Mixture of Balls With Different Volumes

NIANG Mohamed
KAINA Mohamed Abdellah

20 December 2019

Contents

1 Context

Let us consider a vector of p random variables $x^1, ..., x^p$ independent, normal, all with mean 0 and variance σ^2. The random vector $x = (x^1, ..., x^p)'$ is normal with mean vector $(0, ..., 0)^\intercal$, and covariance matrix $\sigma^2 I_p$. This distribution defines a gaussian ball with mean vector $(0, ..., 0)^\intercal$, and covariance matrix $\sigma^2 I_p$. Let us consider a mixture of K gaussian balls:

$$p(x|\pi, \mu_1, ..., \mu_K, \sigma_1, ..., \sigma_K) = \sum_{k=1}^{K} \pi_k \mathcal{N}_p(x|\mu_k,\ \Sigma_k = \sigma_k^2 I_p)$$

Where $\pi = \pi_k$ are the proportions of the mixture. In the following, we will consider:

- a sample $X = (x_1, ..., x_n)$ from the above defined mixture ;
- latent variables $Z = (z_1, ..., z_n)$ indicating from which component of the mixture each x_i originates ;
- the vector of parameters is denoted $\theta = (\pi, \mu_1, ..., \mu_K, \sigma_1, ..., \sigma_K)$. For simplicity, $\theta_j = \{\pi_j, \mu_j, \Sigma_j\}$.

2 Problem

2.1 Load Library

```
library(knitr) # Markdown
library(kableExtra)
# For Exercise 1
library(mvtnorm)
library(ggplot2)
# For Exercise 2
library(mclust)
```

```
## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

```
## The following object is masked from 'package:mvtnorm':
##
##     dmvnorm
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidyverse 1
```

```
## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v tibble  2.1.3     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------ tidyverse_conflic
## x dplyr::filter()     masks stats::filter()
## x dplyr::group_rows() masks kableExtra::group_rows()
## x dplyr::lag()        masks stats::lag()
## x purrr::map()        masks mclust::map()
```

```
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(fpc)
# For Exercise 5
library(NbClust)
library(cluster)
library(e1071)
```

2.2 Exercise 1: Simulation

2.2.1 Data Simulation

```
set.seed(1234)
X1 <- rmvnorm(1000,mean = c(1,2),sigma = diag(c(1,1)))
X2 <- rmvnorm(1000,mean = c(1,2),sigma = 4*diag(c(1,1)))
Y <- rep(c(1,2),c(1000,1000))
p1 <- 0.5
p2 <- 1 - p1
mixed_proportion <- c(p1,p2)
```

2.2.2 Display the Sample

```
mydata <- data.frame(cbind(rbind(X1,X2),Y))
names(mydata) <- c("X1","X2","Y")
mydata$Y <- as.factor(mydata$Y)
attach(mydata)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     X1, X2, Y
```

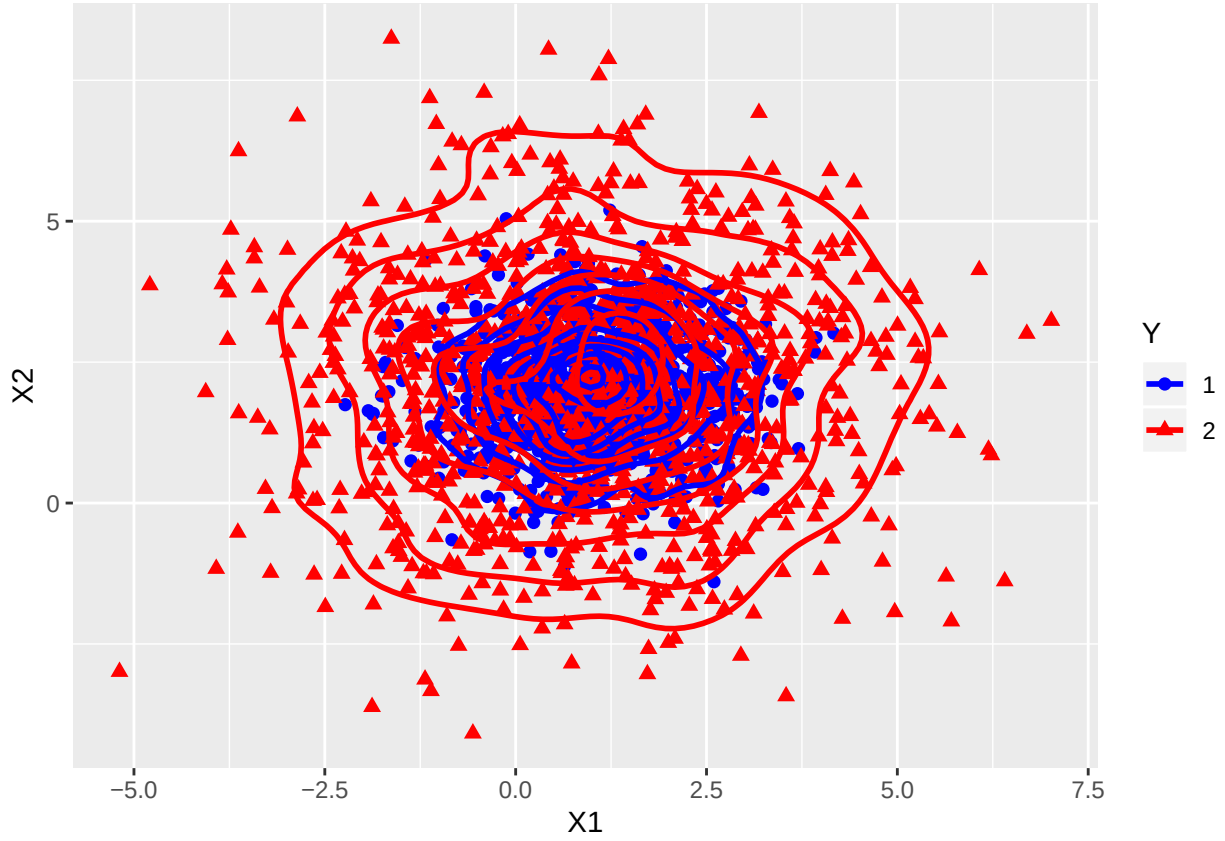
```
# Display the Sample
head(mydata)
```

```
##           X1         X2 Y
## 1 -0.2070657  2.2774292 1
## 2  2.0844412 -0.3456977 1
## 3  1.4291247  2.5060559 1
## 4  0.4252600  1.4533681 1
```

```
## 5  0.4355480  1.1099622 1
## 6  0.5228073  1.0016136 1
```

2.2.3 Display the contour plot of the two dimensional density

```
ggplot(mydata, aes(X1, X2, color = Y)) +
  geom_point(aes(shape=Y), size = 2) +
  geom_density_2d(aes(colour = Y), size = 1) +
  scale_color_manual(values = c("blue", "red"))
```



2.3 Exercise 2: Mclust Versus Kmeans

2.3.1 Running Mclust on the Simulated Data From The Exercise 1

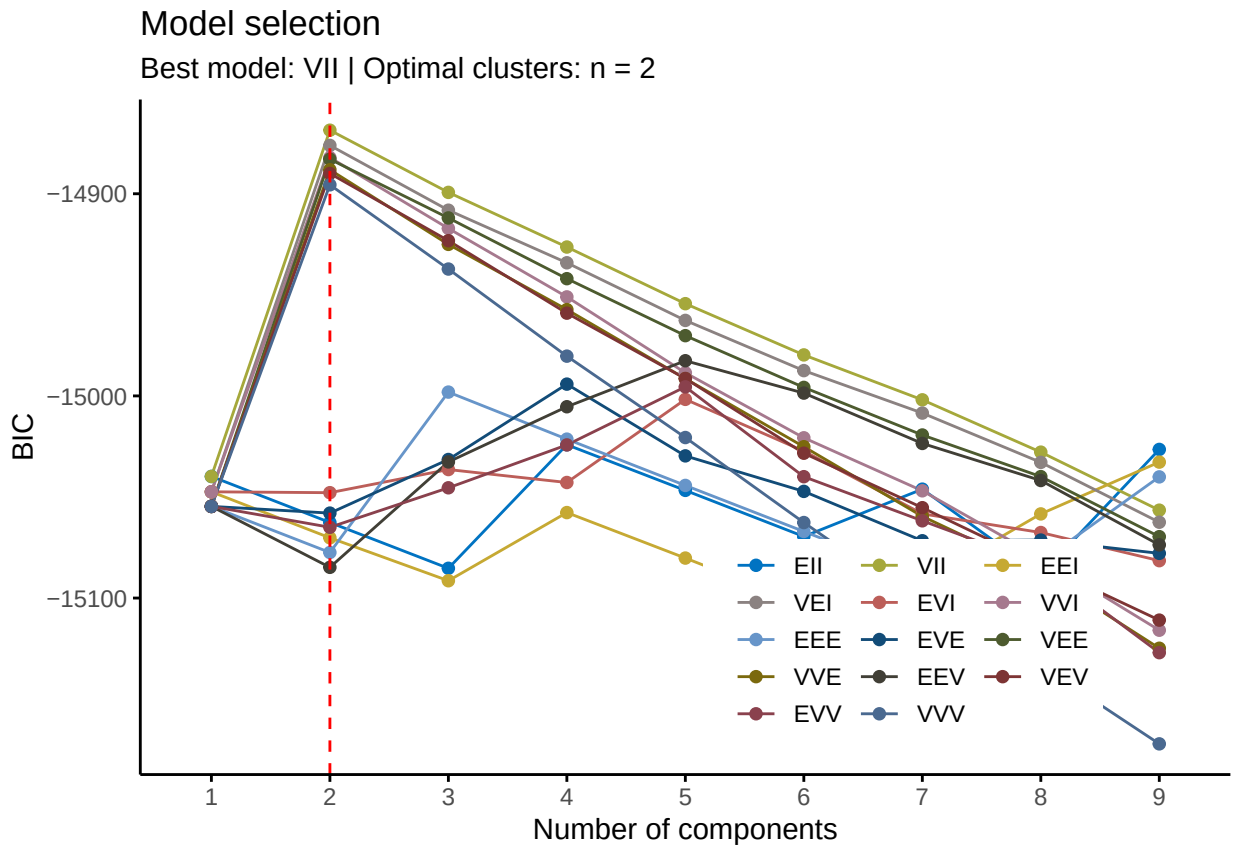
```
mclust <- Mclust(mydata[,c("X1","X2")])
```

