Cluster analysis - Practice 3

José Luis Romero Béjar - Guillermo Arturo Cañadas De la Fuente

Junio, 2024

In this practice, two examples of cluster analysis are illustrated using, first, a **hierarchical method (Ward)** and then, using a **non-hierarchical (K-means)** method.

To carry out this practice you must download the following files available on the PRADO platform of the course:

- *CA_3_en.Rmd*

This brief guide is intended to familiarize the reader with the following:

- R packages required.
- Data preparation.
- Some distances for cluster analysis.
- Hierarchical clustering algorithm.
- Non-hierarchical clustering algorithm.

This material is an adaptation of a Bradley Boehmke R course, specifically, the part that refers to cluster analysis.

Loading packages and data set

Loading and installing R packages

The following source code module is responsible for loading, if they are already installed, all the packages that will be used in this R session. While an R package can be loaded at any time when it is to be used, it is advisable to optimize its calls with this code chunk at the beginning.

Loading a package into an R session **requires it to be already installed**. If it is not, the first step is to run the sentence:

install.packages("name_of_the_library")

```
########################################
# Loading necessary packages and reason #
########################################

# This is an example of the first installation of a package
# Only runs once if the package is not installed
# Once it is installed this sentence has to be commented (not to run again)
# install.packages("factoextra")

# Package required to call 'mutate' function
library(tidyverse)
```

```r
# Package required to call 'clusGap' function
library(cluster)

# Package required to call 'get_dist', 'fviz_cluster' and 'fviz_dist' functions
library(factoextra)

# Package required to call 'ggdendrogram' function
library(ggdendro)

# Package required to call 'grid.arrange' function
library(gridExtra)
```

Data preparation

To perform a cluster analysis with R, the data must be prepared as follows:

- You must ensure that the **rows** are **records of observations** and that the **columns** are the **variables** of interest (data.frame type structure).
- The **missing values** must be adequately treated (eliminate or replace their value).
- It must be decided whether to **work with standardized data** to make variables measured on different scales comparable.

For this practice we will work with the *USArrests* data set from the R base repository. This data set contains statistics on arrests per 100,000 inhabitants according to the crime: assault, murder and rape in each of the 50 states of the USA in the year 1973. It also includes the percentage of the population residing in urban areas. For this example it is decided to **omit all missing values** and **the data is standardized**.

```r
# Data loading
df<-USArrests
# Visualization of a few records
head(df)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```r
# Decide to delete not available data
df<-na.omit(df)

# To prevent the cluster analysis from being influenced by any arbitrary variable, the data are standar
df<-as.data.frame(scale(df))

# Visualization of standardized data
head(df)
```

```
##                Murder   Assault   UrbanPop         Rape
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
## California 0.27826823 1.2628144  1.7589234  2.067820292
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

Cluster analysis

Some distances for cluster analysis

To classify observations into groups it is necessary to choose appropriate measures of **similarity**, or **distance** (dissimilarity), that provide information on how similar any two observations are. In fact, this choice influences the size and shape of the clusters. This choice is a **fundamental step** in clustering.

Some **classical distance measures** frequently used are the **Euclidean** distance or the **Manhattan** distance.

Other measures of dissimilarity widely used, for example, in the analysis of gene expression data are **correlation-based distances**. These distances are obtained by subtracting the corresponding correlation measure from the value 1. Among these distance measures, the following stand out: the distance based on the **Pearson** correlation, on the **Spearman** correlation, and **Kendall** correlation, etc.

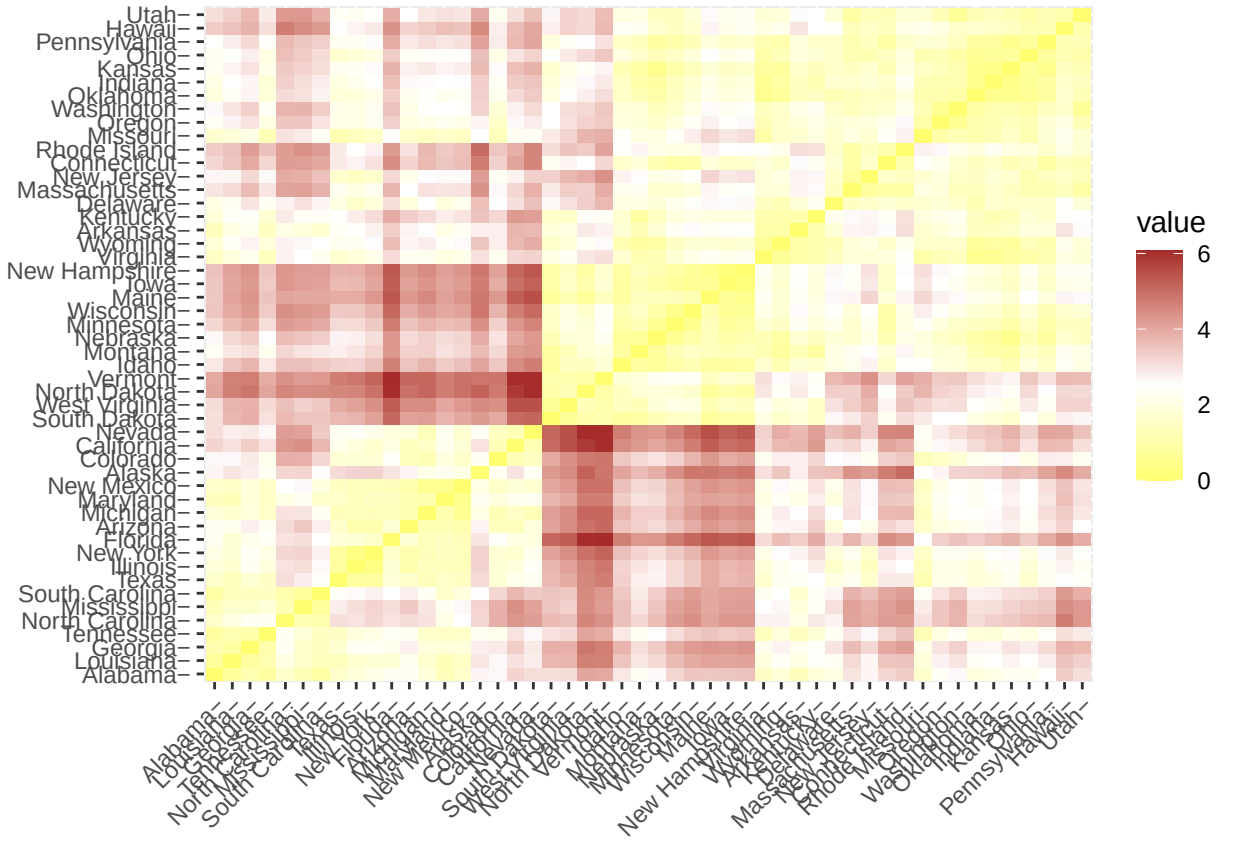
Click this link to view a formal expression of these distances.

As said before, the choice of distance is very important. Almost all the usual software for cluster analysis uses the Euclidean distance, although depending on the type of data and the research questions posed, another measure of dissimilarity or distance may be of interest.

In R language it is easy to calculate and visualize the distance matrix between observations with the *get_dist* and *fviz_dist* functions, respectively, included in the *factoextra* package. In fact, although by default, *get_dist* calculates the Euclidean distance, it also accepts all the distances mentioned above as parameter.

The following distance matrix shows in **brown those states that present large dissimilarities** (distances), compared to those that seem **closer in yellow**. The color white is used to refer to those states with distances that are not so extreme as to be considered low or high.

