

Clustering sull'olio d'oliva italiano

Mariachiara Manoccio, Marta calasso, Daniele Cristofori, Giovanni Tripicchio,

About Dataset

Questo lavoro si basa sulla descrizione di modelli di clustering applicati al dataset open source “olive” presente nel pacchetto “pgmm” del software R. In questo dataset sono riportati i valori della composizione percentuale di otto acidi grassi rilevati nella frazione lipidica di un campione di oli d’oliva italiani. Il dataset si compone di 572 osservazioni distribuite in 10 colonne. La prima colonna riguarda la zona: 1) Sud Italia ; 2) Sardegna; 3) Nord Italia. La seconda colonna riguarda l’area: 1) Nord Puglia; 2) Calabria; 3) Sud Puglia; 4) Sicilia; 5) entroterra Sardegna; 6) costa Sardegna; 7) Est Liguria; 8) Ovest Liguria; 9) Umbria. Le prime due colonne, Region e Area che sono in origine variabili categoriche, ma sono etichettate in numeriche per facilità di manipolazione. Le altre colonne sono variabili numeriche riferite acidi grassi analizzati negli oli osservati: *palmitic, palmitoleic, stearic, oleic, linoleic,linolenic, arachidic, eicosenoic*. Nelle chunk di codice successive, si importano le librerie e si procede all’ esporazione dei dati.

```
# import librerie
library(dplyr)
library(pgmm)
library(MASS)
library(DataExplorer)
library(caret)
library(FactoMineR)
library(factoextra)
library(mixtools)
library(plyr)
library(caret)
library(tidyselect)
library(forcats)
library(janitor)
library(gdata)
library(tools)
library(mclust)
library(ggplot2)
library(dplyr)
library(tidyr)
library(tibble)
library(readr)
library(stringr)
library(tidyverse)
library(clustertend)
library(stats)
library(fpc)
library(c1Valid)
library(NbClust)
#import dataset
data(olive)
```

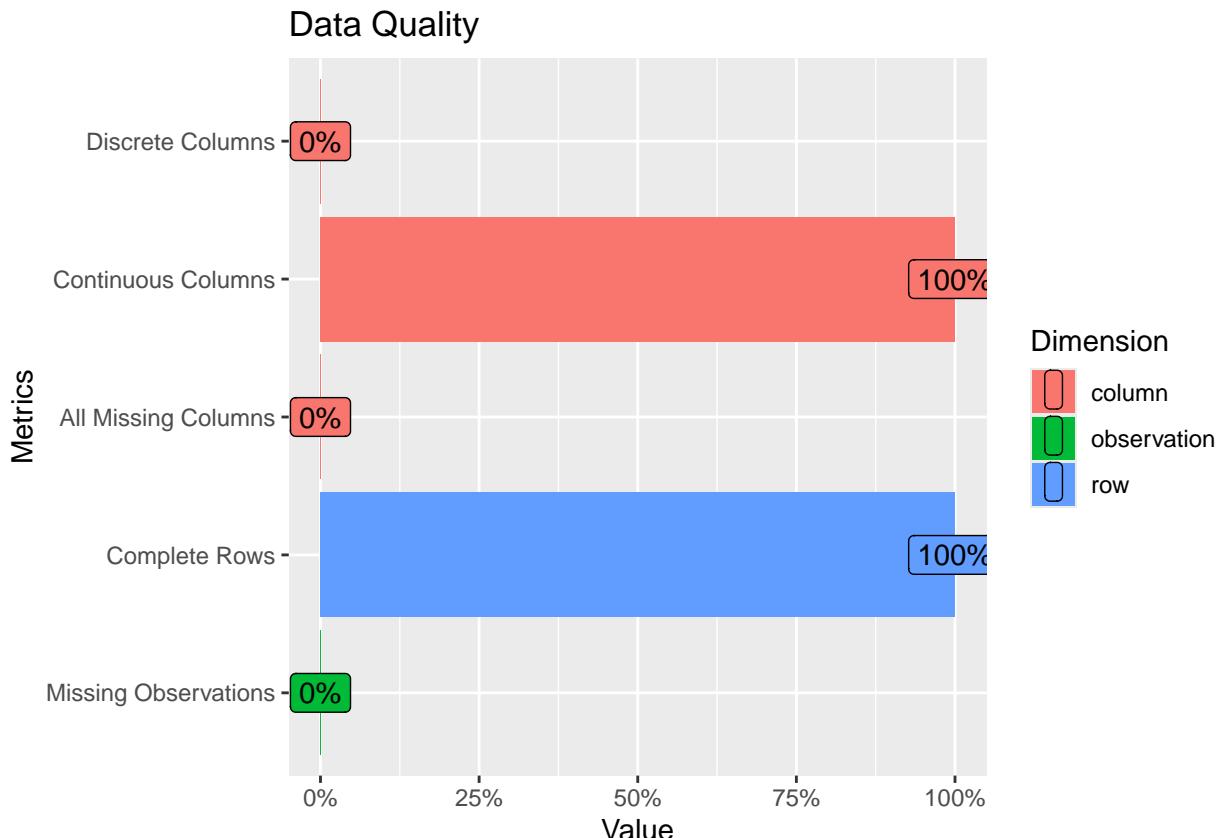
```
#vizzualizzazione prime righe del dataset
head(olive)
```

```
##   Region Area Palmitic Palmitoleic Stearic Oleic Linoleic Linolenic Arachidic
## 1      1    1     1075        75     226    7823     672      36      60
## 2      1    1     1088        73     224    7709     781      31      61
## 3      1    1      911       54     246    8113     549      31      63
## 4      1    1      966       57     240    7952     619      50      78
## 5      1    1     1051       67     259    7771     672      50      80
## 6      1    1      911       49     268    7924     678      51      70
##   Eicosenoic
## 1      29
## 2      29
## 3      29
## 4      35
## 5      46
## 6      44
```

Dall'analisi sulla qualità del dataset si osserva che:

- il dataset non ha osservazioni mancanti;
- le variabili sono tutte continue.

```
attach(olive)
plot_intro(olive, title ="Data Quality" )
```



Dal dataset iniziale vengono rimosse le variabili target "Area" e "Region" al fine di applicare tutte le tecniche che saranno approfondite nei paragrafi successivi. Di seguito sono riportate le statistiche descrittive relative

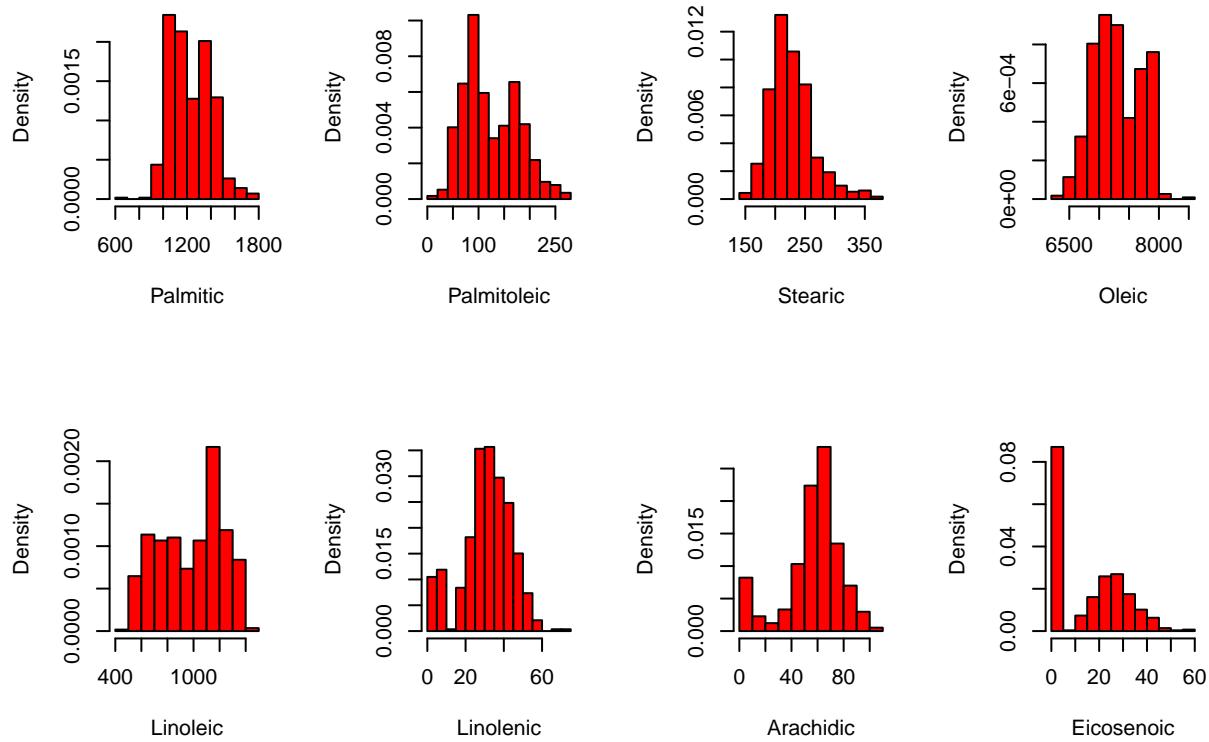
alle variabili.

```
#statistiche descrittive variabili
#rimozione variabili target Region e Area
print(summary(subset(olive, select = -c( Region, Area))))
```

	Palmitic	Palmitoleic	Stearic	Oleic
## Min.	610	Min. : 15.00	Min. :152.0	Min. :6300
## 1st Qu.	1095	1st Qu.: 87.75	1st Qu.:205.0	1st Qu.:7000
## Median	1201	Median :110.00	Median :223.0	Median :7302
## Mean	1232	Mean :126.09	Mean :228.9	Mean :7312
## 3rd Qu.	1360	3rd Qu.:169.25	3rd Qu.:249.0	3rd Qu.:7680
## Max.	1753	Max. :280.00	Max. :375.0	Max. :8410
	Linoleic	Linolenic	Arachidic	Eicosenoic
## Min.	448.0	Min. : 0.00	Min. : 0.0	Min. : 1.00
## 1st Qu.	770.8	1st Qu.:26.00	1st Qu.: 50.0	1st Qu.: 2.00
## Median	1030.0	Median :33.00	Median : 61.0	Median :17.00
## Mean	980.5	Mean :31.89	Mean : 58.1	Mean :16.28
## 3rd Qu.	1180.8	3rd Qu.:40.25	3rd Qu.: 70.0	3rd Qu.:28.00
## Max.	1470.0	Max. :74.00	Max. :105.0	Max. :58.00

Dall'analisi sulle distribuzioni si osserva un andamento tendenzialmente normale di quasi tutte le variabili, ad eccezione degli acidi grassi oleico, linoleico, palmitoleico, che hanno una distribuzione più asimmetrica.

```
#Distribuzioni sulle variabili
par(mfrow=c(2,4))
hist(Palmitic, freq=F, col="red", xlab="Palmitic", main="")
hist(Palmitoleic, freq=F, col="red", xlab="Palmitoleic", main="")
hist(Stearic, freq=F, col="red", xlab="Stearic", main="")
hist(Oleic, freq=F, col="red", xlab="Oleic", main="")
hist(Linoleic, freq=F, col="red", xlab="Linoleic", main="")
hist(Linolenic, freq=F, col="red", xlab="Linolenic", main="")
hist(Arachidic, freq=F, col="red", xlab="Arachidic", main="")
hist(Eicosenoic, freq=F, col="red", xlab="Eicosenoic", main="")
```

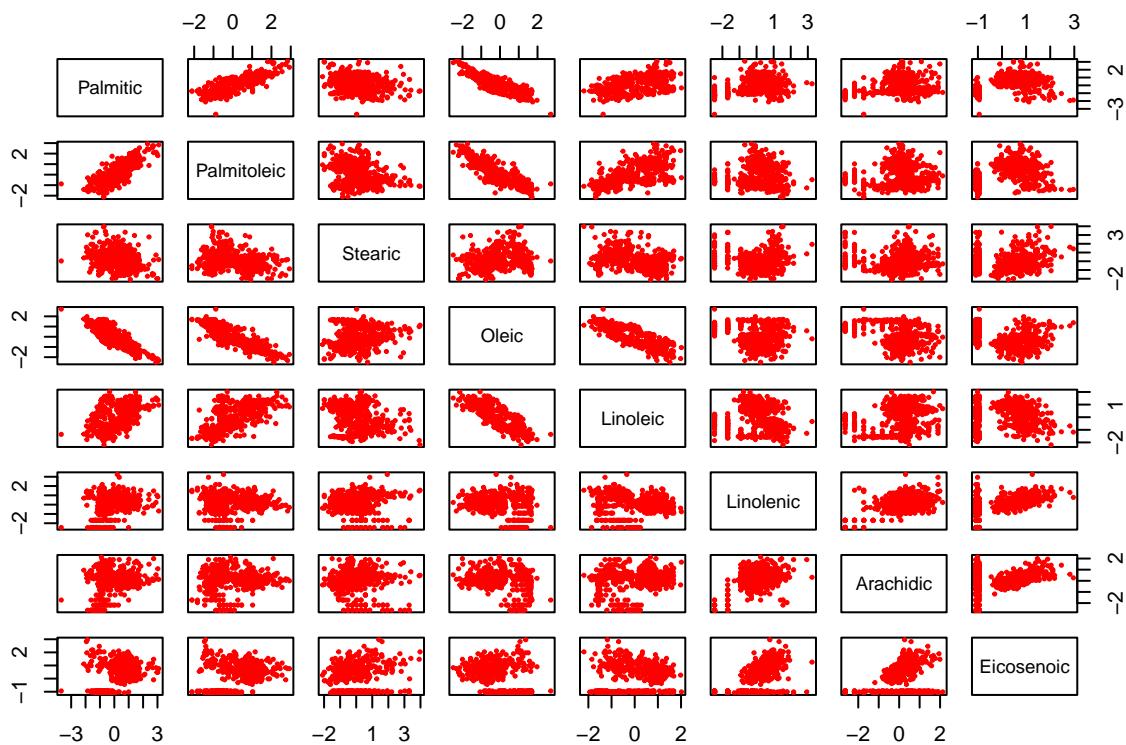


Le variabili vengono standardizzate utilizzando la funzione “scale”, garantendo così che ciascuna variabile abbia lo stesso peso nella determinazione della distanza tra le osservazioni. Da questo punto in poi nel testo, il dataset con le variabili standardizzate (denominato DB2S) verrà utilizzato per tutti i modelli e le tecniche che richiedono la standardizzazione delle variabili.

Dall’analisi del pairplot effettuato sulle variabili emerge la presenza di cluster distinti. In particolare, l’acido eicosenoico sembra formare, a prima vista, due cluster in relazione alle altre variabili. Un ragionamento simile può essere applicato alle variabili Linolenico e Arachidico.

```
#standardizzazione delle variabili
DB2S <- scale(subset(olive, select = -c( Region, Area)))
#pairplot
pairs(DB2S, col="red", pch=20, cex=0.5)
title("Distribuzione dei cluster in funzione delle variabili", line= 3.35)
```

Distribuzione dei cluster in funzione delle variabili



I due grafici successivi mostrano la distribuzione degli oli per macroaree geografiche e aree territoriali, rispettivamente. Dal primo si evince una forte prevalenza della provenienza del Sud Italia, del 57% circa, seguita da Nord Italia (26%) e Sardegna (17%). La zona con la più alta produzione di olio è il sud della Puglia, con una percentuale molto alta del 36%, seguita dalle regioni successive con percentuali molto più basse.

```
#DISTRIBUZIONE PER MACROAREE
DBs <- data.frame(olive)
osservArea <- table(DBs$Region)
osservRegioni <- table(DBs$Area)
osservAreadf <- as.data.frame(osservArea)
colnames(osservAreadf) <- c("Area", "Osservazioni")
osservRegionidf <- as.data.frame(osservRegioni)
colnames(osservRegionidf) <- c("Regione", "Osservazioni")

osservAreadf <- osservAreadf %>%
  mutate(NomeArea = case_when(
    Area == 1 ~ "Sud Italia",
    Area == 2 ~ "Sardegna",
    Area == 3 ~ "Nord Italia",
    TRUE ~ as.character(Area)
  ))

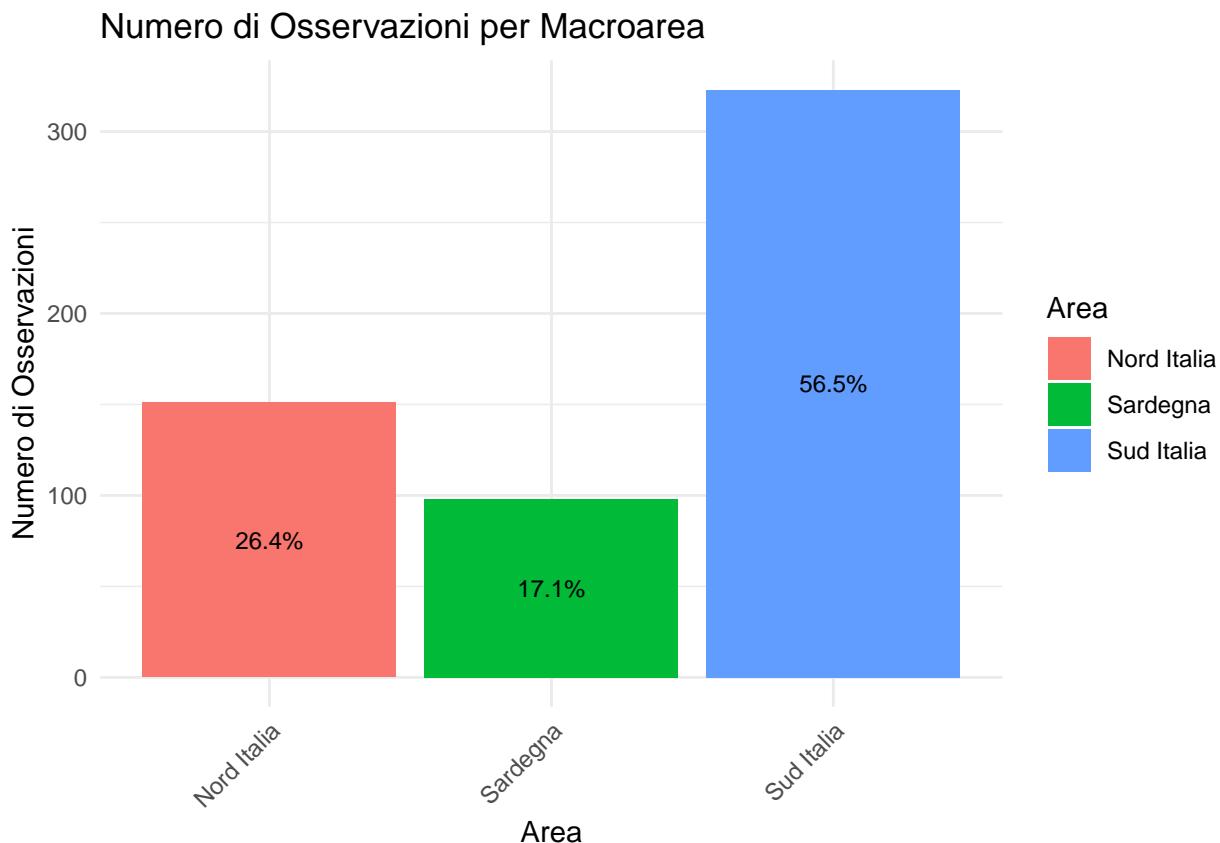
# barplot utilizzando ggplot2
barplotArea <- ggplot(osservAreadf, aes(x=NomeArea, y=Osservazioni,
                                         fill=as.factor(NomeArea))) +
  geom_bar(stat="identity") +
  geom_text(aes(label = paste0(round(Osservazioni / sum(Osservazioni) * 100, 1), "%")),
            position = position_stack(vjust = 0.5), color = "black", size = 3) +
```

```

  labs(title="Numero di Osservazioni per Macroarea", x="Area", y="Numero di Osservazioni",
       fill="Area") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(barplotArea)