```
summary(mclust) # Model = VII and Number of Components = 2
```

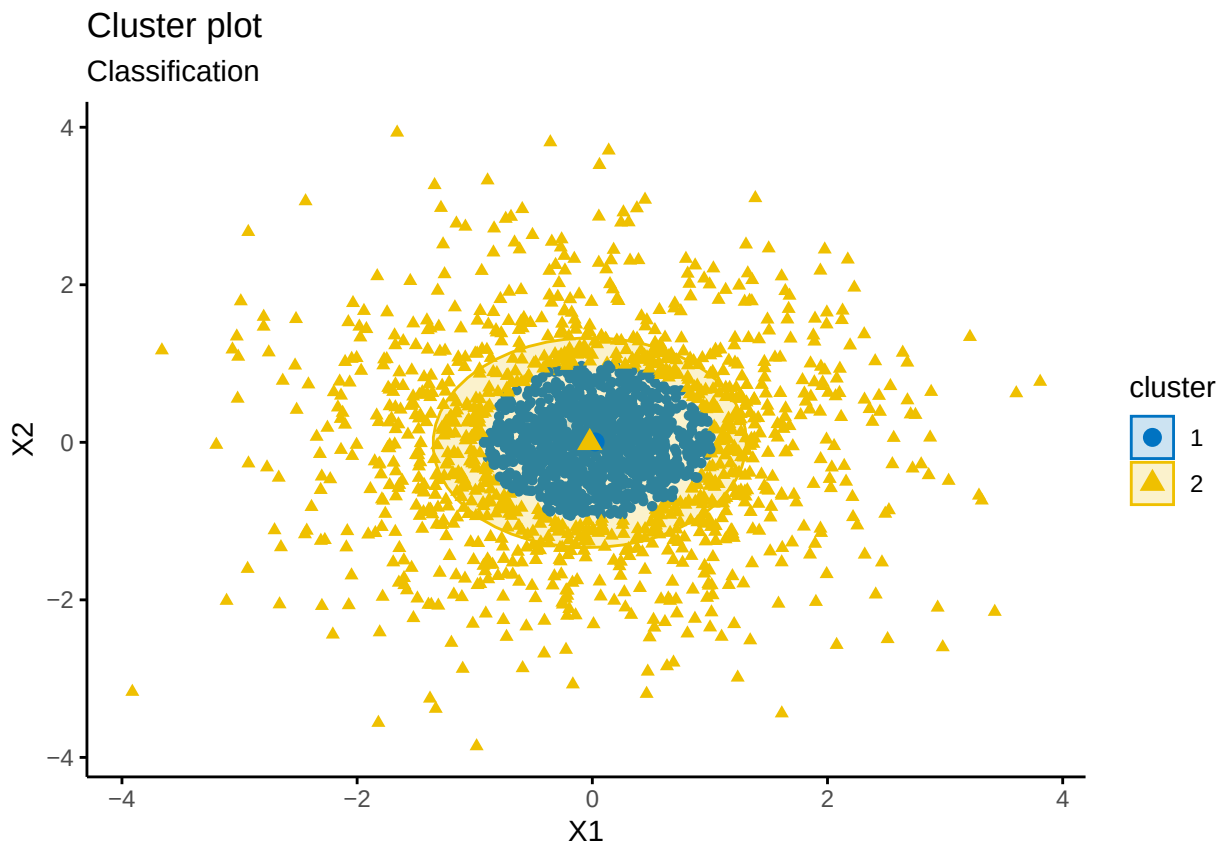
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VII (spherical, varying volume) model with 2 components:
##
##  log-likelihood    n df       BIC       ICL
##       -7407.678 2000  7 -14868.56 -16173.73
##
## Clustering table:
```

```
##    1    2
##  958 1042
```

```
# BIC values used for choosing the number of clusters
fviz_mclust(mclust, "BIC", palette = "jco")
```




```
# Classification: plot showing the clustering
fviz_mclust(mclust, "classification", geom = "point",
pointsize = 1.5, palette = "jco")
```



Comments: The model given by 'Mclust' object is $(VII, 2)$. Indeed, in the first exercise, the variance in the two distributions is proportional to the identity matrix. Thus the analysis of the results shows that the best model is the VII because it has the largest BIC. And we observe a sharp drop in the BIC from the second component. As a result, we take $K = 2$ as the number of classes.

2.3.2 Parameter estimation

```
# The estimated parameters
mclust_VII <- Mclust(mydata[,c("X1","X2")],modelNames = "VII",G=2)

param <- mclust_VII$parameters
param
```

```
## $pro
## [1] 0.4117122 0.5882878
##
## $mean
##        [,1]      [,2]
## X1 1.042765 0.9623319
## X2 2.027610 2.0110169
##
## $variance
## $variance$modelName
## [1] "VII"
##
## $variance$d
## [1] 2
```

```
##
## $variance$G
## [1] 2
##
## $variance$sigma
## , , 1
##
##           X1        X2
## X1 0.8232515 0.0000000
## X2 0.0000000 0.8232515
##
## , , 2
##
##          X1       X2
## X1 3.672472 0.000000
## X2 0.000000 3.672472
##
##
## $variance$sigmasq
## [1] 0.8232515 3.6724725
##
## $variance$scale
## [1] 0.8232515 3.6724725
```

```
param_mean <- mclust_VII$parameters$mean
param_mean
```

```
##        [,1]      [,2]
## X1 1.042765 0.9623319
## X2 2.027610 2.0110169
```

```
param_sigma <- mclust_VII$parameters$variance$sigma
param_sigma
```

```
## , , 1
##
##           X1        X2
## X1 0.8232515 0.0000000
## X2 0.0000000 0.8232515
##
## , , 2
##
##          X1       X2
## X1 3.672472 0.000000
## X2 0.000000 3.672472
```