```
distance<- get_dist(df)
fviz_dist(distance, gradient = list(low ="yellow", mid = "white", high = "brown"))
```

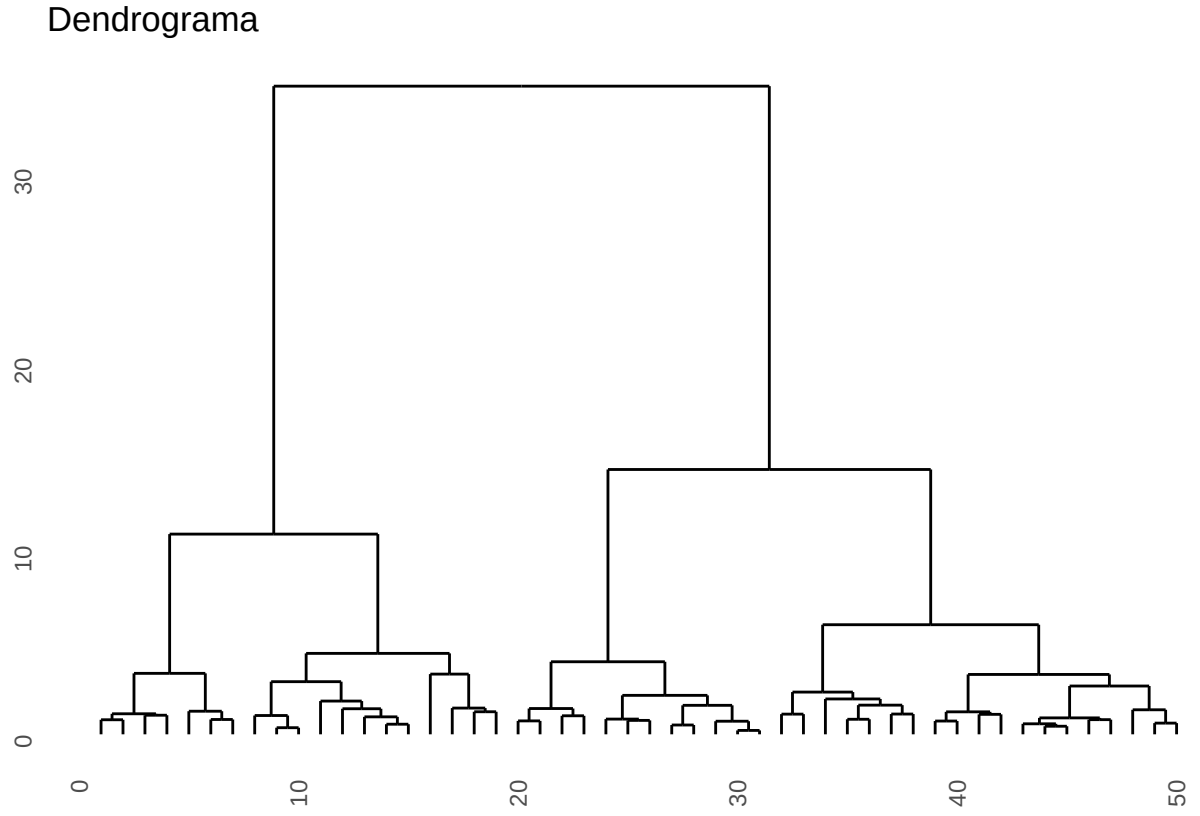


Hierarchical clustering: Ward's method

Hierarchical clustering is interested in finding a hierarchy based on the closeness or similarity of the data according to the distance considered. In the **agglomerative** case, we start from a group with the closest observations. The next closest pairs are then calculated and groups are generated in an ascending manner. This construction can be observed visually by means of a **dendrogram**.

Below it will be illustrated how the groups are defined by the number of vertical lines in the dendrogram, and the selection of the optimal number of groups can be estimated from this same graph.

