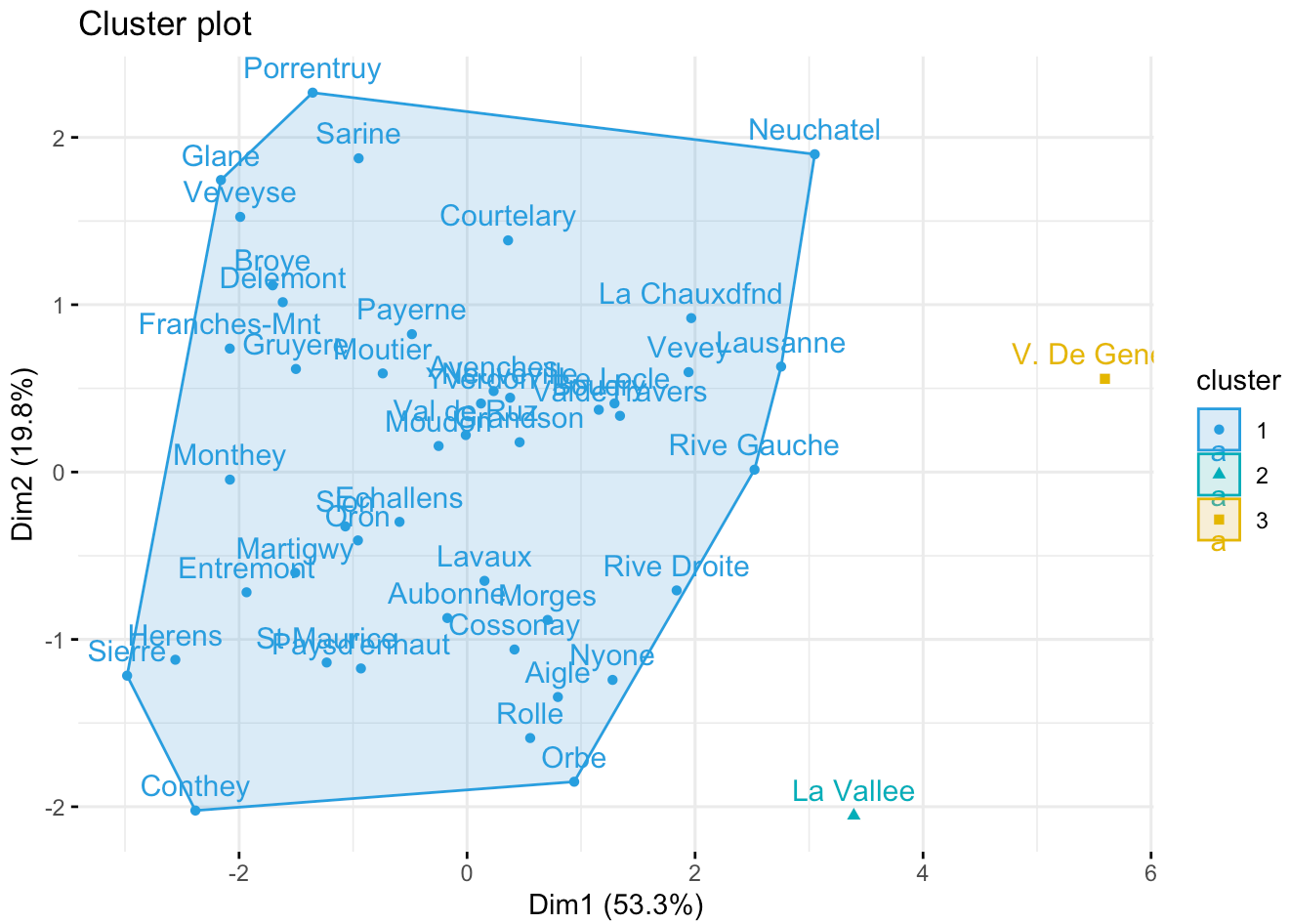
```
res.hc3 <- hclust(d = res.dist, method = "complete")
# Cut tree into 3 groups
grp.hc3 <- cutree(res.hc3, k = 3)
fviz_cluster(list(data = df, cluster = grp.hc3),
palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
ellipse.type = "convex", # Concentration ellipse
show.clust.cent = FALSE, ggtheme = theme_minimal())
```




```
# Number of members in each cluster
table(grp.hc3)
```

```
## grp.hc3
##  1  2  3
## 36 10  1
```

```
res.hc4 <- hclust(d = res.dist, method = "single")
# Cut tree into 3 groups
grp.hc4 <- cutree(res.hc4, k = 3)
fviz_cluster(list(data = df, cluster = grp.hc4),
palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
ellipse.type = "convex", # Concentration ellipse
show.clust.cent = FALSE, ggtheme = theme_minimal())
```




```
# Number of members in each cluster
table(grp.hc4)
```

```
## grp.hc4
##  1  2  3
## 45  1  1
```

Results and Interpretations

1. Ward.D2 Linkage

The **Ward.D2** method minimizes within-cluster variance, tending to form compact and homogeneous clusters. The cluster sizes are as follows:

- **Cluster 1**: 25 observations
- **Cluster 2**: 16 observations
- **Cluster 3**: 6 observations

Interpretation: The **Ward.D2** method results in clusters that are compact and homogeneous. The largest cluster (Cluster 1) contains 25 observations, while the smallest cluster (Cluster 3) includes only 6 observations. This could indicate a few outliers or specific patterns in the data.

2. Ward.D Linkage

The **Ward.D** method, like **Ward.D2**, also minimizes within-cluster variance, but with a slightly different calculation of variance for merging clusters. The results are as follows:

- **Cluster 1**: 24 observations
- **Cluster 2**: 16 observations
- **Cluster 3**: 7 observations

Interpretation: Similar to **Ward.D2**, the **Ward.D** method forms relatively balanced clusters. The main difference between the two methods is in the variance calculation, which might lead to slightly different cluster compositions. The clusters remain fairly balanced with just a slight difference in the number of observations per cluster.

3. Average Linkage

The **Average** linkage method merges clusters based on the average pairwise distance between clusters. The resulting clusters are:

- **Cluster 1**: 45 observations
- **Cluster 2**: 1 observation
- **Cluster 3**: 1 observation

Interpretation: The **Average** linkage method tends to create one large cluster that includes the majority of the observations. The small clusters (Cluster 2 and Cluster 3) contain only a single observation each. This suggests that the method is merging distant clusters based on the average distance, leading to isolated data points being treated as separate clusters.

4. Complete Linkage

The **Complete** linkage method merges clusters based on the maximum pairwise distance between clusters. The resulting clusters are:

- **Cluster 1**: 36 observations
- **Cluster 2**: 10 observations
- **Cluster 3**: 1 observation

Interpretation: The **Complete** linkage method produces tighter, more compact clusters by avoiding the merging of highly dissimilar clusters. Cluster 3, consisting of a single observation, suggests that extreme outliers are treated as separate clusters. This method tends to produce smaller and more isolated clusters.

5. Single Linkage

The **Single** linkage method merges clusters based on the minimum pairwise distance between clusters. The results are as follows:

- **Cluster 1**: 45 observations
- **Cluster 2**: 1 observation
- **Cluster 3**: 1 observation

Interpretation: The **Single** linkage method produces one large cluster (Cluster 1) with 45 observations, similar to the **Average** linkage method, and two isolated clusters with only one observation each. This method is sensitive to chaining, where clusters are formed by progressively merging the closest observations, even if they are distant from the rest of the cluster.

Cluster Characteristics Across Linkage Methods

- **Ward.D2 and Ward.D**: Both **Ward.D2** and **Ward.D** tend to form relatively balanced clusters with a compact structure. These methods are effective at detecting groups that share similar characteristics and are ideal for minimizing within-cluster variance.

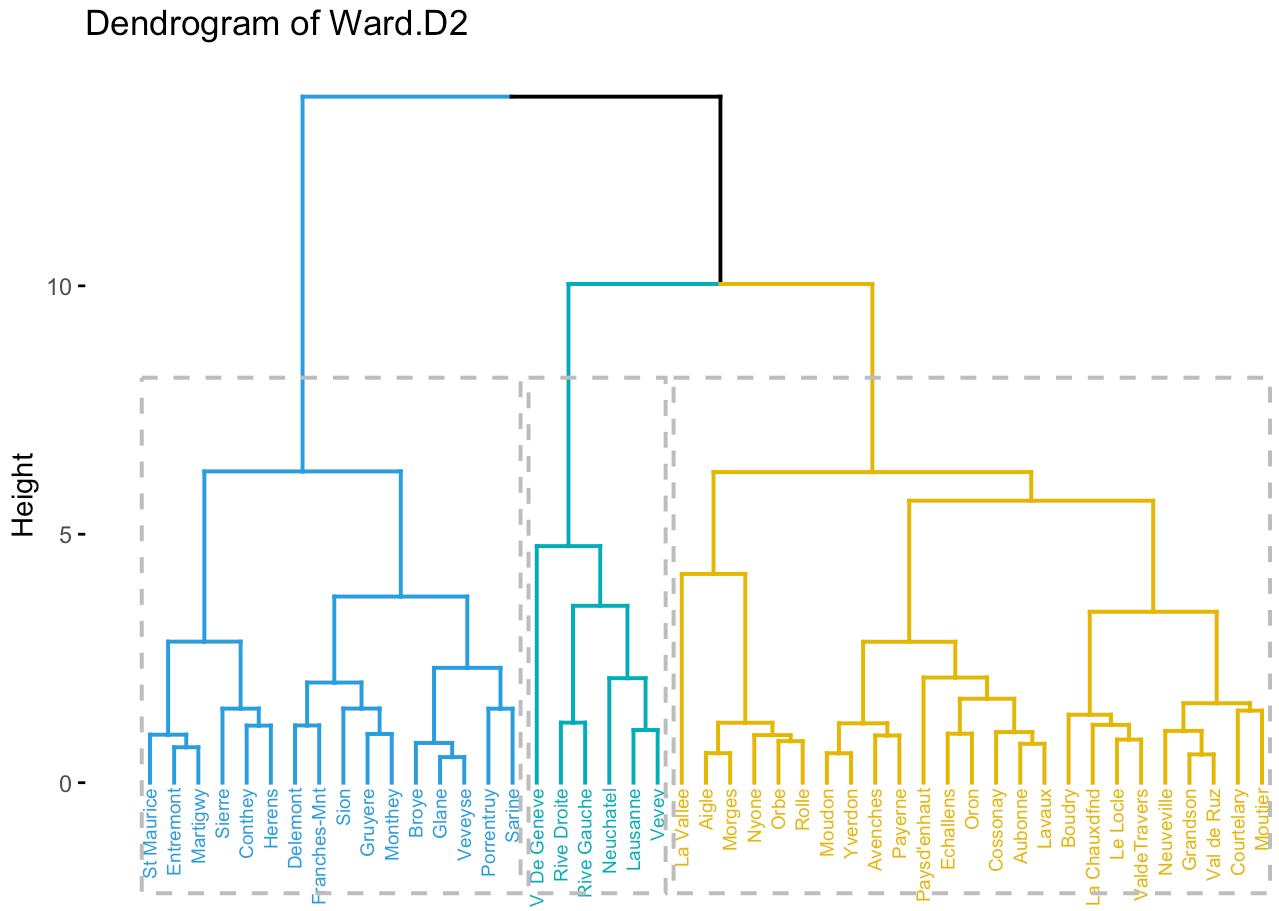
- **Average and Single**: Both **Average** and **Single** methods tend to create large clusters that merge distant points, often leading to isolated observations being treated as separate clusters. This suggests that these methods are less sensitive to the internal structure of the data and might treat outliers as separate clusters.
- **Complete Linkage**: The **Complete** linkage method produces tighter, smaller clusters by avoiding the merging of highly dissimilar points. This is useful when the goal is to have more distinct clusters, but it can result in isolated data points being treated as individual clusters.

Conclusion

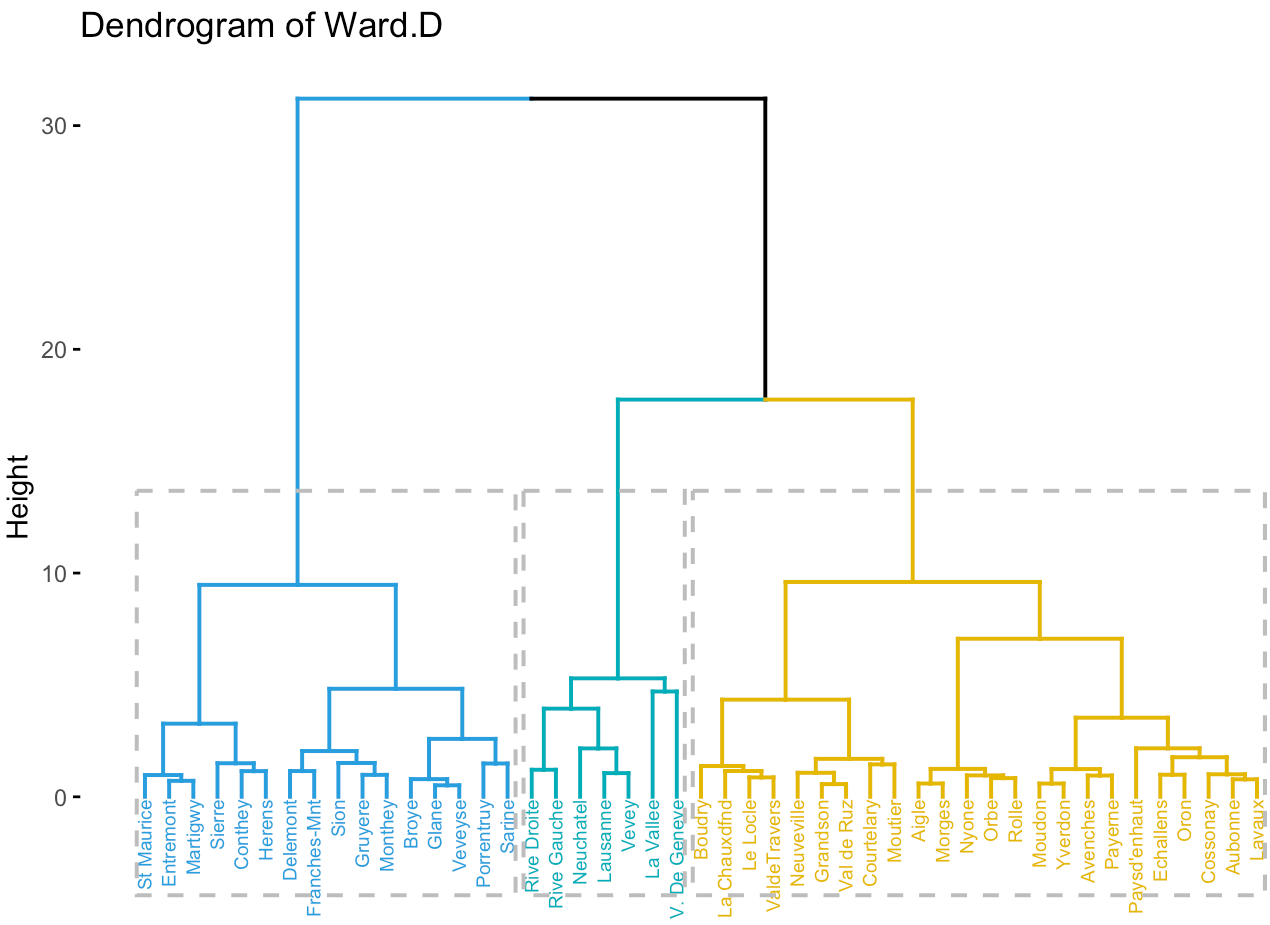
Each linkage method affects the structure of the clusters differently. **Ward.D2** and **Ward.D** are effective when the goal is to minimize variance within clusters and form compact, balanced groups. **Average** and **Single** methods merge distant points, leading to large clusters and isolated observations, while **Complete** linkage is suitable for producing smaller, tighter, and more distinct clusters.

