**Partitioning clustering** are clustering methods used to classify observations, within a data set, into multiple groups based on their similarity. The algorithms require the analyst to specify the number of clusters to be generated.

• **K-means clustering** (MacQueen, 1967), in which, each cluster is represented by the center or means of the data points belonging to the cluster. The K-means method is sensitive to anomalous data points and outliers.

• **K-medoids clustering** or **PAM** (*Partitioning Around Medoids*, Kaufman & Rousseeuw, 1990), in which, each cluster is represented by one of the objects in the cluster. PAM is less sensitive to outliers compared to k-means.

• **CLARA algorithm** (*Clustering Large Applications*), which is an extension to PAM adapted for large data sets.

**K-Means Clustering**

**K-means clustering** (MacQueen, 1967) is the most commonly used unsupervised machine learning algorithm for partitioning a given data set into a set of k groups (i.e. *k clusters*), where k represents the number of groups pre-specified by the analyst. It classifies objects in multiple groups (i.e., clusters).

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

K-means algorithm can be summarized as follow:

1. Specify the number of clusters (K) to be created (by the analyst)

2. Select randomly k objects from the data set as the initial cluster centers or means

3. Assigns each observation to their closest centroid, based on the Euclidean distance between the object and the centroid

4. For each of the k clusters update the *cluster centroid* by calculating the new mean values of all the data points in the cluster. The centoid of a *Kth* cluster is a vector of length *p* containing the means of all variables for the observations in the *kth* cluster; *p* is the number of variables.

5. Iteratively minimize the total within sum of square. That is, iterate steps 3 and 4 until the cluster assignments stop changing or the maximum number of iterations is reached. By default, the R software uses 10 as the default value for the maximum number of iterations.

**Computing k-means clustering in R**

**1 Data**

> data("USArrests") #load the data

> df = scale(USArrests) # scalling the data

> head(df) # view the first 6 rows of the data

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

> head(df, n = 2) # view the first 2 rows of the data

Murder Assault UrbanPop Rape

Alabama 1.2425641 0.7828393 -0.5209066 -0.003416473

Alaska 0.5078625 1.1068225 -1.2117642 2.484202941

**2) Required R packages and functions**

The standard R function for k-means clustering is *kmeans*() [*stats* package], which

simplified format is as follow:

> kmeans(x, centers = 2, iter.max = 8, nstart = 2)

K-means clustering with 2 clusters of sizes 25, 25

Cluster means:

[,1] [,2]

1 -0.1956978 -0.1848774

2 3.3339737 -4.0761910

Clustering vector:

[1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[41] 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:

[1] 65.40068 63.20595

(between\_SS / total\_SS = 72.8 %)

Available components:

[1] "cluster" "centers" "totss" "withinss" "tot.withinss"

[6] "betweenss" "size" "iter" "ifault"

• **x**: numeric matrix, numeric data frame or a numeric vector

• **centers**: Possible values are the number of clusters (k) or a set of initial (distinct) cluster centers. If a number, a random set of (distinct) rows in x is chosen as the initial centers.

• **iter.max**: The maximum number of iterations allowed. Default value is 10.

• **nstart**: The number of random starting partitions when centers is a number. Trying nstart > 1 is often recommended.

> library(factoextra)

**Estimating the optimal number of clusters**

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

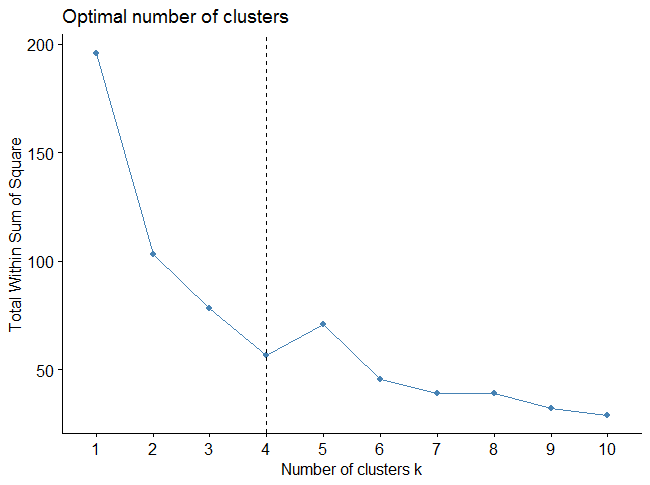
One fundamental question is: How to choose the right number of expected clusters (k)?

The R function *fviz\_nbclust*() [in *factoextra* package] provides a convenient solution

to estimate the optimal number of clusters.