```
dendrogram <- hclust(dist(df, method = 'euclidean'), method = 'ward.D')
ggdendrogram(dendrogram, rotate = FALSE, labels = FALSE, theme_dendro = TRUE) +
  labs(title = "Dendrograma")
```



On the **horizontal axis** of the dendrogram we have *each of the data* that make up the input set, while on the **vertical axis** the *Euclidean distance* that exists between each group is represented, as that these are becoming hierarchical.

Each **vertical line** in the diagram represents *a grouping*. As you go up the dendrogram you end up with a single large group determined by the upper horizontal line. So, **as you go down the hierarchy**, you see that from a single group it goes to 2, then to 3, then to 6, and so on.

One way to determine the appropriate ***K* number of groups** is to cut the dendrogram at that height of the diagram that best represents the input data.

Non-hierarchical clustering: K-means algorithm

Click this link to see a detailed description of this algorithm.

The R language implements the **K-means algorithm** with the function of the same name. This function receives as input parameters the data and the number of groupings to be performed (*centers* parameter). To address the problem of **choosing initial seed points** it incorporates the *nstart* parameter that tests multiple initial configurations and reports on the best one. For example, if *nstart = 25*, it will generate 25 initial configurations. The use of this parameter is recommended.

For this first example the *kmeans* function builds two clusters.

```
k2 <- kmeans(df, centers = 2, nstart = 25)

# Displaying all the fields of the object k2
str(k2)
```

```
## List of 9
```

```
##  $ cluster     : Named int [1:50] 2 2 2 1 2 2 1 1 2 2 ...
##   ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ centers     : num [1:2, 1:4] -0.67 1.005 -0.676 1.014 -0.132 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ totss       : num 196
##  $ withinss    : num [1:2] 56.1 46.7
##  $ tot.withinss: num 103
##  $ betweenss   : num 93.1
##  $ size        : int [1:2] 30 20
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

The output provided by the *kmeans* function is a list of information, including the following:

- *cluster*: is a vector of integers, from 1 to K (K=2 in this case), which indicates the cluster in which each observation has been located.
- *centers*: a matrix with the successive centers of the clusters.
- *totss*: the total sum of squares.
- *withinss*: sum of squares vector within each cluster (one component per cluster).
- *tot.withinss*: total sum of squares of the clusters, i.e. sum(withinss).
- *betweens*: sum of squares between groups, i.e. totss-tot.withinss.
- *size*: the number of observations in each cluster.

When displaying the variable *k2* it is seen how the groupings result in 2 grouping sizes of 30 and 20 states. The centers of each group (means) in the four variables (Murder, Assault, UrbanPop, Rape) are also seen. And finally the group assignment for each observation (i.e., Alabama was assigned to group 2, Arkansas was assigned to group 1, etc.).

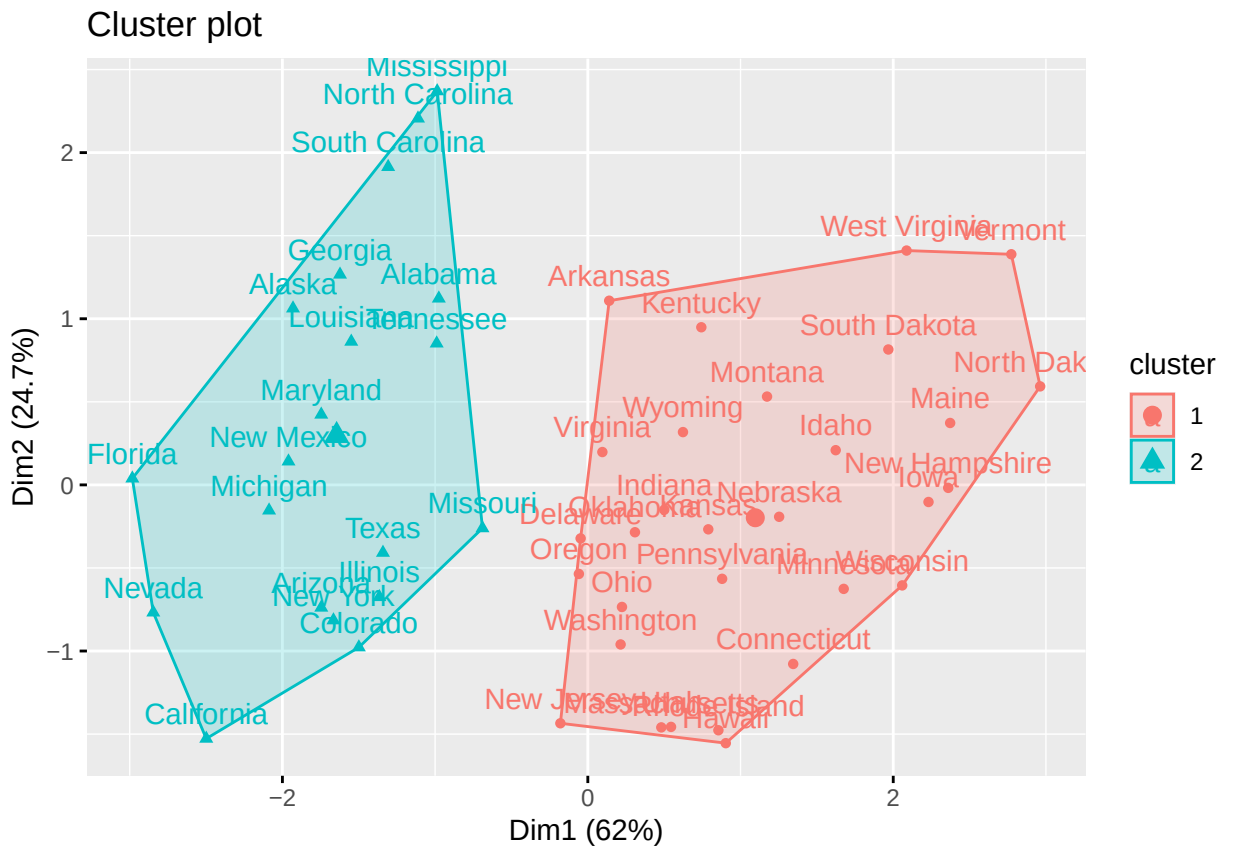
```
k2
```

```
## K-means clustering with 2 clusters of sizes 30, 20
##
## Cluster means:
##      Murder    Assault   UrbanPop       Rape
## 1 -0.669956 -0.6758849 -0.1317235 -0.5646433
## 2  1.004934  1.0138274  0.1975853  0.8469650
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              2              2              2              1              2
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              1              1              2              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              1              1              2              1              1
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              1              1              2              1              2
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              1              2              1              2              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              1              1              2              1              1
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              2              2              2              1              1
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
```

```
##              1              1              1              1              2
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              1              2              2              1              1
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              1              1              1              1              1
##
## Within cluster sum of squares by cluster:
## [1] 56.11445 46.74796
##  (between_SS / total_SS =  47.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

A visual way to summarize the results elegantly and with straightforward interpretation is by using the *fviz_cluster* function.