Visualizing the cluster tree - Dendrograms

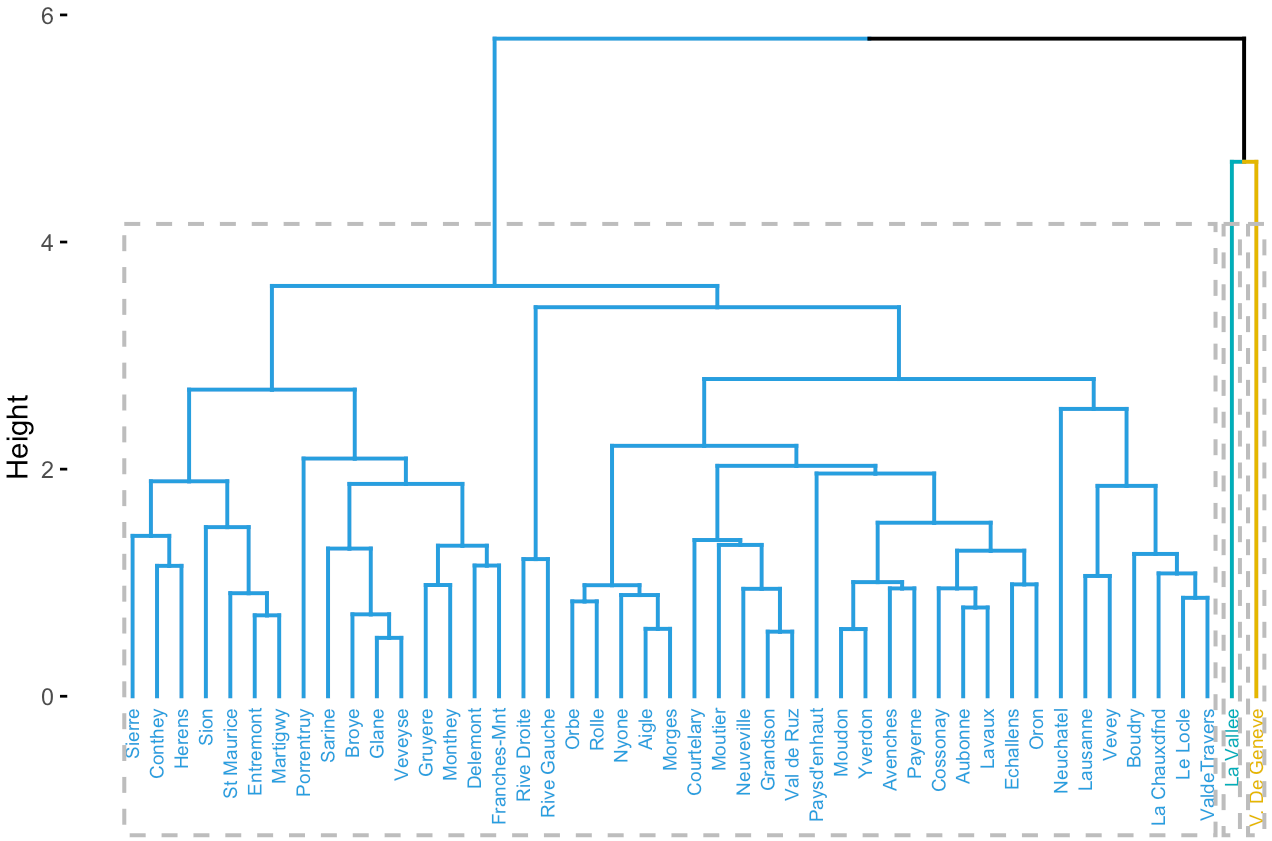
```
fviz_dend(res.hc, k = 3, # Cut in three groups
main = "Dendrogram of Ward.D2",
cex = 0.5, # label size
k_colors = c("#2E9FDF", "#00AFBB", "#E7B800"),
color_labels_by_k = TRUE, # color labels by groups
rect = TRUE # Add rectangle around groups
)
```



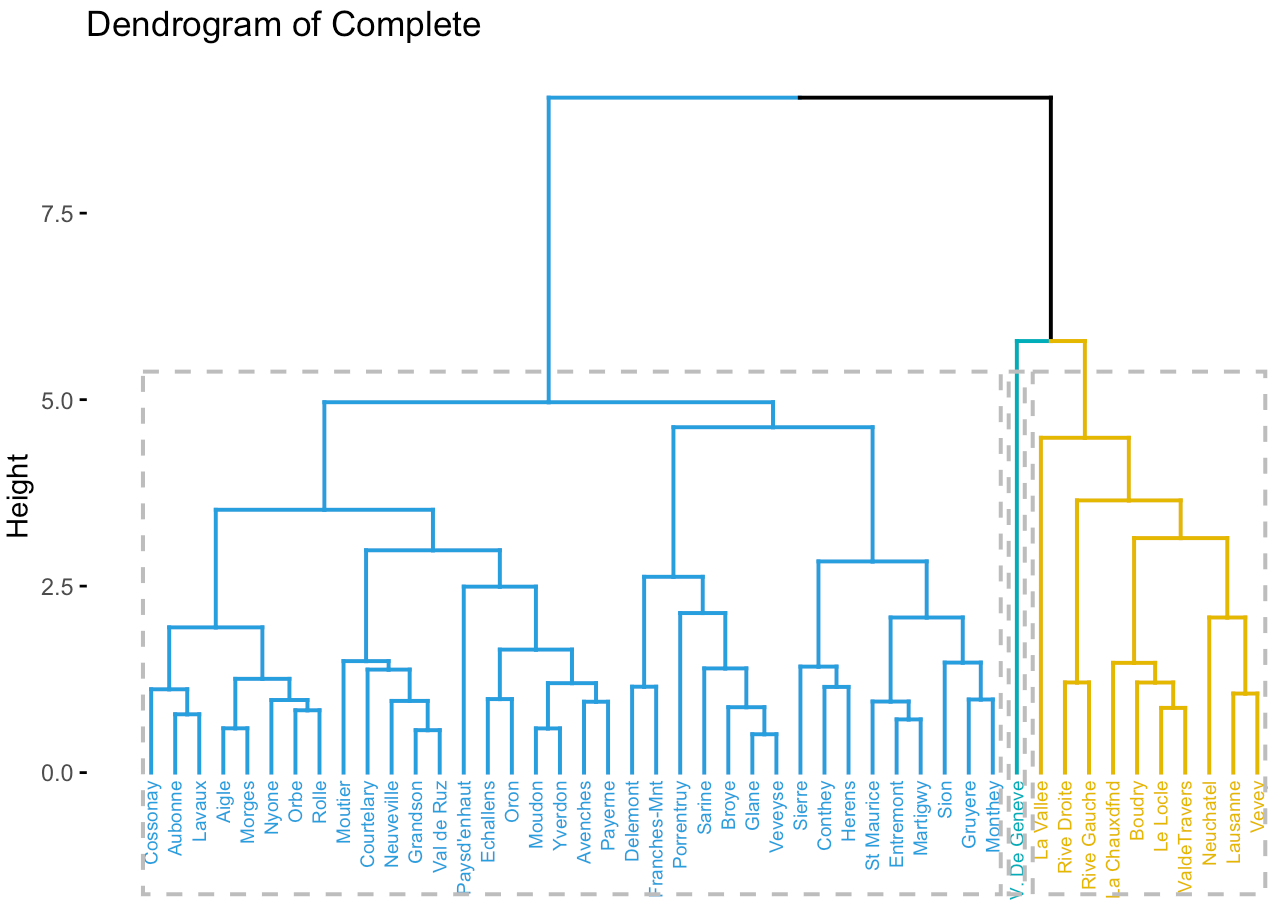
```
fviz_dend(res.hc1, k = 3, # Cut in three groups
main = "Dendrogram of Ward.D",
cex = 0.5, # label size
k_colors = c("#2E9FDF", "#00AFBB", "#E7B800"),
color_labels_by_k = TRUE, # color labels by groups
rect = TRUE # Add rectangle around groups
)
```



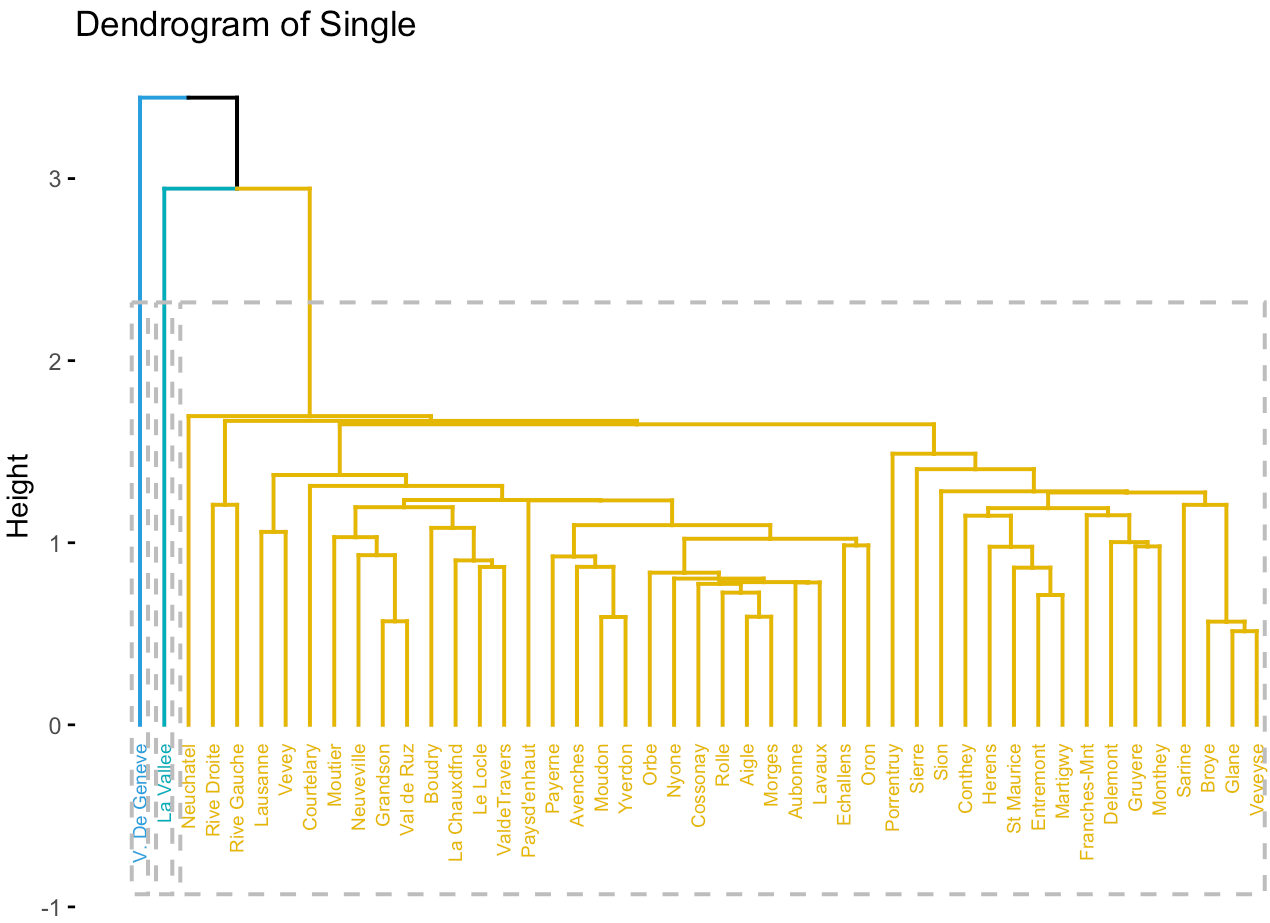

```
fviz_dend(res.hc2, k = 3, # Cut in three groups
main = "Dendrogram of Average",
cex = 0.5, # label size
k_colors = c("#2E9FDF", "#00AFBB", "#E7B800"),
color_labels_by_k = TRUE, # color labels by groups
rect = TRUE # Add rectangle around groups
)
```




```
fviz_dend(res.hc3, k = 3, # Cut in three groups
main = "Dendrogram of Complete",
cex = 0.5, # label size
k_colors = c("#2E9FDF", "#00AFBB", "#E7B800"),
color_labels_by_k = TRUE, # color labels by groups
rect = TRUE # Add rectangle around groups
)
```



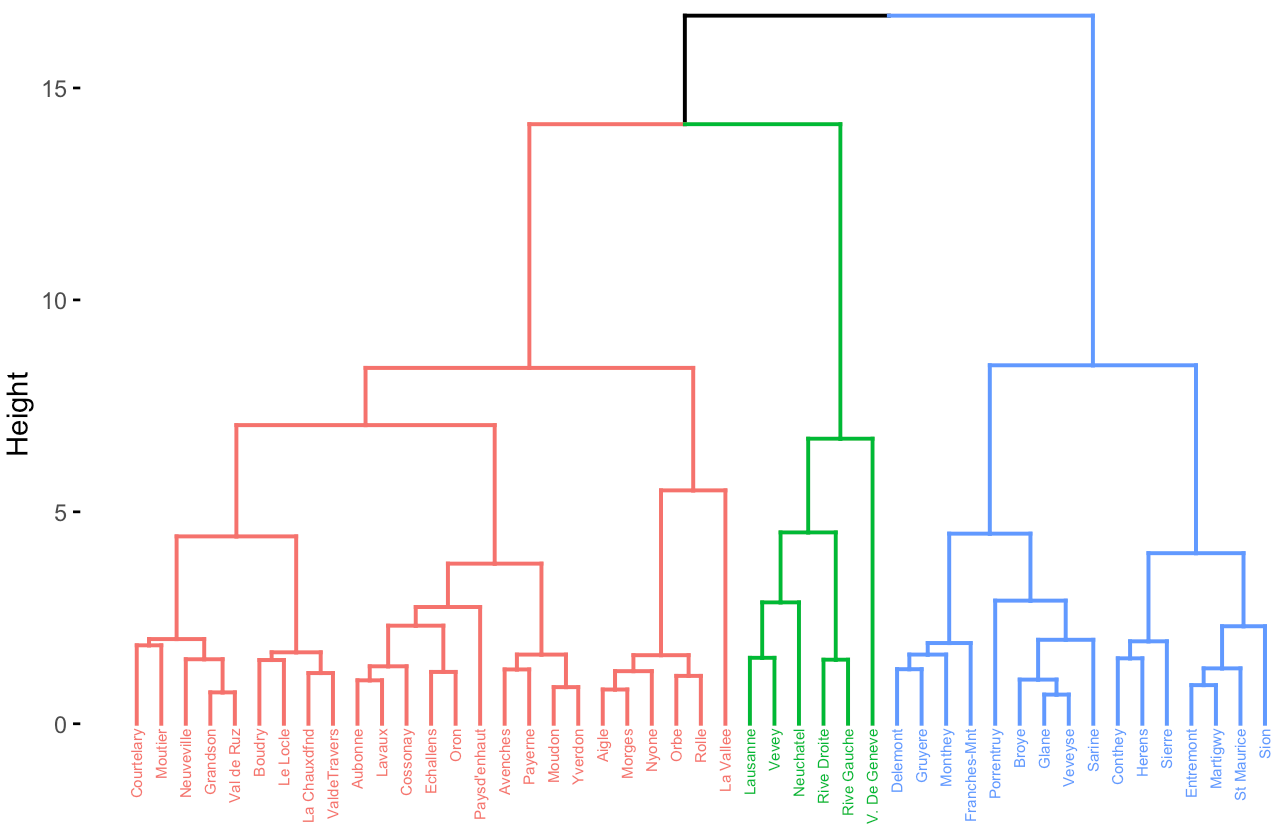

```
fviz_dend(res.hc4, k = 3, # Cut in three groups
main = "Dendrogram of Single",
cex = 0.5, # label size
k_colors = c("#2E9FDF", "#00AFBB", "#E7B800"),
color_labels_by_k = TRUE, # color labels by groups
rect = TRUE # Add rectangle around groups
)
```



Agnes - Agglomerative Nesting Clustering

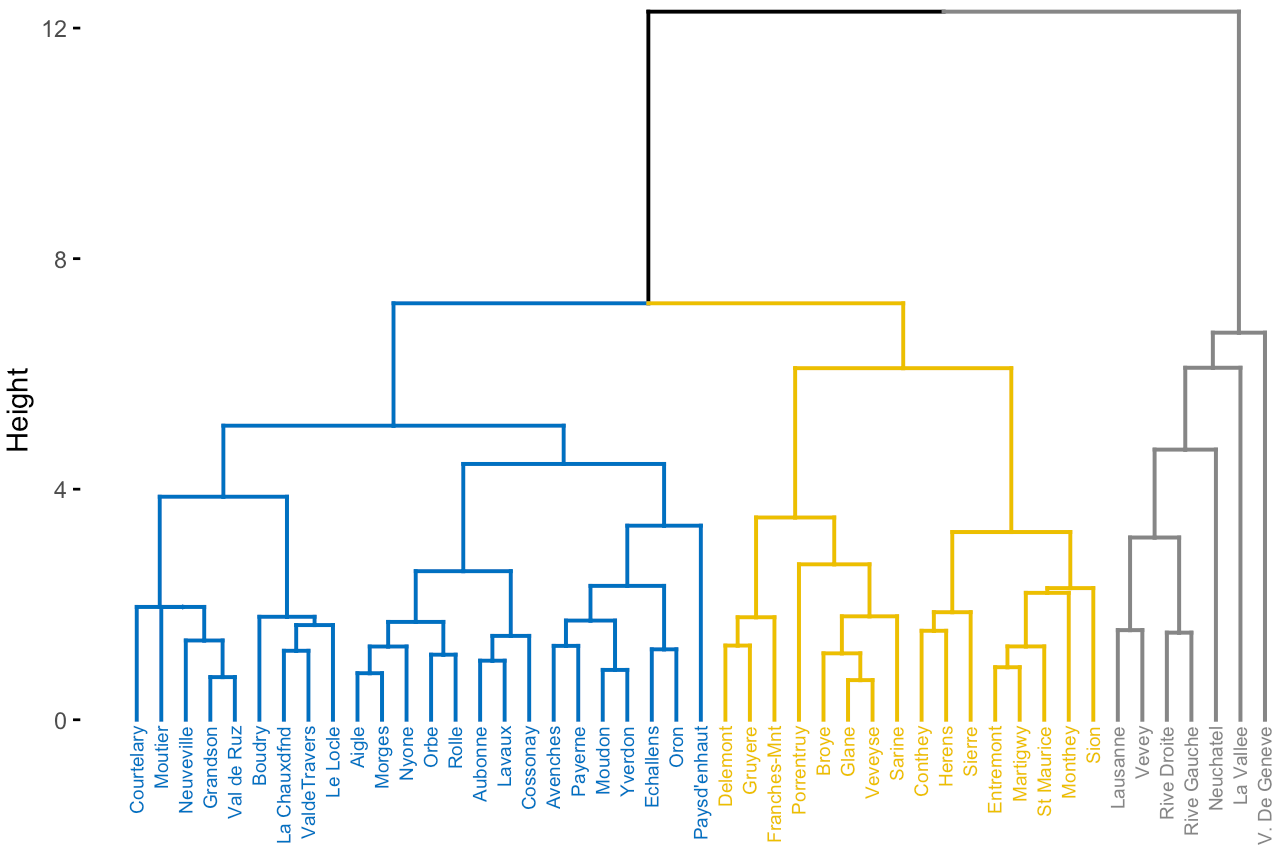
```
library("cluster")
## # Agglomerative Nesting (Hierarchical Clustering)
res.agnes <- agnes(x = df, # data matrix
stand = TRUE, # Standardize the data
metric = "euclidean", # metric for distance matrix
method = "ward" # Linkage method
)
```

```
fviz_dend(res.agnes, cex = 0.4, k = 3, main = "Cluster Dendrogram for AGNES")
```



Divisive Analysis Clustering

```
# Compute diana()
library(cluster)
res.diana <- diana(df, stand = TRUE)
# Plot the dendrogram
library(factoextra)
fviz_dend(main = "Cluster Dendrogram for Divisive Analysis (DIANA)", res.diana, cex = 0.
5,
k = 3, # Cut in four groups
palette = "jco" # Color palette
)
```