```



```

#OSSERVAZIONI PER REGIONE
# Aggiunge una colonna relativa al nome dell'area relativa al dataframe con i nomi delle
# regioni
osservRegionidf <- osservRegionidf %>%
  mutate(NomeRegione = case_when(
    Regione == 1 ~ "Nord Puglia",
    Regione == 2 ~ "Calabria",
    Regione == 3 ~ "Sud Puglia",
    Regione == 4 ~ "Sicilia",
    Regione == 5 ~ "Entroterra Sardegna",
    Regione == 6 ~ "Costa Sardegna",
    Regione == 7 ~ "Est Liguria",
    Regione == 8 ~ "Ovest Liguria",
    Regione == 9 ~ "Umbria",
    TRUE ~ as.character(Regione)
  ))
osservRegionidf

```

```

##   Regione Osservazioni      NomeRegione
## 1          1              25      Nord Puglia

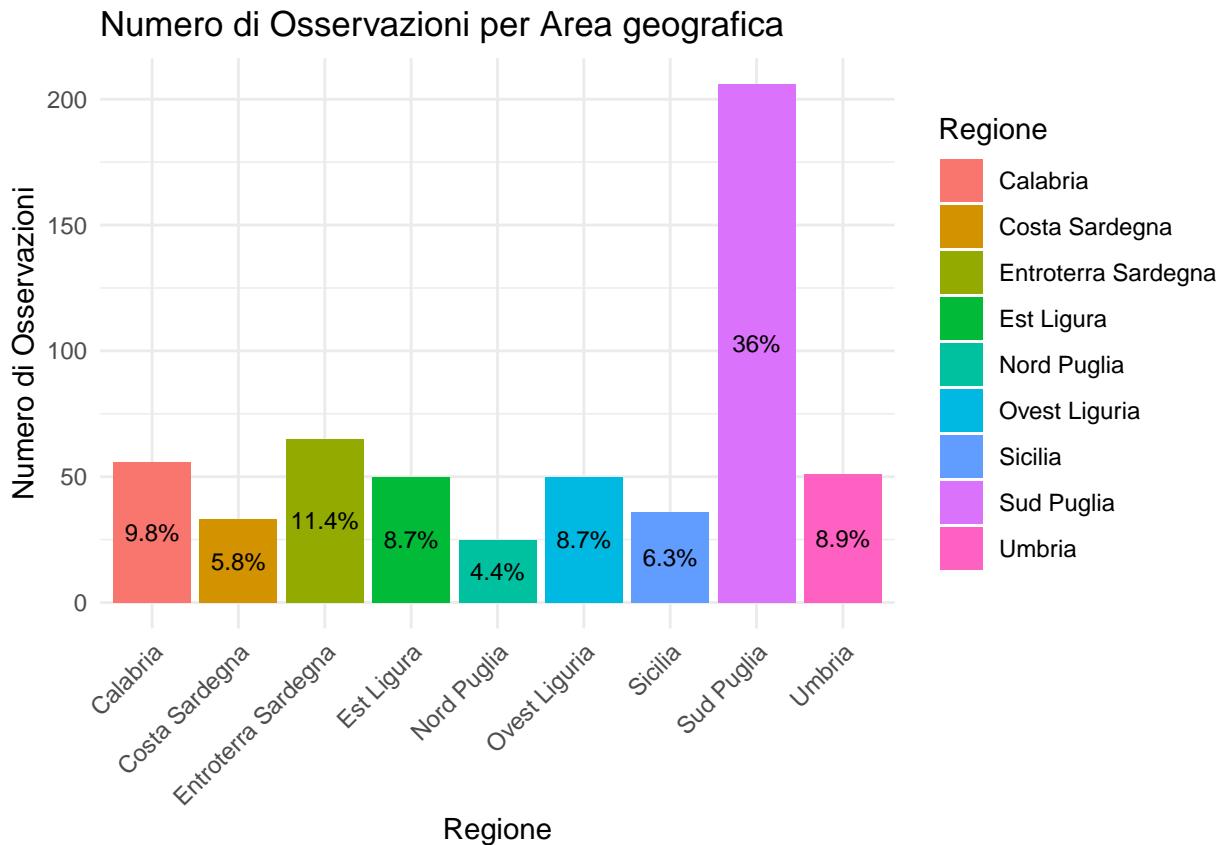
```

```

## 2      2      56      Calabria
## 3      3     206     Sud Puglia
## 4      4      36      Sicilia
## 5      5     65 Entroterra Sardegna
## 6      6     33 Costa Sardegna
## 7      7      50      Est Ligura
## 8      8     50 Ovest Liguria
## 9      9      51      Umbria

#barplot
barplotRegione <- ggplot(osservRegionidf, aes(x=NomeRegione, y=Osservazioni,
                                               fill=as.factor(NomeRegione))) +
  geom_bar(stat="identity") +
  geom_text(aes(label = paste0(round(Osservazioni / sum(Osservazioni) * 100, 1), "%")),
            position = position_stack(vjust = 0.5), color = "black", size = 3) +
  labs(title="Numero di Osservazioni per Area geografica", x="Regione",
       y="Numero di Osservazioni", fill="Regione") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(barplotRegione)

```



Il boxplot seguente mostra la presenza di outliers ad eccezione delle variabili: eicosenoic, linolenic, oleic e palmitoleic. Da notare che le medie sono distribuite intorno allo zero perché il grafico è stato costruito sui dati standardizzati per una migliore visualizzazione.

```

#costruzione boxplot
DB2S <- as.data.frame(DB2S)
data_long <- gather(DB2S, key = "Caratteristiche", value = "Value")

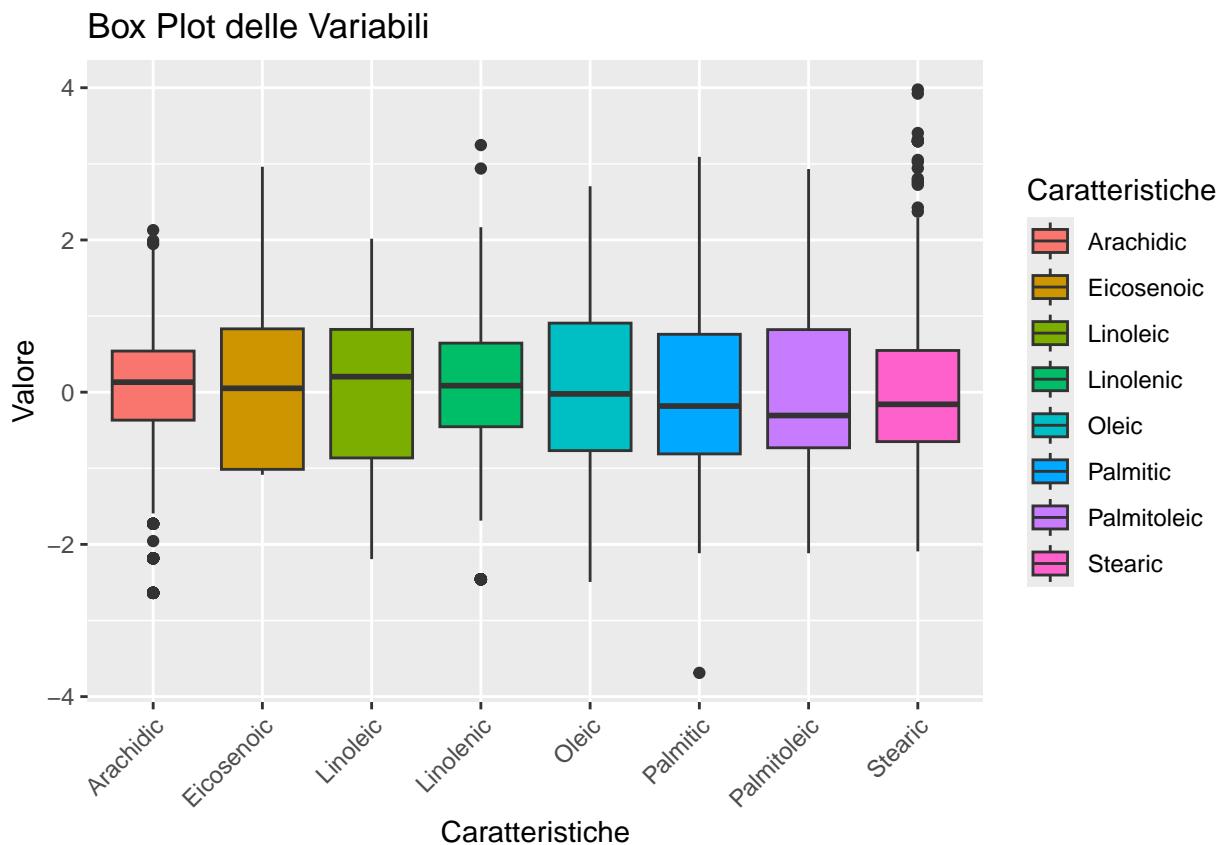
```

```

attach(data_long)

ggplot(data_long, aes(x = Caratteristiche, y = Value, fill=Caratteristiche)) +
  geom_boxplot() +
  labs(x = "Caratteristiche", y = "Valore", title = "Box Plot delle Variabili")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```



Principal component Analysis - PCA

Si applica la funzione per l'analisi delle componenti principali (PCA) per avere informazioni sull' impatto delle variabili sul modello. La PCA è un metodo che viene spesso utilizzato nel caso di set di dati con un ingente numero di variabili per ridurne la dimensionalità, trasformando il dataset di partenza in un sistema più piccolo che contiene ancora la maggior parte delle informazioni originali. Le componenti principali sono nuove variabili costruite come combinazioni lineari o miscele delle variabili iniziali non correlate tra loro. Per tale analisi si applica la funzione `PCA()`, che effettua automaticamente lo scaling dei dati. Le componenti principali ottenute tramite PCA sono distribuite in base alla varianza dei dati originali. In altre parole, le prime componenti principali spiegano la massima variazione nei dati, mentre le successive spiegano via via sempre meno questa variazione. Questa distribuzione è determinata dagli autovalori associati a ciascuna componente principale.

```

# funzione PCA
res.pca <- PCA(olive, graph = FALSE)
#informazioni risultanti dall' applicazione della funzione PCA
#calcolo autovalori
eig.val <- get_eigenvalue(res.pca)
eig.val

##          eigenvalue variance.percent cumulative.variance.percent
## Dim.1    5.159467378      51.59467378            51.59467
## Dim.2    1.932591302      19.32591302            70.92059
## Dim.3    1.040435578      10.40435578           81.32494
## Dim.4    0.798095093      7.98095093            89.30589
## Dim.5    0.512693700      5.12693700            94.43283
## Dim.6    0.251483337      2.51483337            96.94766
## Dim.7    0.156374451      1.56374451            98.51141
## Dim.8    0.119292031      1.19292031            99.70433
## Dim.9    0.027595080      0.27595080            99.98028
## Dim.10   0.001972049      0.01972049            100.00000

```

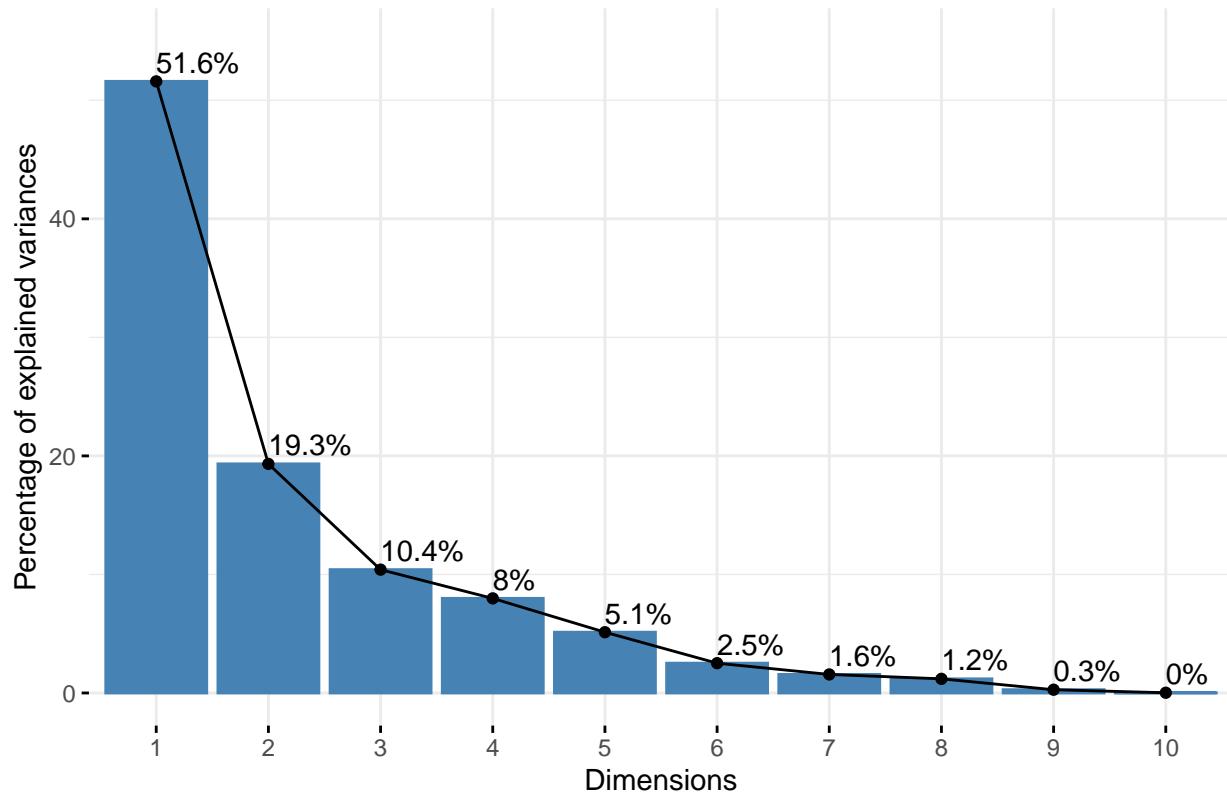
Con il calcolo degli autovalori è possibile determinare il numero di componenti principali misurando la variazione mantenuta da ogni componente principale (PC). I pesi (autovettori) vengono scelti in modo tale che il valore di un'unità campionaria su un asse non sia correlato al valore della stessa unità campionaria su un altro asse. Dall'output del calcolo degli autovalori si può notare che autovalore più alto presenta una varianza cumulativa al 51.6%. La PCA cerca di inserire la massima informazione possibile nella componente 1. La seconda componente insieme alla prima formano il 70% della varianza cumulativa totale e così via per le successive componenti principali, come mostrato nello scree plot sottostante.

```

#Screeplot sulle PCA, metodo utilizzato per visualizzare le componenti principali.
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 55))

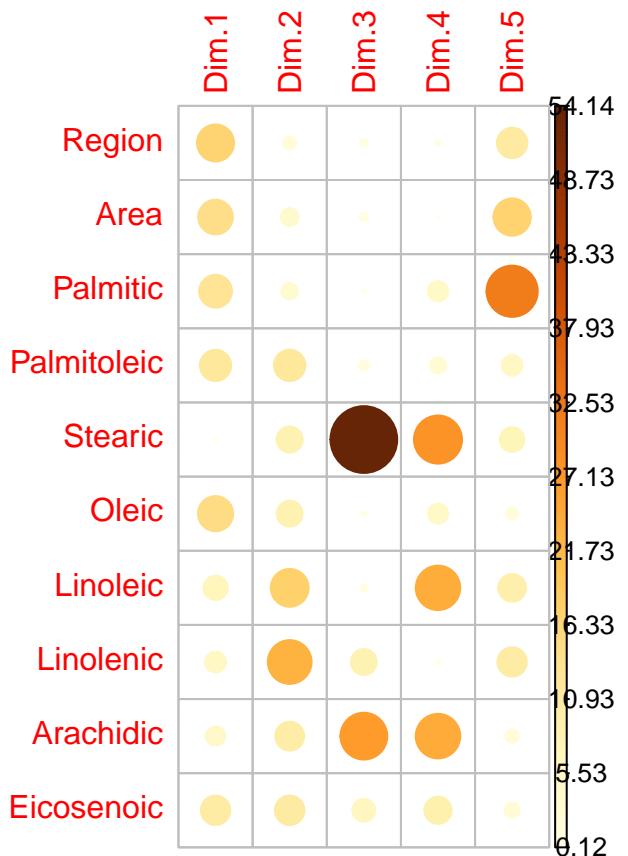
```

Scree plot



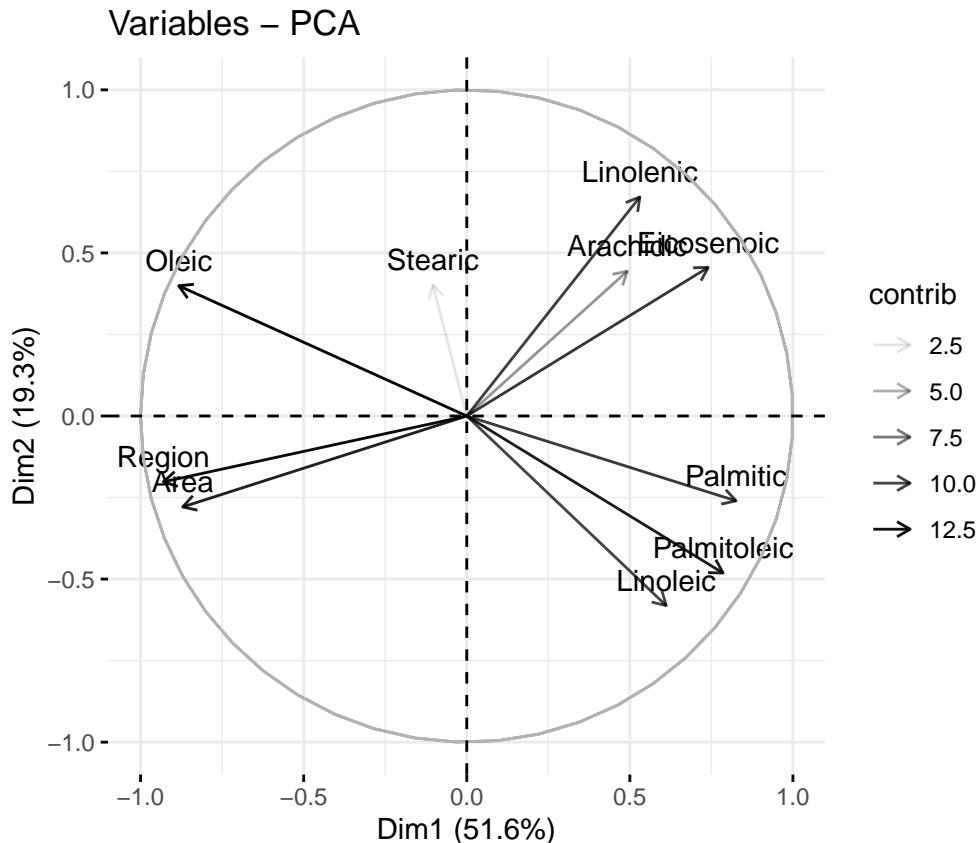
Se l'obiettivo è la riduzione dei dati, e ci si vuole concentrare su un numero limitato di componenti principali delle osservazioni, possiamo fermarci alla componente 5. Il seguente plot mostra il contributo in percentuale delle variabili nelle prime 5 componenti principali.