```
fviz_cluster(k2,data=df)
```

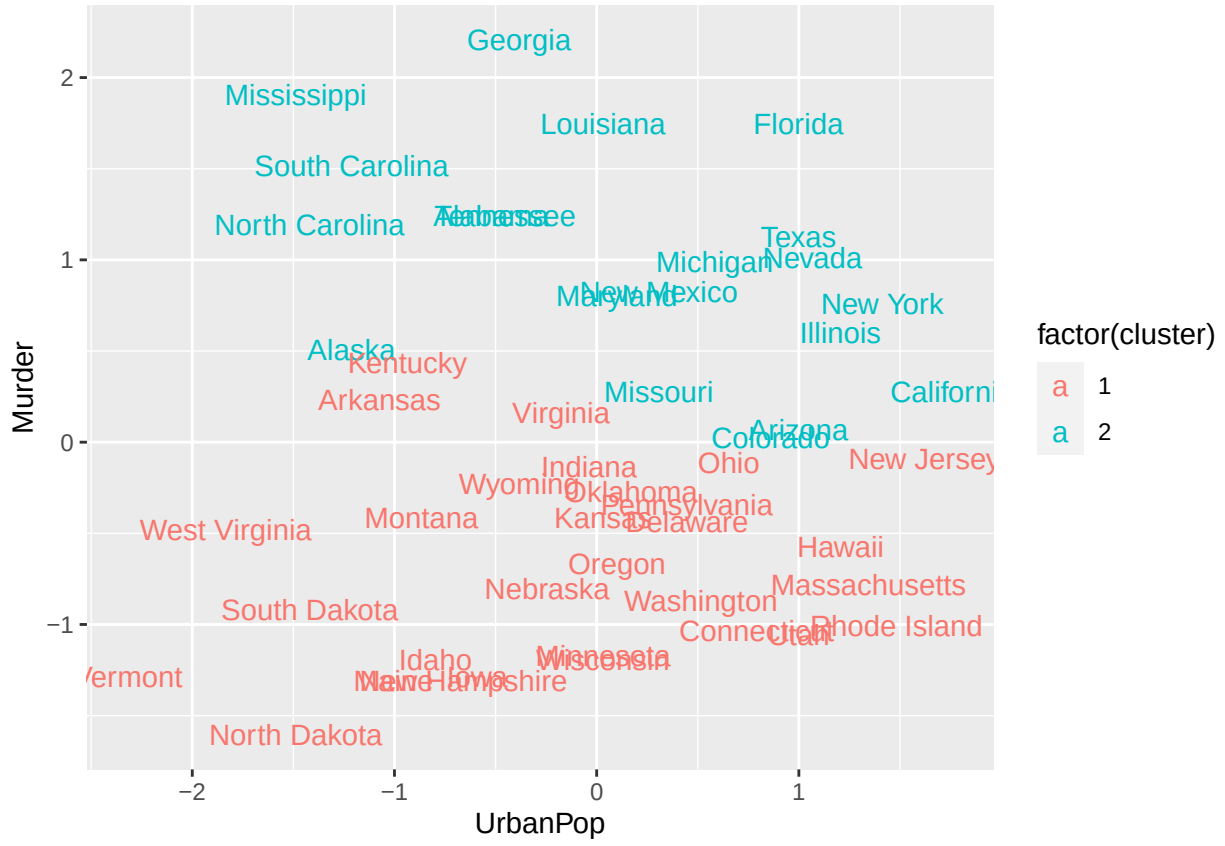


Remark:

If there are more than two dimensions (variables), this function will first perform a principal component analysis (PCA) and draw the points according to the first two principal components obtained (the ones that explain most of the variance). This is why in the graph above, *Dim1* and *Dim2* refer to these two main components.

Alternatively a pairwise scatterplot can be used to illustrate the groups in comparison to the original variables.