Comparing the linkage methods by verifying the cluster tree

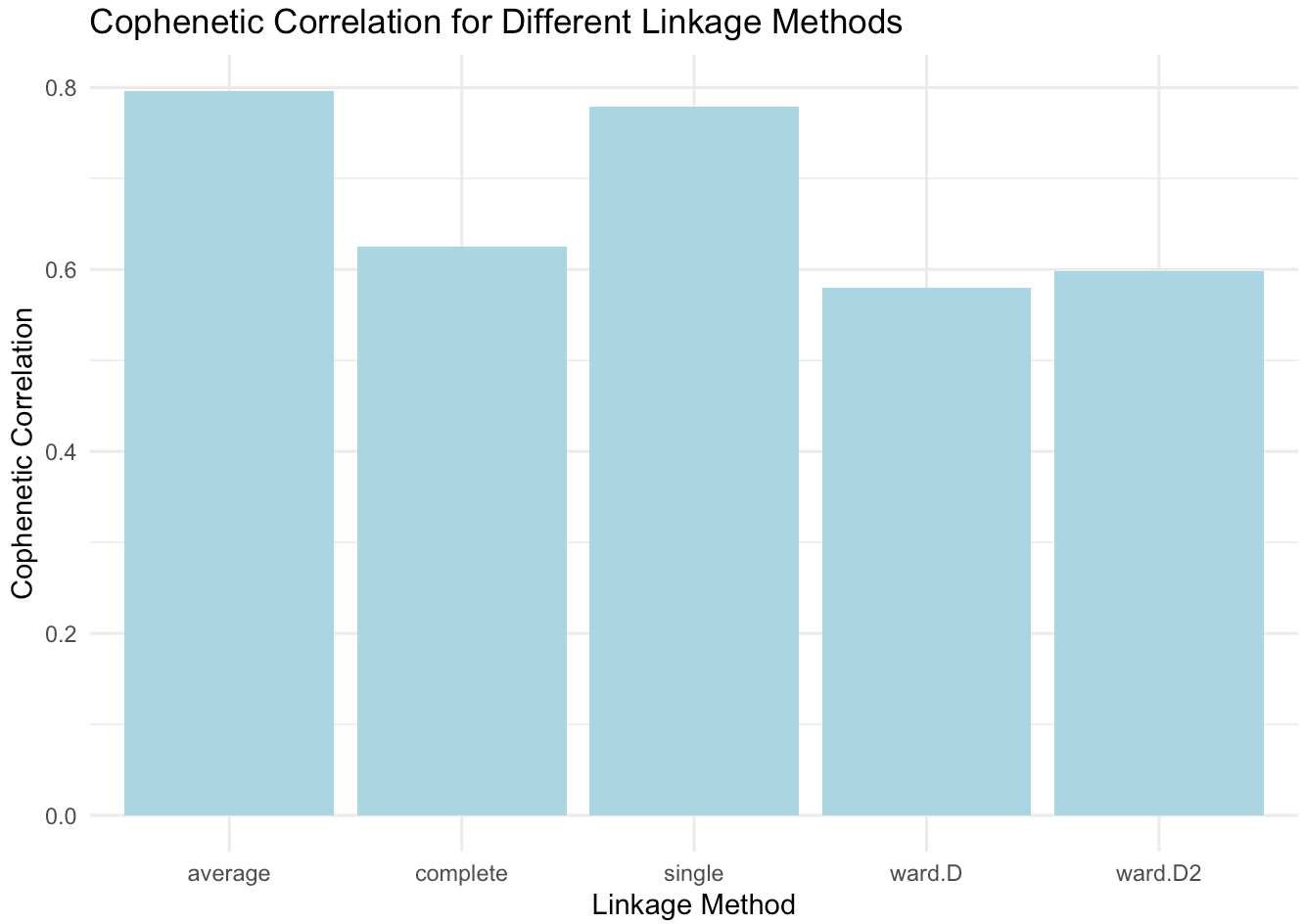
One way to measure how well the cluster tree generated by the hclust() function reflects your data is to compute the correlation between the cophenetic distances and the original distance data generated by the dist() function.

If the clustering is valid, the linking of objects in the cluster tree should have a strong correlation with the distances between objects in the original distance matrix.

```
# List of linkage methods to compare
linkage_methods <- c("ward.D", "single", "complete", "average", "ward.D2")

# Compute cophenetic correlation coefficients
cophenetic_results <- data.frame(
  Method = linkage_methods,
  Cophenetic_Correlation = sapply(linkage_methods, function(method) {
    # Perform hierarchical clustering
    hc <- hclust(dist(df), method = method)
    # Compute cophenetic correlation
    cor(dist(df), cophenetic(hc))
  })
)

# View results
print(cophenetic_results)
```

```
##            Method Cophenetic_Correlation
## ward.D     ward.D              0.5799125
## single     single              0.7791025
## complete complete              0.6247571
## average   average              0.7963038
## ward.D2   ward.D2              0.5981558
```

```
# Plot cophenetic correlation coefficients
ggplot(cophenetic_results, aes(x = Method, y = Cophenetic_Correlation)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  labs(
    title = "Cophenetic Correlation for Different Linkage Methods",
    x = "Linkage Method",
    y = "Cophenetic Correlation"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text())  # Rotate x-axis labels
```



Insights from the correlation

1. **Ward.D2**: Correlation = **0.598**

- Moderate correlation, indicating some distortion in preserving the original pairwise distances.

2. **Ward.D**: Correlation = **0.580**

- Slightly lower correlation than Ward.D2, suggesting more distortion in preserving the original distances.

3. Average: Correlation = 0.796

- High correlation, indicating good preservation of the pairwise distances.

4. Complete: Correlation = 0.625

- Moderate correlation, showing reasonable preservation of the structure but some distortion.

5. Single: Correlation = 0.779

- High correlation, indicating good preservation of the original structure, similar to Average linkage.

Summary:

- **High Correlation**: Average (0.80) and Single (0.78) preserve the structure well.
- **Moderate Correlation**: Complete (0.63) and Ward.D2 (0.60) show moderate preservation.
- **Lower Correlation**: Ward.D (0.58) introduces more distortion compared to other methods.
- Based on these insights, we decided to move ahead with the **Average linkage** method as it offers the best balance between preserving pairwise distances and generating meaningful clusters.

Average linkage

```
# Cut tree into 3 groups
grp <- cutree(res.hc2, k = 3)
head(grp, n = 3)
```

```
##   Courtelary     Delemont Franches-Mnt
##            1            1            1
```

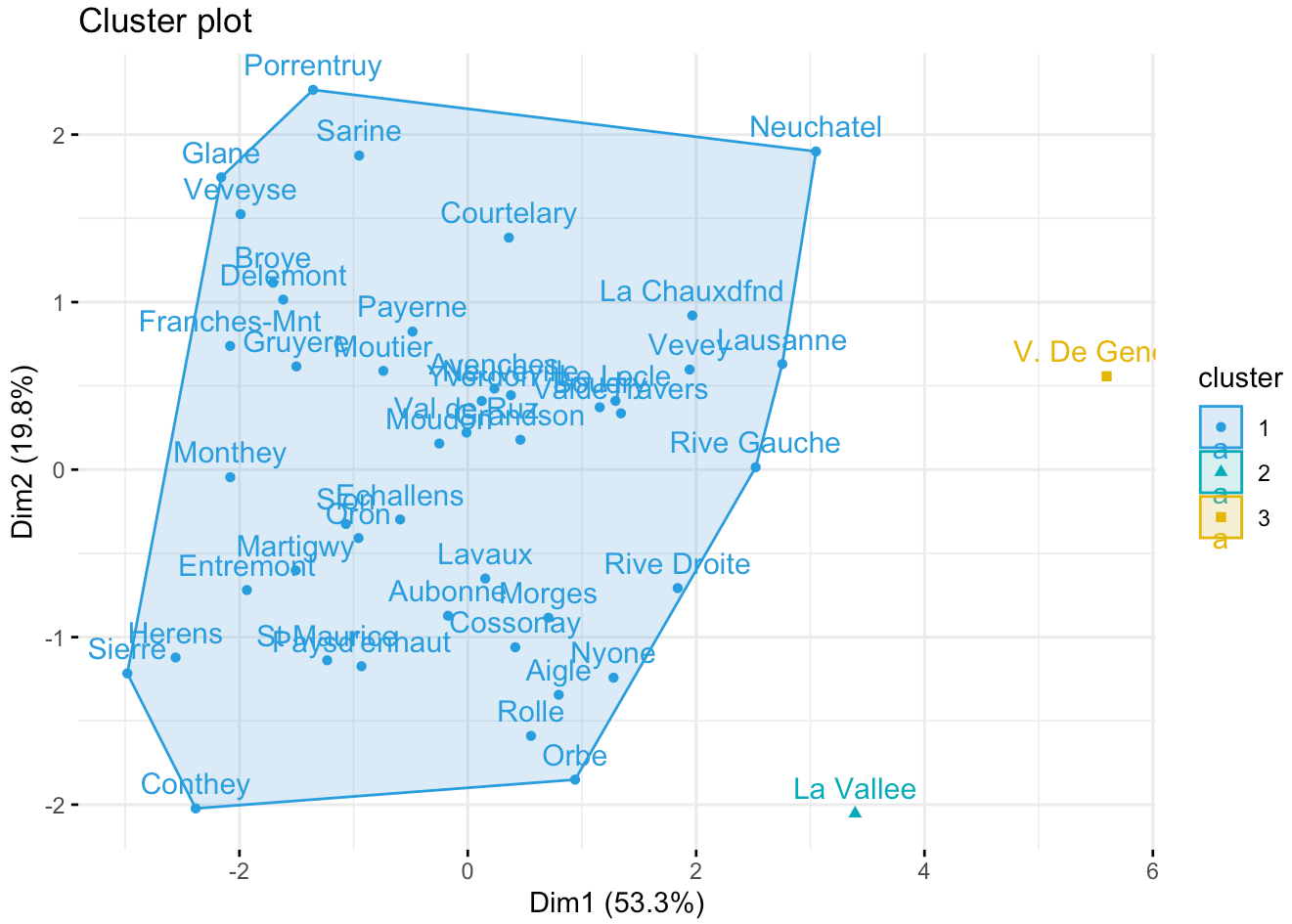
```
# Number of members in each cluster
table(grp)
```

```
## grp
##  1  2  3
## 45  1  1
```

```
# Get the names for the members of cluster 1
rownames(df)[grp == 1]
```

```
##  [1] "Courtelary"   "Delemont"     "Franches-Mnt" "Moutier"      "Neuveville"
##  [6] "Porrentruy"   "Broye"        "Glane"        "Gruyere"      "Sarine"
## [11] "Veveyse"      "Aigle"        "Aubonne"      "Avenches"     "Cossonay"
## [16] "Echallens"    "Grandson"     "Lausanne"     "Lavaux"       "Morges"
## [21] "Moudon"       "Nyone"        "Orbe"         "Oron"         "Payerne"
## [26] "Paysd'enhaut" "Rolle"        "Vevey"        "Yverdon"      "Conthey"
## [31] "Entremont"    "Herens"       "Martigwy"     "Monthey"      "St Maurice"
## [36] "Sierre"       "Sion"         "Boudry"       "La Chauxdfnd" "Le Locle"
## [41] "Neuchatel"    "Val de Ruz"   "ValdeTravers" "Rive Droite"  "Rive Gauche"
```

```
## ----cluster-plot, fig.width=6, fig.height=6----------------------------
fviz_cluster(list(data = df, cluster = grp),
palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
ellipse.type = "convex", # Concentration ellipse
show.clust.cent = FALSE, ggtheme = theme_minimal())
```



Comparison of the linkage methods using *their dendrograms*

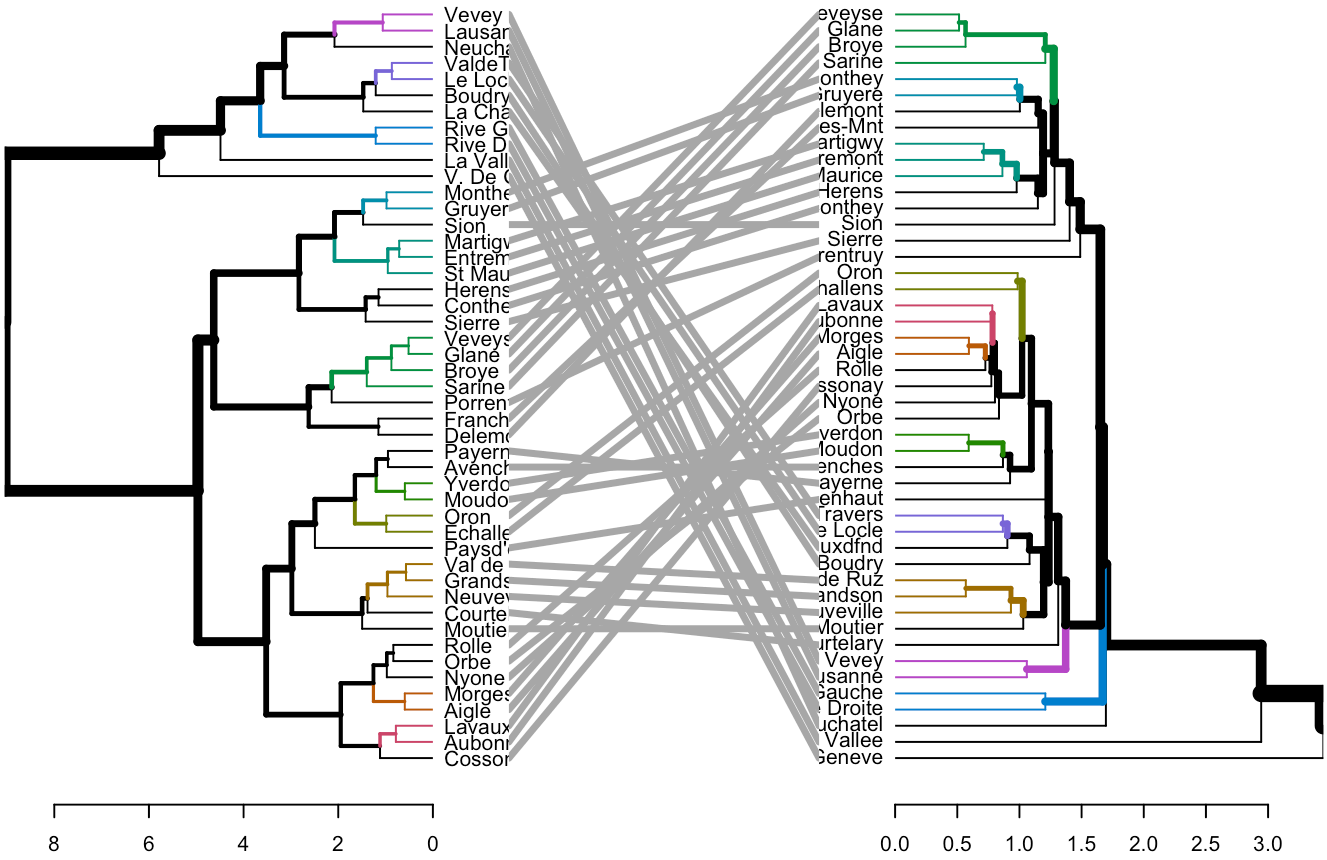
Note that, conclusions about the proximity of two objects can be drawn only based on the height where branches containing those two objects first are fused. We cannot use the proximity of two objects along the horizontal axis as a criteria of their similarity.

Comparing and Visualizing Dendrograms

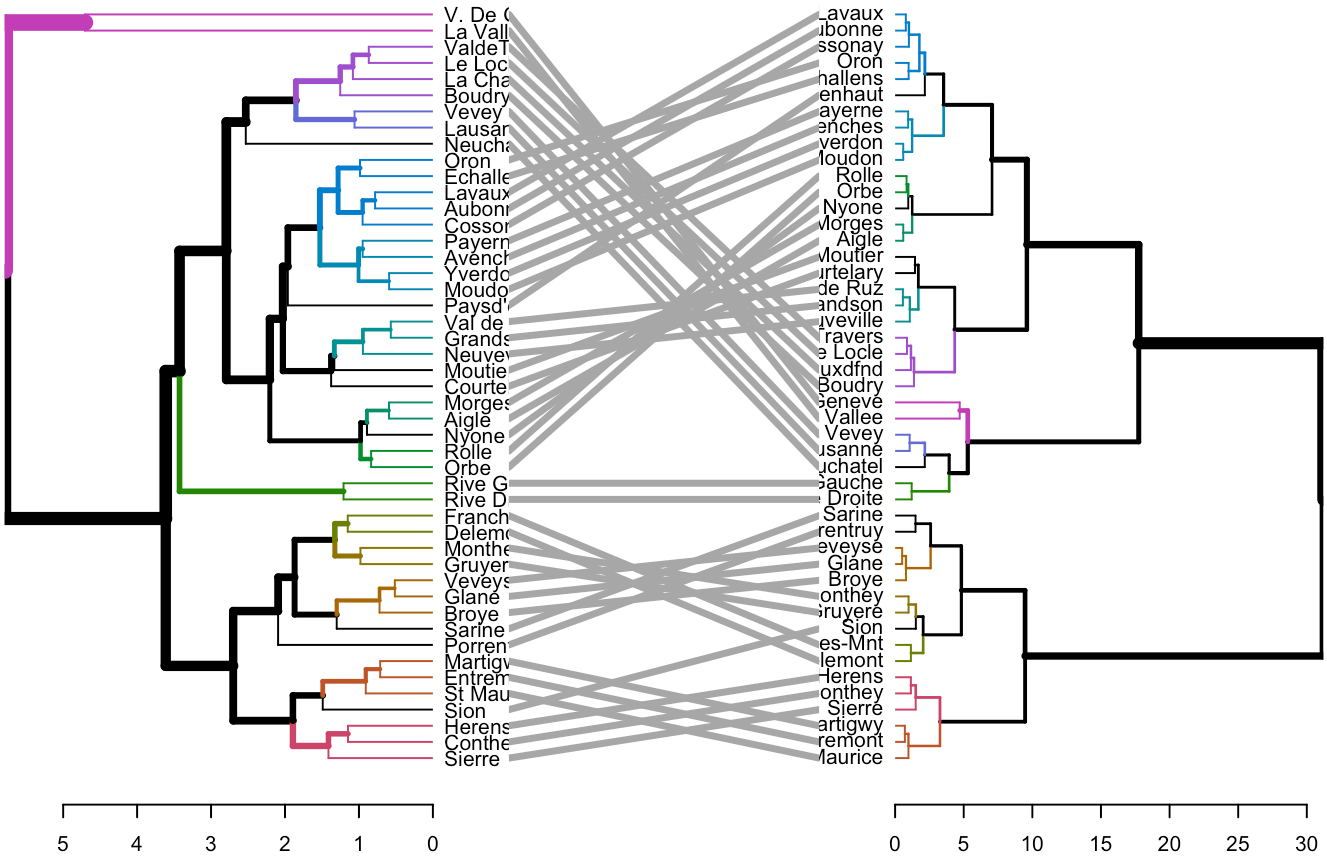
```
library(dendextend)

# Compute 2 hierarchical clusterings
hc1 <- hclust(res.dist, method = "average")
hc2 <- hclust(res.dist, method = "ward.D")

# Create two dendrograms
dend1 <- as.dendrogram (hc1)
dend2 <- as.dendrogram (hc2)

# Create a list to hold dendrograms
dend_list <- dendlist(dend1, dend2)
```

```
# Compute 2 hierarchical clusterings
hcco <- hclust(res.dist, method = "complete")
hcsin <- hclust(res.dist, method = "single")

# Create two dendrograms
dendco <- as.dendrogram (hcco)
dendsin <- as.dendrogram (hcsin)

# Create a list to hold dendrograms *complte and single
dend_listcs <- dendlist(dendco, dendsin)
```