```
# Contributo delle variabili
library("corrplot")
var <- get_pca_var(res.pca)
corrplot(var$contrib, is.corr=FALSE)
```



Un' altra rappresentazione indicativa delle componenti principali è il biplot, calcolato con il seguente script:

```
#Biplot. La trasparenza varia in base al peso delle componenti.  
fviz_pca_var(res.pca, alpha.var = "contrib")
```



Nel grafico gli assi x e y sono la componente principale 1 e 2, rispettivamente.

Ogni vettore corrisponde a una variabile e punta nella direzione in cui tale variabile è più fortemente correlata linearmente. La lunghezza della linea e la sua vicinanza al cerchio indicano quanto bene la variabile è rappresentata nel grafico. Inoltre, l'angolo (α) tra i vettori è un'approssimazione della correlazione tra le variabili. In particolare:

- $\alpha < 90^\circ$ le variabili sono correlate positivamente (nel grafico sono raggruppate insieme);
- $\alpha = 90^\circ$, le variabili non sono correlate;
- $\alpha \sim 180^\circ$ indica che le variabili sono correlate negativamente e posizionate nei quadanti opposti;
- I vettori perpendicolari rappresentano variabili non correlate.

In questo progetto, poiché il numero di variabili di interesse ai fini dell'applicazione dei successivi modelli di clustering è sostenibile, sceglieremo di non applicare la PCA ma verranno utilizzate le variabili originali.

Cluster Analysis

Cluster tendency

Prima di procedere ad applicare qualsiasi algoritmo di clusterizzazione al dataset, bisogna valutare la tendenza al clustering del dataset. Le principali metodologie per effettuare tale valutazione sono:

1. *metodo statistico*: Hopkins statistics;
2. *metodo grafico*: VAT.

La statistica H di Hopkins è una misura della tendenza al clustering che assume valori tra [0, 1]. Se assume valori vicino a zero indica dati che i dati possono essere clusterizzati, mentre se si aggirano intorno a 0.5 indica che i dati sono uniformemente distribuiti

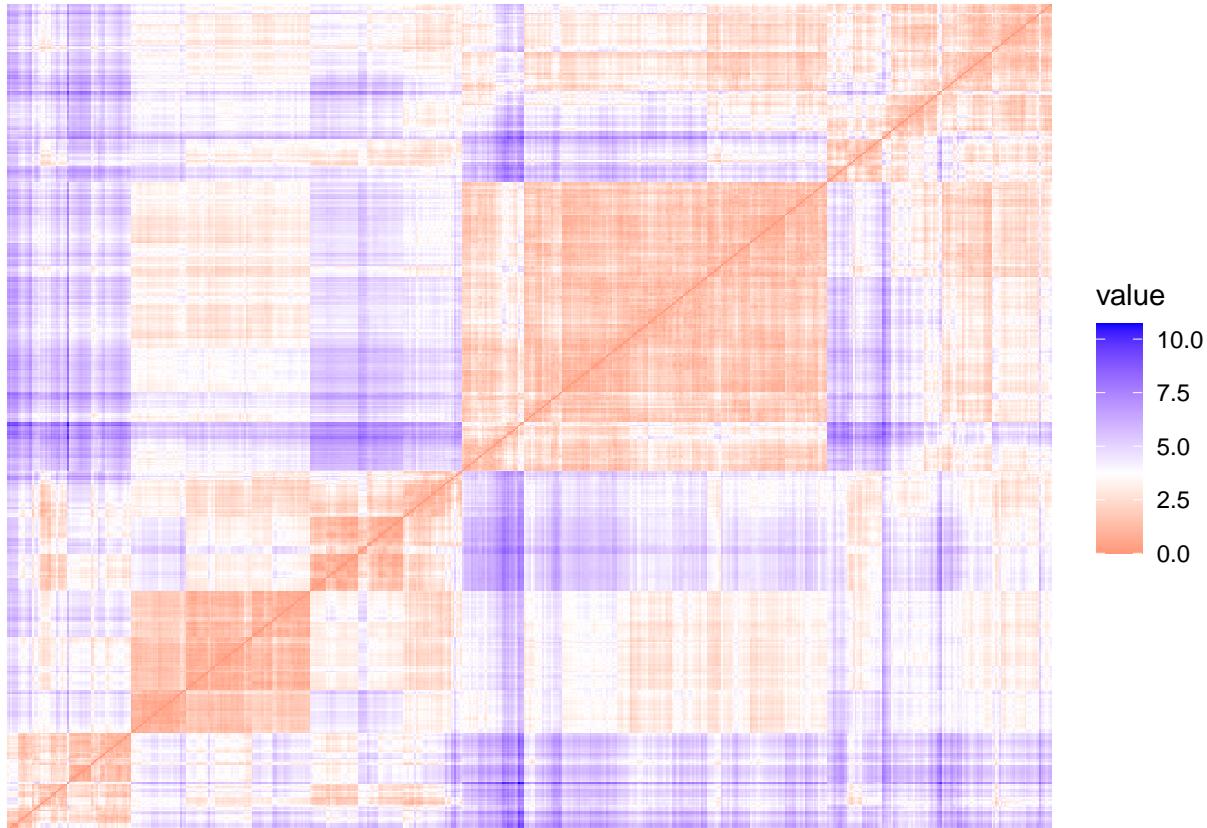
```
hop <- hopkins(DB2S, n=nrow(DB2S)-1)
hop
```

```
## $H
## [1] 0.1742754
```

Il valore ottenuto della statistica di Hopkins ($hop = 0.17$) è vicino a zero; questo indica che il dataset è “clusterizzabile”.

Il metodo grafico adotta l’algoritmo VAT. La misura di distanza adottata è quella euclidea.

```
fviz_dist(dist(DB2S), show_labels = FALSE)
```



L’immagine della matrice di dissimilarità mostra la formazione di “quadrati” di dimensioni crescenti sulla diagonale, confermando ulteriormente la presenza di strutture a grappolo nel set di dati.

Numero di cluster

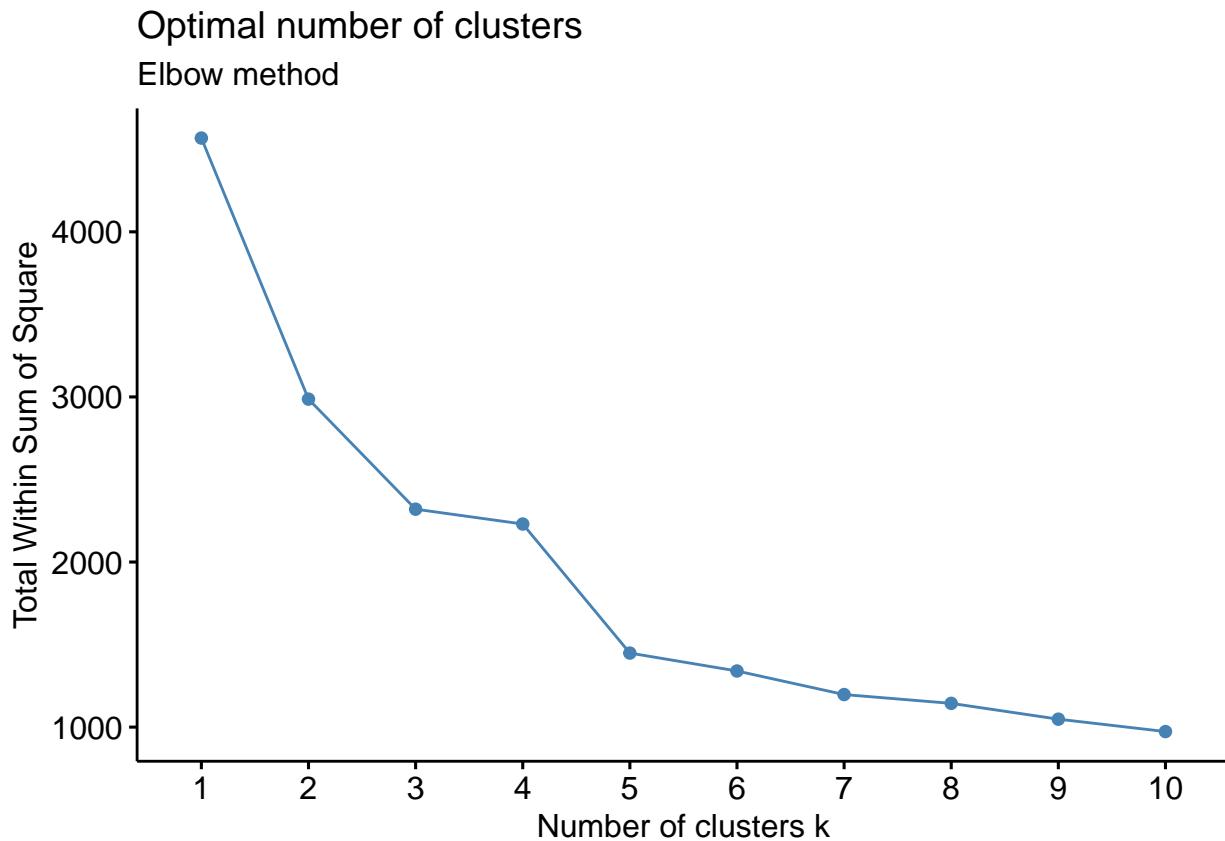
Determinare il numero ottimale di cluster K è di fondamentale importanza per l’ applicazione di diversi algoritmi di clustering e, a seconda dei metodi è possibile ottenere diverse soluzioni. Ai fini degli algoritmi che verranno presentati nei paragrafi successivi si utilizzano le seguenti metodologie per il rilevamento dei numeri di cluster:

1. *metodi diretti*: metodo del gomito e della silhouette;
2. *metodi statistici*: gap statistic.

Applicheremo questi metodi al dataset olive considerando come modello di clustering il K-means. Il metodo del gomito misura approssimativamente la qualità del cluster attraverso un’analisi della sua coesione in termini di somma dei quadrati all’interno del cluster (WSS), che rappresenta una misura di coesione o compattezza

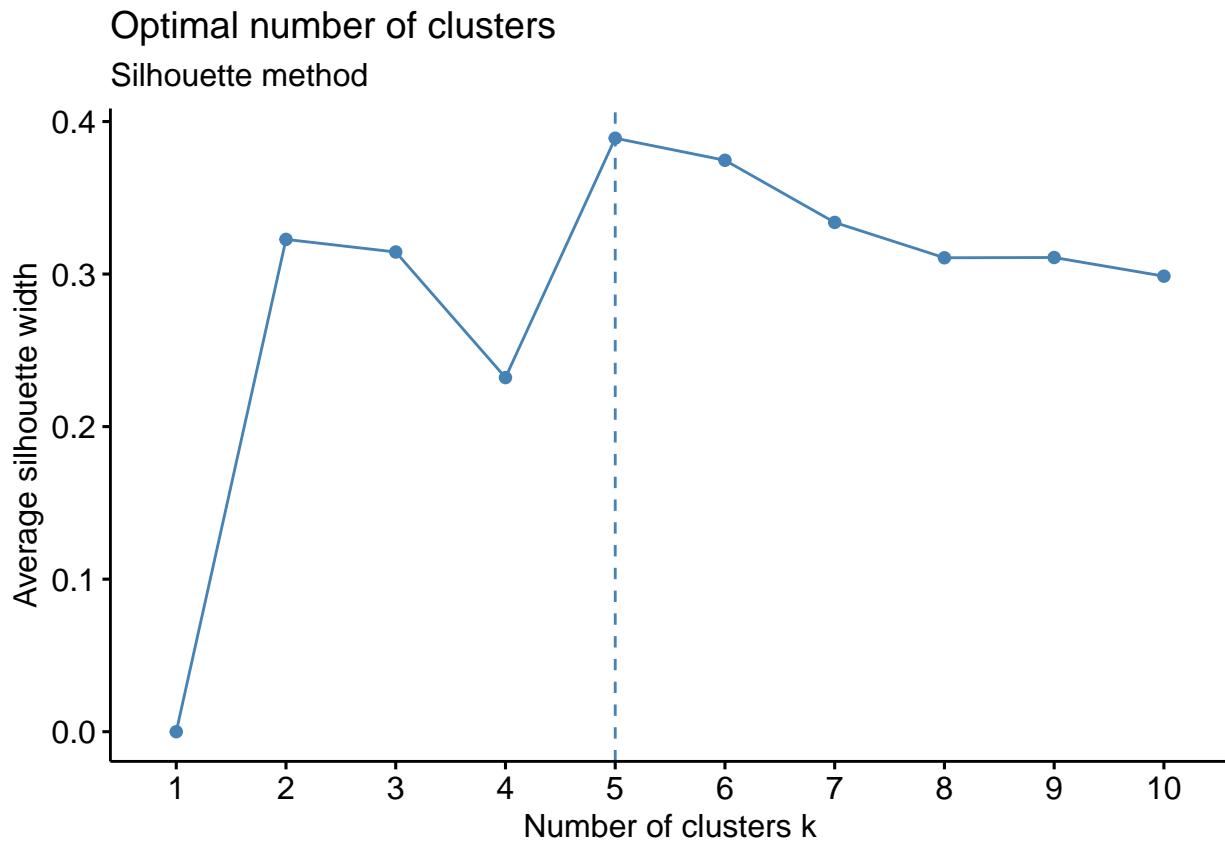
dei cluster. La funzione fviz_nbclust utilizza la distanza euclidea di default come misura di dissimilarità. Nel grafico sottostante, si mostra la WSS infunzione del numero di cluster K. Si osserva come WSS diminuisce all'aumentare di K. Per K=5 si può osservare un “gomito” che indica come all'aumentare di K il valore di WSS non tende più a diminuire vertiginosamente. Pertanto K=5 risulta essere il numero di cluster ottimale per tale metodologia.

```
fviz_nbclust(DB2S, #data scaled
            kmeans, #function
            method = "wss")+
  labs(subtitle = "Elbow method")
```



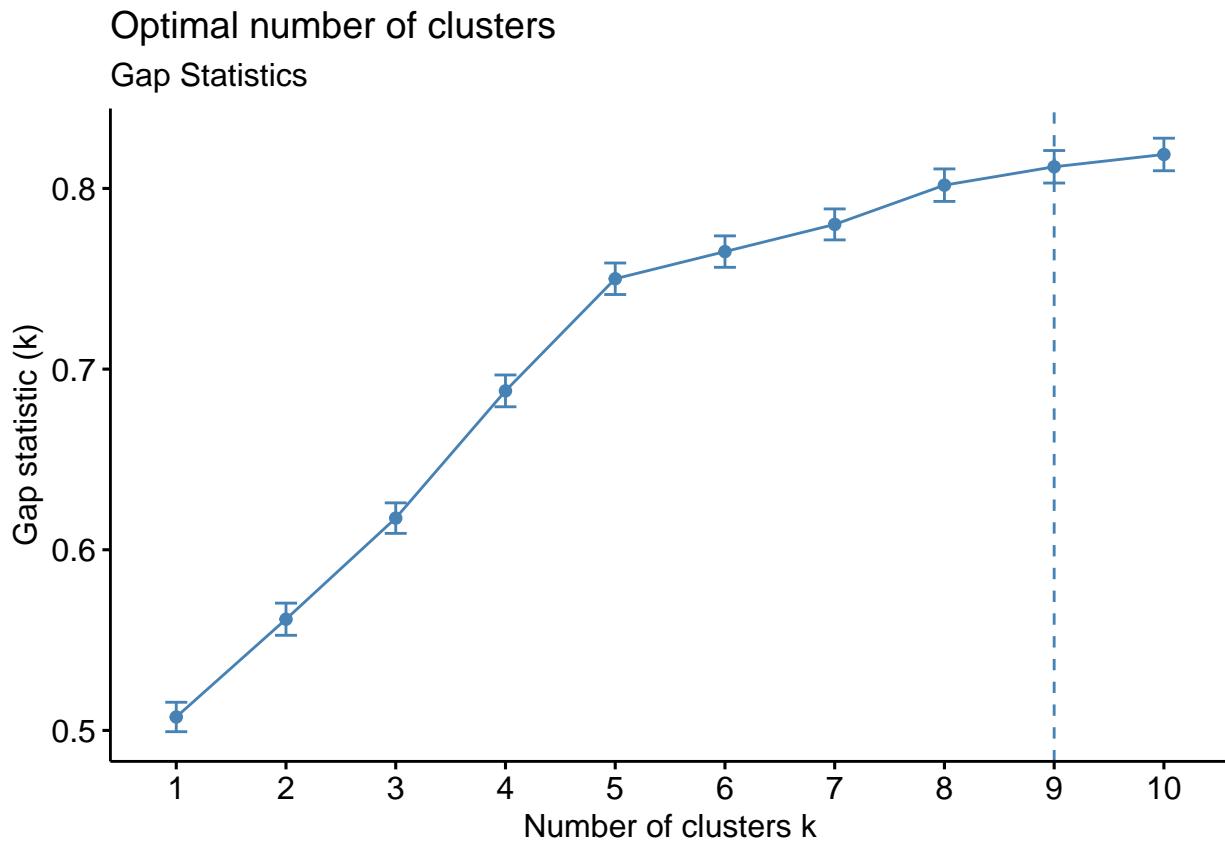
Il grafico seguente mostra il numero di cluster individuato con metodo diretto della silhouette, rappresentato dal numero corrispondente al punto in cui il valore di “avarage silhouette” è massimo. Il numero di cluster “consigliato” risulta essere 5.