```
df %>%
  as_tibble() %>%
  mutate(cluster = k2$cluster,
         state = row.names(USArrests)) %>%
  ggplot(aes(UrbanPop, Murder, color = factor(cluster), label = state)) +
  geom_text()
```

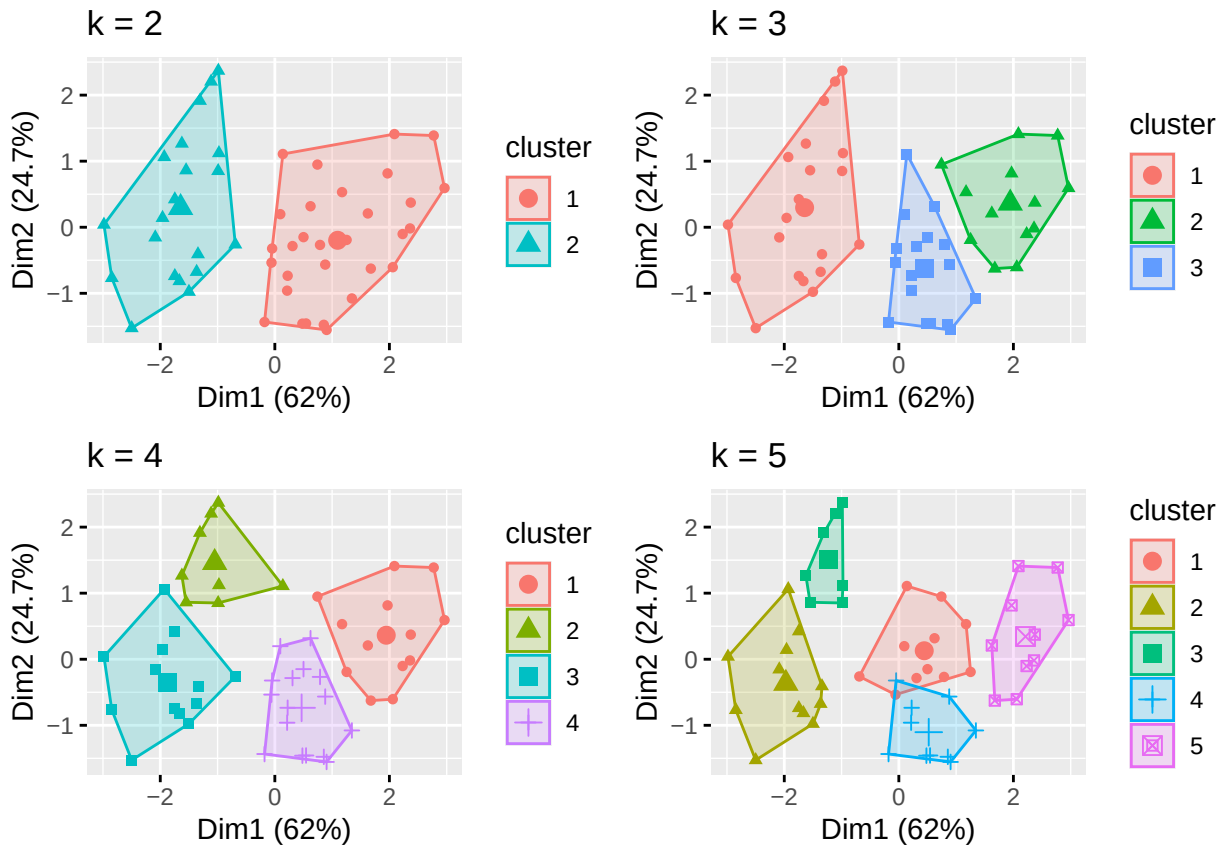


To use the K-means algorithm, the **K number of clusters must be fixed in advance**. This is why it is advisable to run the same process with other values of K (in this example for K= 3, 4 and 5) to compare and examine the differences between the results.

```
set.seed(123)
k3 <- kmeans(df, centers = 3, nstart = 25)
k4 <- kmeans(df, centers = 4, nstart = 25)
k5 <- kmeans(df, centers = 5, nstart = 25)

# Plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = df) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = df) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = df) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = df) + ggtitle("k = 5")