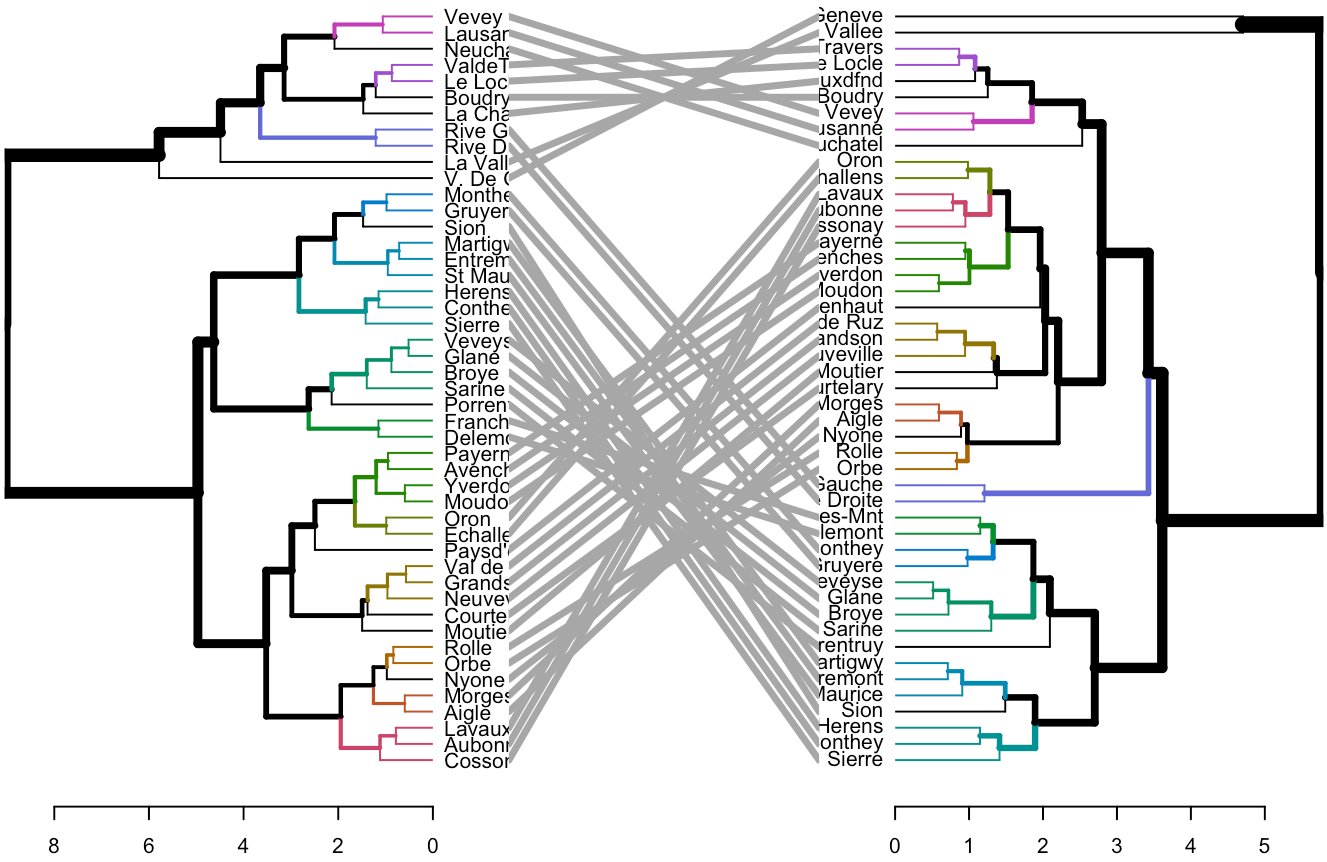
```
tanglegram(dendco, dendsin,
highlight_distinct_edges = FALSE, # Turn-off dashed lines
common_subtrees_color_lines = FALSE, # Turn-off line colors
common_subtrees_color_branches = TRUE, # Color common branches
main = paste("entanglement of single vs complete =", round(entanglement(dend_listcs),
2))
)
```



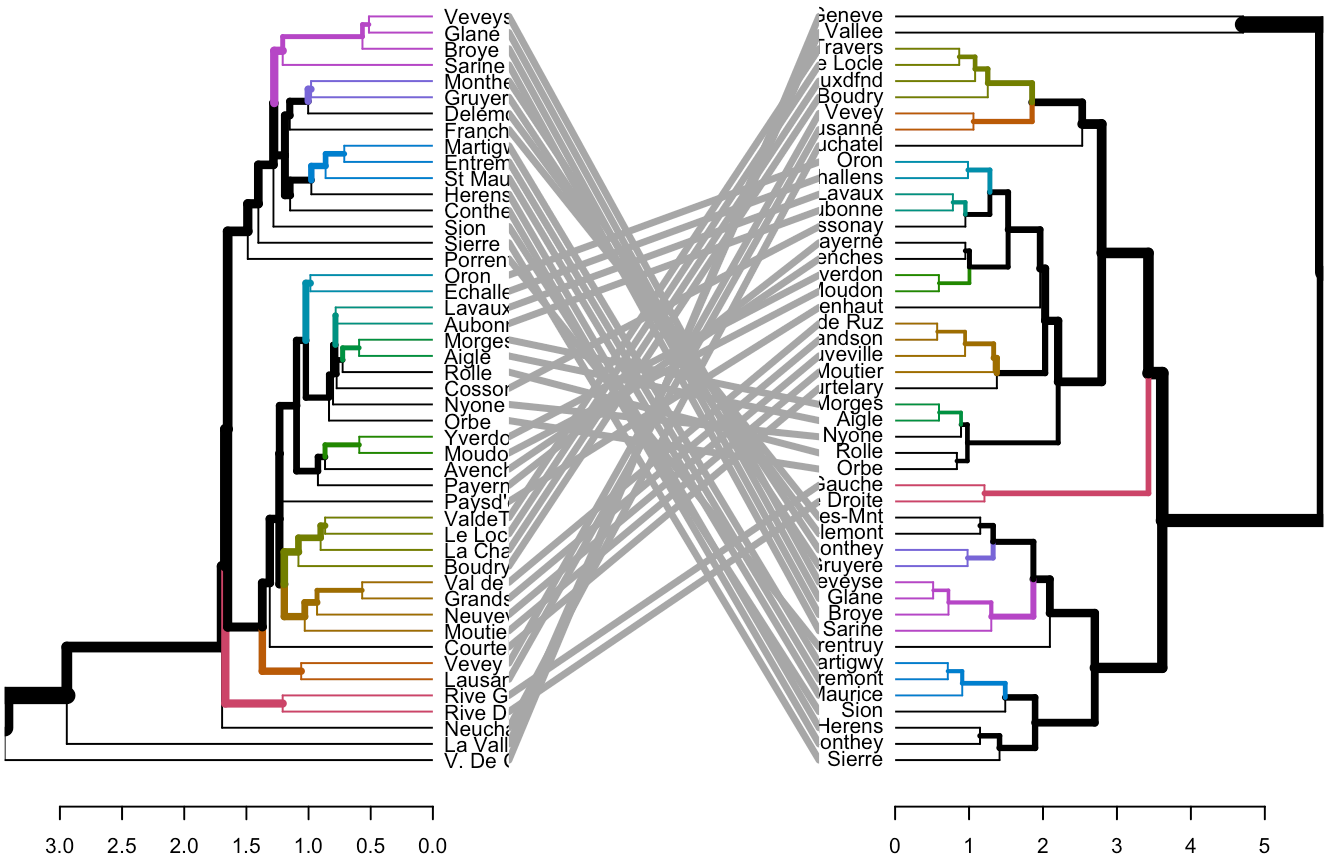

```
tanglegram(dend1, dend2,
highlight_distinct_edges = FALSE, # Turn-off dashed lines
common_subtrees_color_lines = FALSE, # Turn-off line colors
common_subtrees_color_branches = TRUE, # Color common branches
main = paste("entanglement of average vs ward.D=", round(entanglement(dend_list), 2))
)
```




```
# Create a list to hold dendrograms *complte and average
dend_listca <- dendlist(dendco, dend1)
tanglegram(dendco, dend1,
highlight_distinct_edges = FALSE, # Turn-off dashed lines
common_subtrees_color_lines = FALSE, # Turn-off line colors
common_subtrees_color_branches = TRUE, # Color common branches
main = paste("entanglement of average vs complete=", round(entanglement(dend_listca),
2))
)
```




```
dend_listsa <- dendlist(dendsin, dend1)
tanglegram(dendsin, dend1,
highlight_distinct_edges = FALSE, # Turn-off dashed lines
common_subtrees_color_lines = FALSE, # Turn-off line colors
common_subtrees_color_branches = TRUE, # Color common branches
main = paste("entanglement of average vs single=", round(entanglement(dend_listsa), 2))
)
```



The quality of the alignment of the two trees can be measured using the function entanglement(). Entanglement is a measure between 1 (full entanglement) and 0 (no entanglement).

A lower entanglement coefficient corresponds to a good alignment.

Average and Ward.D has the lowest entanglement, making it most similar to one other.

Correlation matrix between a list of dendrogams

The function cor.dendlist() is used to compute “Baker” or “Cophenetic” correlation matrix between a list of trees. The value can range between -1 to 1. With near 0 values meaning that the two trees are not statistically similar.

```
# Cophenetic correlation matrix
cor.dendlist(dend_list, method = "cophenetic")
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.6345401
## [2,] 0.6345401 1.0000000
```

```
# Baker correlation matrix
cor.dendlist(dend_list, method = "baker")
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.8226455
## [2,] 0.8226455 1.0000000
```

```
# Cophenetic correlation coefficient
cor_cophenetic(dend1, dend2)
```

```
## [1] 0.6345401
```

```
# Baker correlation coefficient
cor_bakers_gamma(dend1, dend2)
```

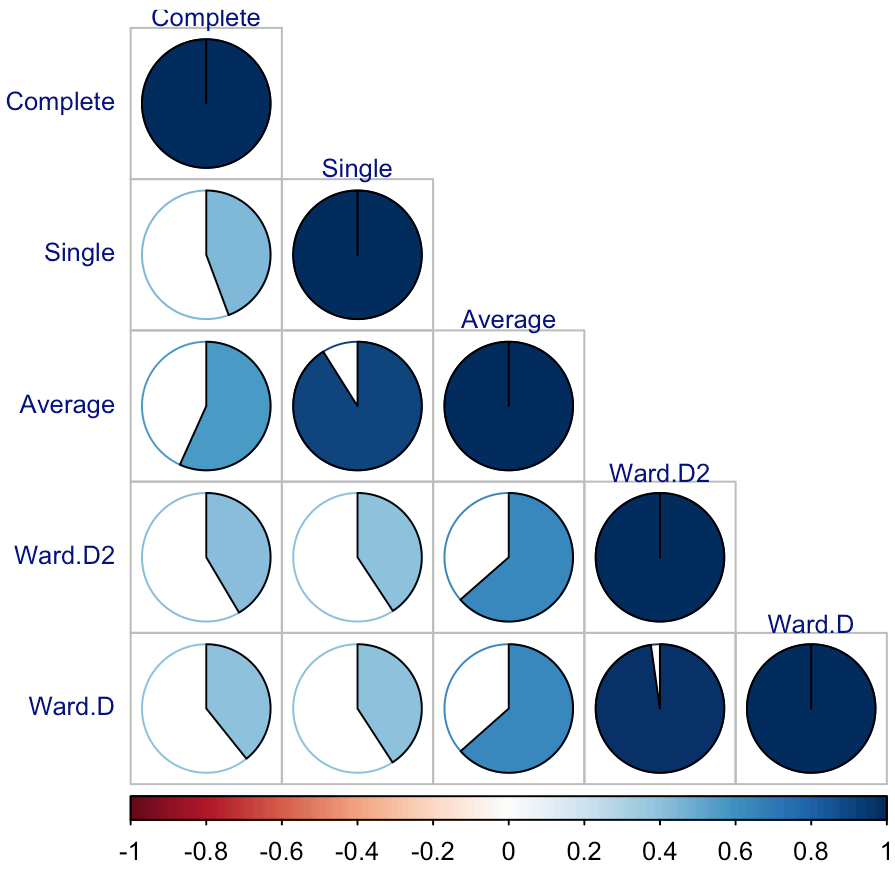
```
## [1] 0.8226455
```

It's also possible to compare simultaneously multiple dendrograms. A chaining operator %>% is used to run multiple function at the same time. It's useful for simplifying the code:

```
# Create multiple dendrograms by chaining
dend1 <- df %>% dist %>% hclust("complete") %>% as.dendrogram
dend2 <- df %>% dist %>% hclust("single") %>% as.dendrogram
dend3 <- df %>% dist %>% hclust("average") %>% as.dendrogram
dend4 <- df %>% dist %>% hclust("ward.D2") %>% as.dendrogram
dend5 <- df %>% dist %>% hclust("ward.D") %>% as.dendrogram
# Compute correlation matrix
dend_list <- dendlist("Complete" = dend1, "Single" = dend2,
                      "Average" = dend3, "Ward.D2" = dend4, "Ward.D" = dend5)
cors <- cor.dendlist(dend_list)
# Print correlation matrix
round(cors, 2)
```

```
##          Complete Single Average Ward.D2 Ward.D
## Complete     1.00   0.44    0.57    0.41   0.39
## Single       0.44   1.00    0.91    0.41   0.41
## Average      0.57   0.91    1.00    0.64   0.63
## Ward.D2      0.41   0.41    0.64    1.00   0.98
## Ward.D       0.39   0.41    0.63    0.98   1.00
```

```
# Visualize the correlation matrix using corrplot package
library(corrplot)
corrplot(cors, "pie", "lower", tl.col = "navyblue", tl.srt = 0, title = "Pearson Correla
tion between the dendrograms",mar = c(0, 0, 2,0), tl.cex = 0.8)
```



Clustering Methods Correlation Insights - Pearson

The graph presents the pairwise correlation between different clustering methods. Here's a summary of the key findings:

1. **Ward.D vs Ward.D2 (0.98)**: These methods produce nearly identical clustering results, showing high similarity.
2. **Single vs Average (0.91)**: These methods are closely related, producing similar cluster compositions, though they use different approaches (Single based on minimum distance, Average based on the average distance).
3. **Average vs Complete (0.57)**: Moderate correlation, suggesting some similarity in clustering, but noticeable differences in how they treat distant points.
4. **Complete vs Single (0.44)**: Low correlation, indicating these methods produce very different clusterings. Complete forms tighter clusters, while Single merges distant points.
5. **Ward.D vs Average (0.63)**: Moderate correlation, showing some overlap, but differences in how clusters are formed based on variance minimization and distance calculation.

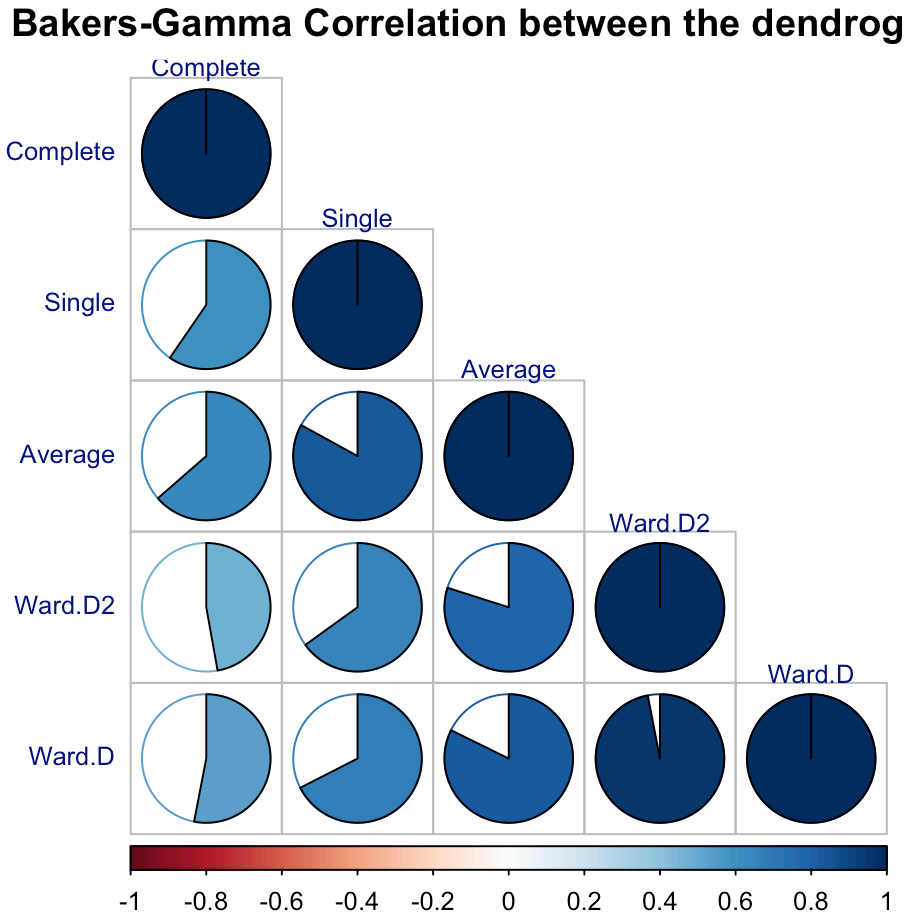
Conclusion:

- **Ward.D** and **Ward.D2** are nearly identical.
- **Single** and **Average** are highly similar.
- **Complete** and **Single** show significant differences in their clustering approaches.

```
cors_bakers <- cor.dendlist(dend_list, method = "baker")
# Print correlation matrix
round(cors_bakers, 2)
```

```
##          Complete Single Average Ward.D2 Ward.D
## Complete     1.00   0.60    0.64    0.47   0.53
## Single       0.60   1.00    0.83    0.65   0.67
## Average      0.64   0.83    1.00    0.80   0.82
## Ward.D2      0.47   0.65    0.80    1.00   0.97
## Ward.D       0.53   0.67    0.82    0.97   1.00
```

```
library(corrplot)
corrplot(cors_bakers, "pie", "lower", tl.col = "navyblue", tl.srt = 0, title = "Bakers-G
amma Correlation between the dendrograms",mar = c(0, 0, 2,0), tl.cex = 0.8)
```



Clustering Methods Correlation Insights - Baker's Gamma

1. **Higher Baker's Gamma Correlations Across All Methods:**

- Baker's Gamma values are generally higher than Pearson correlations, showing that the clustering methods maintain a higher agreement in their ranked pairwise relationships compared to their direct distance mappings.