2.3.3 A Partition of The Simulated Data Using Mclust

```
mclust_part <- MclustDR(mclust_VII)
```

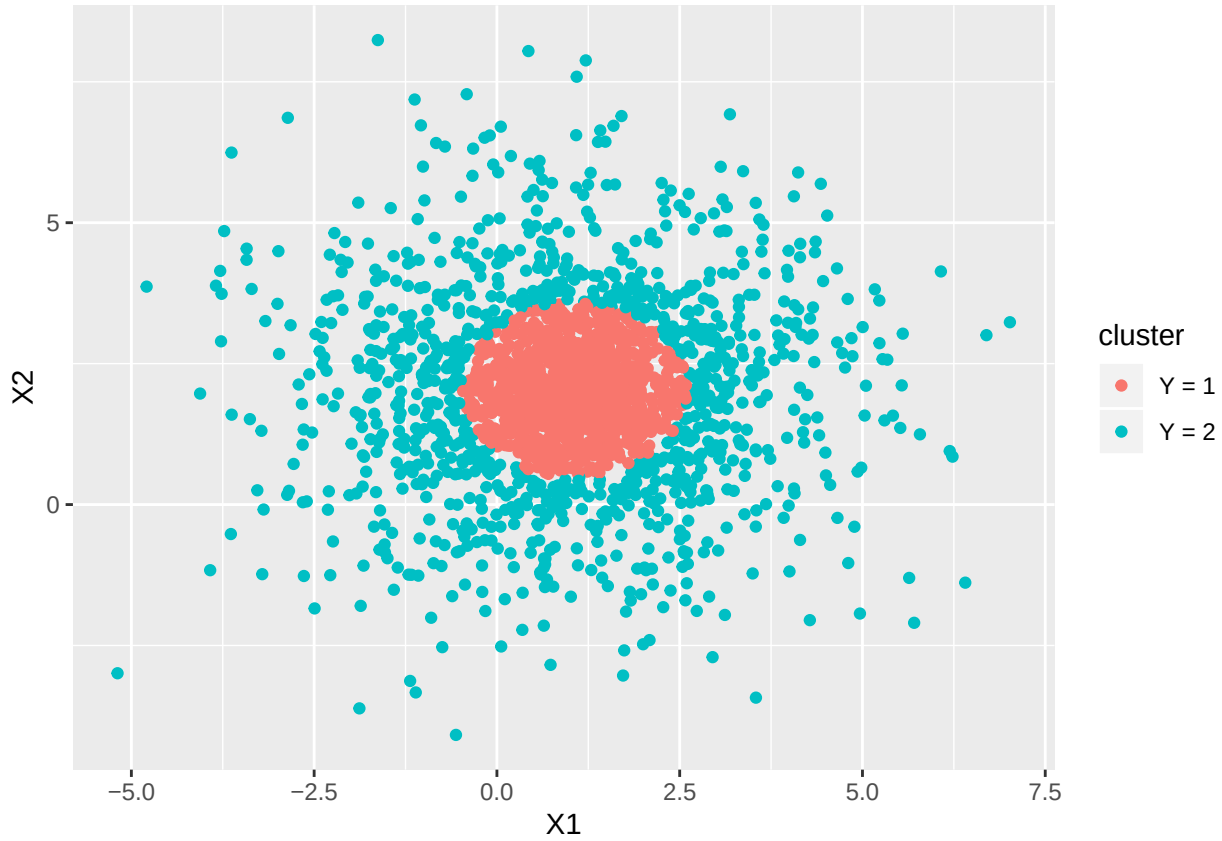
```
summary(mclust_part)
```

```
## ----------------------------------------------------------------
## Dimension reduction for model-based clustering and classification
## ----------------------------------------------------------------
##
## Mixture model type: Mclust (VII, 2)
```

```
##
## Clusters    n
##        1  958
##        2 1042
##
## Estimated basis vectors:
##        Dir1    Dir2
## X1 -0.72352 0.69030
## X2  0.69030 0.72352
##
##                 Dir1      Dir2
## Eigenvalues  0.32341   0.30603
## Cum. %      51.37998 100.00000
```

```
clusterPlots <- data.frame(X1 = mydata[,c("X1")],
                           X2 = mydata[,c("X2")],
                           cluster = factor(mclust_part$classification,
    levels = c(1, 2), labels = c("Y = 1", "Y = 2")))

clusterPlots.gg <- ggplot(clusterPlots)
clusterPlots.gg + geom_point(aes(x = X1, y = X2, color = cluster))
```



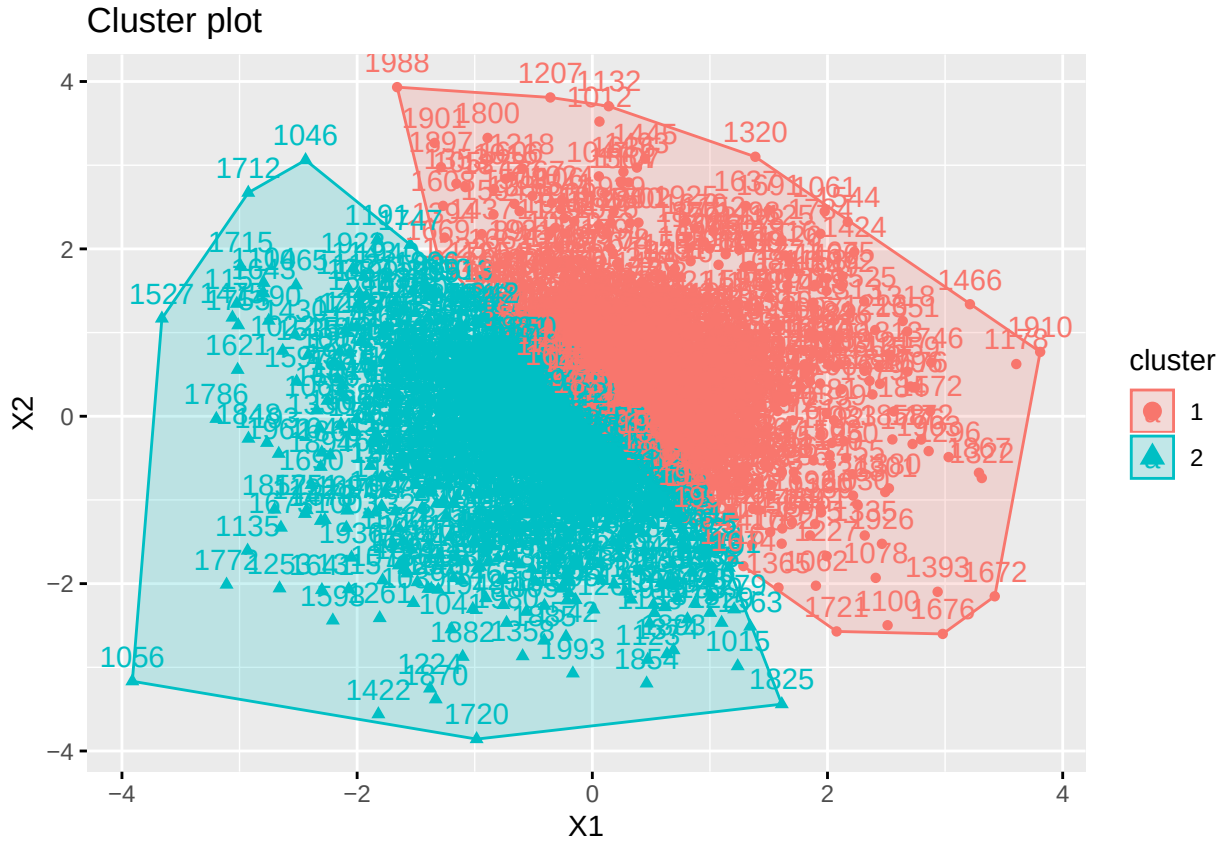
2.3.4 A Partition of The Simulated Data Using Kmeans

```
kmeans_part <- kmeans(mydata[,c("X1","X2")], centers = 2)

summary(kmeans_part)
```

```
##              Length Class  Mode
## cluster      2000   -none- numeric
## centers         4   -none- numeric
## totss           1   -none- numeric
## withinss        2   -none- numeric
## tot.withinss    1   -none- numeric
## betweenss       1   -none- numeric
## size            2   -none- numeric
## iter            1   -none- numeric
## ifault          1   -none- numeric
```

```
fviz_cluster(kmeans_part, data = mydata[,c("X1","X2")])
```



2.3.5 Comparison of The Results of The Two Partitions (Kmeans and Mclust)

To compare the two models, we use the *cluster.stats*() function of the fpc library. Among the values returned by the function *cluster.stats*(), there are two indices to compare the performance of two clusters, namely the within.cluster.ss and the means.silwidth.

Most often, we focus on using within.cluster.ss and avg.silwidth to validate the clustering method. The within.cluster.ss measurement stands for the within clusters sum of squares, and avg.silwidth represents the average silhouette width.

- within.cluster.ss measurement shows how closely related objects are in clusters; the smaller the value, the more closely related objects are within the cluster.
- avg.silwidth is a measurement that considers how closely related objects are within the cluster and how clusters are separated from each other. The silhouette value usually ranges from 0 to 1; a value closer to 1 suggests the data is better clustered.

```
d <- dist(mydata[,c("X1","X2")], method ="euclidean")
stat_mclust <- cluster.stats(d, mclust_VII$classification)
stat_kmeans <- cluster.stats(d, kmeans_part$cluster)

within_mclust <- stat_mclust$within.cluster.ss
avg_mclust <- stat_mclust$avg.silwidth

within_kmeans <- stat_kmeans$within.cluster.ss
avg_kmeans <- stat_kmeans$avg.silwidth

statsmodelsMclust <- c(within_mclust, avg_mclust)
statsmodelsKmeans <- c(within_kmeans, avg_kmeans)
resultsMclust <- data.frame("Mclust" = c("within.cluster.ss","avg.silwidth"), "Stats Mclust" = statsmod
resultsKmeans <- data.frame("Kmeans" = c("within.cluster.ss","avg.silwidth"), "Stats Kmeans" = statsmod
resultfinal <- cbind(resultsMclust,resultsKmeans)
```

```
# Comparison Table
kable(arrange(resultfinal,desc(statsmodelsMclust),desc(statsmodelsKmeans)), digits = 2) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = F,
                font_size = 18,
                position = "center")
```

Mclust	Stats.Mclust	Kmeans	Stats.Kmeans
within.cluster.ss	9998.22	within.cluster.ss	7023.85
avg.silwidth	0.13	avg.silwidth	0.29

Comments: Based on the results of the above table, we conclude that the Kmeans is **the best model because it has within.cluster.ss smaller and one avg.silwidth larger than the Mclust**.