grid.arrange(p1, p2, p3, p4, nrow = 2)
```



Although this visualization allows us to deduce where the true differences occur (or do not occur, as in clusters 1 and 4 in the graph for K=5), it **does not tell us what the optimal number of clusters is**.

Determination of the optimal number of clusters

As indicated before, when a non-hierarchical method such as K-means is applied to perform a cluster analysis, the researcher must inform a priori of the desired number of clusters. In this sense, this researcher will be interested in **providing an optimal number of groups** to form.

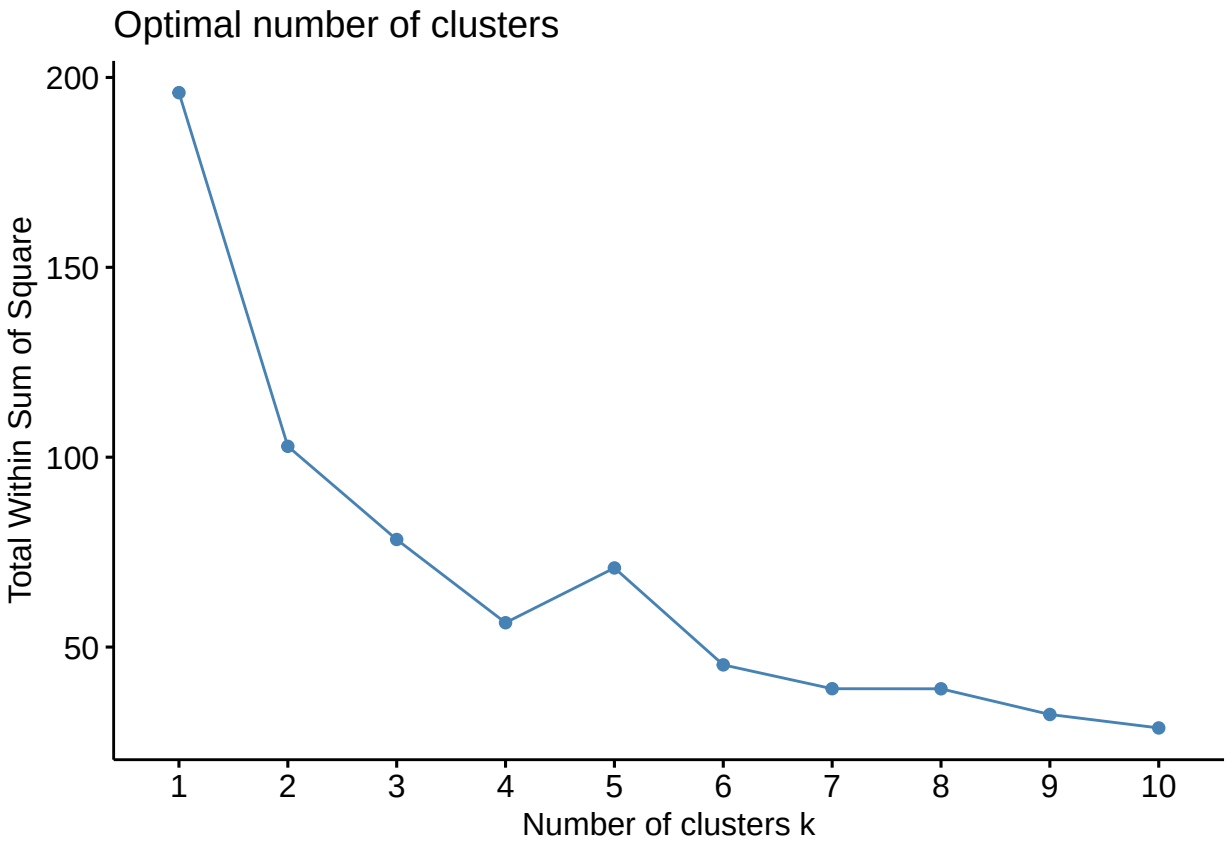
Below are three of the most used methods to determine this optimal number of groups: Elbow method, Silhouette method and the Gap statistic.

Elbow method Keeping in mind that the idea behind a division into K clusters is to obtain these groupings so that the total intra-cluster variance is minimal (total within-cluster variation or total within-cluster sum of square), you can use the following algorithm to identify the optimal number of clusters:

- Run a clustering algorithm (such as K-means) for different values of K (for example K=1,...,10).
- For each K the total intra-cluster variation is calculated (total within-cluster sum of square, which we denote here by *wss*).
- The *wss* curve is drawn according to the number of clusters K.
- The location on this curve of a 'cubit or elbow' is taken as the most appropriate indicator of the number of clusters.

Although we can program this algorithm in R language, Elbow's method is implemented in the *fviz_nbclust* function.

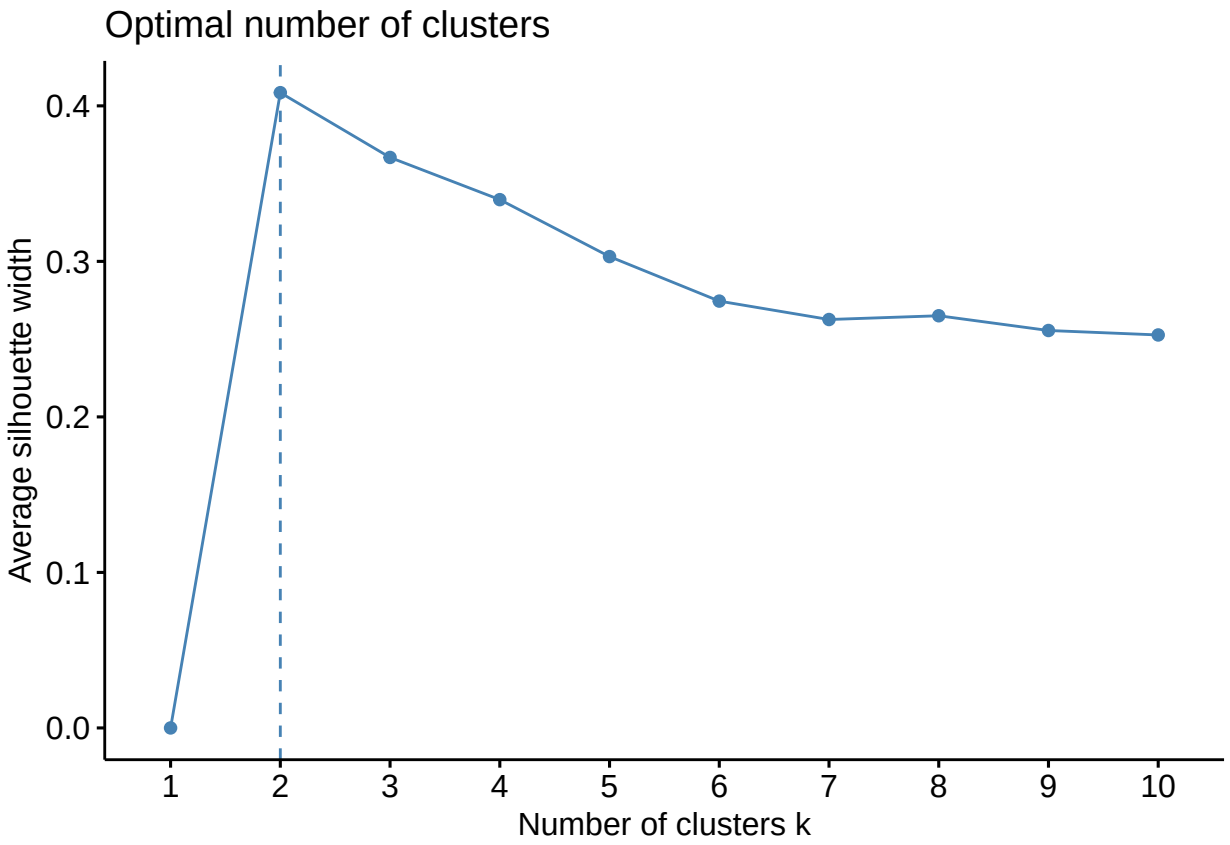
```
set.seed(123)
fviz_nbclust(df, kmeans, method = "wss")
```



Silhouette Method This approach, *average silhouette*, measures the quality of a cluster. That is, it determines how suitable an object is within its cluster. The optimal number of clusters according to this approach is, among a range of possible values for K, the one that maximizes the average silhouette.

As before, this algorithm can be programmed in R language, but the *fviz_nbclust* function also includes it.

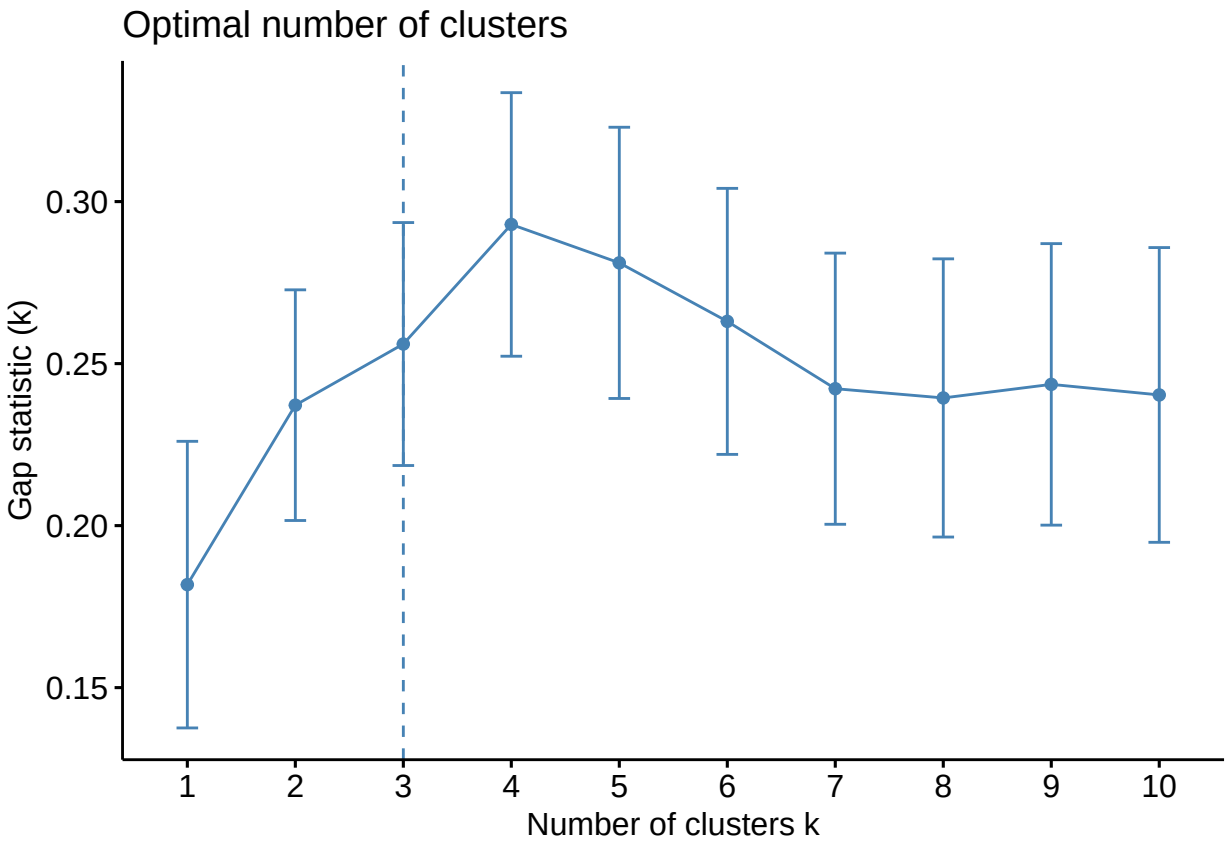
```
set.seed(123)
fviz_nbclust(df, kmeans, method = "silhouette")
```



Gap Statistical Method This method compares the total intracluster variation for different values of K. It uses Monte Carlo simulation in its algorithm.

The *clusGap* function provides the GAP statistic and its standard error for an output. With the function *fviz_gap_stat* you obtain a graphical representation that suggests a number of clusters.