2. **Complete vs. Average Linkage:**

- Consistent with the Pearson correlation, Complete and Average linkage methods maintain a high Baker's Gamma correlation, further confirming their similarity.

3. **Improved Single Linkage Correlation:**

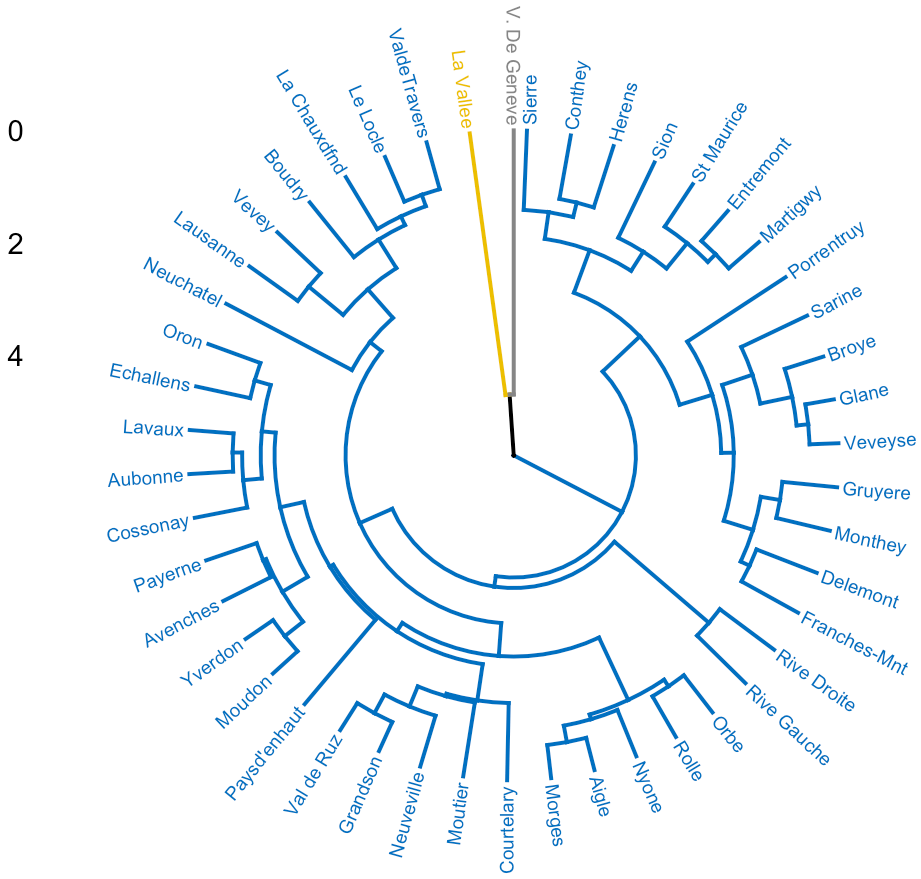
- Baker's Gamma correlation for Single linkage improves compared to Pearson's, indicating that while Single linkage has a distinct approach, it still maintains some consistency in the ranked clustering relationships.

4. **Ward Methods:**

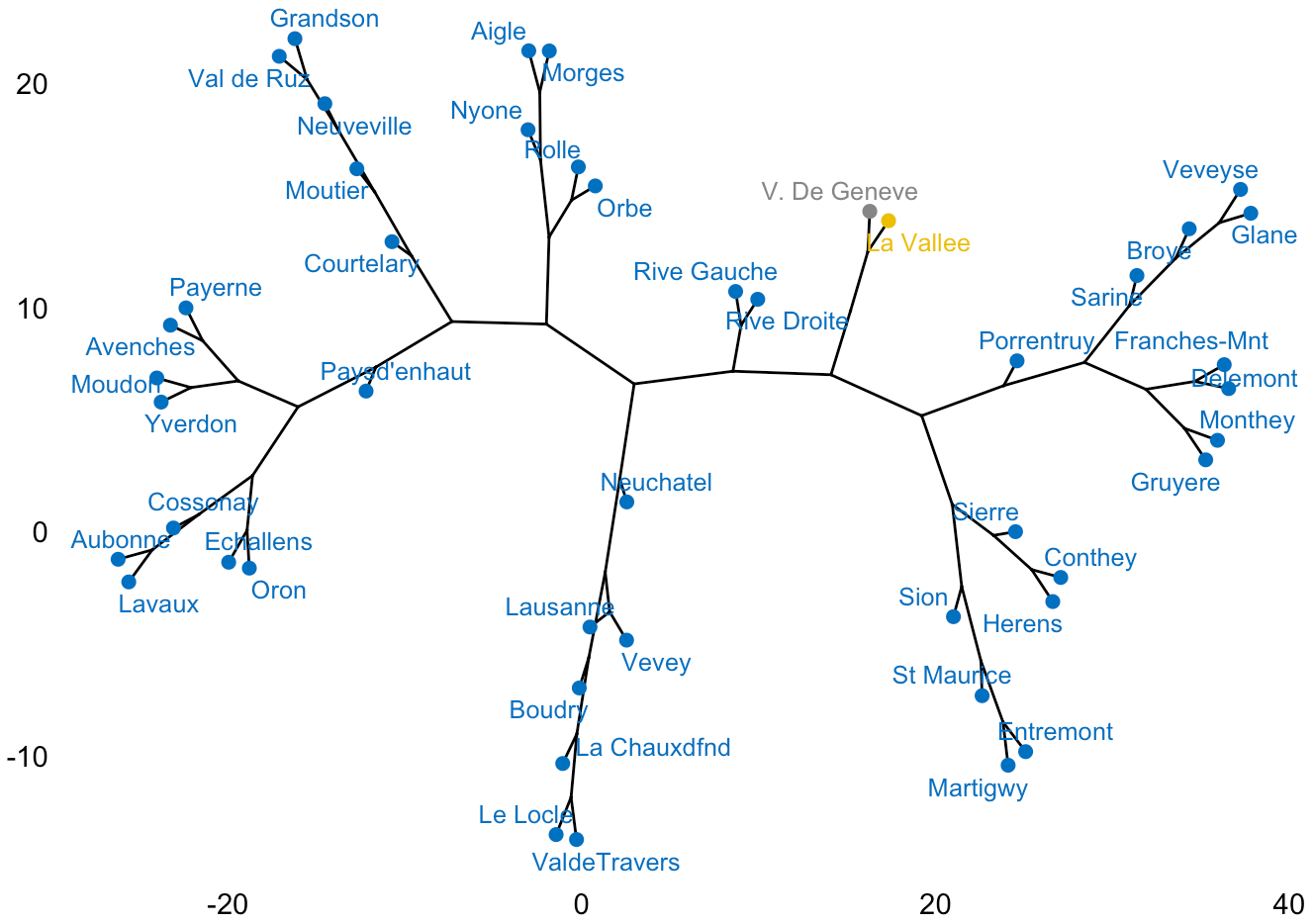
- The correlation between Ward.D and Ward.D2 is near-perfect in both Pearson and Baker's Gamma, reinforcing their equivalency for most practical applications.

Visualizing Dendrograms - Circular

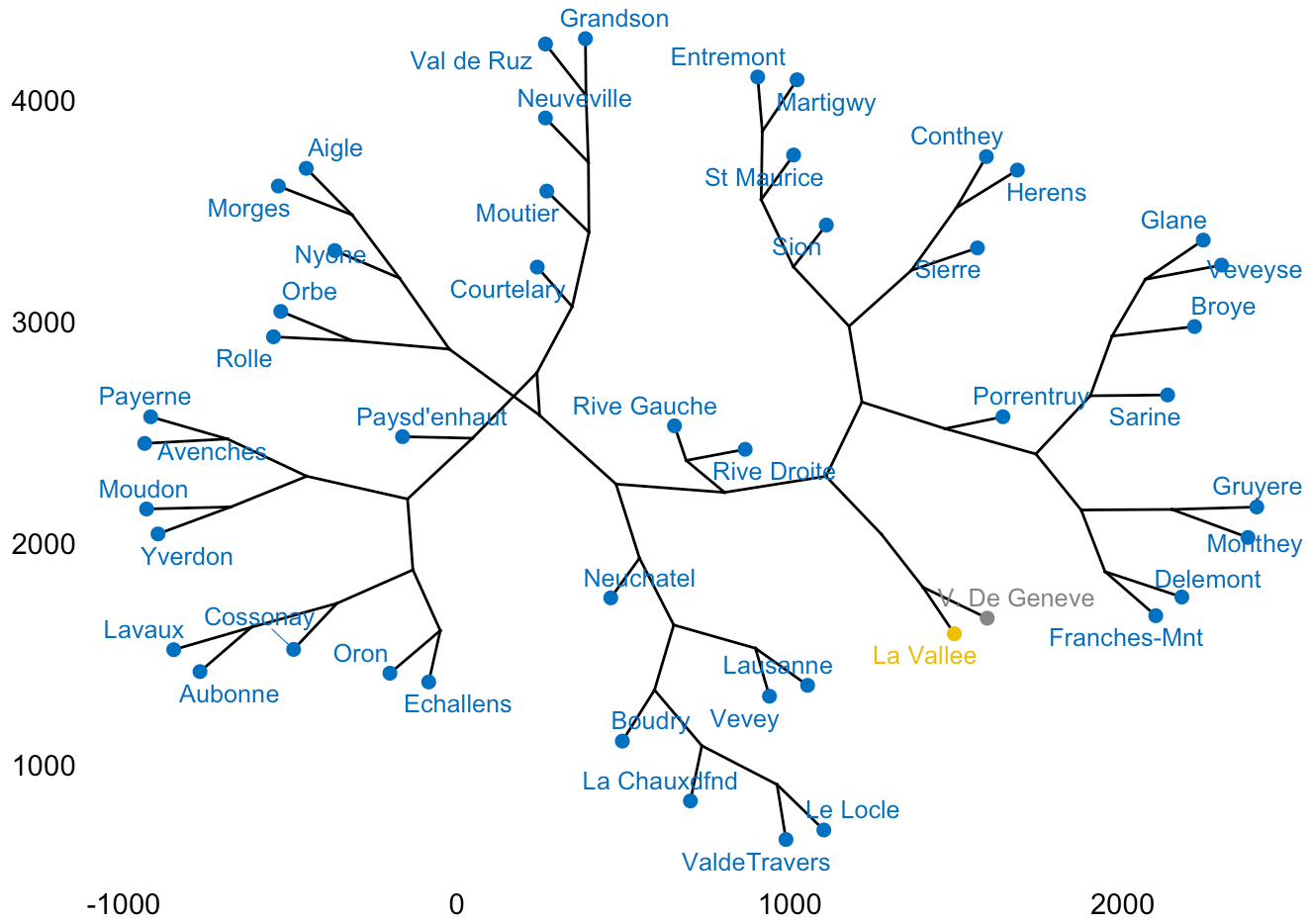
```
fviz_dend(res.hc2, cex = 0.5, k = 3,
k_colors = "jco", type = "circular")
```




```
require("igraph")
fviz_dend(res.hc2, k = 3, k_colors = "jco",
type = "phylogenic", repel = TRUE)
```



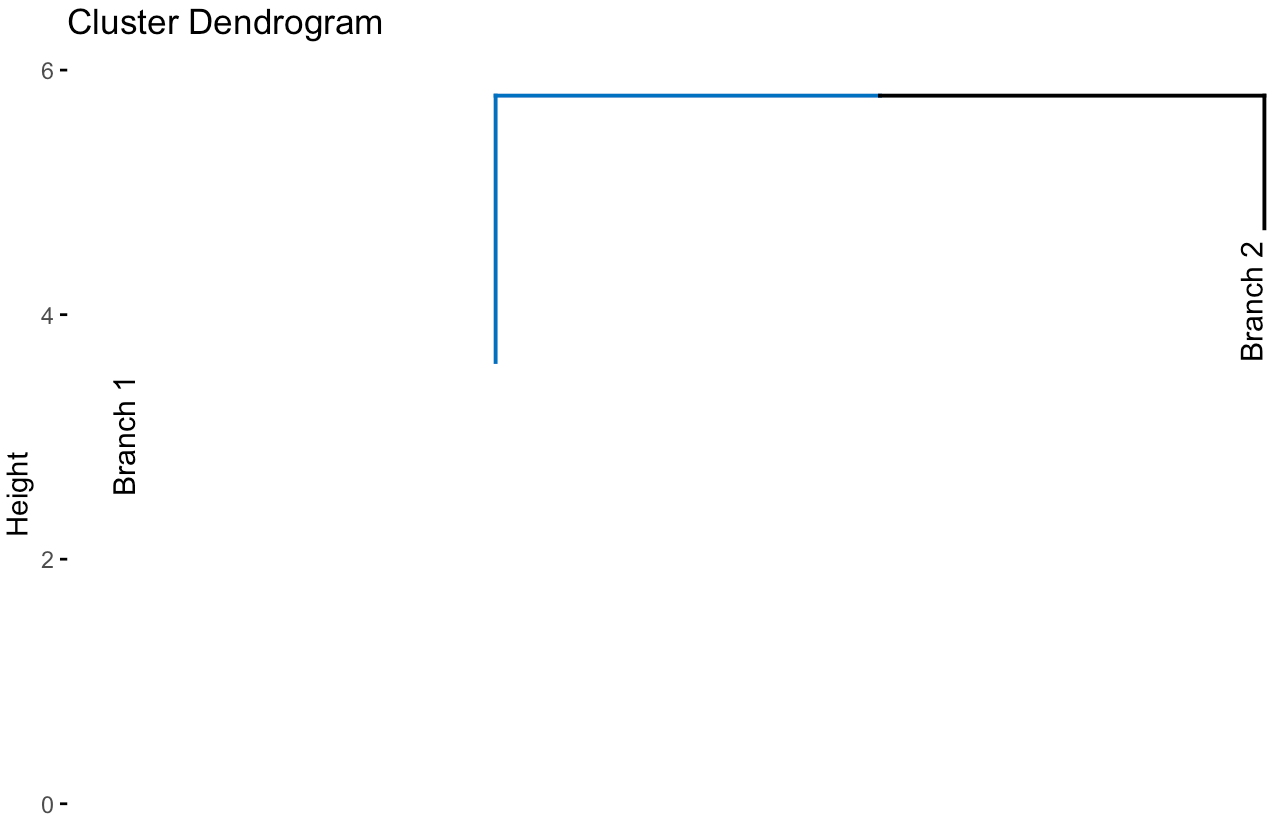

```
require("igraph")
fviz_dend(res.hc2, k = 3, # Cut in three groups
k_colors = "jco",
type = "phylogenic", repel = TRUE,
phylo_layout = "layout.gem")
```



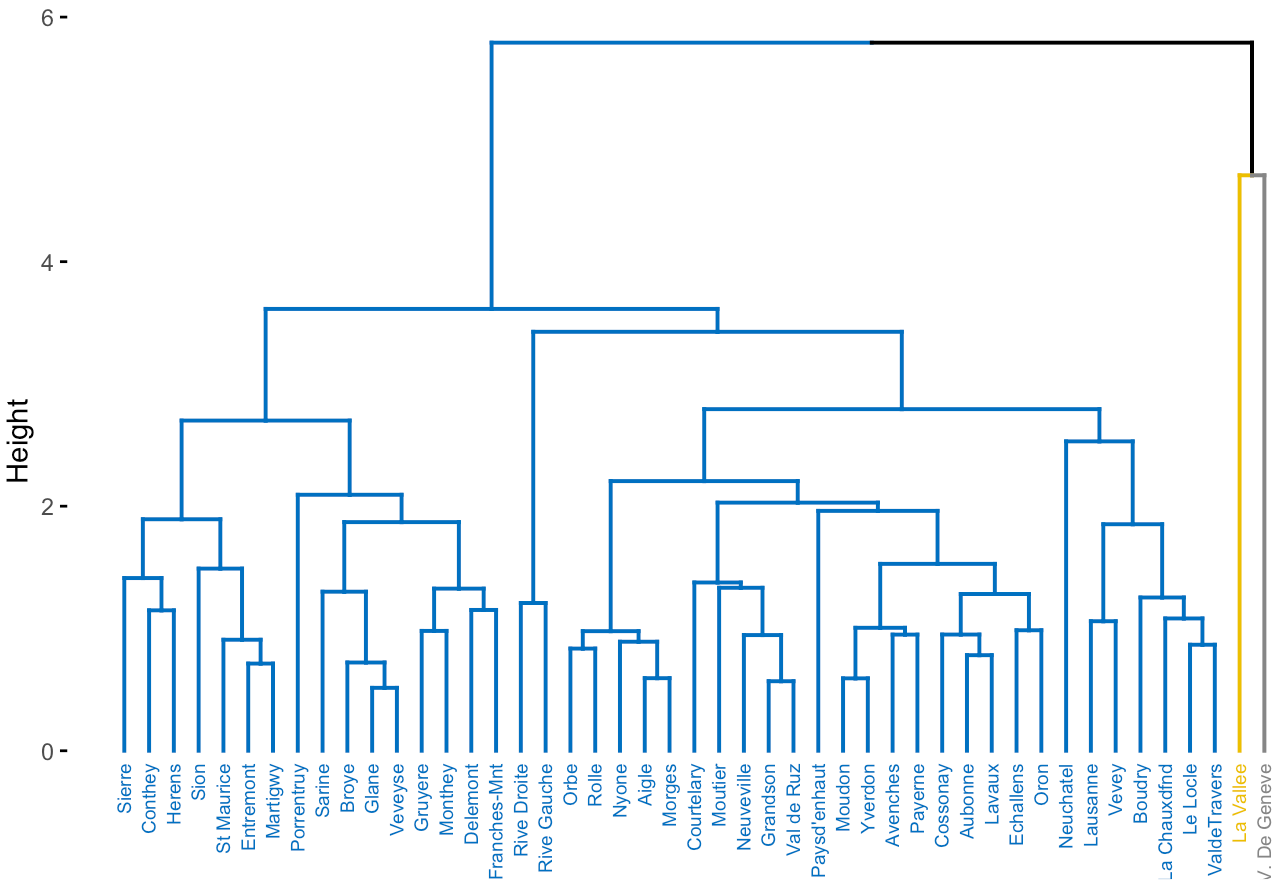
Plotting a sub-tree of dendrograms

```
# Create a plot of the whole dendrogram,
# and extract the dendrogram data
dend_plot <- fviz_dend(res.hc2, k = 3, # Cut in three groups
cex = 0.5, # label size
k_colors = "jco"
)
dend_data <- attr(dend_plot, "dendrogram") # Extract dendrogram data
```