2.4 Exercise 3: EM Algorithm for a Mixture of Balls

2.4.1 Detail the Computation

Given our current estimate of the parameters $\theta^{(q)}$, the conditional distribution of the z_i is determined by Bayes theorem to be normalized Gaussian density weighted by π_j:

$$
\begin{aligned}
& t_{ik}^{(q)} \\
= \; & p(z_i = k \mid x_i; \boldsymbol{\theta}^{(q)}) \\
= \; & \frac{p(x_i, z_i = k; \boldsymbol{\theta}^{(q)})}{p(x_i; \boldsymbol{\theta}^{(q)})} \\
= \; & \frac{p(x_i, z_i = k; \boldsymbol{\theta}^{(q)})}{\sum_{j=1}^{K} p(x_i, z_i = j; \boldsymbol{\theta}^{(q)})} \\
= \; & \frac{f\left(x_i; \mu_k^{(q)}, \Sigma_k^{(q)}\right) \cdot \pi_k^{(q)}}{\sum_{j=1}^{K} f\left(x_i; \mu_j^{(q)}, \Sigma_j^{(q)}\right) \cdot \pi_j^{(q)}}
\end{aligned}
$$

2.4.2 Express The Expectation Step

Expectation-Step

Define $Q(\boldsymbol{\theta}^{(q)} \mid \boldsymbol{\theta})$

$$L(\mathbf{X}, \mathbf{Z}; \theta) = \sum_{i=1}^{n} \log f\left(x_i; \mu_k, \Sigma_k\right) \cdot \pi_k$$

Where

$$f\left(x_i; \mu_k, \Sigma_k\right) = \frac{1}{(2\pi)^{d/2} \left|\Sigma_k\right|^{1/2}} \exp\left(-\frac{1}{2}\left(x_i - \mu_k\right)^T \Sigma_k^{-1}\left(x_i - \mu_k\right)\right)$$

Then

$$Q(\boldsymbol{\theta^{(q)}} \mid \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{k=1}^{K} p(z_i = k \mid x_i; \boldsymbol{\theta}^{(q)}) \cdot \log f\left(x_i; \mu_k, \Sigma_k\right) \cdot \pi_k$$

2.4.3 Detail The Computation of Maximization Step

Maximization-Step

We need to maximize, with respect to our parameters $\boldsymbol{\theta}^{(q)}$, the quantity

$$\begin{aligned}
& Q(\boldsymbol{\theta}^{(q)} \mid \boldsymbol{\theta}) \\
= & \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(q)} \log \frac{1}{(2\pi)^{d/2} \left|\Sigma_k\right|^{1/2}} \exp\left(-\frac{1}{2}\left(x_i - \mu_k\right)^T \Sigma_k^{-1}\left(x_i - \mu_k\right)\right) \cdot \pi_k \\
= & \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(q)} \left[\log \pi_k - \frac{1}{2} \log \left|\Sigma_k\right| - \frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k) - \frac{d}{2} \log(2\pi)\right]
\end{aligned}$$

- Update π_k

Grouping together only the terms that depend on π_k, we find that we need to maximize

$$\sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(q)} \log \pi_k$$

With subject to

$$\sum_{k=1}^{K} \pi_k = 1$$

So we construct the Lagrangian

$$\mathcal{L}(\pi) = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(q)} \log \pi_k + \lambda \left(\sum_{k=1}^{K} \pi_k - 1\right)$$

Where λ is the Lagrange multiplier. Taking derivative, we find

$$\frac{\partial}{\partial \pi_k} \mathcal{L}(\pi) = \sum_{i=1}^{n} \frac{t_{ik}^{(q)}}{\pi_k} + \lambda$$

Setting this to zero and solving, we get

$$\pi_k = \frac{\sum_{i=1}^{n} t_{ik}^{(q)}}{-\lambda}$$

Using the constraint that $\sum_{k=1}^{K} \pi_k = 1$ and knowing the fact that $\sum_{k=1}^{K} t_{ik}^{(q)} = 1$ (probabilities sum to 1), we easily find:

$$-\lambda = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(q)} = \sum_{i=1}^{n} 1 = n$$

We therefore have updates for the parameters π_k:

$$\pi_k = \frac{1}{n} \sum_{i=1}^{n} t_{ik}^{(q)}$$

- Update μ_k

$$\begin{aligned}
& \frac{\partial}{\partial \mu_k} Q(\boldsymbol{\theta}^{(q)} \mid \boldsymbol{\theta}) \\
= \; & \frac{\partial}{\partial \mu_k} \sum_{i=1}^{n} -\frac{1}{2} t_{ik}^{(q)} (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k) \\
= \; & \frac{1}{2} \sum_{i=1}^{n} t_{ik}^{(q)} \frac{\partial}{\partial \mu_k} \left(2\mu_k^T \Sigma_k^{-1} x_i - \mu_k^T \Sigma_k^{-1} \mu_k \right) \\
= \; & \sum_{i=1}^{n} t_{ik}^{(q)} \left(\Sigma_k^{-1} x_i - \Sigma_k^{-1} \mu_k \right)
\end{aligned}$$

Setting this to zero and solving for μ_k therefore yields the update rule

$$\mu_k = \frac{\sum_{i=1}^{n} t_{ik}^{(q)} x_i}{\sum_{i=1}^{n} t_{ik}^{(q)}}$$

- Update Σ_k

$$\begin{aligned}
& \frac{\partial}{\partial \Sigma_k} Q(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}) \\
= \; & \frac{\partial}{\partial \Sigma_k} \sum_{i=1}^{n} -\frac{1}{2} t_{ik}^{(q)} \left[\log |\Sigma_k| + (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k) \right] \\
= \; & -\frac{1}{2} \sum_{i=1}^{n} t_{ik}^{(q)} \frac{\partial}{\partial \Sigma_k} \left[\log |\Sigma_k| + (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k) \right] \\
= \; & -\frac{1}{2} \sum_{i=1}^{n} t_{ik}^{(q)} \left(\Sigma_k^{-1} - (x_i - \mu_k)(x_i - \mu_k)^T \Sigma_k^{-2} \right)
\end{aligned}$$

Setting the partial derivative to zero and solving for Σ_k therefore yields the update rule

$$\Sigma_k = \frac{\sum_{i=1}^{n} t_{ik}^{(q)} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} t_{ik}^{(q)}}$$

2.4.4 Write The Pseudo-Code of An EM Algorithm

```
# Initialization Function
init.EM <- function(X,K=n){
  mus <- X[sample(1:nrow(X),K),1]
  sds <- rep(sd(X[,1]),K)
  pis <- rep(1/K,K)
  return(parameters = list(mus=mus,sds=sds,pis=pis))
}
```

2.4.5 Write a Expectation Step Function

```
# Expectation Function
E.step <- function(X,parameters){
  K <- length(parameters$mus)
  Tik <- matrix(0,nrow(X),K)
  for(k in 1:K){
    Tik[,k] <- parameters$pis[k]*dnorm(X[,1],
                                       mean=parameters$mus[k],
                                       sd=parameters$sds[k])
  }
  return(Tik <- Tik/rowSums(Tik))
}
```

2.4.6 Write a Maximization Step Function

```
# Maximization Function
M.step <- function(X, Tik, parameters){
  K <- length(parameters$mus)
  parameters$pis <- colSums(Tik)/nrow(X)
  for (k in 1:K){
    parameters$mus[k] <- sum(Tik[,k]*X) / sum(Tik[,k])
    parameters$sds[k] <- sqrt(sum(Tik[,k]*(X-parameters$mus[k])^2)/ sum(Tik[,k]))
  }
  return(parameters)
}
```

2.4.7 Program The EM Algorithm

```
# EM Algorithm
EM <- function(X,K=n){
  parameters <- init.EM(X,K)
  iter <- 0
  parameters.new <- parameters
  repeat{
    Tik <- E.step(X,parameters)
    parameters.new <- M.step(X,Tik,parameters)
    if ((sum(unlist(parameters.new) - unlist(parameters))^2)/
        sum(unlist(parameters.new))^2 < 1e-20) break
    parameters<-parameters.new
  }
  return(list(parameters=parameters.new,Tik=Tik))
}
```

2.4.8 Application On The Simulated Data From The Exercise 1

```
# Define Mixture Data From Exercise 1
mixture_data <- mydata[,c("X1","X2")]
```

```
# Application
pc <- proc.time()
result1 <- EM(mixture_data,2)$parameters
proc.time() - pc
```

```
##    user  system elapsed
##    0.09    0.00    0.10
```

```
result1
```

```
## $mus
## [1] 3.013296 3.013296
##
## $sds
## [1] 3.172237 3.172237
##
## $pis
## [1] 0.4743226 0.5256774
```

2.5 Exercise 4: Mixture of Balls Using The Kernel Trick

Assume a transformation $\Phi : \mathbb{R} \leftarrow \mathcal{V}$ and denote $K(.,.)$ the scalar product in $\mathcal{V}$.