```
set.seed(123)
gap_stat <- clusGap(df, FUN = kmeans, nstart = 25,K.max = 10, B = 50)
fviz_gap_stat(gap_stat)
```



Analysis of results

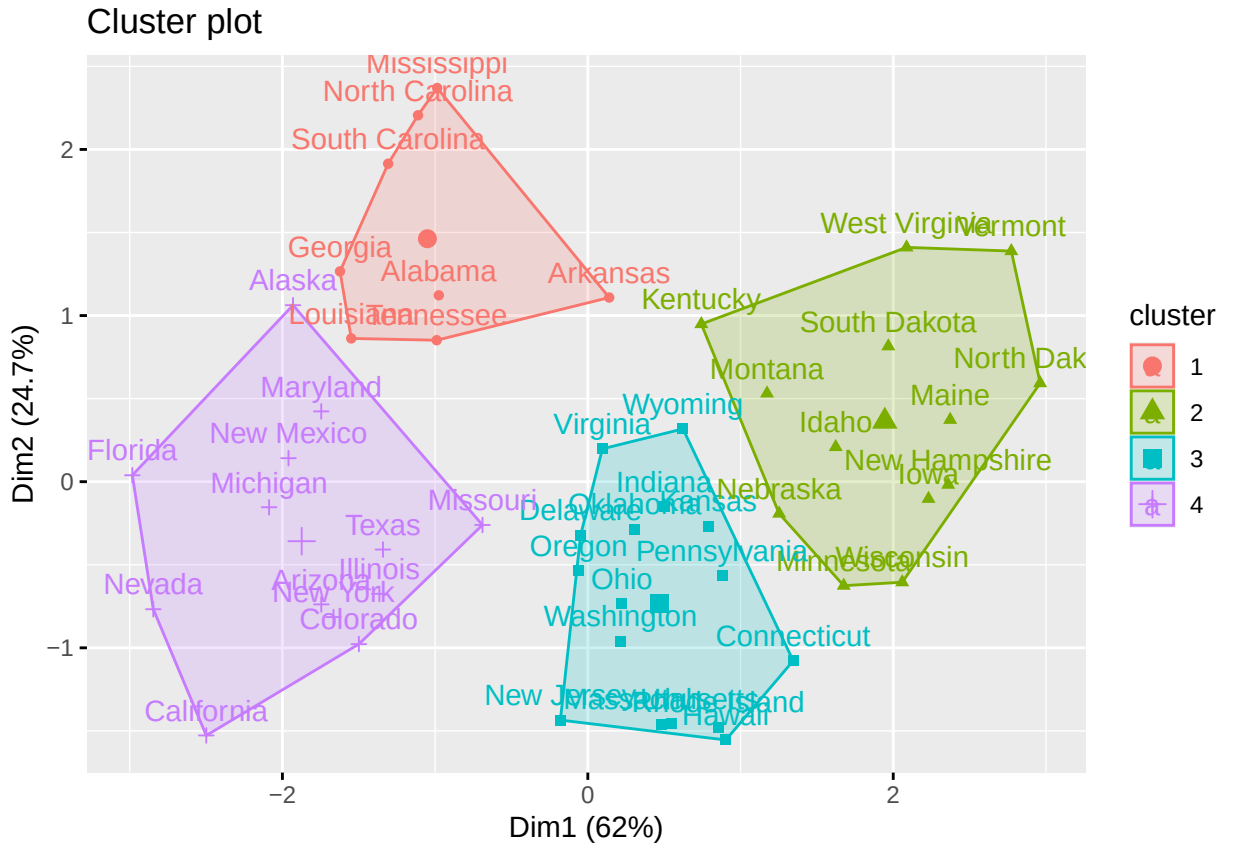
After the detailed analysis of the optimal number of clusters, carried out in the previous section, **K=4** seems to be the most appropriate number of groups for this analysis.