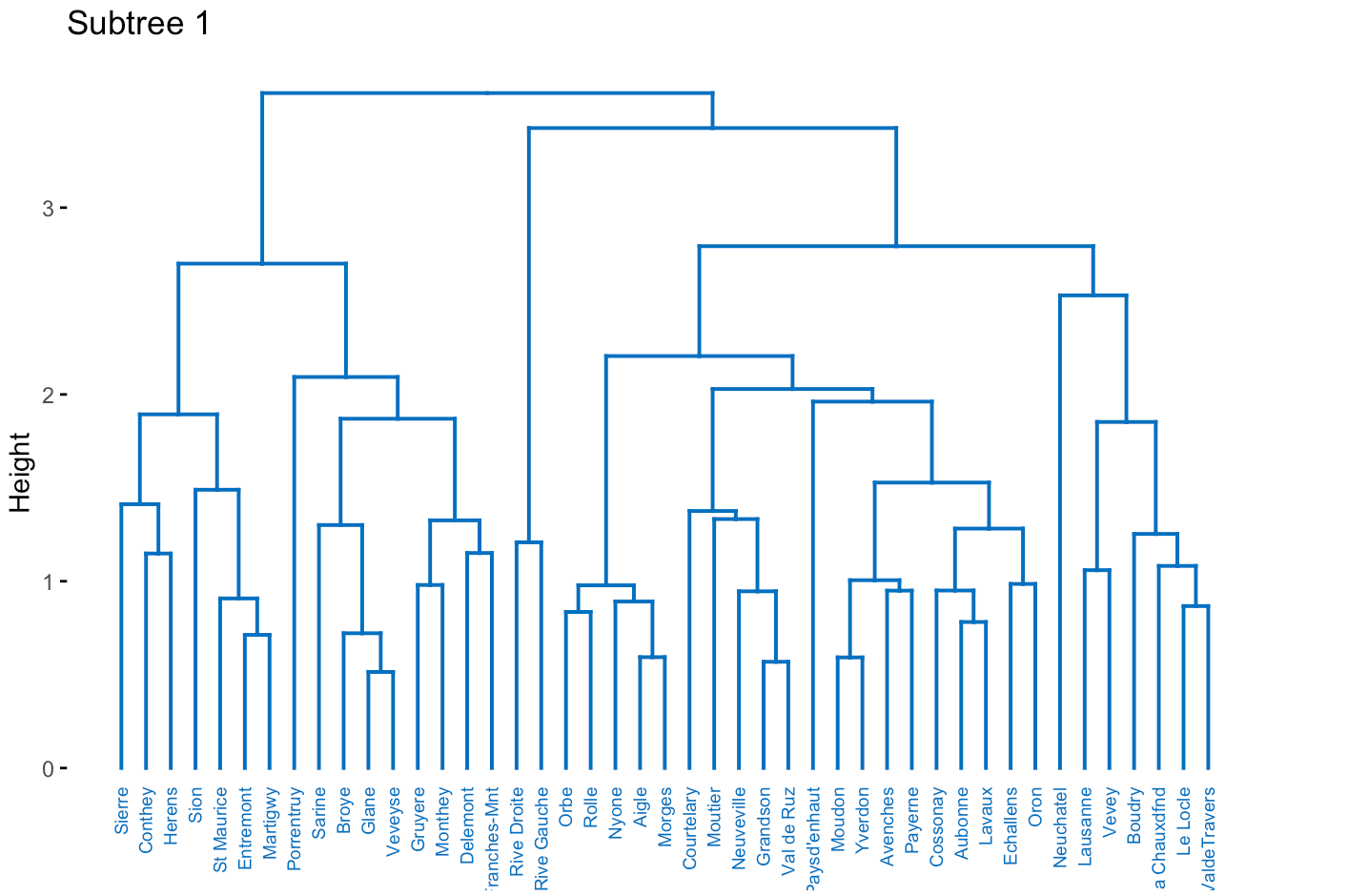
```
# Cut the dendrogram at height h = 10
dend_cuts <- cut(dend_data, h = 10)
# Visualize the truncated version containing
# two branches
fviz_dend(dend_cuts$upper)
```



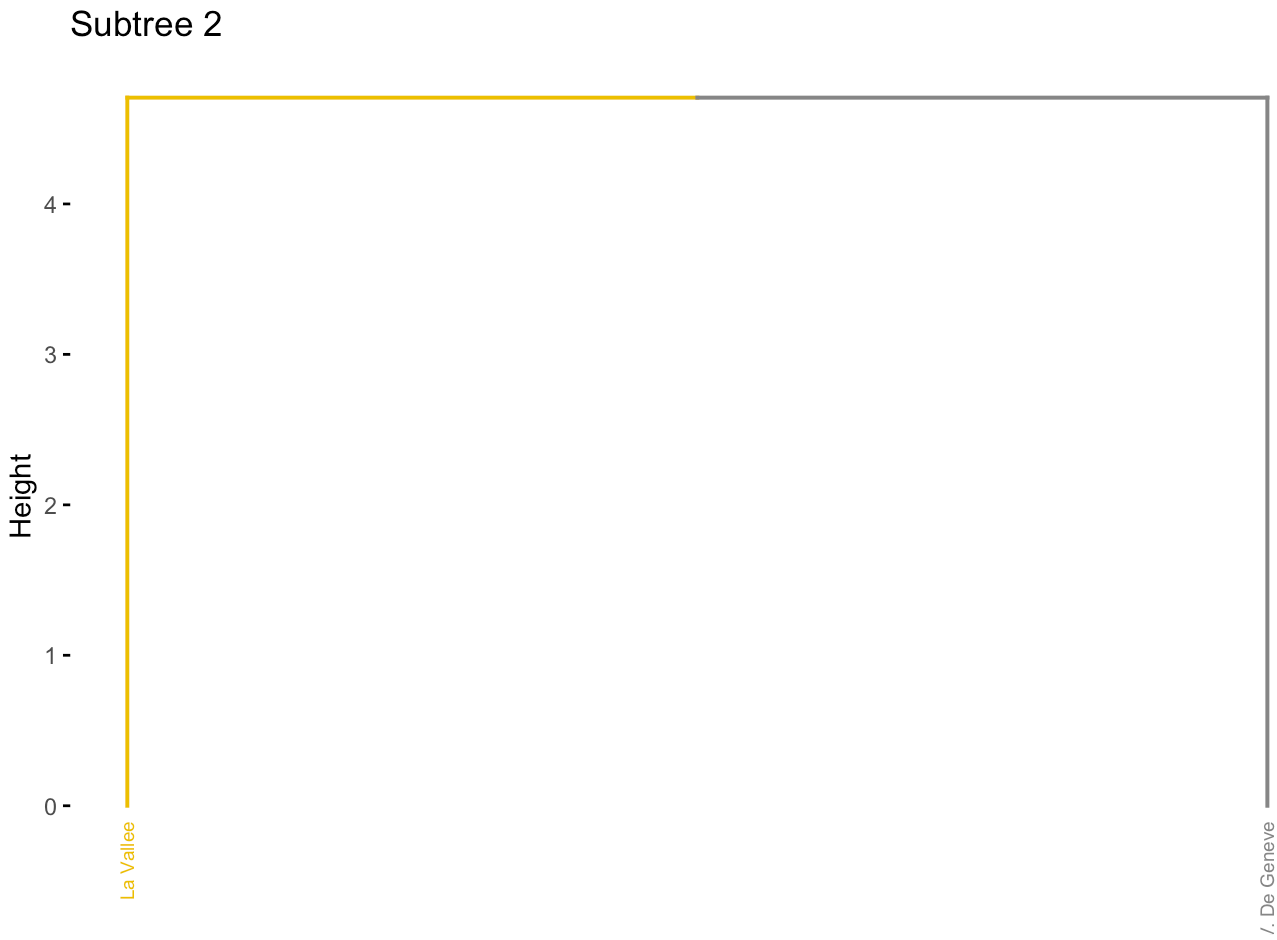

```
# Plot the whole dendrogram
print(dend_plot)
```



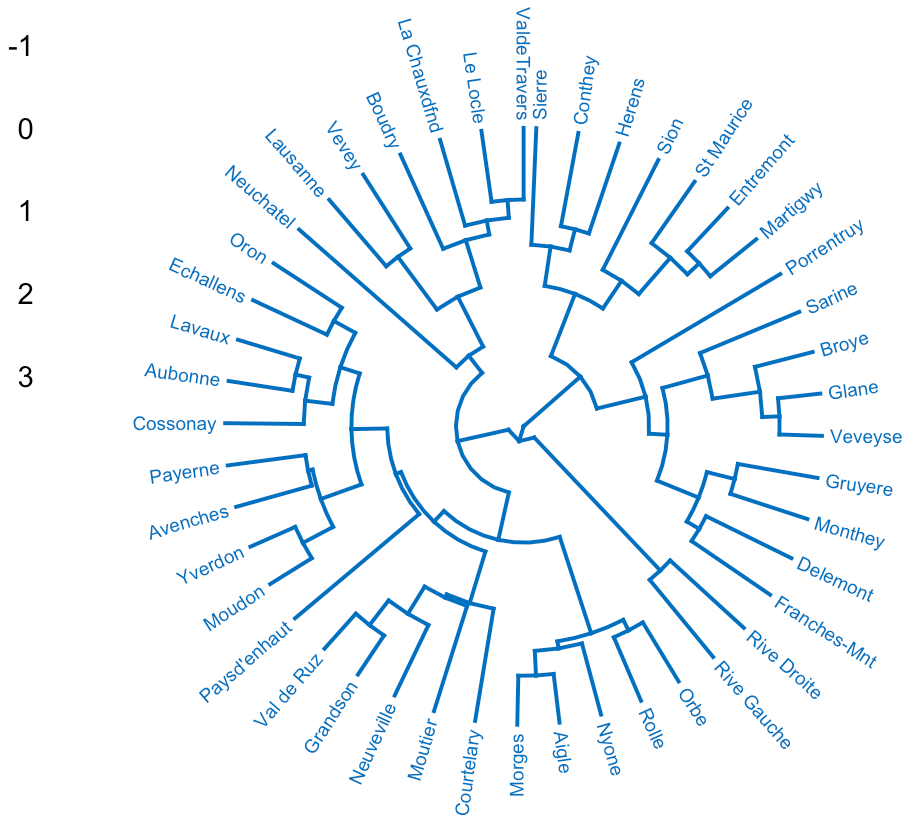

```
# Plot subtree 1
fviz_dend(dend_cuts$lower[[1]], main = "Subtree 1")
```




```
# Plot subtree 2
fviz_dend(dend_cuts$lower[[2]], main = "Subtree 2")
```

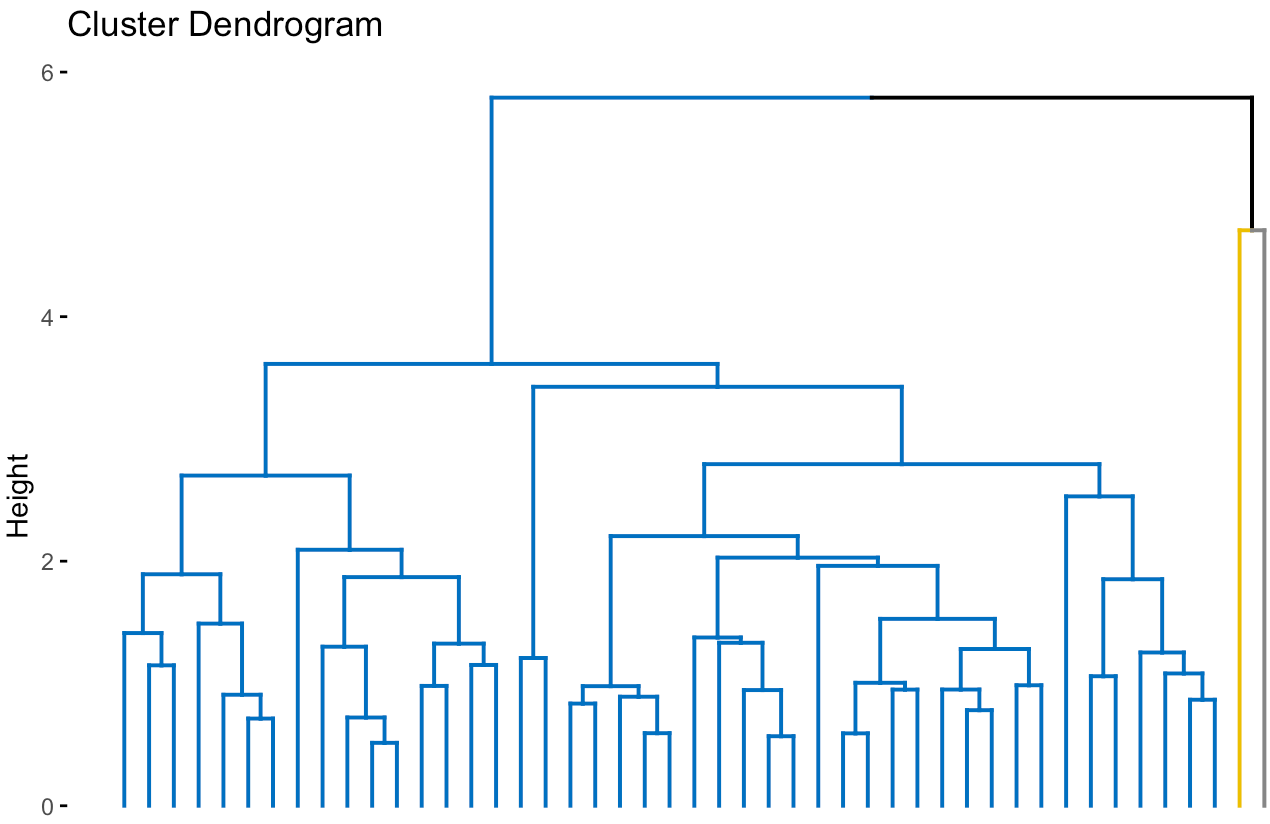



```
# Plot subtree 1
fviz_dend(dend_cuts$lower[[1]], type = "circular", main = "Subtree 1")
```



Cluster Validation Statistics

```
# Hierarchical clustering
hc.res <- eclust(df, "hclust", k = 3, hc_metric = "euclidean",
hc_method = "average", graph = FALSE)
# Visualize dendrograms
fviz_dend(hc.res, show_labels = FALSE,
palette = "jco", as.ggplot = TRUE)
```



Cluster Validation

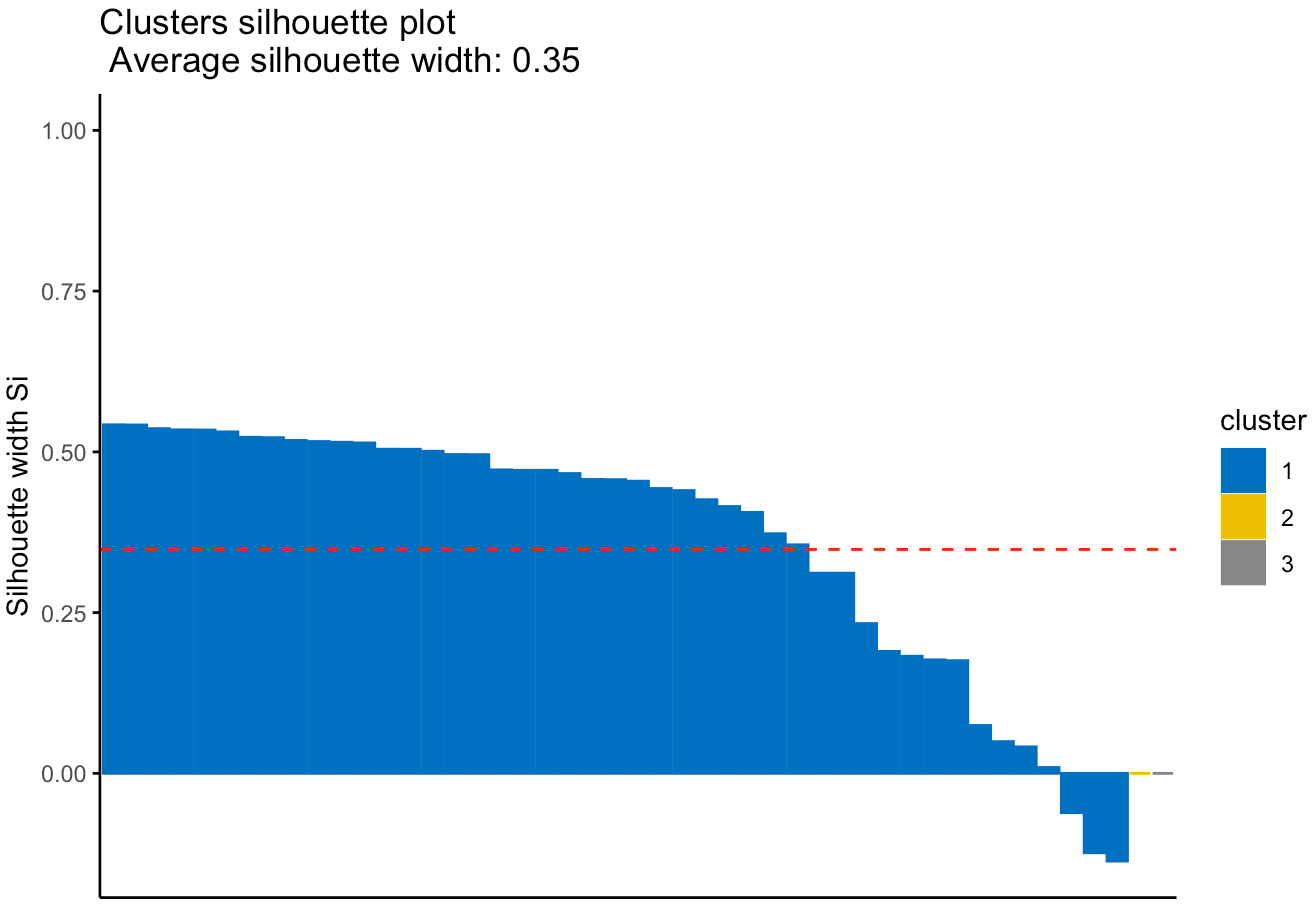
Silhouette coefficient (Si) measures how similar an object i is to the the other objects in its own cluster versus those in the neighbor cluster. Si values range from 1 to - 1:

• A value of Si close to 1 indicates that the object is well clustered. In the other words, the object i is similar to the other objects in its group.