2.5.1 Compute The Distance in The Transformed Space

Let $\mu_k = \frac{1}{\sum_j t_{jk}} \sum_j t_{jk} x_j$. Let's show that the distance between x_i and μ_k in the transformed space is :

$$\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2 = K(x_i, x_i) - \frac{2}{\sum_j t_{jk}} \sum_j t_{jk} K(x_i, x_j) + \frac{1}{\sum_j t_{jk} \sum_h t_{hk}} \sum_j \sum_h t_{jk} t_{hk} K(x_j, x_h)$$

Based on the definition of the scalar product, we have

$$\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2 = \langle x_i - \mu_k, x_i - \mu_k \rangle_{\mathcal{V}}$$

$$\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2 = \langle x_i, x_i \rangle_{\mathcal{V}} + \langle x_i, -\mu_k \rangle_{\mathcal{V}} + \langle -\mu_k, x_i \rangle_{\mathcal{V}} + \langle -\mu_k, -\mu_k \rangle_{\mathcal{V}}$$

By developing the scalar product, we obtain

$$\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2 = \langle x_i, x_i \rangle_{\mathcal{V}} - 2\langle x_i, \mu_k \rangle_{\mathcal{V}} + \langle \mu_k, \mu_k \rangle_{\mathcal{V}}$$

Yet

$$\langle x_i, x_i \rangle_{\mathcal{V}} = K(x_i, x_i) \qquad (1)$$

$$\langle x_i, \mu_k \rangle_{\mathcal{V}} = \langle x_i, \frac{1}{\sum_j t_{jk}} \sum_j t_{jk} x_j \rangle_{\mathcal{V}}$$

$$\langle x_i, \mu_k \rangle_{\mathcal{V}} = \frac{1}{\sum_j t_{jk}} \sum_j t_{jk} \langle x_i, x_j \rangle_{\mathcal{V}}$$

$$\langle x_i, \mu_k \rangle_{\mathcal{V}} = \frac{1}{\sum_j t_{jk}} \sum_j t_{jk} K(x_i, x_j) \qquad (2)$$

And

$$\langle \mu_k, \mu_k \rangle_{\mathcal{V}} = \langle \frac{1}{\sum_j t_{jk}} \sum_j t_{jk} x_j, \frac{1}{\sum_h t_{hk}} \sum_h t_{hk} x_h \rangle_{\mathcal{V}}$$

$$\langle \mu_k, \mu_k \rangle_{\mathcal{V}} = \frac{1}{\sum_j t_{jk}} \sum_j t_{jk} \frac{1}{\sum_h t_{hk}} \sum_h t_{hk} \langle x_j, x_h \rangle_{\mathcal{V}}$$

$$\langle \mu_k, \mu_k \rangle_{\mathcal{V}} = \frac{1}{\sum_j t_{jk} \sum_h t_{hk}} \sum_j \sum_h t_{jk} t_{hk} K(x_j, x_h) \qquad (3)$$

According to (1), (2) and (3), we can see that

$$\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2 = K(x_i, x_i) - \frac{2}{\sum_j t_{jk}} \sum_j t_{jk} K(x_i, x_j) + \frac{1}{\sum_j t_{jk} \sum_h t_{hk}} \sum_j \sum_h t_{jk} t_{hk} K(x_j, x_h)$$

2.5.2 Express The Estimation of Variance

The estimate of σ_k in the transformed space is given by

$$\sigma_k = \frac{1}{\sum_i t_{ik}} \sum_i \sum_k t_{ik} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2$$

2.5.3 Compute The Responsabilities In The Transformed Space

Let the objective function $J = \sum_i \sum_k t_{ik} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2$. The computation of responsabilities is given by

$$\frac{\partial J}{\partial t_{ik}} = \sum_i \sum_k \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2$$

This gives you

$$t_{ik} = \begin{cases} 1 & \text{if } k = \operatorname{argmin} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_{\mathcal{V}}^2 \\ 0 & \text{otherwise} \end{cases}$$

2.6 Exercise 5: The Iris Data

We will be using the **Iris Dataset** for the comparison of clustering algorithms. The dataset consists of 150 observations with 4 variables: Sepal Length, Sepal Width, Petal Length and Petal Width. The entire dataset has three different species of Setosa, Versicolor and Virginica with 50 samples each.

2.6.1 Run Unsupervised Algorithms on The iris dataset

Clustering is an unsupervised machine learning technique to identify groups in the dataset which contain observations with similar profiles according to the specified criteria. Similarity between the observations are defined using distance measures such as Euclidian, Manhattan Distances and some correlation based distance measures.

There are different clustering methodologies, which can be subdivided as the five general strategies with examples:

- Partitioning Methods- K Means Clustering
- Hierarchical Methods- Hierarchical Clustering
- Fuzzy Clustering- Fuzzy C-Means Clustering
- Model- Based Clustering- Normal Mixture Model Clustering

Before we move on to Clustering methodologies, we need to find the optimal number of clusters as certain methods require prior knowledge of number of clusters before performing the clustering methodology.

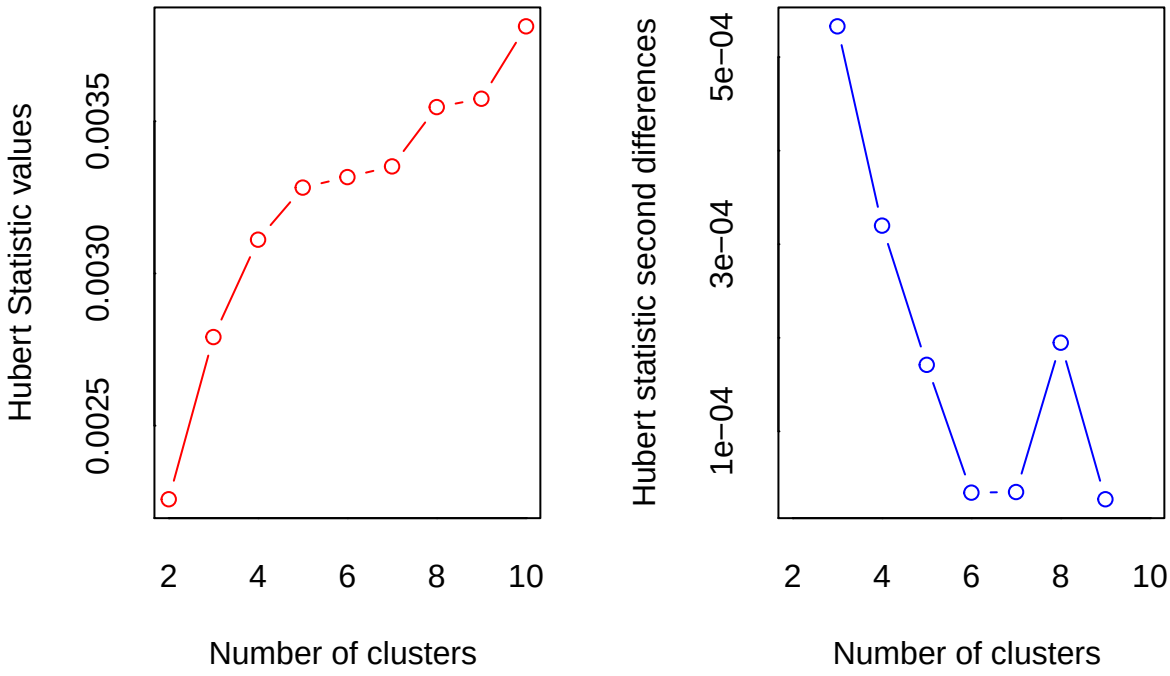
2.6.1.1 Kmeans

```
set.seed(1234)
irisdata <- iris[,-5]
iris.scaled <- scale(irisdata)
```