```
set.seed(123)
final <- kmeans(df, 4, nstart = 25)
print(final)
```

```
## K-means clustering with 4 clusters of sizes 8, 13, 16, 13
##
## Cluster means:
##       Murder    Assault   UrbanPop        Rape
## 1  1.4118898  0.8743346 -0.8145211  0.01927104
## 2 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 3 -0.4894375 -0.3826001  0.5758298 -0.26165379
## 4  0.6950701  1.0394414  0.7226370  1.27693964
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              4              4              1              4
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              4              3              3              4              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              2              4              3              2
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              2              1              2              4
```

```
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              3              4              2              1              4
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              2              2              4              2              3
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              4              4              1              2              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              3              3              3              3              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              2              1              4              3              2
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              3              3              2              2              3
##
## Within cluster sum of squares by cluster:
## [1]  8.316061 11.952463 16.212213 19.922437
##  (between_SS / total_SS =  71.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Final output
fviz_cluster(final, data = df)
```



Finally, it is possible to extract the clusters and add them to our initial data to provide some cluster-level descriptive statistics.

```r
USArrests %>%
  mutate(Cluster = final$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
```

```
## # A tibble: 4 x 5
##   Cluster Murder Assault UrbanPop  Rape
##     <int>  <dbl>   <dbl>    <dbl> <dbl>
## 1       1  13.9    244.      53.8  21.4
## 2       2   3.6     78.5     52.1  12.2
## 3       3   5.66   139.      73.9  18.8
## 4       4  10.8    257.      76    33.2
```