> fviz\_nbclust(df, kmeans, method = "wss") +

+ geom\_vline(xintercept = 4, linetype = 2)



The plot above represents the variance within the clusters. It decreases as k increases, but it can be seen a bend (or “elbow”) at k = 4.

**4 Computing k-means clustering**

As k-means clustering algorithm starts with k randomly selected centroids, it’s always recommended to use the *set.seed()* function in order to set a seed for *R’s random number generator*. The aim is to make reproducible the results, so that the reader of this article will obtain exactly the same results as those shown below.

The R code below performs *k-means clustering* with k = 4:

> km.res = kmeans(df, 4, nstart = 25)

> print(km.res)

K-means clustering with 4 clusters of sizes 8, 16, 13, 13

Cluster means:

Murder Assault UrbanPop Rape

1 1.4118898 0.8743346 -0.8145211 0.01927104

2 -0.4894375 -0.3826001 0.5758298 -0.26165379

3 -0.9615407 -1.1066010 -0.9301069 -0.96676331

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 2 2 4 1

Hawaii Idaho Illinois Indiana Iowa

2 3 4 2 3

Kansas Kentucky Louisiana Maine Maryland

2 3 1 3 4

Massachusetts Michigan Minnesota Mississippi Missouri

2 4 3 1 4

Montana Nebraska Nevada New Hampshire New Jersey

3 3 4 3 2

New Mexico New York North Carolina North Dakota Ohio

4 4 1 3 2

Oklahoma Oregon Pennsylvania Rhode Island South Carolina

2 2 2 2 1

South Dakota Tennessee Texas Utah Vermont

3 1 4 2 3

Virginia Washington West Virginia Wisconsin Wyoming

2 2 3 3 2

Within cluster sum of squares by cluster:

[1] 8.316061 16.212213 11.952463 19.922437

(between\_SS / total\_SS = 71.2 %)

Available components:

[1] "cluster" "centers" "totss" "withinss" "tot.withinss"

[6] "betweenss" "size" "iter" "ifault"

The printed output displays:

• the cluster means or centers: a matrix, which rows are cluster number (1 to 4) and columns are variables.

• the clustering vector: A vector of integers (from 1:k) indicating the cluster to which each point is allocated.

# Cluster number for each of the observations

> km.res$cluster

Alabama Alaska Arizona Arkansas California

1 4 4 1 4

Colorado Connecticut Delaware Florida Georgia

4 2 2 4 1

Hawaii Idaho Illinois Indiana Iowa

2 3 4 2 3

Kansas Kentucky Louisiana Maine Maryland

2 3 1 3 4

Massachusetts Michigan Minnesota Mississippi Missouri

2 4 3 1 4

Montana Nebraska Nevada New Hampshire New Jersey

3 3 4 3 2

New Mexico New York North Carolina North Dakota Ohio

4 4 1 3 2

Oklahoma Oregon Pennsylvania Rhode Island South Carolina

2 2 2 2 1

South Dakota Tennessee Texas Utah Vermont

3 1 4 2 3

Virginia Washington West Virginia Wisconsin Wyoming

2 2 3 3 2

> head(km.res$cluster, 4)

Alabama Alaska Arizona Arkansas

1 4 4 1

> km.res$size # Cluster size

[1] 8 16 13 13

> # Cluster means

> km.res$centers

Murder Assault UrbanPop Rape

1 1.4118898 0.8743346 -0.8145211 0.01927104

2 -0.4894375 -0.3826001 0.5758298 -0.26165379

3 -0.9615407 -1.1066010 -0.9301069 -0.96676331

4 0.6950701 1.0394414 0.7226370 1.27693964

**6 Visualizing k-means clusters**

It is a good idea to plot the cluster results. These can be used to assess the choice of the number of clusters as well as comparing two different cluster analyses.

Now, we want to visualize the data in a scatter plot with coloring each data point according to its cluster assignment.

The problem is that the data contains more than 2 variables and the question is what variables to choose for the xy scatter plot.

A solution is to reduce the number of dimensions by applying a dimensionality reduction algorithm, such as **Principal Component Analysis (PCA)**, that operates on the four variables and outputs two new variables (that represent the original variables) that you can use to do the plot.

In other words, if we have a multi-dimensional data set, a solution is to perform Principal Component Analysis (PCA) and to plot data points according to the first two principal components coordinates.

# Visualizing k-means clusters

> fviz\_cluster(km.res, data = df,

+ palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),

+ ellipse.type = "euclid", # Concentration ellipse

+ star.plot = TRUE, # Add segments from centroids to items

+ repel = TRUE, # Avoid label overplotting (slow)

+ ggtheme = theme\_minimal()

+ )