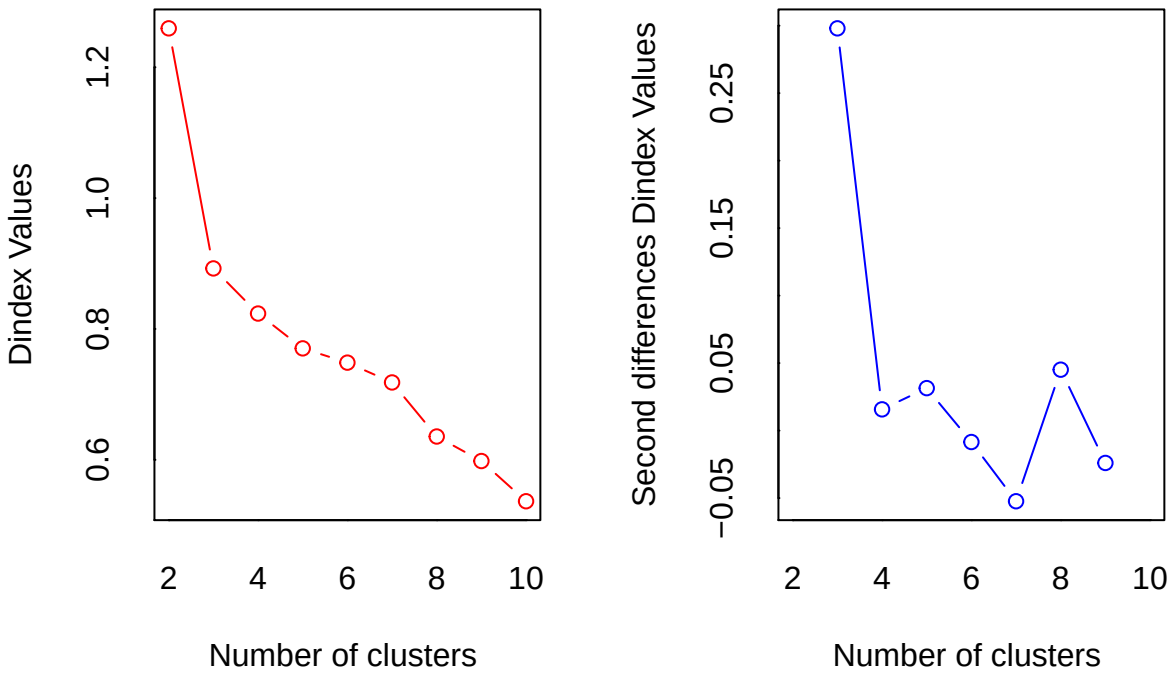
```
head(iris.scaled)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]   -0.8976739  1.01560199    -1.335752   -1.311052
## [2,]   -1.1392005 -0.13153881    -1.335752   -1.311052
## [3,]   -1.3807271  0.32731751    -1.392399   -1.311052
## [4,]   -1.5014904  0.09788935    -1.279104   -1.311052
## [5,]   -1.0184372  1.24503015    -1.335752   -1.311052
## [6,]   -0.5353840  1.93331463    -1.165809   -1.048667
```

```
# The Number of Clusters Required
nb <- NbClust(iris.scaled, distance = "euclidean", min.nc = 2,
              max.nc = 10, method = "complete", index ="all")
```




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

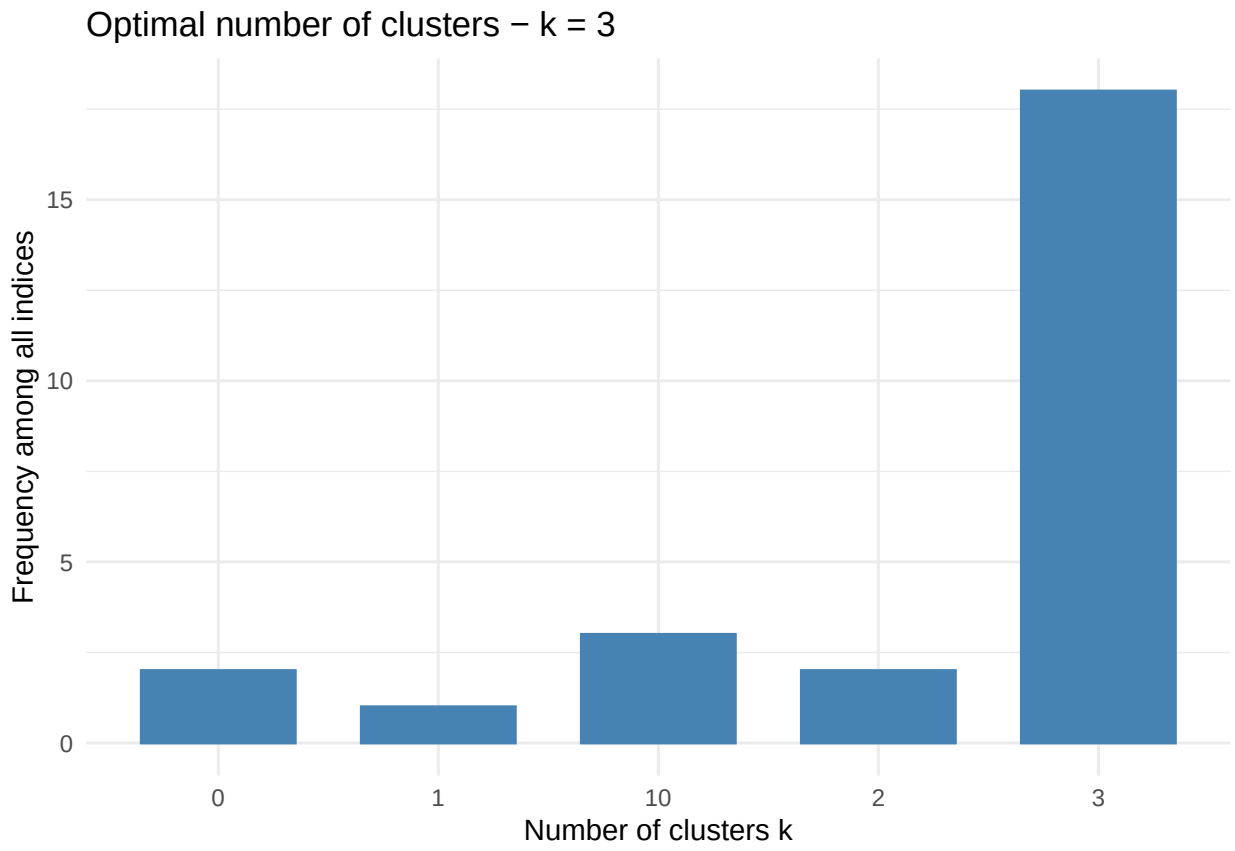



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 2 proposed 2 as the best number of clusters
## * 18 proposed 3 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  3
##
##
## *******************************************************************
```

```
fviz_nbclust(nb) + theme_minimal()
```

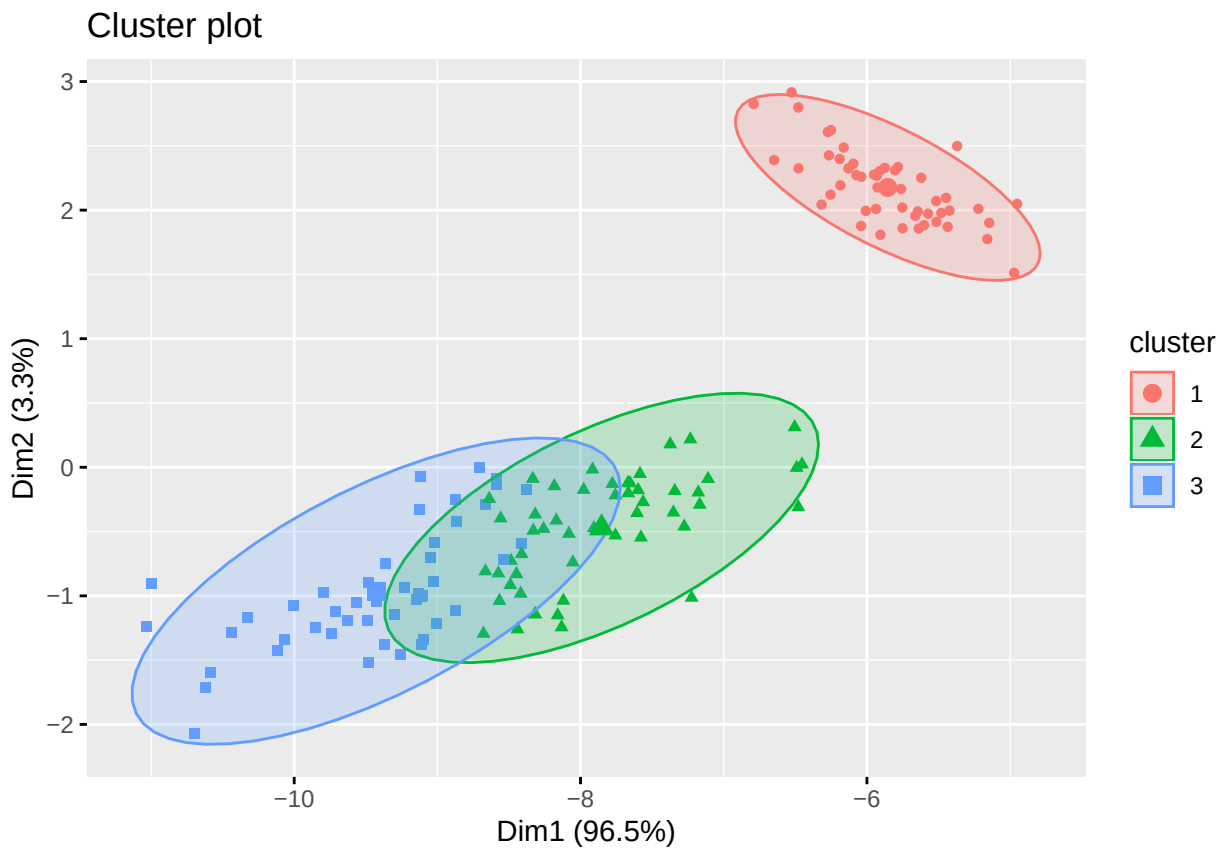
```
## Among all indices:
## ==================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 2 proposed  2 as the best number of clusters
## * 18 proposed  3 as the best number of clusters
## * 3 proposed  10 as the best number of clusters
```