```
fviz_nbclust(DB2S , kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



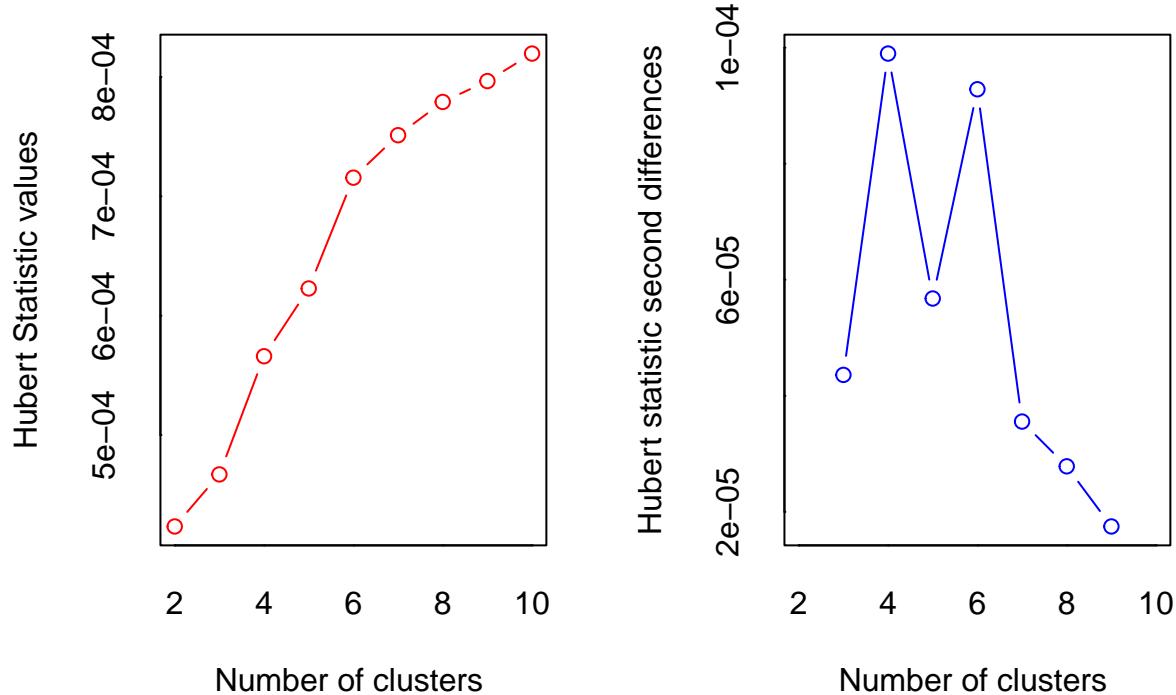
Si procede all'applicazione della seconda metodologia, la Gap Statistic. L'obiettivo di tale tecnica è fornire una procedura per formalizzare il metodo euristico del gomito. Per individuare il numero ottimale di cluster, si cerca il punto in cui il valore della gap statistic raggiunge il massimo o si stabilizza per la prima volta, che nel caso in esame risulta essere K=9.

```
set.seed(123)
fviz_nbclust(DB2S , kmeans, nstart=25,method = "gap_stat",nboot=50) +
  labs(subtitle = "Gap Statistics")
```



Esiste anche una funzione NBClust che analizza contemporaneamente diverse tecniche di individuazione del numero di cluster e propone come soluzione migliore il numero di cluster proposto dalla maggior parte delle tecniche. Nel caso in esame, il numero di cluster restituito dalla funzione è $k=5$.

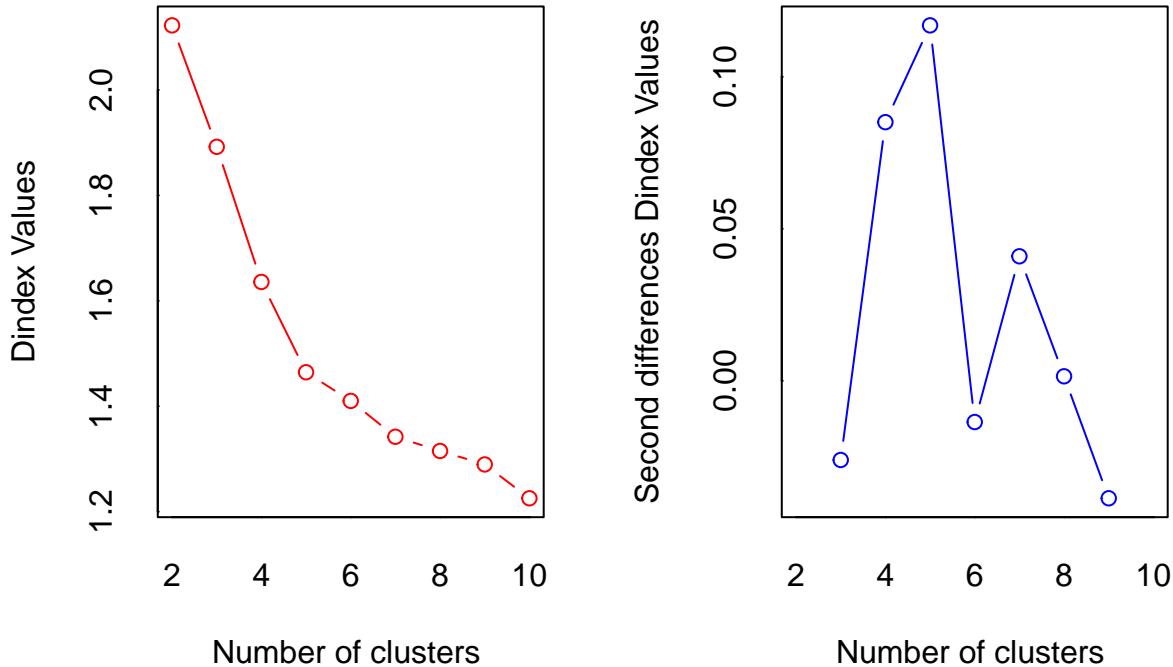
```
km <- NbClust(DB2S,distance = "euclidean", min.nc = 2, max.nc = 10, method = "kmeans")
```



```

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

```



```

## *** : The D index is a graphical method of determining the number of clusters.
## In the plot of D index, we seek a significant knee (the significant peak in Dindex
## second differences plot) that corresponds to a significant increase of the value of
## the measure.
##
## *****
## * Among all indices:
## * 4 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 3 proposed 4 as the best number of clusters
## * 9 proposed 5 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 3 proposed 10 as the best number of clusters
## *
## ***** Conclusion *****
## *
## * According to the majority rule, the best number of clusters is 5
## *
## *****

```

Clusterizzazione Gerarchica: metodo agglomerativo

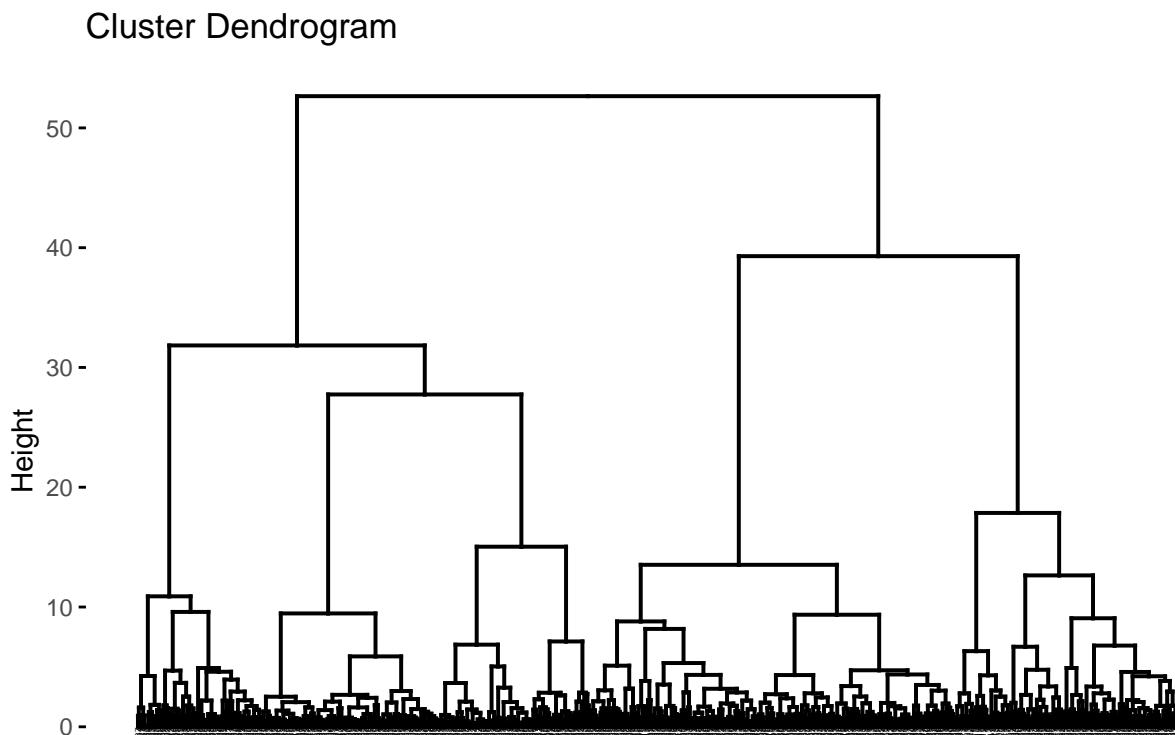
Il clustering gerarchico è una metodologia di clusterizzazione che costruisce una gerarchia di cluster e può essere di due tipi: agglomerativo o divisivo. In questo progetto verrà applicato il clustering gerarchico agglomerativo, con un approccio “Bottom Up”, che consiste nell’attribuire inizialmente ogni osservazione a cluster singoli e successivamente raggrupparli a coppie formando dei “grappoli” che vengono a loro volta uniti, in base alle criterio di similarità scelto, fino a convergere in un unico grande cluster. Tale metodo non presuppone di stabilire un numero di cluster a priori.

Gli algoritmi di clustering gerarchico differiscono tra loro in funzione del metodo scelto per misurare la distanza tra le osservazioni e per il metodo di collegamento scelto per definire e valutare la dissimilarità tra i cluster. Si procede con la creazione dell’albero gerarchico utilizzando la funzione “*hclust*”. Occorre inoltre stabilire quale metodo utilizzare per raggruppare le coppie di osservazioni in cluster rispetto alla loro similarità. Tra i metodi da poter utilizzare troviamo:

- *Single linkage method*: la distanza tra due cluster A e B viene definita come la distanza più piccola individuata tra le unità di A e le unità di B;
- *Complete linkage method*: la distanza tra due cluster A e B viene definita come la distanza maggiore individuata tra le unità di A e le unità di B;
- *Average linkage method*: la distanza tra i cluster A e B viene definita come la media tra le distanze delle unità di A e le unità di B;
- *Centroid linkage method*: la distanza tra cluster A e B viene definita come la distanza tra i centri di A e B determinati dai vettori medi campionari (x_A e x_B). Una volta fusi il baricentro del nuovo insieme può essere calcolato in funzione di x_A e x_B utilizzando la proprietà associativa della media aritmetica;
- *Ward’s (minimum deviance) method*: basato sulla somma delle deviazioni di ciascuna variabile.

Il grafico sottostante mostra il dendogramma ottenuto utilizzando il metodo Ward per misurare la distanza.

```
DF_dd <- dist(DB2S, method = "euclidean")
DF_dd_hc <- hclust(d = DF_dd, method = "ward.D2")
fviz_dend(DF_dd_hc, cex = 0.2)
```



Per valutare se la clusterizzazione è appropriata è possibile effettuare la correlazione tra “cophenetic distances”, misura di quanto due unità devono essere simili per essere raggruppate nello stesso cluster, e la distanza

originale generata dalla funzione “dist()”. Più il coefficiente di correlazione sarà vicino al valore 1, più il metodo di clusterizzazione sarà efficiente.

```
DF_dd_coph <- cophenetic(DF_dd_hc)
cor(DF_dd, DF_dd_coph)
```

```
## [1] 0.6521319
```

Il coefficiente di correlazione non è molto alto, si procede quindi con il calcolare la correlazione in funzione degli altri metodi e valutare quello che offre una migliore rappresentazione dei dati.

```
DF_dd_hc2 <- hclust(d = DF_dd, method = "single") #metodo1
DF_dd_hc3 <- hclust(d = DF_dd, method = "complete") #metodo2
DF_dd_hc4 <- hclust(d = DF_dd, method = "mcquitty") #metodo3
DF_dd_hc5 <- hclust(d = DF_dd, method = "median") #metodo4
DF_dd_hc6 <- hclust(d = DF_dd, method = "average") #metodo5

DF_dd_coph2 <- cophenetic(DF_dd_hc2)
DF_dd_coph3 <- cophenetic(DF_dd_hc3)
DF_dd_coph4 <- cophenetic(DF_dd_hc4)
DF_dd_coph5 <- cophenetic(DF_dd_hc5)
DF_dd_coph6 <- cophenetic(DF_dd_hc6)

cor(DF_dd, DF_dd_coph2) #correlazione metodo1
```

```
## [1] 0.6294245
```

```
cor(DF_dd, DF_dd_coph3) #correlazione metodo2
```

```
## [1] 0.7017031
```

```
cor(DF_dd, DF_dd_coph4) #correlazione metodo3
```

```
## [1] 0.6022032
```

```
cor(DF_dd, DF_dd_coph5) #correlazione metodo4
```

```
## [1] 0.6837586
```

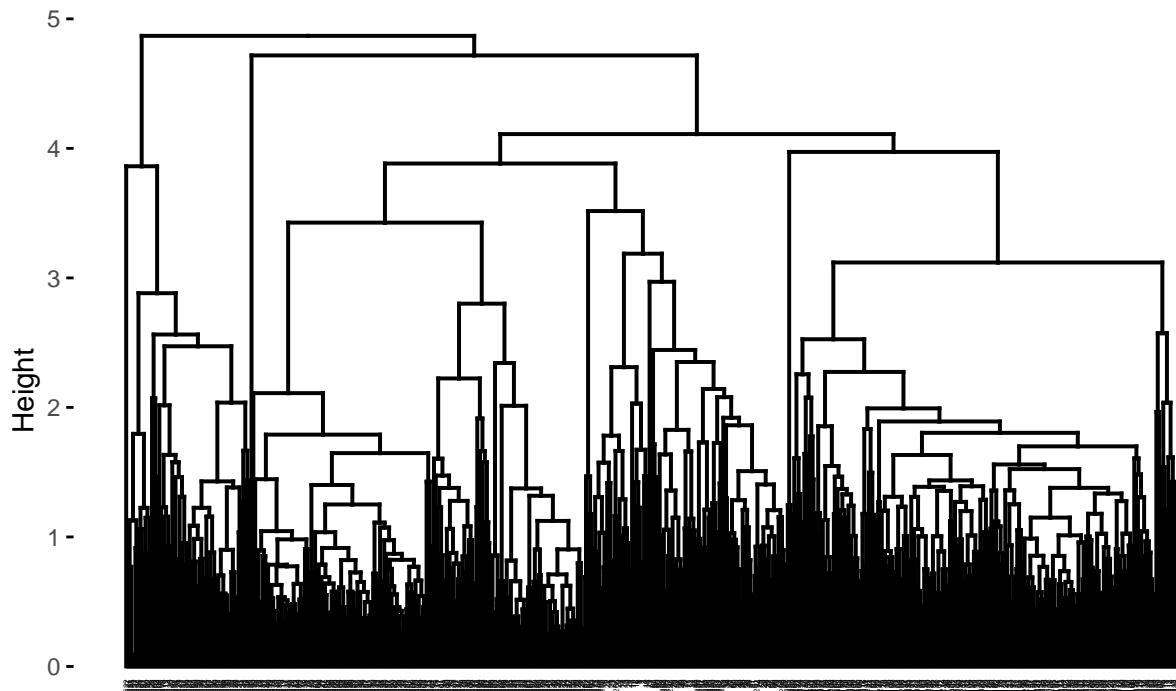
```
cor(DF_dd, DF_dd_coph6) #correlazione metodo5
```

```
## [1] 0.7240629
```

Alla luce dei risultati si può affermare che l’average linkage method sia il metodo migliore da adottare per la clusterizzazione. In grafico seguente mostra il dendogramma ottenuto utilizzando l’average linkage method come metodo di misura della distanza, ed è anche meno sensibile agli outliers rispetto ad altri metodi più comuni.

```
fviz_dend(DF_dd_hc6, cex = 0.2)
```