• A value of Si close to -1 indicates that the object is poorly clustered, and that assignment to some other cluster would probably improve the overall results.

```
fviz_silhouette(hc.res, palette = "jco",
ggtheme = theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1   45          0.36
## 2       2    1          0.00
## 3       3    1          0.00
```



Silhouette information can be extracted as follow:

```
## # Silhouette information
silinfo <- hc.res$silinfo
names(silinfo)
```

```
## [1] "widths"          "clus.avg.widths" "avg.width"
```

```
## # Silhouette widths of each observation
head(silinfo$widths[, 1:3], 10)
```

```
##           cluster neighbor sil_width
## Payerne         1        2 0.5421558
## Broye           1        2 0.5419111
## Gruyere         1        2 0.5361706
## Veveyse         1        2 0.5342623
## Delemont        1        2 0.5340691
## Monthey         1        2 0.5311546
## Oron            1        2 0.5226978
## Echallens       1        2 0.5221496
## Moudon          1        2 0.5177667
## Glane           1        2 0.5161145
```

```
## # Average silhouette width of each cluster
silinfo$clus.avg.widths
```

```
## [1] 0.3637332 0.0000000 0.0000000
```

```
## # The total average (mean of all individual silhouette widths)
silinfo$avg.width
```

```
## [1] 0.3482552
```

```
## # The size of each clusters
hc.res$size
```

```
## [1] 45  1  1
```

It can be seen that several samples, in cluster 1,2,and 3, have a negative silhouette coefficient. 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 <- hc.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
## Lausanne          1        2 -0.06118869
## Rive Gauche       1        3 -0.12347202
## Neuchatel         1        3 -0.13642612
```

There are three negative silhouette width, which means these cities are placed in the wrong cluster.

Computing Dunn index and other cluster validation statistics

```
library(fpc)
# Statistics for HC clusters
hc_stats <- cluster.stats(dist(df), hc.res$cluster)
# Dun index
hc_stats$dunn
```

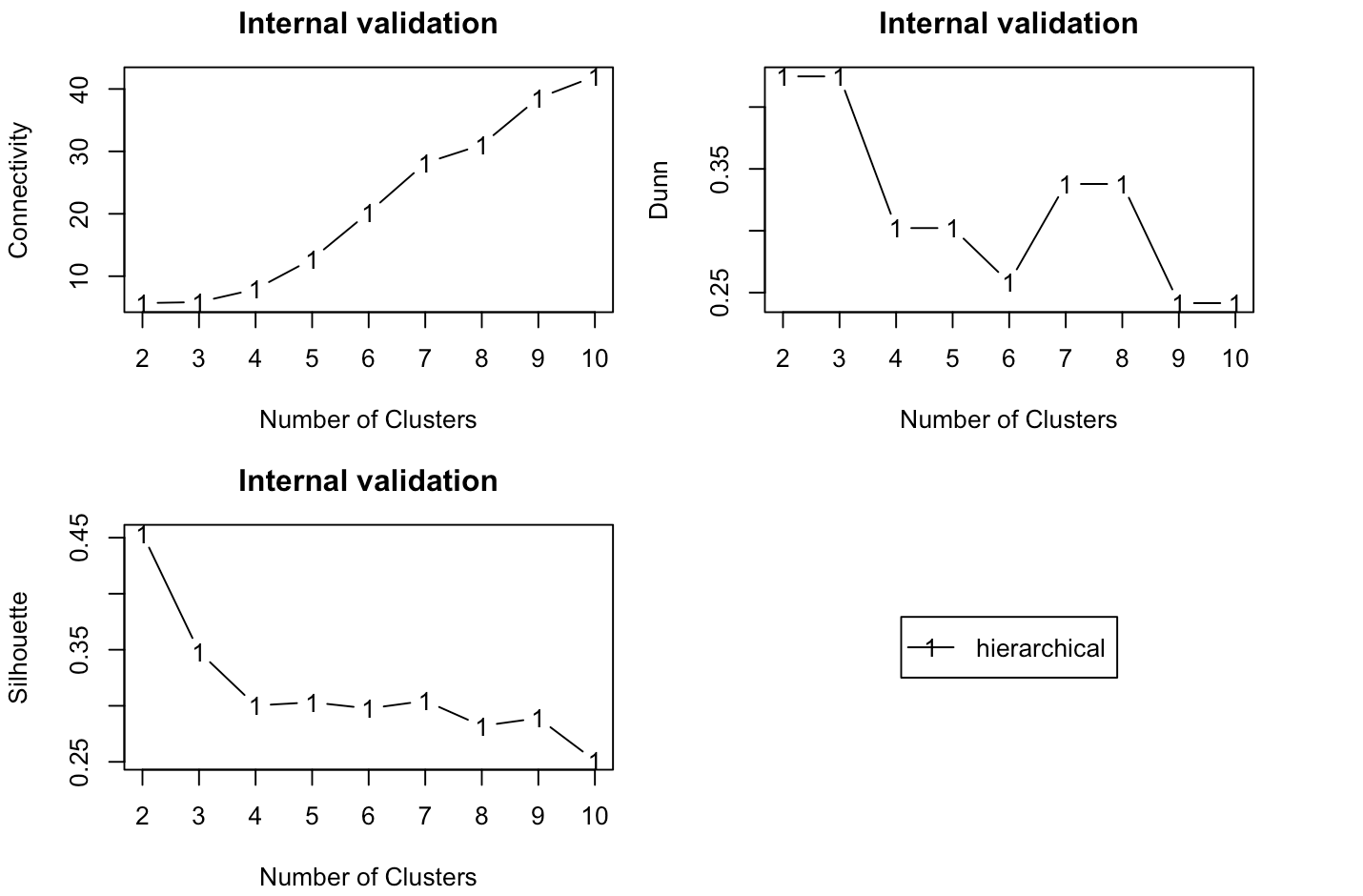
```
## [1] 0.4247781
```

Internal Validation

```
# Load necessary libraries
library(clValid)

# Perform clustering validation with connectivity metric
connectivity_results <- clValid(df, nClust = 2:10,
                                clMethods = "hierarchical",
                                validation = "internal", method = "average")

# Extract connectivity scores
summary(connectivity_results)
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2 3 4 5 6 7 8 9 10
##
## Validation Measures:
##                                  2       3       4       5       6       7       8
9      10
##
## hierarchical Connectivity   5.6913  5.8579  7.9171 12.7429 20.1988 28.1226 31.0488 3
8.5437 42.0171
##              Dunn           0.4248  0.4248  0.3022  0.3022  0.2581  0.3378  0.3378
0.2415  0.2415
##              Silhouette     0.4534  0.3483  0.3003  0.3031  0.2976  0.3044  0.2821
0.2890  0.2511
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 5.6913 hierarchical 2
## Dunn         0.4248 hierarchical 2
## Silhouette   0.4534 hierarchical 2
```

```
op = par(no.readonly=TRUE)
par(mfrow=c(2,2),mar=c(4,4,3,1))
plot(connectivity_results, legend = FALSE)
plot(nClusters(connectivity_results),measures(connectivity_results,"Dunn")[,,1],type
="n",axes=F,xlab="",ylab="")
legend("center", clusterMethods(connectivity_results),col=1:9, lty=1:9,pch=paste(1:9))
```




```
par(op)
```

Connectivity:

- Connectivity measures the compactness of clusters, where lower values indicate better clustering quality.
- The connectivity value increases steadily as the number of clusters increases (from 5.69 for 2 clusters to 42.01 for 10 clusters).
- This suggests that larger cluster sizes may lead to reduced compactness, implying that fewer clusters (e.g., 2 or 3) provide more cohesive results.

Dunn Index:

- The Dunn index evaluates cluster separation and compactness, where higher values indicate better clustering.
- The Dunn index is highest at 2 clusters (0.4248) and decreases with increasing clusters, reaching its lowest value at 9 and 10 clusters (0.2415).
- This indicates that having 2 clusters results in the best balance between cluster separation and compactness.

Silhouette Width:

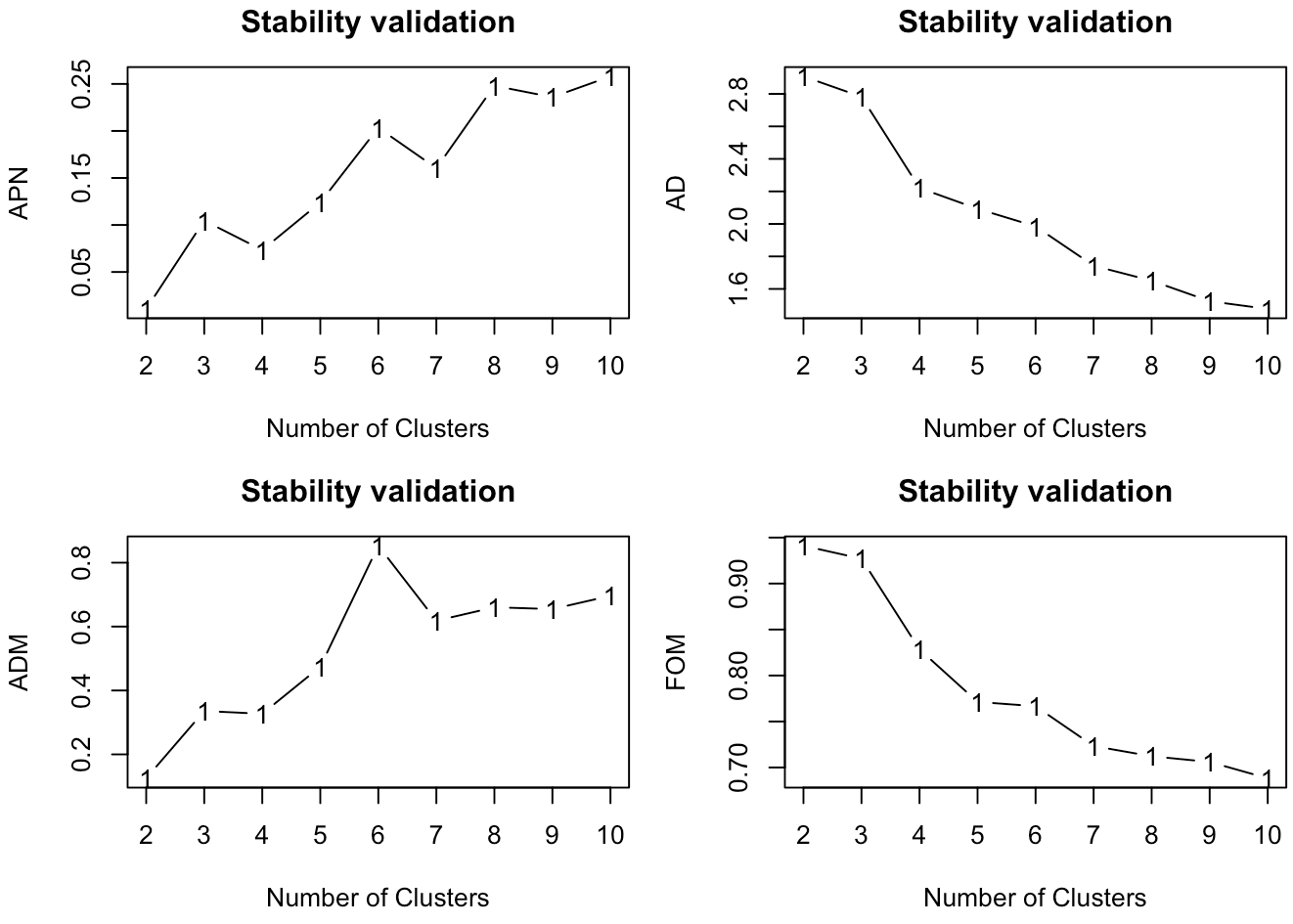
- The silhouette width measures how similar an object is to its cluster compared to other clusters, with higher values indicating better clustering.
- The silhouette value is highest at 2 clusters (0.4534) and steadily decreases as the number of clusters increases (0.2511 for 10 clusters).
- This supports the idea that fewer clusters, such as 2 or 3, offer better-defined clustering.

Stability

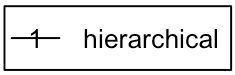
```
# Load necessary libraries
library(clValid)

# Perform clustering validation with connectivity metric
stability_results <- clValid(df, nClust = 2:10,
                             clMethods = "hierarchical",
                             validation = "stability", method = "average")

# Extract connectivity scores
optimalScores(stability_results)
```

```
##         Score      Method Clusters
## APN 0.0106383 hierarchical        2
## AD  1.4766666 hierarchical       10
## ADM 0.1250787 hierarchical        2
## FOM 0.6882230 hierarchical       10
```

```
par(mfrow=c(2,2),mar=c(4,4,3,1))
plot(stability_results,measure = c("APN","AD","ADM","FOM"), legend=FALSE)
```



```
plot(nClusters(stability_results),measures(stability_results,"APN")[,,1],type="n",axes=
F,xlab="",ylab="")
legend("center", clusterMethods(stability_results),col=1:9, lty=1:9,pch=paste(1:9))
par(op)
```



1. APN (Average Proportion of Non-overlap):

- Lower values indicate more stable clusters.
- The APN score is lowest for 2 clusters (0.0106), indicating that 2 clusters yield the most stable solution.

2. AD (Average Distance Between Clusters):

- Higher values suggest better separation between clusters.
- The AD score is highest for 10 clusters (1.4767), but this is misleading as smaller clusters tend to artificially increase inter-cluster distances.

3. ADM (Average Distance Between Members):

- Lower values indicate that objects within a cluster are more tightly grouped.
- The ADM score is lowest for 2 clusters (0.125), reinforcing the compactness of 2 clusters.

4. FOM (Figure of Merit):

- Lower values indicate better clustering quality.
- The FOM score is lowest for 2 clusters (0.6882), again supporting the conclusion that 2 clusters provide the best results.

Results -

While exploratory analysis and clustering metrics (Elbow, Silhouette, Gap) suggested 3 clusters, internal validation metrics (Dunn, Connectivity, Silhouette) and stability analysis supported 2 clusters as more robust.

- If compactness is more important (WSS), go with 3 clusters.
- If separation is more critical (Dunn index, silhouette width), go with 2 clusters.

Research Component

Anomaly Detection

Anomaly detection refers to the process of identifying data points, transactions, or patterns that deviate from the standard norm.
With financial transactions happening at breakneck speed these days, faster anomaly detection and resolution in transaction data become critical in safeguarding the integrity of financial systems.

Anomaly Detection using DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is widely used in anomaly detection in finance by identifying outliers or unusual transactions that deviate from typical patterns. Here's how DBSCAN is applied:

Fraud Detection

DBSCAN clusters normal transactions based on features like transaction amounts, time, and location.
Transactions that do not belong to any cluster (outliers) are flagged as potentially fraudulent.

Market Anomalies

DBSCAN can detect unusual stock price movements or abnormal trading volumes. Clusters represent typical market behaviors, while points outside these clusters are considered anomalous, signaling potential market manipulation or unexpected events.

Credit Risk Assessment

DBSCAN can be used to identify unusual customer behaviors, such as unexpected loan repayment patterns or account activities, which may indicate credit risk or potential fraud.

Anomaly Detection Using K-Means

K-Means is another popular clustering algorithm used in anomaly detection in finance. Here's how K-Means can be applied:

Fraud Detection

- K-Means can be used to cluster normal transactions based on attributes like transaction amount, time, and user behavior.
- Transactions that are far from the cluster centers (centroids) are considered outliers and flagged as potentially fraudulent.

Market Anomalies

- In stock market data, K-Means can be applied to cluster stocks based on price movements, volatility, and trading volumes.

- Stocks or assets that significantly deviate from the centroids of the clusters can be flagged as anomalous, indicating potential market manipulation or unusual events.

Credit Risk Assessment

- K-Means helps in grouping customers based on features like loan repayment history, credit scores, and other financial behaviors.
- Customers whose behavior significantly differs from the typical groups (clusters) may be identified as high-risk or exhibiting irregular patterns.

Customer Segmentation in Retail

- By clustering customers based on purchasing behavior, K-Means can identify those with abnormal spending patterns.
- These anomalies can potentially point to fraudulent activity or shifts in customer preferences.

Clustering in Cybersecurity

With the growing complexity and volume of network traffic, faster anomaly detection and resolution have become critical in safeguarding the integrity of cybersecurity systems.

Anomaly Detection Using DBSCAN in Cybersecurity

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an effective clustering algorithm for detecting anomalies in cybersecurity. It is particularly useful in identifying outliers in network traffic and system activities. Here's how DBSCAN can be applied:

Intrusion Detection

- DBSCAN can cluster network activities based on features such as packet size, connection time, and protocol type.
- Activities that do not belong to any cluster (outliers) are flagged as potential intrusions, such as unauthorized access attempts.

DDoS Attack Detection

- By analyzing traffic volume, request rates, and response times, DBSCAN identifies dense clusters of normal network traffic.
- Excessive or irregular traffic patterns that fall outside these clusters can indicate Distributed Denial of Service (DDoS) attacks.

Data Exfiltration

- DBSCAN can be used to cluster network sessions based on data transfer size and duration.
- Outliers, representing unusually high data transfer volumes or abnormal durations, may signal unauthorized data exfiltration attempts.

Malware Communication Detection

- Communication patterns between systems are clustered using DBSCAN, based on attributes like destination IP, port numbers, and traffic intervals.
- Anomalous communication patterns outside typical clusters are flagged, potentially identifying malware communicating with command-and-control (C2) servers.

Anomaly Detection Using K-Means in Cybersecurity

K-Means is a popular clustering algorithm widely used in cybersecurity to detect anomalies that might indicate malicious activity. Here's how K-Means can be applied:

Intrusion Detection

- K-Means can cluster network activities based on attributes like packet size, response time, and connection duration.
- Unusual network sessions or activities that are far from the cluster centroids are flagged as potential intrusions.

DDoS Attack Detection

- By analyzing the number of requests, session durations, and traffic volume, K-Means can group normal network traffic.
- Clusters with excessively high traffic or unusually short durations may indicate Distributed Denial of Service (DDoS) attacks.

Data Exfiltration

- K-Means can cluster sessions based on data transfer volume and session duration.
- Anomalous sessions with large data transfers or unusually long durations can signal potential data exfiltration attempts.

Malware Communication Detection

- Network communication patterns can be clustered using attributes like destination IP addresses, port usage, and packet intervals.
- Outliers in these clusters often represent unauthorized or malicious communication channels, such as malware communicating with command-and-control (C2) servers.

Example code - Anomaly Detection in transctions

dataset using DBSCAN

```
library(dbscan)
library(ggplot2)

set.seed(123)