```
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  3 .
```




```
irisKmeans <- kmeans(iris.scaled, 3, nstart = 20)

fviz_cluster(irisKmeans, data = irisdata, geom = "point",
             stand = FALSE, ellipse.type = "norm")
```




```
# Stats Kmeans
iris.dist <- dist(iris.scaled)
stat_kmeans <- cluster.stats(iris.dist, irisKmeans$cluster)
within_kmeans <- stat_kmeans$within.cluster.ss
avg_kmeans <- stat_kmeans$avg.silwidth
```

2.6.1.2 Heirarchical Clustering With Ward's Method

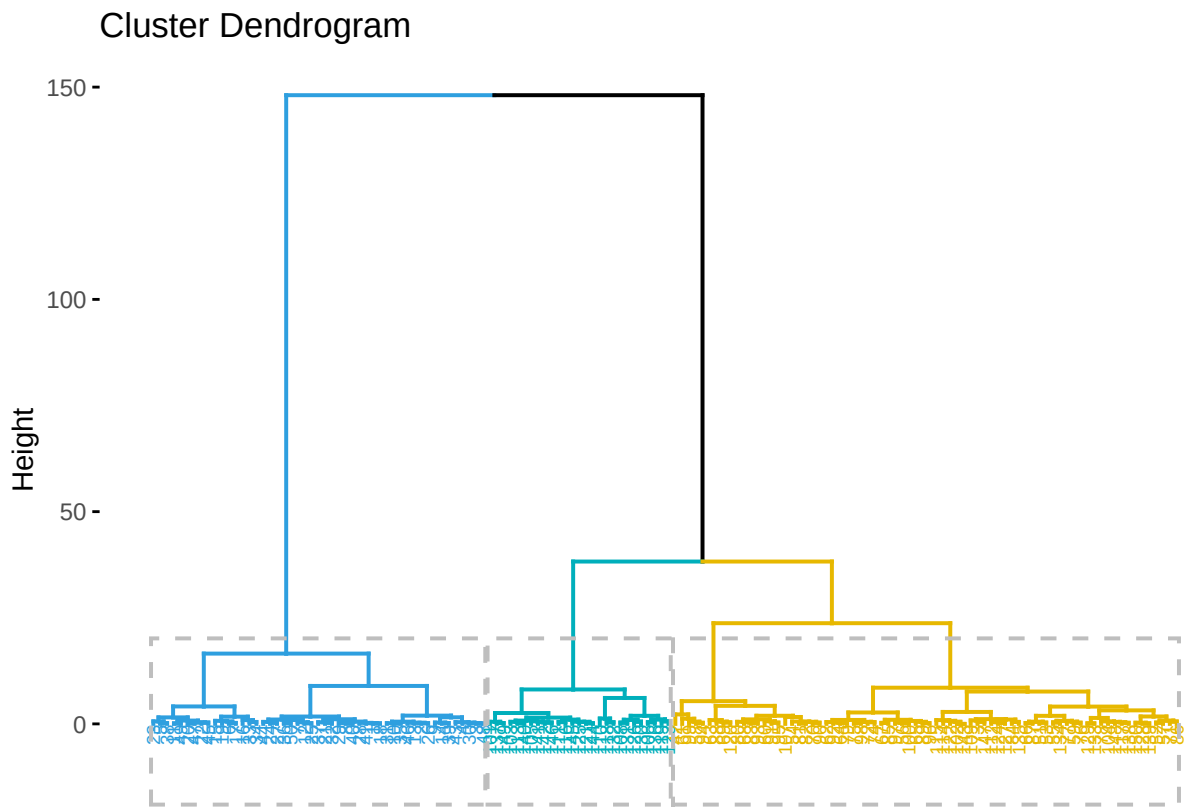
Hierarchical clustering can be divided into agglomerative and divisive clustering. The former starts with a single element and builds the nodes based on similar clusters to form a final big cluster or root. The latter starts with a big cluster and breaks it until all data points are in different clusters.

Ward's Method

This minimizes the total within-cluster variance. At each of the steps, the pairs of clusters with minimum between-cluster distances are merged.

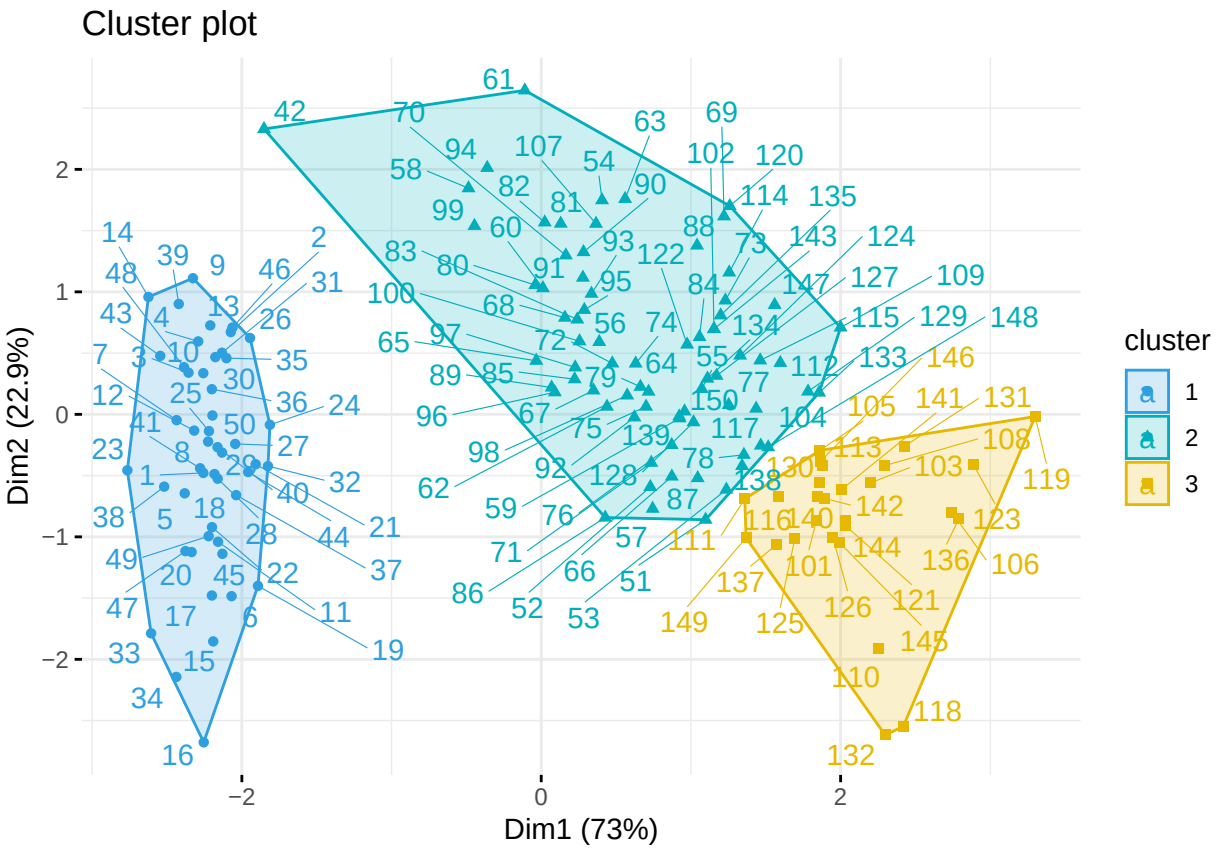
```
irisHclust <- hclust(iris.dist, method="ward.D")
```

```
fviz_dend(irisHclust, k = 3, # Cut in three groups
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
)
```




```
# Cut tree into 3 groups
grp <- cutree(irisHclust, k = 3)