Cluster Dendrogram



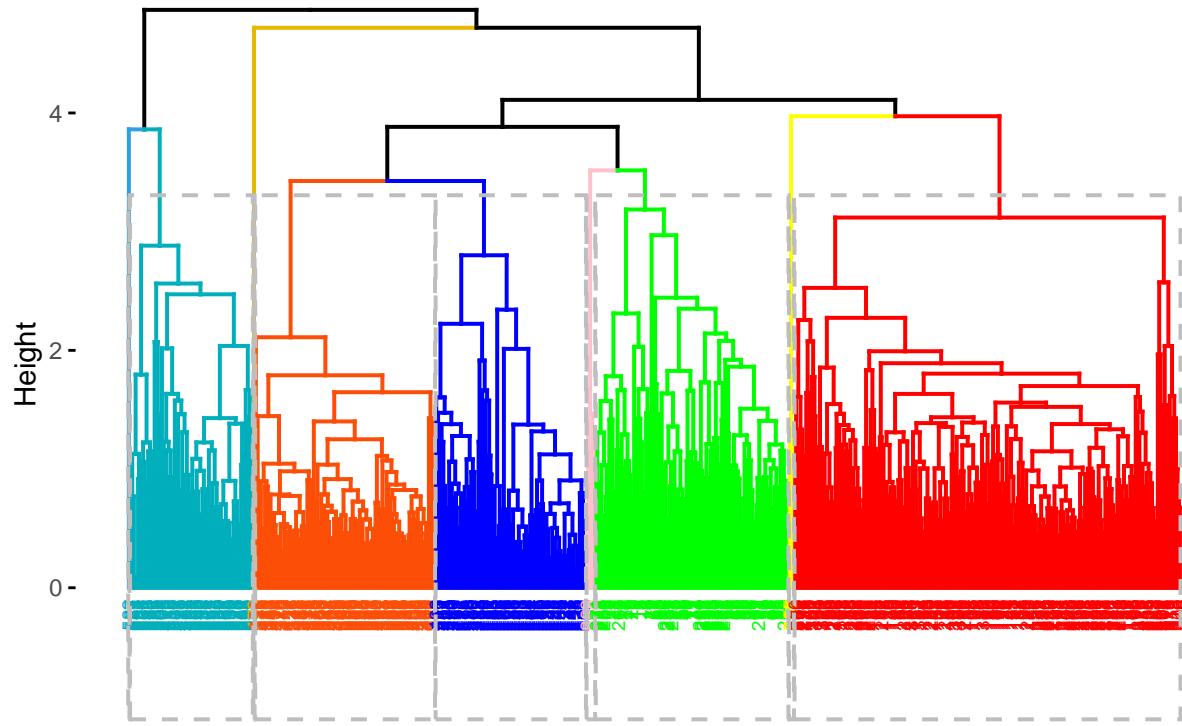
La clusterizzazione gerarchica non dà indicazioni in merito al numero di cluster da individuare, si può decidere liberamente dove tagliare l'albero specificando il numero di gruppi desiderati. Di seguito un esempio con 9 cluster:

```
grp<- cutree(DF_dd_hc6, k=9)
table(grp)

## grp
##   1   2   3   4   5   6   7   8   9
## 105 210  3  5 98  82  1 67  1

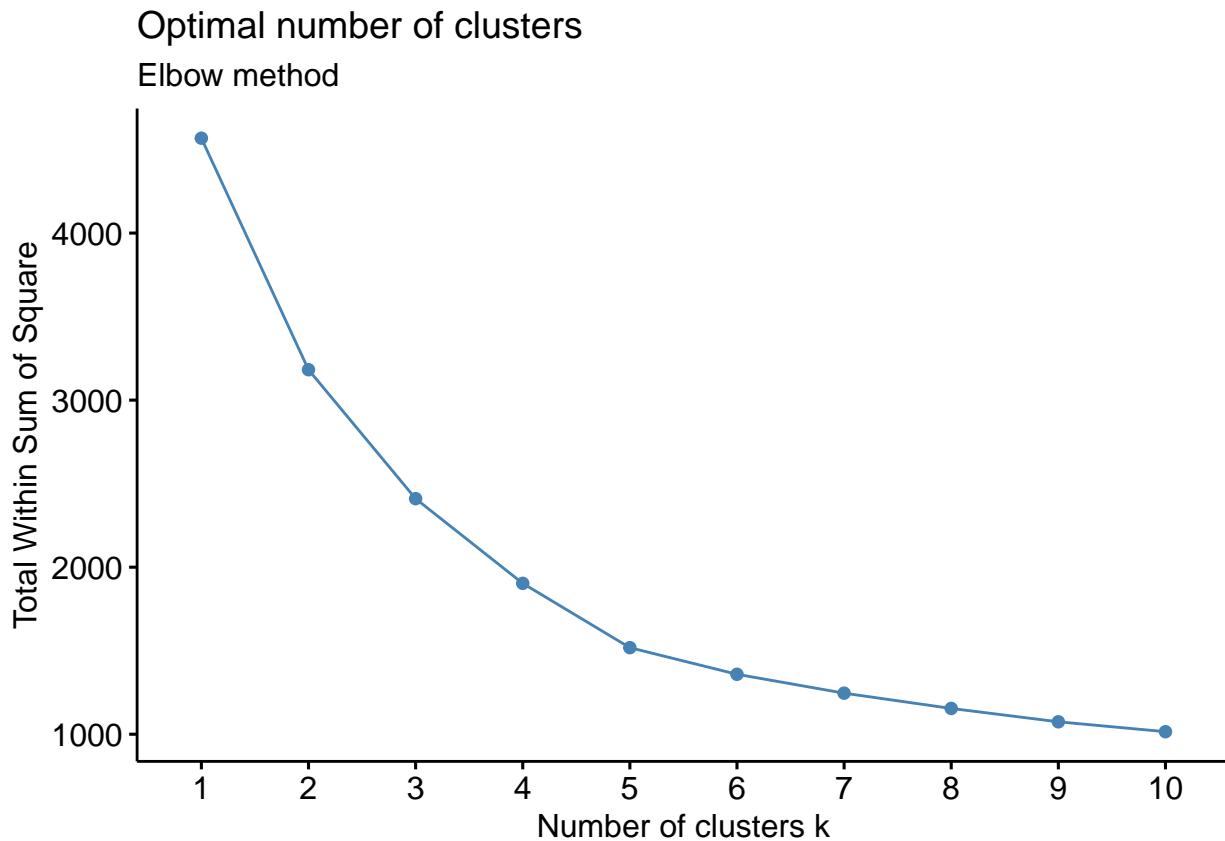
fviz_dend(DF_dd_hc6, k = 9, # Cut in four groups
          cex = 0.5, # label size
          k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07", "blue", "pink",
                      "green", "yellow", "red"),
          color_labels_by_k = TRUE, # color labels by groups
          rect = TRUE # Add rectangle around groups
)
```

Cluster Dendrogram



Si vuole individuare il numero ottimale di cluster in cui dividere il dendogramma. Tra le metodologie da poter utilizzare si decide di concentrarsi su “elbow method” e “silhouette method”.

```
fviz_nbclust(DB2S, hcut, method = "wss")+
  labs(subtitle = "Elbow method")
```



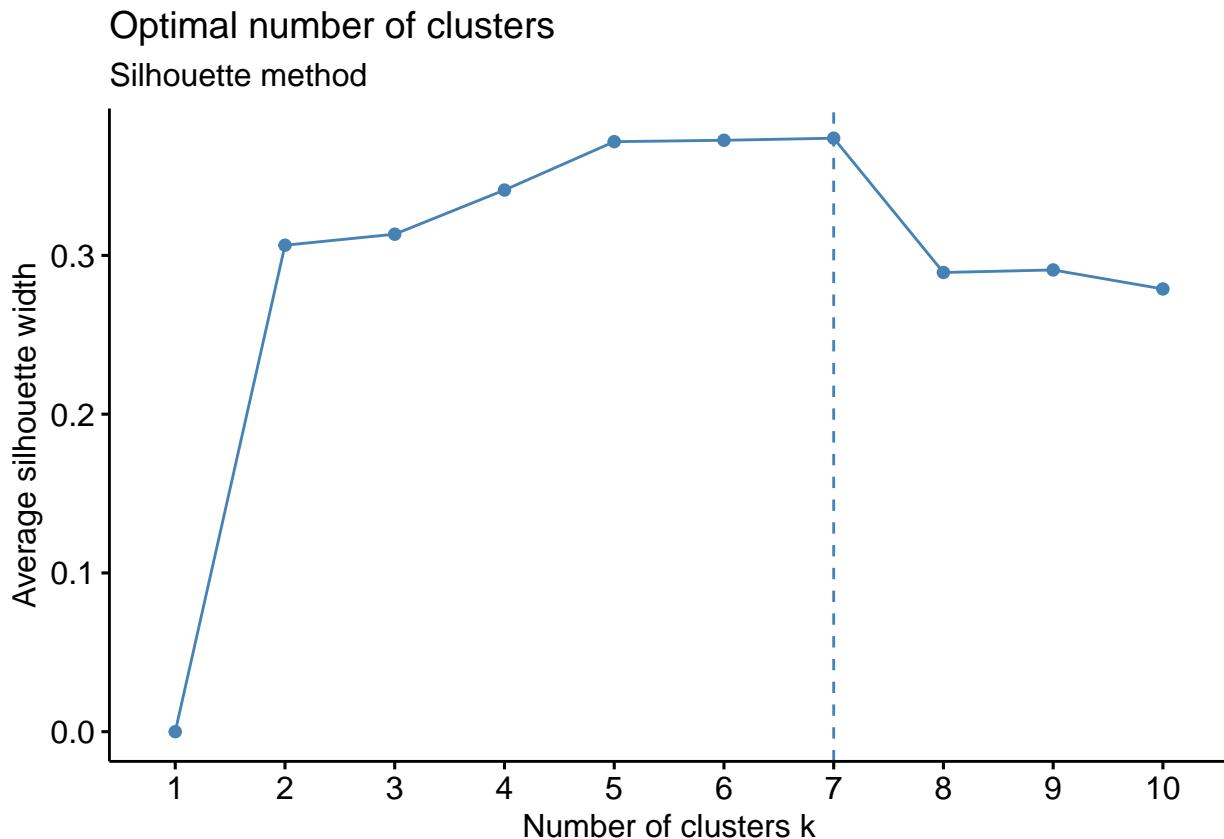
Nel caso del metodo di “Elbow” si ottiene un numero di cluster ottimale pari a 5 e le osservazioni sono distribuite nei cluster come nella seguente tabella:

```
grp<- cutree(DF_dd_hc6, k=5)
table(grp)
```

```
## grp
##   1   2   3   4   5
## 290 210   3   1  68
```

Nel caso del metodo di “Silhouette” otteniamo un numero di cluster ottimale pari a 7 e le osservazioni sono distribuite come nella tabella seguente:

```
fviz_nbclust(DB2S, hcut, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



```
grp <- cutree(DF_dd_hc6, k=7)
table(grp)
```

```
## grp
##   1   2   3   4   5   6   7
## 110 210  3 180   1  67   1
```

Algoritmi di partizione

K-means e K-medoids sono algoritmi che appartengono al gruppo degli algoritmi di partizione. In entrambe le metodologie il numero di cluster K si stabilisce a priori.

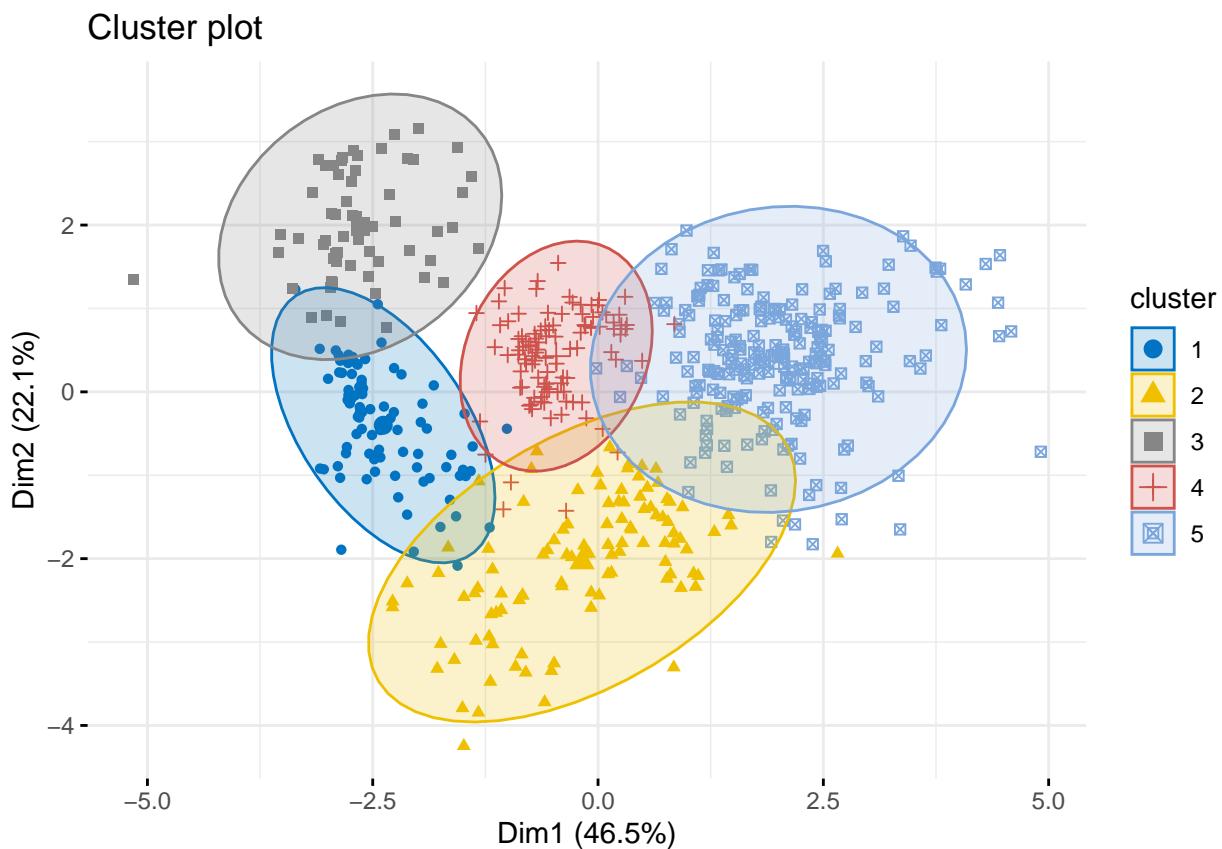
Il clustering K-Means rappresenta uno dei più diffusi algoritmi di partizionamento utilizzati per l'analisi dei dati. L'obiettivo principale di questo algoritmo è suddividere le n unità di dati in K cluster in modo che le unità all'interno dello stesso cluster siano il più simili possibile, garantendo così un'elevata coesione dei cluster, mentre le unità appartenenti a cluster diversi siano il più dissimili possibile, garantendo una elevata separazione tra i cluster. Per raggiungere questo obiettivo, l'algoritmo utilizza la somma dei quadrati delle distanze euclidi (WSS) tra i punti assegnati al cluster e la media di quei punti. È importante ricordare che il WSS rappresenta una misura della dissimilarità all'interno dei cluster. Tale algoritmo seleziona casualmente K centroidi e assegna ciascun punto al cluster del centroide più vicino. Successivamente, i centroidi vengono aggiornati iterativamente fino a quando non viene raggiunta una convergenza, minimizzando così il WSS complessivo e producendo i cluster finali. Inoltre, il K-means è sensibile agli outliers, a differenza del K-medoids che è più robusto. Nei grafici successivi si applica il modello utilizzando come numeri di cluster K=5 e K=9 ottenuti nei paragrafi precedenti.

```
#K=5
set.seed(123)
km.res <- kmeans(DB2S, 5, nstart = 25)
```

```

cluster.km.1 <- km.res$cluster
km.ress <- eclust(DB2S,"kmeans",k=5, nstart=25,graph = FALSE)
fviz_cluster(km.ress,geom="point", ellipse.type = "norm", palette="jco", ggtheme = theme_minimal())

```



```

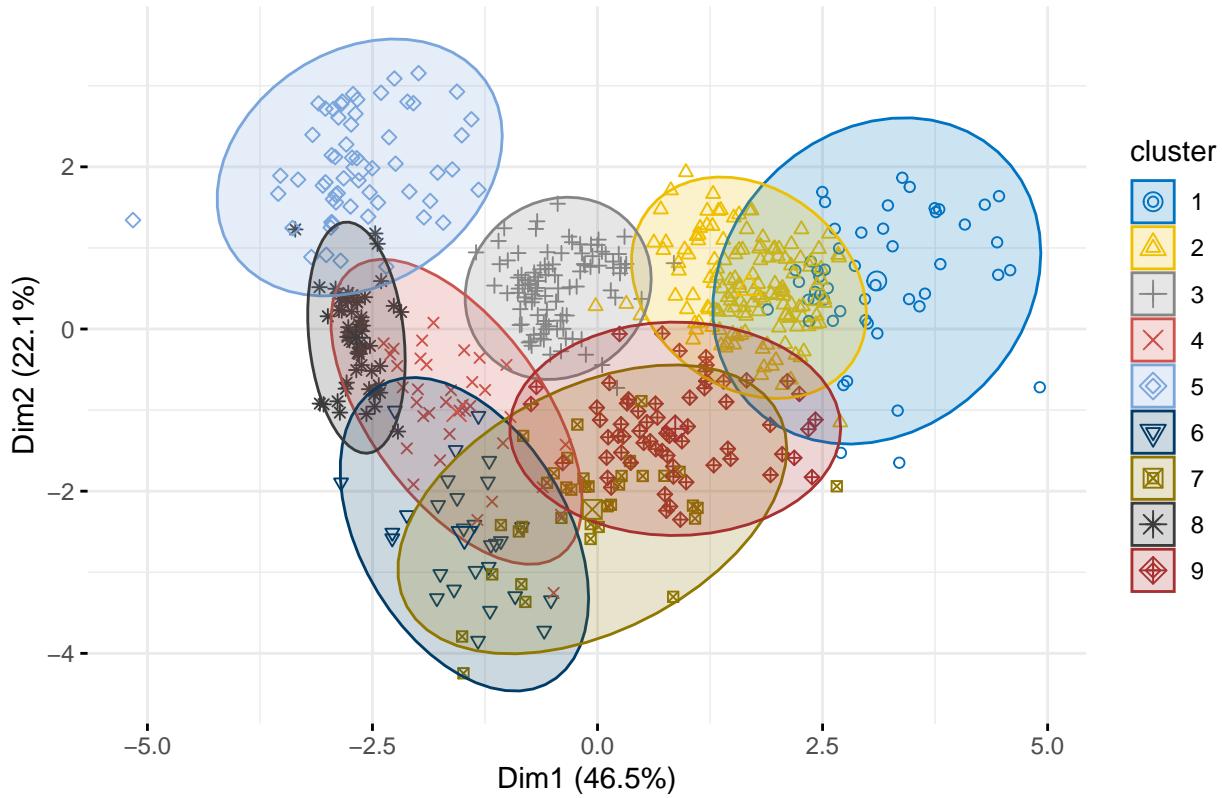
# K=9
set.seed(123)
km.res9 <- kmeans(DB2S,9,nstart = 25)