# 1. Synthetic dataset for fraud detection (transactions)
transactions <- data.frame(
  amount = c(runif(100, 50, 500), runif(5, 5000, 10000)),  # normal & fraudulent transac
tion amounts
  time = c(runif(100, 1, 24), runif(5, 25, 48)),             # normal & fraudulent trans
action times (hours)
  location = c(runif(100, 1, 10), runif(5, 15, 20))          # normal & fraudulent trans
action locations
)

# 2. Compute the k-distance for each point (with minPts = 5)
kdist <- dbscan::kNNdist(transactions, k = 5)

# 3. Sort the k-distances
sorted_kdist <- sort(kdist)

# 4. Print the sorted k-distances
print("Sorted k-distances:")
```

```
## [1] "Sorted k-distances:"
```

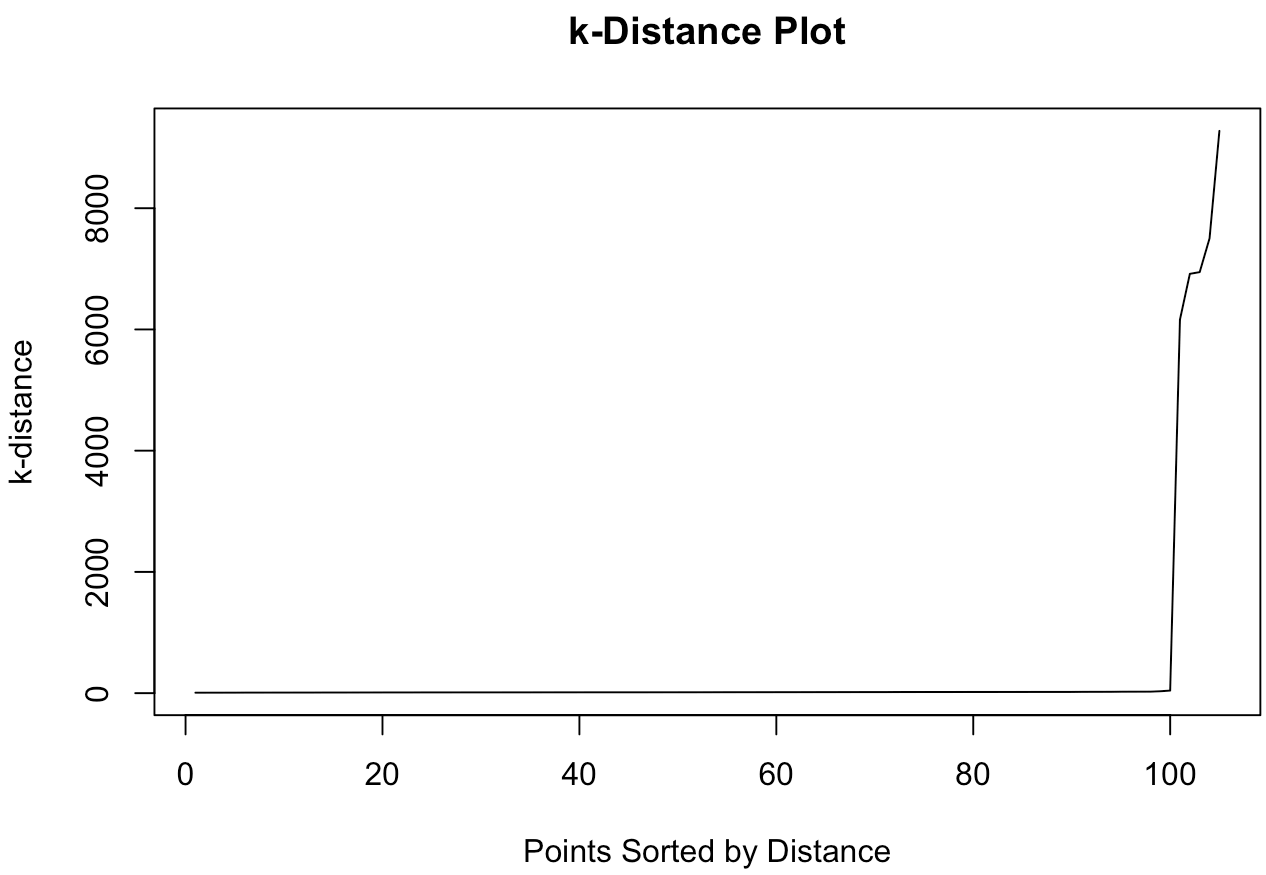
```
print(sorted_kdist)
```

```
##   [1]    7.619940    7.627715    7.803521    8.024496    8.208236    9.000158
##   [7]    9.235477    9.255146    9.336858    9.454578    9.691776    9.698174
##  [13]    9.883450   10.385479   10.493595   10.493595   10.636689   11.068822
##  [19]   11.249826   11.563285   11.789174   11.789174   11.799081   12.239803
##  [25]   12.239803   12.289851   12.480410   12.556942   12.556942   12.655827
##  [31]   12.824995   12.833826   13.073878   13.073958   13.073958   13.438618
##  [37]   13.503867   13.523121   13.675190   13.807756   13.831541   13.856952
##  [43]   13.856952   13.985084   14.012115   14.012115   14.100025   14.406463
##  [49]   14.415523   14.485002   14.570503   14.750531   14.848738   15.331108
##  [55]   15.542321   15.718488   15.725584   15.777497   15.777497   16.140385
##  [61]   16.146923   16.329075   16.519625   16.736748   16.829426   16.966648
##  [67]   17.289988   17.361321   17.428328   17.457647   17.497004   18.148751
##  [73]   18.458212   18.594289   18.657929   18.745323   18.829199   18.945199
##  [79]   19.327797   19.466099   19.596888   19.607238   19.651886   19.651886
##  [85]   19.776614   20.093610   20.177710   20.230512   20.230512   20.295569
##  [91]   21.291072   21.743599   21.839102   22.082469   23.474258   24.039105
##  [97]   24.211611   24.660074   31.383444   42.351201 6166.785243 6917.151478
## [103] 6945.718581 7502.651175 9274.990716
```

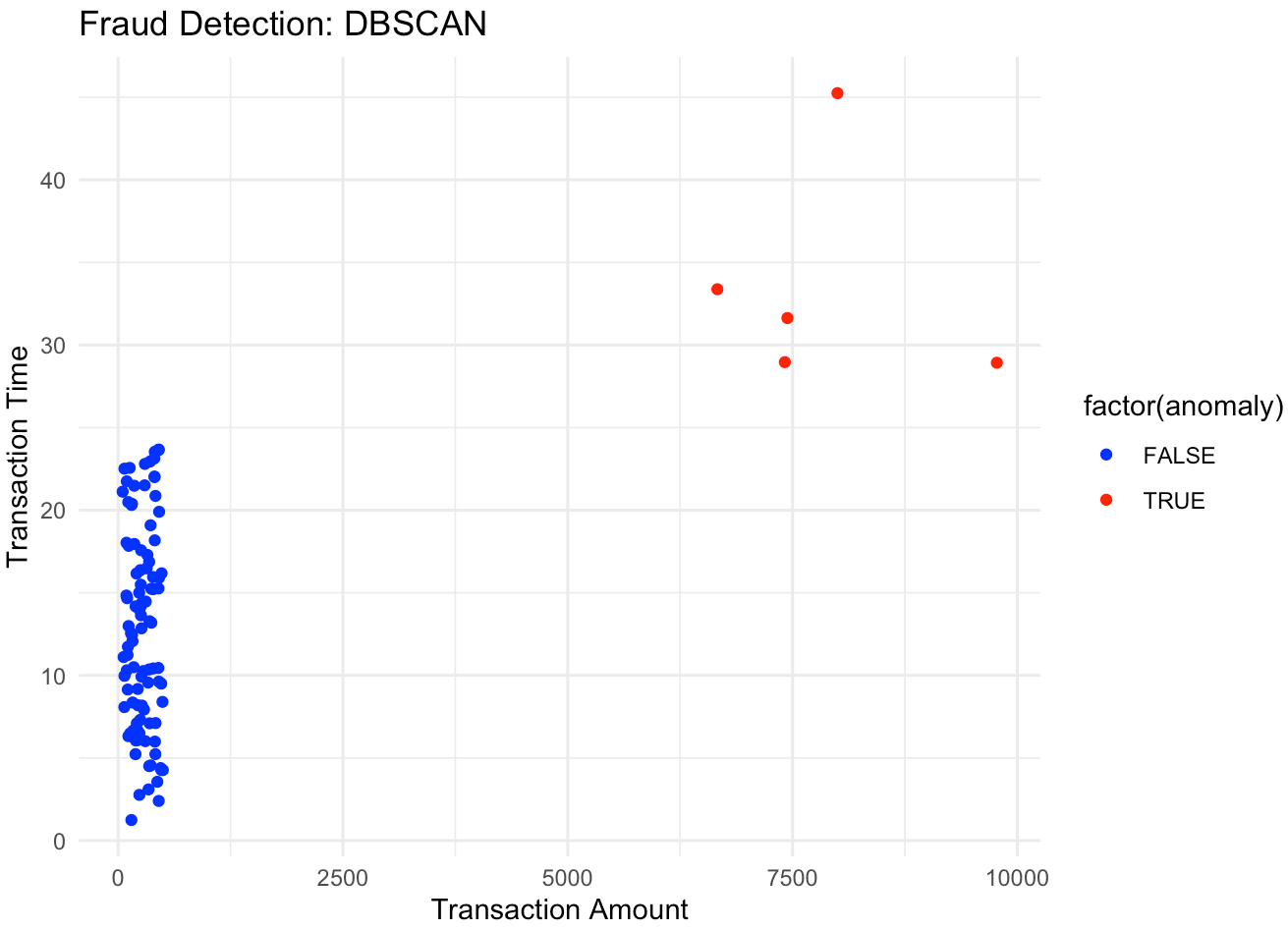
```
# 5. Calculate the differences between consecutive k-distances
kdist_diff <- diff(sorted_kdist)

# 6. Find the index of the largest difference (elbow)
elbow_index <- which.max(kdist_diff)

# 7. Suggested eps value (just before the sharp rise)
suggested_eps <- sorted_kdist[elbow_index]

# 8. Print the suggested eps value
cat("Suggested eps value (based on k-distance plot):", suggested_eps, "\n")
```

```
## Suggested eps value (based on k-distance plot): 42.3512
```

```
# 9. Plot the k-distance
plot(sorted_kdist, type = "l", main = "k-Distance Plot", xlab = "Points Sorted by Distan
ce", ylab = "k-distance")
```



```r
library(dbscan)
library(ggplot2)
set.seed(123)

# Apply DBSCAN to the transactions dataset with adjusted parameters
dbscan_transactions <- dbscan(transactions, eps = 42.5, minPts = 5)
transactions$anomaly <- dbscan_transactions$cluster == 0  # Mark outliers as anomalies

#Visualize Results - Transactions Dataset
ggplot(transactions, aes(x = amount, y = time, color = factor(anomaly))) +
  geom_point() +
  labs(title = "Fraud Detection: DBSCAN", x = "Transaction Amount", y = "Transaction Tim
e") +
  scale_color_manual(values = c("blue", "red")) +
  theme_minimal()
```



```r
library(dbscan)
library(ggplot2)

set.seed(123)

# 1. Synthetic dataset for cybersecurity anomaly detection
cyber_data <- data.frame(
  packets = c(runif(100, 100, 1000), runif(5, 5000, 10000)),  # normal & abnormal packet counts
  duration = c(runif(100, 1, 60), runif(5, 70, 120)),         # normal & abnormal session durations (seconds)
  requests = c(runif(100, 10, 100), runif(5, 200, 500))       # normal & abnormal request rates
)

# 2. Compute the k-distance for each point (with minPts = 5)
kdist <- dbscan::kNNdist(cyber_data, k = 5)

# 3. Sort the k-distances
sorted_kdist <- sort(kdist)

# 4. Print the sorted k-distances
print("Sorted k-distances:")
```

```
## [1] "Sorted k-distances:"
```

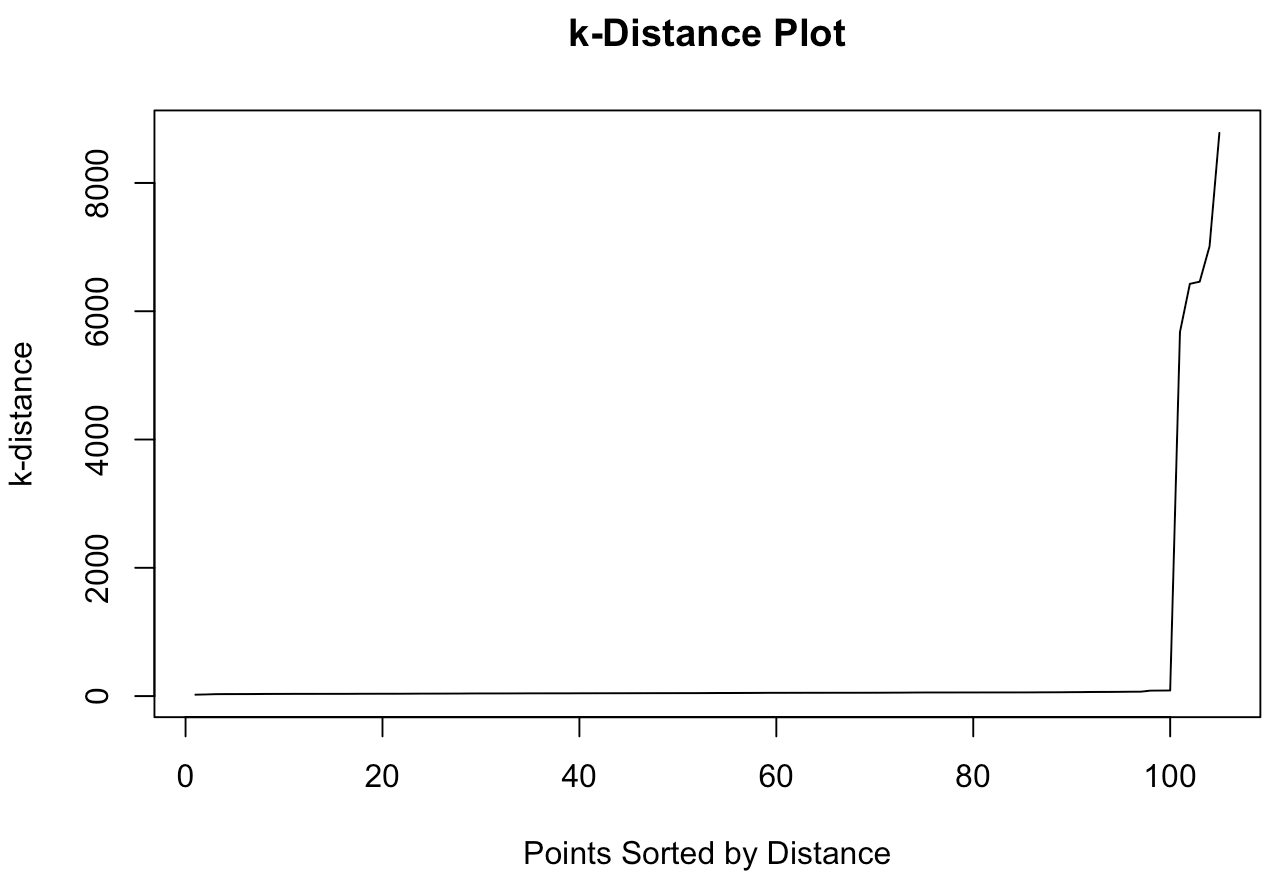
```r
print(sorted_kdist)
```

```
##   [1]   21.84494   25.39144   30.05896   31.18827   31.34104   31.41507
##   [7]   31.87762   33.51269   33.90070   34.06823   34.56901   34.61240
##  [13]   34.71173   34.76789   34.79542   34.79542   35.37245   36.16159
##  [19]   36.16493   36.49291   36.53957   36.56225   37.53243   38.41042
##  [25]   38.41042   38.78428   39.26476   39.70326   40.98463   41.31397
##  [31]   41.33808   41.41398   41.77581   42.24380   42.94690   42.94690
##  [37]   43.10096   43.12576   43.25228   43.91075   44.02642   44.49849
##  [43]   44.49849   44.71900   44.87078   45.33662   45.48729   45.86307
##  [49]   45.93643   46.08761   46.10318   46.40681   47.09759   47.87181
##  [55]   47.93720   48.65044   49.22174   49.53594   50.63783   50.87667
##  [61]   50.92113   51.14140   51.30402   51.51556   51.51556   51.62998
##  [67]   51.63046   51.87730   51.87730   52.22940   52.77617   53.49590
##  [73]   54.04393   55.45299   56.15956   56.22858   56.32610   56.35119
##  [79]   56.57704   56.57704   56.84125   56.87231   57.16653   57.19346
##  [85]   57.51557   57.84064   59.01113   59.27447   60.00551   61.64310
##  [91]   62.13146   64.25098   64.73640   65.18384   66.36461   67.94773
##  [97]   68.50160   85.05343   86.09917   87.40863 5679.08065 6426.64198
## [103] 6460.97367 7012.37292 8780.10732
```

```
# 5. Calculate the differences between consecutive k-distances
kdist_diff <- diff(sorted_kdist)

# 6. Find the index of the largest difference (elbow)
elbow_index <- which.max(kdist_diff)

# 7. Suggested eps value (just before the sharp rise)
suggested_eps <- sorted_kdist[elbow_index]

# 8. Print the suggested eps value
cat("Suggested eps value (based on k-distance plot):", suggested_eps, "\n")
```

```
## Suggested eps value (based on k-distance plot): 87.40863
```

```
# 9. Plot the k-distance
plot(sorted_kdist, type = "l", main = "k-Distance Plot",
     xlab = "Points Sorted by Distance",
     ylab = "k-distance")
```



```r
# 10. Apply DBSCAN using the suggested eps value
dbscan_model <- dbscan(cyber_data, eps = suggested_eps, minPts = 5)
cyber_data$cluster <- dbscan_model$cluster

# 11. Mark anomalies (outliers have cluster 0)
cyber_data$anomaly <- dbscan_model$cluster == 0

# 12. Visualize the results
ggplot(cyber_data, aes(x = packets, y = duration, color = factor(anomaly))) +
  geom_point(size = 3) +
  labs(title = "DBSCAN: Cybersecurity Anomaly Detection",
       x = "Packets",
       y = "Session Duration (seconds)",
       color = "Anomaly") +
  scale_color_manual(values = c("blue", "red"), labels = c("Normal", "Anomalous")) +
  theme_minimal()
```