fviz_cluster(list(data = iris.scaled, cluster = grp),
palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
ellipse.type = "convex", # Concentration ellipse
repel = TRUE, # Avoid label overplotting (slow)
show.clust.cent = FALSE, ggtheme = theme_minimal())
```



2.6.1.3 Fuzzy C-Means Clustering

In K-means, the data is divided into distinct clusters, where each element is affected exactly to one cluster. This type of clustering is also known as hard clustering or non-fuzzy clustering.

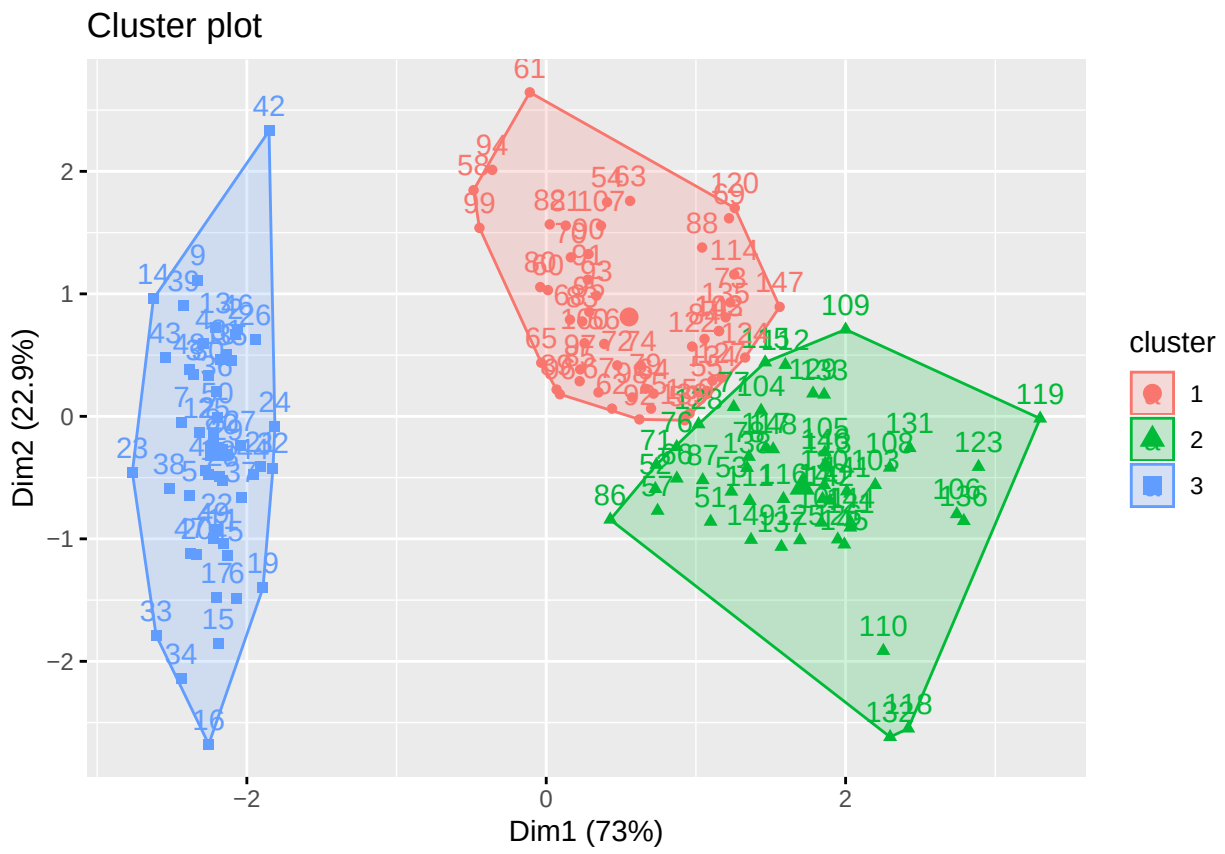
Unlike K-means, Fuzzy clustering is considered as a soft clustering, in which each element has a probability of belonging to each cluster. In other words, each element has a set of membership coefficients corresponding to the degree of being in a given cluster.

Points close to the center of a cluster, may be in the cluster to a higher degree than points in the edge of a cluster. The degree, to which an element belongs to a given cluster, is a numerical value in [0, 1].

Fuzzy c-means (FCM) algorithm is one of the most widely used fuzzy clustering algorithms. It was developed by Dunn in 1973 and improved by Bezdek in 1981. It's frequently used in pattern recognition.

```
irisCmeans <- cmeans(iris.scaled, 3)
```

```
fviz_cluster(list(data = iris.scaled, cluster=irisCmeans$cluster))
```




```
# Stats Cmeans
stat_cmeans <- cluster.stats(iris.dist, irisCmeans$cluster)
within_cmeans <- stat_cmeans$within.cluster.ss
avg_cmeans <- stat_cmeans$avg.silwidth
```

2.6.1.4 Normal Mixture Model Clustering

The traditional clustering methods such as hierarchical clustering and partitioning algorithms (k-means and others) are heuristic and are not based on formal models.

An alternative is to use model-based clustering, in which, the data are considered as coming from a distribution that is mixture of two or more components (i.e. clusters) (Chris Fraley and Adrian E. Raftery, 2002 and 2012).

Each component k (i.e. group or cluster) is modeled by the normal or Gaussian distribution which is characterized by the parameters: mean vector and covariance matrix, an associated probability in the mixture. Each point has a probability of belonging to each cluster.

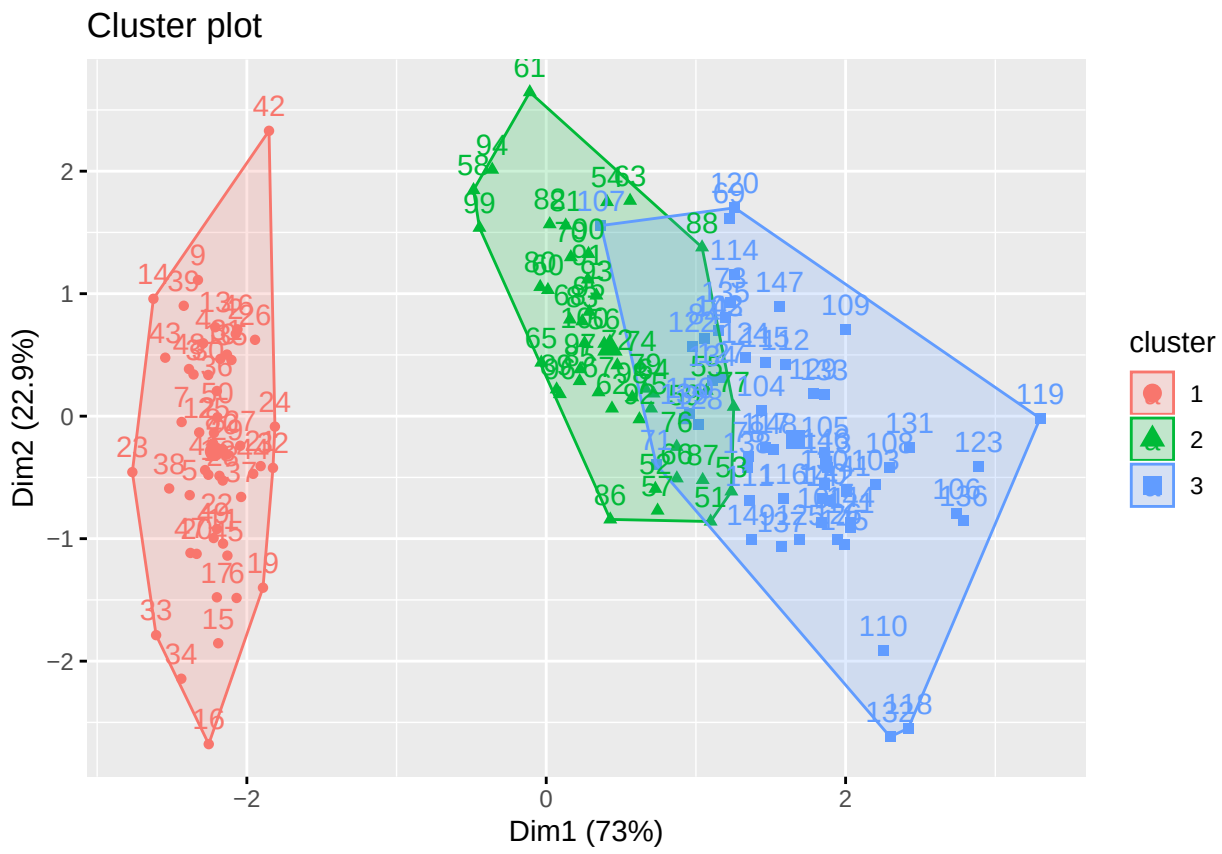
```
irisMclust <- Mclust(iris.scaled,G=3)
```

```
summary(irisMclust)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model
## with 3 components:
##
```

```
##  log-likelihood   n df      BIC       ICL
##       -288.5255 150 44 -797.519 -800.7391
##
## Clustering table:
##  1  2  3
## 50 45 55
```

```
fviz_cluster(list(data = iris.scaled, cluster=irisMclust$classification))
```




```
# Stats Mclust
stat_mclust <- cluster.stats(iris.dist, irisMclust$classification)
within_mclust <- stat_mclust$within.cluster.ss
avg_mclust <- stat_mclust$avg.silwidth
```

2.6.2 Comment : Kmeans Versus Cmeans

```
statsmodelsCmeans <- c(within_cmeans, avg_cmeans)
statsmodelsKmeans <- c(within_kmeans, avg_kmeans)
resultsCmeans <- data.frame("Cmeans" = c("within.cluster.ss","avg.silwidth"), "Stats Cmeans" = statsmode
resultsKmeans <- data.frame("Kmeans" = c("within.cluster.ss","avg.silwidth"), "Stats Kmeans" = statsmode
resultfinal <- cbind(resultsCmeans,resultsKmeans)
```

```
# Comparison Table
kable(arrange(resultfinal,desc(statsmodelsCmeans),desc(statsmodelsKmeans)), digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = F,
                font_size = 18,
                position = "center")
```

Cmeans	Stats.Cmeans	Kmeans	Stats.Kmeans
within.cluster.ss	139.106	within.cluster.ss	138.888
avg.silwidth	0.458	avg.silwidth	0.460

Comments: Based on the results of the above table, we conclude that the Kmeans is **the best model because it has within.cluster.ss smaller and one avg.silwidth larger than the Cmeans**.