cluster.km.9 <- km.res9$cluster

km.ress9 <- eclust(DB2S,"kmeans",k=9, nstart=25,graph = FALSE)
fviz_cluster(km.ress9,geom="point", ellipse.type = "norm", palette="jco", ggtheme = theme_minimal())

```

Cluster plot

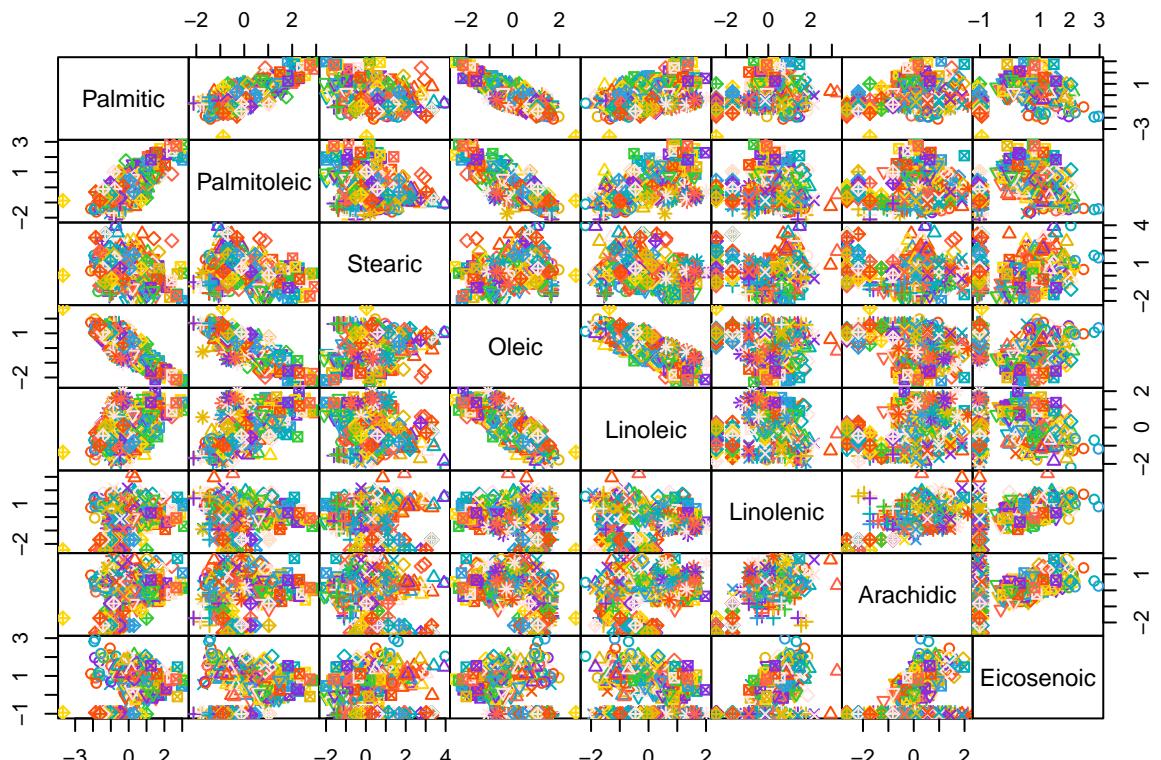
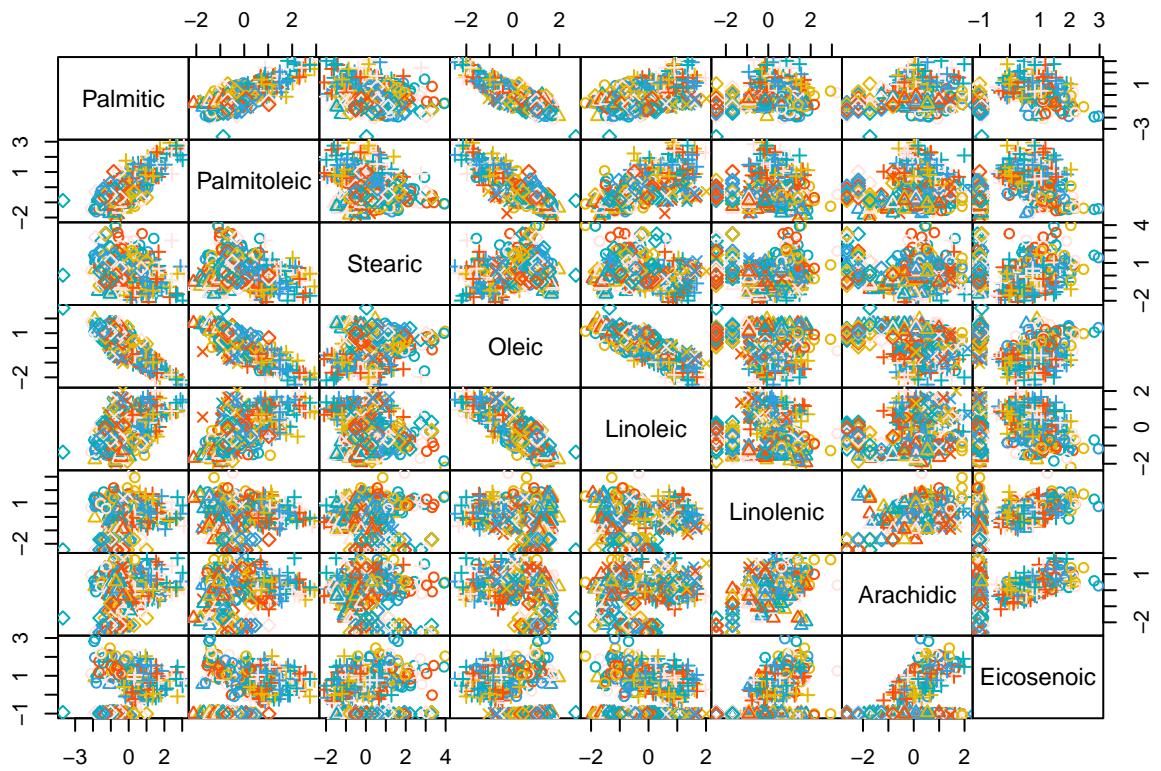


E' possibile osservare come con K=9 il 77.7% della variabilità totale dei dati è spiegato dalla distribuzione dei dati tra i cluster rispetto al 68% che si ottiene con K=5. Un possibile soluzione è l'adozione del PAM il quale risulta essere meno sensibile agli outliers. A differenza del K-means, che utilizza i centroidi, nel clustering K-medoids ogni cluster è rappresentato da uno dei punti dati all'interno del cluster stesso, chiamato medoide del cluster. Il termine "medoide" si riferisce a un'unità all'interno di un cluster per cui la dissimilarità media tra questa e tutti gli altri membri del cluster è minima. Il medoide corrisponde al punto posizionato più al centro del cluster. Queste unità possono essere considerate esempi rappresentativi dei membri del cluster in cui si trovano.

Il PAM (Partitioning Around Medoids) è un algoritmo di clustering il cui obiettivo è di trovare i medoidi dei cluster in modo che la somma delle dissimilarità tra i punti dati e i rispettivi medoidi sia minimizzata. L'algoritmo inizia selezionando casualmente K medoidi e quindi assegna iterativamente ciascun punto al medoide più vicino. Successivamente, aggiorna i medoidi per minimizzare la somma delle dissimilarità. Il processo continua fino a quando non viene raggiunta la convergenza e i medoidi rimangono stabili. Il PAM è particolarmente utile quando i dati hanno una struttura non lineare o quando i cluster hanno forme irregolari, poiché utilizza i medoidi che sono punti reali nel dataset. Si applica il modello utilizzando come numeri di cluster K=5 e K=9 mostrati nei seguenti pairplot.

```
#K=5
pam.res5 <- pam(DB2S,5,metric = "euclidean")

clusterPam5 <- pam.res5$clustering
pairs(DB2S, gap=0, pch=clusterPam5,col=c("#2E9FDF", "#00AFBB", "#E7B800", "#FFE4E1", "#FC4E07") )
```



Cluster validation statistics

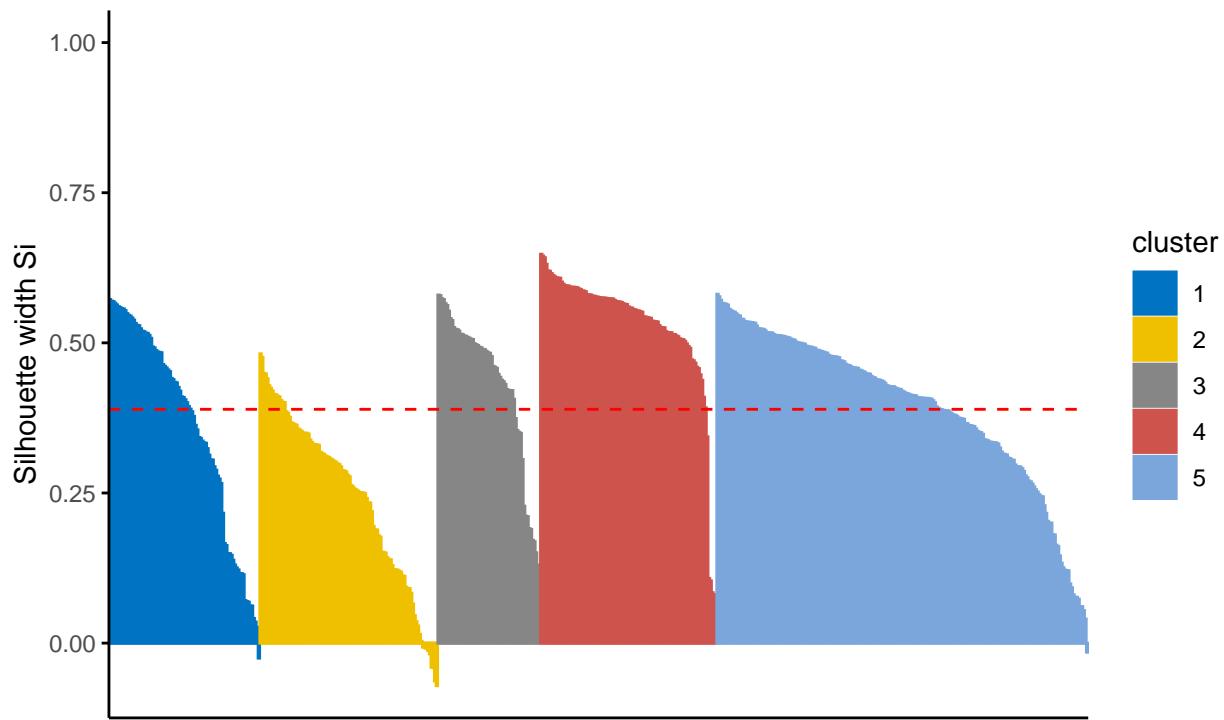
La validazione del clustering è fondamentale per garantire che i risultati ottenuti dall'algoritmo di clustering siano affidabili e significativi. Ciò implica valutare se i cluster formati dall'algoritmo sono coerenti e rappresentano correttamente le strutture presenti nei dati. Esistono diverse tecniche e metriche di validazione del clustering, tra cui l'indice di validità silhouette, l'indice di Dunn, e l'indice Rand. Questi strumenti consentono di valutare la coerenza interna dei cluster e confrontare l'efficacia di diversi algoritmi di clustering nell'identificare strutture nei dati. Inoltre, la validazione del clustering è importante per evitare il fenomeno dell' overfitting, in cui l'algoritmo di clustering trova pattern casuali nei dati anziché veri raggruppamenti significativi. Inoltre consente di confrontare due algoritmi di clustering per determinare quale sia più adatto per un determinato insieme di dati. La validazione interna del clustering utilizza le informazioni interne del processo di clustering per valutare la bontà di una struttura di clustering senza fare riferimento a informazioni esterne. Può essere utilizzata anche per stimare il numero di cluster e l'algoritmo di clustering appropriato senza alcun dato esterno. La validazione esterna del clustering consiste nel confrontare i risultati di un'analisi dei cluster con un risultato noto esternamente, come etichette di classe fornite esternamente. Poiché conosciamo il numero “vero” di cluster in anticipo, questo approccio è principalmente utilizzato per selezionare l'algoritmo di clustering corretto per un determinato insieme di dati. La validazione relativa del clustering tiene conto di diversi valori dei parametri per lo stesso algoritmo (ad esempio, variando il numero di cluster K).

Nel grafico seguente si valida il metodo della silhouette utilizzando il numero di cluster per il K-means applicato precedentemente per K=5 e K=9.

```
#K=5
fviz_silhouette(km.ress, palette="jco", ggtheme=theme_classic())

##   cluster size ave.sil.width
## 1       1    88      0.36
## 2       2   104      0.24
## 3       3    60      0.43
## 4       4   103      0.53
## 5       5   217      0.39
```

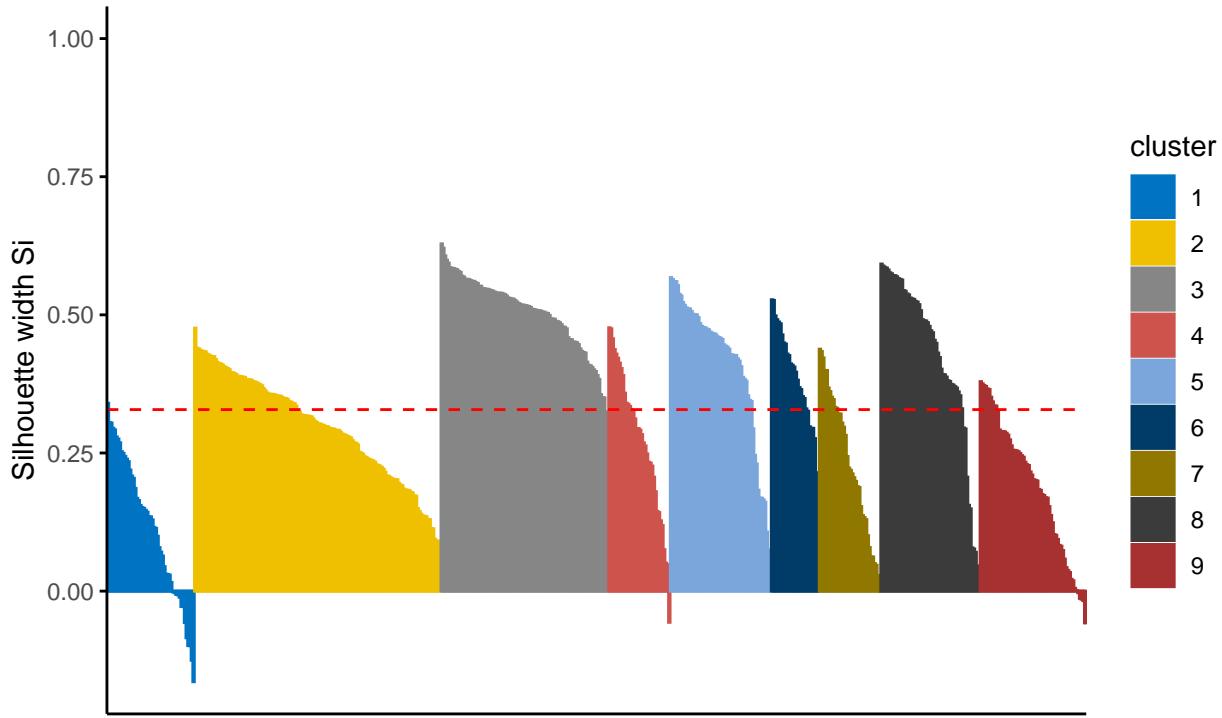
Clusters silhouette plot
Average silhouette width: 0.39



```
#K=9  
fviz_silhouette(km.ress9, palette="jco", ggtheme=theme_classic())
```

```
##   cluster size ave.sil.width  
## 1       1   51      0.11  
## 2       2  144      0.30  
## 3       3   98      0.50  
## 4       4   36      0.28  
## 5       5   59      0.41  
## 6       6   28      0.38  
## 7       7   36      0.23  
## 8       8   58      0.43  
## 9       9   62      0.19
```

Clusters silhouette plot
Average silhouette width: 0.33



La silhouette per k=5 ha valori medi più alti, indicando una migliore struttura di clustering.

Si procede ad esaminare il Dunn Index. Il Dunn Index è una misura di validazione interna utilizzata per valutare la bontà di un clustering. Questa misura valuta la coesione all'interno dei cluster e la separazione tra i cluster. Un valore più alto del Dunn Index indica una migliore qualità del clustering, con cluster più compatti e ben separati.

```
km5stats <- cluster.stats(dist(DB2S), km.ress$cluster)
km9stats <- cluster.stats(dist(DB2S), km.ress9$cluster)

km5stats$dunn

## [1] 0.08810187
km9stats$dunn

## [1] 0.07393023
```

Possiamo osservare come il Dunn Index assumere valori più prossimi a 1 per K=5

La validazione esterna del clustering confronta i risultati con un'annotazione esterna o un insieme di etichette note. Questo può essere particolarmente utile quando si ha a disposizione un insieme di dati di riferimento già etichettati, ad esempio, se si sta valutando un algoritmo di clustering supervisionato o si dispone di informazioni di classe esterne. L'indice di Rand corretto è una misura che valuta quanto i cluster identificati corrispondano alle etichette di classe note, considerando le associazioni corrette e scorrette tra i cluster e le etichette di classe ed è più attendibile tanto più si avvicina al valore 1. L'indice di variazione di Meila è un'altra misura che fornisce una valutazione simile, ma con alcuni adattamenti specifici. Questi strumenti di validazione esterna del clustering consentono di valutare quanto bene i cluster identificati rappresentino le strutture presenti nei dati rispetto alle informazioni esterne disponibili, e risulta più efficace tanto più si avvicina a 0. Inoltre, forniscono un modo per confrontare l'efficacia di diversi algoritmi di clustering nel contesto specifico del dataset e delle informazioni di riferimento disponibili. Facciamo il confronto con la

variabile Areee del dataset in esame per i cluster K=5 e K=9.

```
#External validation
Areee <- as.numeric(olive$Area)

#K=5
clust_statskm5 <- cluster.stats(d= dist(DB2S), Areee, km.ress$cluster)
#indice di variazione di Rand
clust_statskm5$corrected.rand

## [1] 0.760864

#indice di variazione di Meila
clust_statskm5$vi

## [1] 0.8591506

#K=9
clust_statskm9 <- cluster.stats(d= dist(DB2S), Areee, km.ress9$cluster)
#indice di variazione di Rand
clust_statskm9$corrected.rand

## [1] 0.630525

#indice di variazione di Meila
clust_statskm9$vi

## [1] 0.9745949
```

I risultati di entrambi gli indici applicati mostrano che il clustering con K-means sembrano avere prestazioni migliori con numero di cluster pari a 5..

Infine applichiamo in metodi relativi di cluster validation su diversi algoritmi per ottenere ulteriori informazioni in merito alla bontà del modello.

```
rownames(DB2S) <- rownames(olive)

metodi <- c("hierarchical", "kmeans", "pam")
intern <- clValid(DB2S, nClust= 5:9., clMethods=metodi, validation= "internal")
summary(intern)

## 
## Clustering Methods:
##   hierarchical kmeans pam
## 
## Cluster sizes:
##   5 6 7 8 9
## 
## Validation Measures:
##                               5       6       7       8       9
## 
##   ## hierarchical Connectivity 32.2246 33.1341 36.0631 43.2567 44.3524
##   ##           Dunn          0.0969  0.1072  0.1072  0.1072  0.1266
##   ##           Silhouette  0.2543  0.3239  0.3168  0.3028  0.3426
##   ## kmeans      Connectivity 62.7448 89.0885 76.2060 100.2131 103.8754
##   ##           Dunn          0.1011  0.1079  0.0824  0.0827  0.0683
##   ##           Silhouette  0.3277  0.3773  0.3708  0.3451  0.3424
##   ## pam         Connectivity 76.0845 65.3683 67.6829 122.9742 184.8016
##   ##           Dunn          0.0630  0.0881  0.0924  0.0835  0.0587
```

```

##          Silhouette      0.3891    0.3754    0.3743    0.2889    0.2885
##
## Optimal Scores:
##
##          Score   Method   Clusters
## Connectivity 32.2246 hierarchical 5
## Dunn        0.1266 hierarchical 9
## Silhouette   0.3891 pam      5
stab <- clValid(DB2S, nClust= 5:9., clMethods=metodi, validation= "stability")
summary(stab)

##
## Clustering Methods:
##   hierarchical kmeans pam
##
## Cluster sizes:
##   5 6 7 8 9
##
## Validation Measures:
##           5       6       7       8       9
##
##          hierarchical APN  0.2108 0.1705 0.2078 0.2283 0.1312
##                      AD   2.9177 2.6749 2.6107 2.5756 2.1635
##                      ADM  1.0226 0.9088 0.9075 0.9620 0.5107
##                      FOM  0.8577 0.8078 0.7993 0.7967 0.6490
##          kmeans     APN  0.1762 0.1221 0.1983 0.1957 0.2277
##                      AD   2.3679 2.1289 2.1116 2.0173 1.9651
##                      ADM  0.6458 0.4610 0.6542 0.5948 0.5718
##                      FOM  0.6902 0.6613 0.6490 0.6200 0.5898
##          pam       APN  0.0854 0.1057 0.1814 0.1966 0.2447
##                      AD   2.1580 2.0780 2.0180 1.9594 1.9234
##                      ADM  0.2799 0.3447 0.4526 0.4941 0.5845
##                      FOM  0.6385 0.6226 0.6113 0.5856 0.5852
##
## Optimal Scores:
##
##          Score   Method Clusters
## APN  0.0854 pam      5
## AD   1.9234 pam      9
## ADM  0.2799 pam      5
## FOM  0.5852 pam      9

```

Nella seguente tabella sono riportati i valori di:

- APN (Average Proximity to Nearest): Misura della distanza media tra ciascun punto e il suo punto più vicino nel cluster.
- AD (Average Diameter): Misura del diametro medio dei cluster, ovvero la massima distanza tra coppie di punti nello stesso cluster.
- ADM (Average Distance between Means): Misura della distanza media tra i centroidi dei cluster.
- FOM (Figure of Merit): Misura della somiglianza tra i cluster rispetto alla similarità attesa se i dati fossero distribuiti casualmente.

Tutte le analisi condotte, portano a scegliere come miglior numero di cluster, K=5.

Modelli di Mistura Gaussiana

I modelli di mistura gaussiana (GMM) rappresentano una forma di clustering basata su modelli che forniscono una probabilità di appartenenza a ciascun cluster. A differenza dei modelli di clustering come il K-means, i GMM identificano automaticamente il numero ottimale di cluster e consentono di raggruppare una più ampia varietà di forme e orientamenti di cluster. I GMM sono sensibili agli outliers. Per utilizzare i GMM si assume la normalità multivariata dei dati. Si utilizza il pacchetto mclust che verrà utilizzato per applicare i modelli GMM ai dati. Si applica la funzione Mclust che stima automaticamente il miglior modello di mistura gaussiana in base ai dati forniti.

```
#fit del modello GMM utilizzando la funzione Mclust
gmm <- mclust::Mclust(data = DB2S, verbose = FALSE)
summary(gmm)

## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
## 
## Mclust VVE (ellipsoidal, equal orientation) model with 9 components:
## 
## log-likelihood   n   df      BIC      ICL
##           -1823.343 572 180 -4789.531 -4842.973
## 
## Clustering table:
##   1   2   3   4   5   6   7   8   9
## 29  93  80 121  67  31  43  60  48
```

La tabella di clustering mostra il numero di osservazioni assegnate a ciascun cluster. L'output rivela 9 cluster con 29, 93, 80, 121, 67, 31, 43, 60, 48, data points, rispettivamente. Il termine "VVE" sta per "Varianza-equal shape" (varianza con forma uguale), e si riferisce a uno dei modelli di clustering forniti dai modelli GMM. Il modello VVE cerca di raggruppare le osservazioni in cluster in base alla loro somiglianza, assumendo che ciascun cluster abbia una forma ellissoidale e che le ellissi abbiano dimensioni uguali ma orientamenti diversi. Dalla tabella in output i parametri restituiti che misurano il fit del modello sono:

- Log-likelihood (Verosimiglianza): Misura la verosimiglianza dei dati osservati sotto il modello stimato. Un valore maggiore di log-likelihood indica un miglior adattamento del modello ai dati.
- n, Numero di osservazioni: Indica il numero totale di osservazioni nel dataset.
- df, Gradi di libertà: Rappresenta il numero di parametri stimati meno le restrizioni del modello. In un modello VVE, i gradi di libertà includono il numero di medie, varianze e probabilità di appartenenza ai cluster.
- BIC (Bayesian Information Criterion): È un criterio di selezione del modello che penalizza i modelli con un numero maggiore di parametri. Valori più bassi di BIC indicano una migliore adattabilità del modello. Nel package utilizzato Mclust, BIC ha segno negativo.
- ICL (Integrated Completed Likelihood): È un altro criterio di selezione del modello simile a BIC ma è specificamente sviluppato per modelli di mistura gaussiana. Come BIC, valori più bassi di ICL indicano una migliore adattabilità del modello.

Il seguente grafico mostra come il BIC cambia al variare di K (il numero di componenti nei modelli GMM). I modelli migliori sono quelli in cui il valore del BIC (Bayesian Information Criterion) è più basso (in valore assoluto). Questo indica che il modello è più parsimonioso e che si adatta meglio ai dati e ha una migliore capacità di generalizzazione.

```
# Grafico valore Bic in funzione del numero di cluster
gmm_BIC <- matrix(gmm$BIC, nrow = dim(gmm$BIC)[1], ncol = dim(gmm$BIC)[2]) %>%
  data.frame() %>%
  add_column(K = 1:9, .before=-1)
```

```

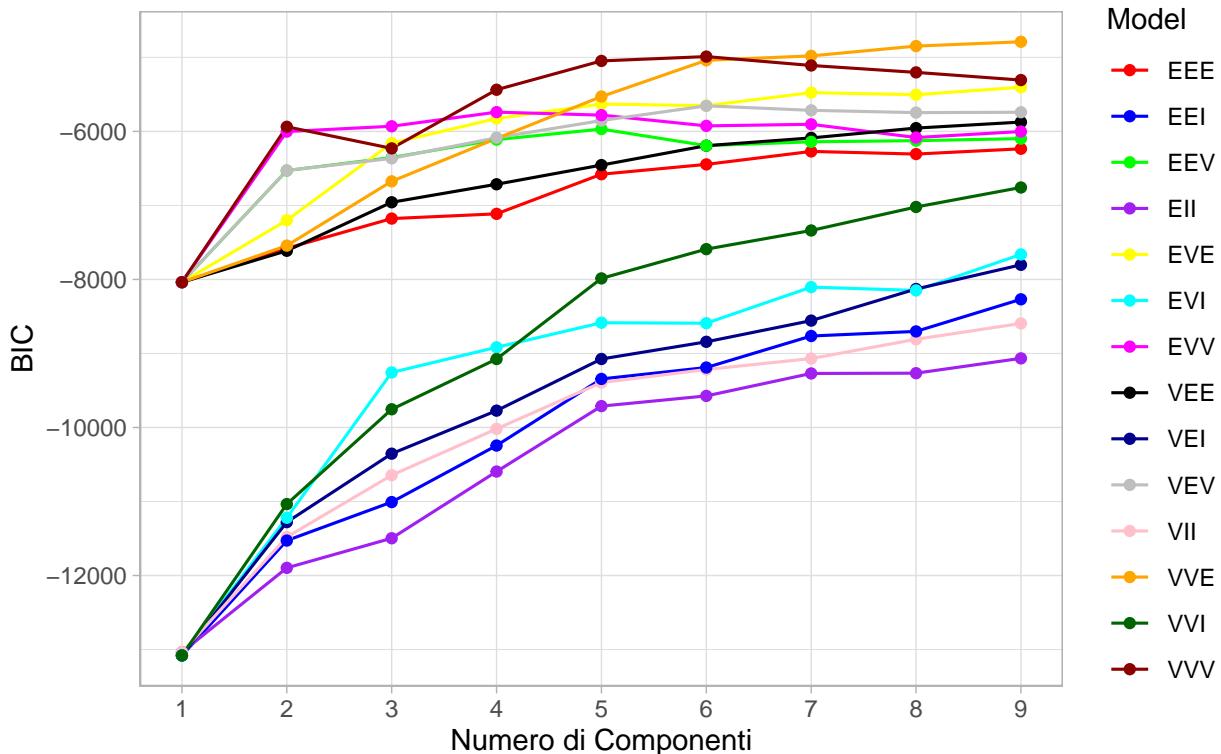
colnames(gmm_BIC) <- c("K", colnames(gmm$BIC))

gmm_BIC %>%
  gather(key = "Model", value = "BIC", -K) %>%
  filter(!is.na(BIC)) %>%
  ggplot(aes(x = K, y = BIC, col = Model)) +
  geom_line(size=0.55) +
  geom_point() +
  scale_color_manual(values = c("red", "blue", "green", "purple", "yellow", "cyan",
                               "magenta", "black", "darkblue", "grey", "pink",
                               "orange", "darkgreen", "darkred")) +
  scale_x_continuous(breaks = 1:9, minor_breaks = NULL) +
  labs(title = "Gaussian Mixture Models vs Cluster Number",
       subtitle = "Grafico BIC per modelli GMM",
       x = "Numero di Componenti") +
  theme_light()

```

Gaussian Mixture Models vs Cluster Number

Grafico BIC per modelli GMM



La funzione mclustBIC calcola i BIC per ogni modello e per ogni cluster. Dall' output si osserva che in base al criterio BIC si possono selezionare 3 modelli migliori, che corrispondono ai VVE che il modello aveva calcolato precedentemente con numeri di cluster preferibili da 7 a 9.

```

#bic per cluster e modello.
bic_values <- mclustBIC(data=DB2S, G=2:9)
bic_values

## Bayesian Information Criterion (BIC):
##          EII          VII          EEI          VEI          EVI          VVI          EEE
## 2 -11896.421 -11480.142 -11527.432 -11278.731 -11219.567 -11034.576 -7578.944

```

```

## 3 -11496.435 -10644.639 -11007.488 -10353.973 -9256.790 -9754.054 -7177.782
## 4 -10595.277 -10018.528 -10243.894 -9772.454 -8916.985 -9075.370 -7114.029
## 5 -9711.278 -9393.291 -9344.248 -9074.594 -8585.140 -7985.818 -6578.677
## 6 -9573.483 -9216.839 -9190.190 -8842.595 -8591.176 -7590.721 -6445.713
## 7 -9270.056 -9068.600 -8764.378 -8556.236 -8103.799 -7338.648 -6270.787
## 8 -9266.493 -8808.382 -8702.059 -8128.963 -8148.397 -7021.325 -6306.220
## 9 -9066.158 -8594.317 -8268.323 -7802.521 -7662.287 -6757.699 -6235.190
##          VEE      EVE      VVE      EEV      VEV      EVV      VVV
## 2 -7612.786 -7200.913 -7542.889 -6529.368 -6527.145 -6002.360 -5937.509
## 3 -6957.466 -6158.774 -6674.559 -6353.144 -6364.607 -5930.640 -6229.408
## 4 -6714.225 -5825.990 -6095.814 -6111.212 -6084.956 -5739.830 -5437.416
## 5 -6455.152 -5630.318 -5529.516 -5968.656 -5853.286 -5780.332 -5048.083
## 6 -6193.205 -5652.339 -5039.592 -6192.349 -5654.893 -5925.323 -4989.423
## 7 -6088.211 -5476.198 -4979.872 -6140.573 -5715.536 -5904.292 -5109.567
## 8 -5955.443 -5504.787 -4847.777 -6125.391 -5746.292 -6082.846 -5202.921
## 9 -5874.468 -5403.088 -4789.531 -6098.213 -5741.297 -6001.914 -5306.613
##
## Top 3 models based on the BIC criterion:
##      VVE,9      VVE,8      VVE,7
## -4789.531 -4847.777 -4979.872

```

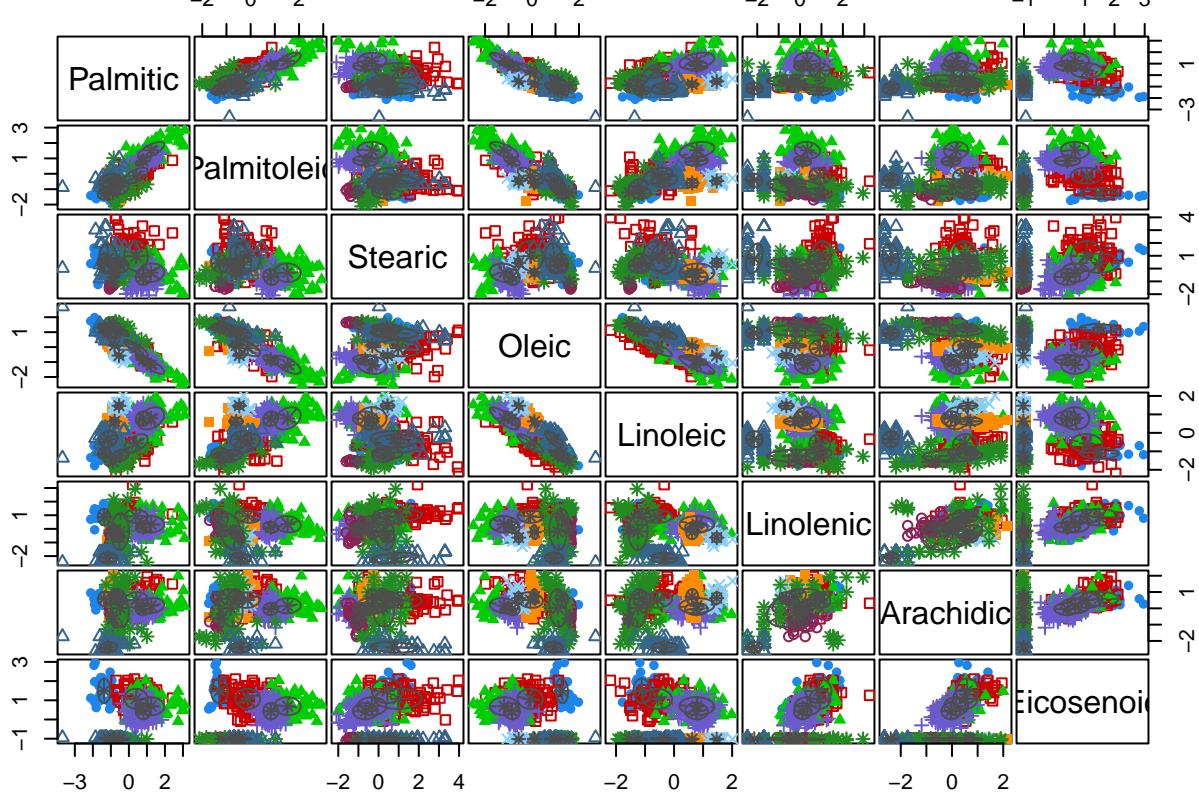
Nel grafico seguente si può osservare graficamente il risultato del clustering ottenuto. Le ellissi mostrano la forma approssimativa dei cluster individuati dal modello. I punti sono raggruppati per colore in base alla classificazione attribuita loro dal modello di clustering. Ogni colore rappresenta una diversa classe assegnata. Aggiungendo la funzione addEllipses = TRUE, verranno disegnate le ellissi che rappresentano la forma e l'orientamento delle distribuzioni gaussiane che approssimano i cluster individuati dal modello. Queste ellissi aiutano a visualizzare la struttura del cluster e la loro distribuzione.

```

#visualizzare graficamente come le osservazioni sono distribuite nei diversi cluster.
plot.Mclust(gmm, what = c("classification"), dimens = NULL, xlab = NULL, ylab = NULL,
            addEllipses = TRUE)
title("Distribuzione dei diversi cluster in funzione delle variabili", line= 3.35)

```

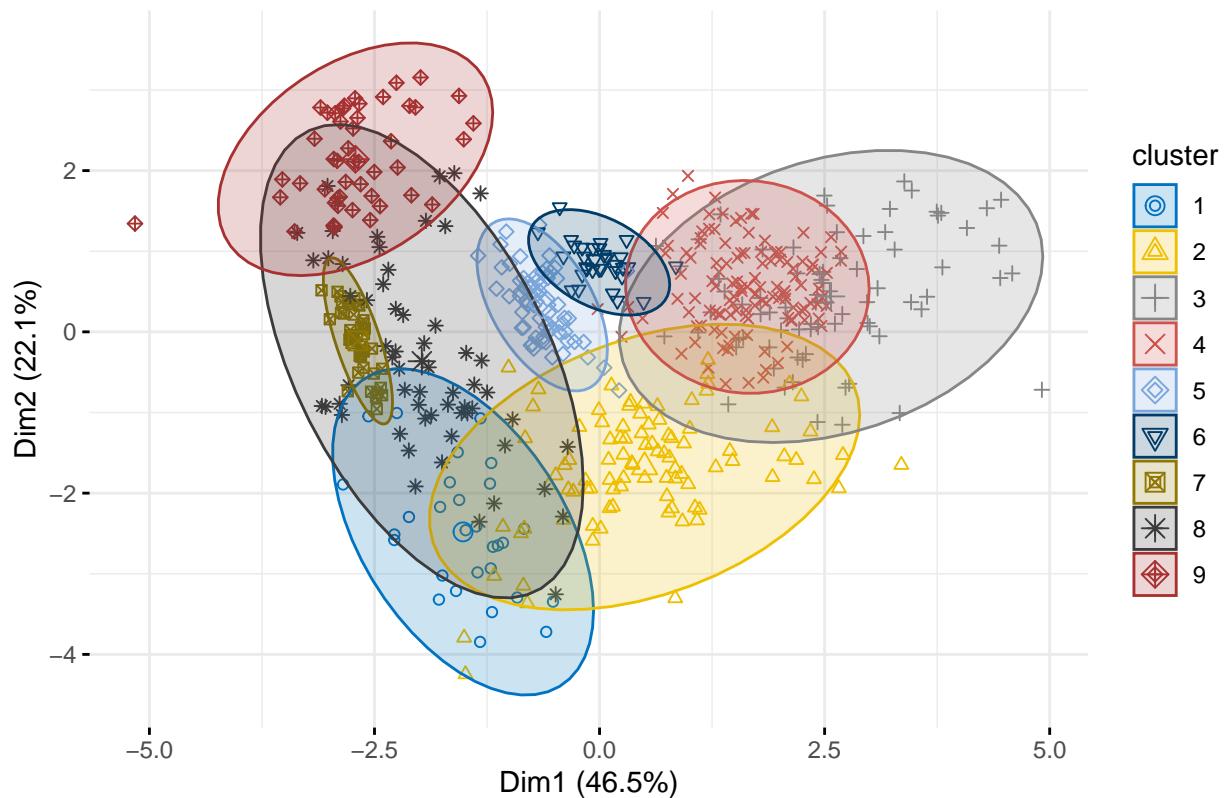
Distribuzione dei diversi cluster in funzione delle variabili



Il grafico seguente mostra la suddivisione in cluster ottenuta utilizzando il modello di mistura gaussiana.

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

Cluster plot



Confusion Matrix

Infine, calcoliamo la confusion matrix per il sistema scelto confrontando i true labels e i predicted labels.

```
# definizione true e predicted labels
predicted_labels <- gmm$classification
true_labels <- olive$Area
# Calcolo matrice di confusione
confusion_matrix <- table(true_labels, predicted_labels)

# Stampa la matrice di confusione
print(confusion_matrix)
```

```
##          predicted_labels
## true_labels   1   2   3   4   5   6   7   8   9
##   1     23   2   0   0   0   0   0   0   0
##   2      0  56   0   0   0   0   0   0   0
##   3      0   6  79 121   0   0   0   0   0
##   4      6  29   1   0   0   0   0   0   0
##   5      0   0   0   0  65   0   0   0   0
##   6      0   0   0   0   0  31   0   0   0
##   7      0   0   0   0   0   0   0   49   1
##   8      0   0   0   0   0   0   0   3  47
##   9      0   0   0   0   0   0  43   8   0
```

Le metriche di Precision e Recall vengono calcolate per valutare il modello applicato considerando la clusterizzazione. I valori presi in considerazione sono le medie di Precision e Recall calcolate per ciascun

cluster. Queste misure rappresentano la performance media del modello rispetto a tutte le classi, fornendo un'indicazione generale di quanto bene il modello sia in grado di predire le diverse classi.

Per ottenere una visione completa delle prestazioni del modello, si confronta l'andamento di queste metriche congiuntamente. Se ci si focalizza sui valori di queste metriche calcolati per ogni classe si può dire che nel caso delle classi 1-2, un valore alto di Precision e basso di Recall potrebbero indicare che il modello effettua poche predizioni positive, ma corrette. Viceversa, un alto valore di Recall e una bassa Precision (ad esempio, per la classe 3) potrebbero indicare che il modello è più incline a fare molte predizioni positive, anche se alcune di esse potrebbero essere errate. Se invece i valori di Precision e Recall sono simili e si avvicinano a 1, allora la predizione del modello rispetto al caso reale è molto vicina, e il numero di falsi positivi (per la Precision) e falsi negativi (per la Recall) è molto basso (ad esempio, per le classi 5-6). Infine, nei casi in cui entrambi i valori si avvicinano a 0, il modello predice in modo sbagliato rispetto al caso reale (ad esempio, per le classi 4-8-9).

Nel caso generale che tiene conto della media di questi valori, entrambi inferiori al 50%, si può dedurre che il valore di falsi positivi (nella Precision) e di Falsi Negativi (nella Recall), sono molto alti e quindi in media riducono di più del 50% le predizioni corrette rispetto al caso reale.

```
# METRICHE
# True positives (TP), false positives (FP) e false negatives (FN) per ogni classe
TP <- diag(confusion_matrix) # Elementi diagonali della matrice di confusione
FP <- rowSums(confusion_matrix) - TP
FN <- colSums(confusion_matrix) - TP

# Precision e recall per ogni classe
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)

# Medie di precision e recall su tutte le classi:
mean_precision <- mean(precision)
mean_recall <- mean(recall)

for (i in 1:length(precision)) {
  cat("Classe", i, "- Precision:", precision[i], "Recall:", recall[i], "\n")
}

## Classe 1 - Precision: 0.92 Recall: 0.7931034
## Classe 2 - Precision: 1 Recall: 0.6021505
## Classe 3 - Precision: 0.3834951 Recall: 0.9875
## Classe 4 - Precision: 0 Recall: 0
## Classe 5 - Precision: 1 Recall: 0.9701493
## Classe 6 - Precision: 0.9393939 Recall: 1
## Classe 7 - Precision: 0 Recall: 0
## Classe 8 - Precision: 0.06 Recall: 0.05
## Classe 9 - Precision: 0 Recall: 0

cat("Media Precision:", mean_precision, "Media Recall:", mean_recall, "\n")

## Media Precision: 0.4780988 Media Recall: 0.4892115
```

Linear Discriminant analisys (LDA)

Questa metodologia di clustering parte dall'assunzione che i cluster siano noti. Si parte da oggetti già classificati (in questo caso si utilizzerà la variabile target Region), e si cerca una regola per poterli discriminare, attraverso una separazione supervisionata. In generale, l'analisi discriminante lineare sfrutta la potenza della riduzione della dimensionalità per migliorare l'accuratezza della classificazione. Partendo dalle distribuzioni

di probabilità a priori delle varabili esplicative si estrapolano i valori in cui queste distribuzioni si intersecano e si assegnano i valori in base al massimo valore della funzione:

$$\delta(x)_k = \ln(\pi_k) + x' \Sigma \mu_k - \frac{1}{2} \mu'_k \Sigma \mu_k \quad (1)$$

dove x definisce l'insieme delle varabili esplicative, e $\delta(x)_k$ è una funzione lineare (matrice) di x . La superficie di separazione dei cluster sono definite da punti, rette o piano in base al numero di variabili esplicative, e nel caso specifico sarà un iperpiano.

Di seguito vengono riportati i valori estratti dall'analisi dell'LDA sui dati, in cui la variabile riposta è data dalla colonna Region nel dataset, che indica le tre aree in cui sono stati estratti i dati. Questo metodo non necessita la standardizzazione dei dati, pertanto si utilizza il dataset originale per il fitting.

```
#LDA FIT
olive2 <- subset(olive, select = -Area)
olive2$Region<-as.factor(olive2$Region)

N<-nrow(olive2)
lda.fit=lda(Region~., data=olive2)
lda.fit

## Call:
## lda(Region ~ ., data = olive2)
##
## Prior probabilities of groups:
##          1          2          3
## 0.5646853 0.1713287 0.2639860
##
## Group means:
##   Palmitic Palmitoleic Stearic    Oleic Linoleic Linolenic Arachidic
## 1 1332.288     154.8019 228.7740 7100.009 1033.4985 38.06502 63.11765
## 2 1111.347      96.7449 226.1837 7268.020 1196.5306 27.09184 73.17347
## 3 1094.801     83.7351 230.8013 7793.053 727.0331 21.78808 37.57616
##   Eicosenoic
## 1 27.321981
## 2 1.938776
## 3 1.973510
##
## Coefficients of linear discriminants:
##              LD1        LD2
## Palmitic   -0.0027354202  0.008973371
## Palmitoleic -0.0128674404  0.018139213
## Stearic     0.0028236870  0.004256022
## Oleic       -0.0005496889  0.006270797
## Linoleic    -0.0010641425 -0.001223012
## Linolenic   -0.0411021440  0.005551704
## Arachidic    0.0175912328 -0.034913021
## Eicosenoic  -0.1632123194  0.010353941
##
## Proportion of trace:
##      LD1      LD2
## 0.7853 0.2147
```

L'output "Proportion of Trace" identifica le due migliori funzioni discriminanti LD1 e LD2 i cui valori che identificano la percentuale di separazione ottenuta da ciascuna funzione discriminante sono rispettivamente 0.7853 e 0.2147.

Ad esempio, la prima funzione discriminante LD1, ad esempio, è una combinazione lineare delle variabili esplicative ed è descritta dai coefficienti lineari dei discriminanti: $-0.0027 * Palmatic - 0.0128 * Pamitoleic - \dots - 0.163 * Elicosenoic$

Viene valutata la matrice di confusione per calcolare l'accuratezza, che risulta essere del 99.12%, nel clustering dei dati.

```
#Confusion Matrix
lda.pred=predict(lda.fit, olive2)
lda.class=lda.pred$class
table(olive2$Region, lda.class)

##      lda.class
##      1   2   3
## 1 322  0  1
## 2   0 98  0
## 3   0  4 147

#Calcolo Accuratezza
accuracy.LDA<-sum(diag(table(lda.class,olive2$Region)))/nrow(olive)*100
accuracy.LDA

## [1] 99.12587
```

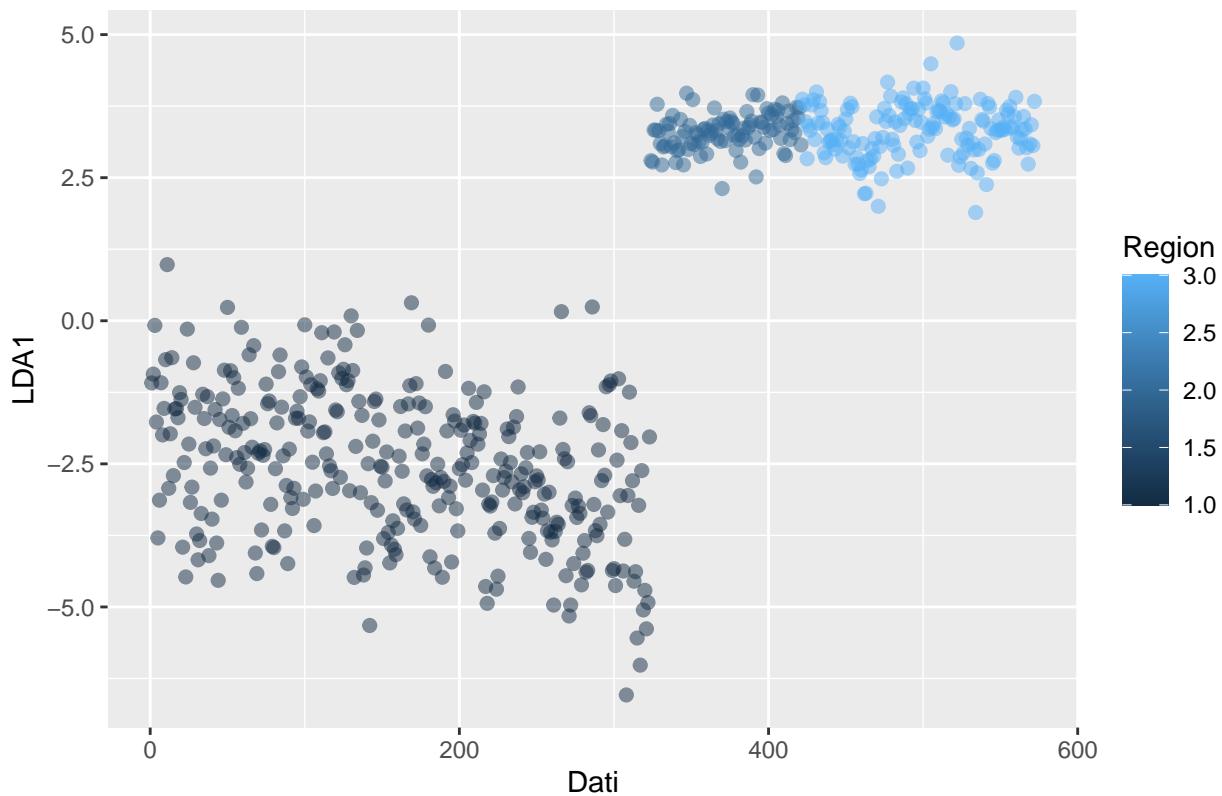
Il grafico successivo mostra lo scatterplot dei valori delle due funzioni LDA1 e LD2 separatamente.

```
library(ggplot2)
x<- 1:length(lda.pred$x[,1])

par(mfrow=c(1,2))
plotdata=data.frame(x,lda.pred$x[,1])
plotdata2=data.frame(x,lda.pred$x[,2])
attach(plotdata)
attach(plotdata2)

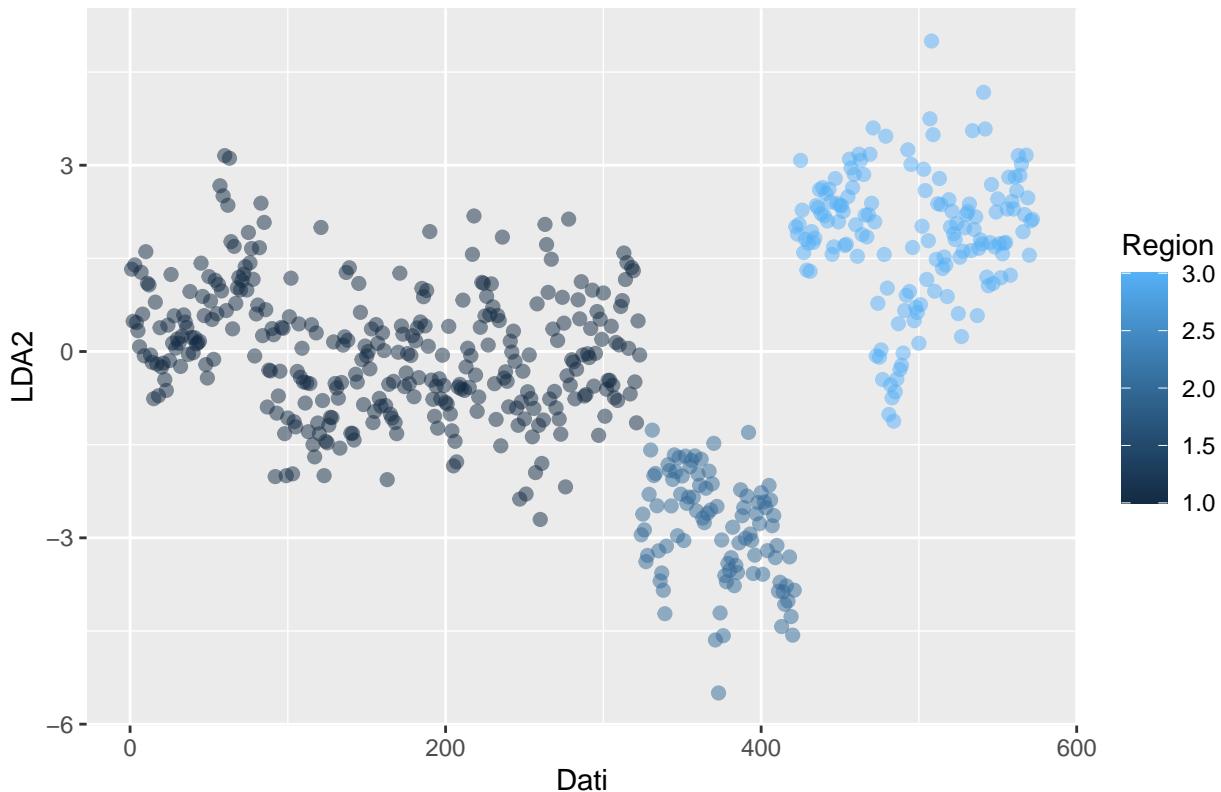
ggplot(data = plotdata,
       mapping = aes(x =x, y =lda.pred.x...1.,
                     color = Region)) +
  geom_point(alpha = .5, size = 2) +
  labs(title = "Clustering attraverso LDA1",
       x = "Dati", y = "LDA1")
```

Clustering attraverso LDA1



```
ggplot(data = plotdata2,
       mapping = aes(x = x,
                     y = lda.pred.x...2.,
                     color = Region)) +
  geom_point(alpha = .5, size = 2) +
  labs(title = "Clustering attraverso LDA2",
       x = "Dati", y = "LDA2")
```

Clustering attraverso LDA2



I dati sono ben suddivisi in base alla regione considerata. Infatti si può vedere come la prima funzione discriminante LDA1 separi perfettamente la regione 1 dalle regioni 2 e 3 ma non perfettamente la regione 2 dalla 3. La seconda funzione discriminante LD2 performa una buona separazione della regione 1 e 2 dalla regione 3 e una discreta separazione tra regione 1 e 2. Per questo motivo per ottenere un'ottima separazione in base alle regioni è opportuno utilizzare tutte e due le funzioni discriminanti contemporaneamente. Tuttavia, alla base dell'applicazione di questa analisi, c'è un'importante ipotesi che deve essere rispettata. Le variabili esplicative devono rispettare l'ipotesi di omoschedasticità ($\sigma_1^2 = \sigma_2^2 = \dots = \sigma_k^2 = \sigma^2$). Dall'analisi esplorativa prima effettuata si è riscontrato che questa ipotesi non è rispettata. Conducendo ugualmente l'analisi si è visto che comunque i tre cluster sono ben separati dalla due LDA, e questa buona suddivisione compensa la presenza di eteroschedasticità nei dati.

Analisi su Train e Test

Una scelta più accurata ai fini della validazione del modello richiede l'analisi su un set di Train e di Test.

Per prima cosa si effettua lo splitting del dataset in test e train.

```
set.seed(1500)
trainIndex <- createDataPartition(olive2$Region, p = 0.7, list = FALSE, times = 1)
# Dataset di train
olive_train<-olive2[trainIndex,]
# Dataset di test
olive_Test<-olive2[-trainIndex,]

table(olive_train$Region)/nrow(olive_train)*100

##
##          1           2           3
## 56.46766 17.16418 26.36816
```

```



```

L'accuratezza ottenuta sul Test set (99.41), molto simile a quella del Train set (98.75), ha convalidato la capacità di generalizzazione del modello, fornendo previsioni accurate sui dati.

Conclusione

L'obiettivo principale è stato utilizzare un dataset open source fornito dal software R, nella fattispecie il dataset olive della libreria pgmm. Inizialmente è stato effettuata un'analisi descrittiva del dataset, seguita dall'applicazione di diverse tecniche di clustering e metodi di valutazione per l'implementazione dei modelli, gradualmente più complessi. L'applicazione della PCA ha confermato la possibilità di poter ridurre la dimensionalità dei dati da 8 variabili a 5 con il 94,4% della varianza cumulativa totale.

Per la cluster tendency assessment sono stati applicati sia la metodologia grafica (VAT) che quella statistica

attraverso il calcolo del coefficiente Hopkins, che hanno dato evidenzia la clusterizzazione del dataset. A seguito dell'applicazione degli algoritmi di clustering gerarchico agglomerativo e “partizionali”, è stato possibile riscontrare un numero di cluster ottimali pari a 5 e 9. Ragion per cui, si è proceduto all'applicazione di modelli di misture gaussiane i cui risultati hanno dato come risultato un numero di cluster ottimali pari a 9 con una precisione del 50%. In ultima analisi è stata condotta la Linear Discriminant Analysis, con la quale si è evinto come i cluster presenti nel dataset siano 3 con un a accuratezza del 99%, come confermata rispettivamente dall'analisi sul train e test set.