MASTER IN ENTREPRENEURSHIP
INNOVATION MANAGEMENT
IN COLLABORATION WITH MIT SLOAN

IN COLLABORATION WITH
MIT MANAGEMENT SLOAN SCHOOL

UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

MASTER MEIM 2021-2022

BIG DATA ANALYTICS

Master 2023

Unsupervised Learning for Innovation

Giuseppe Scandurra

Professor of Economic Statistics @ Parthenope University

BIG DATA ANALYTICS

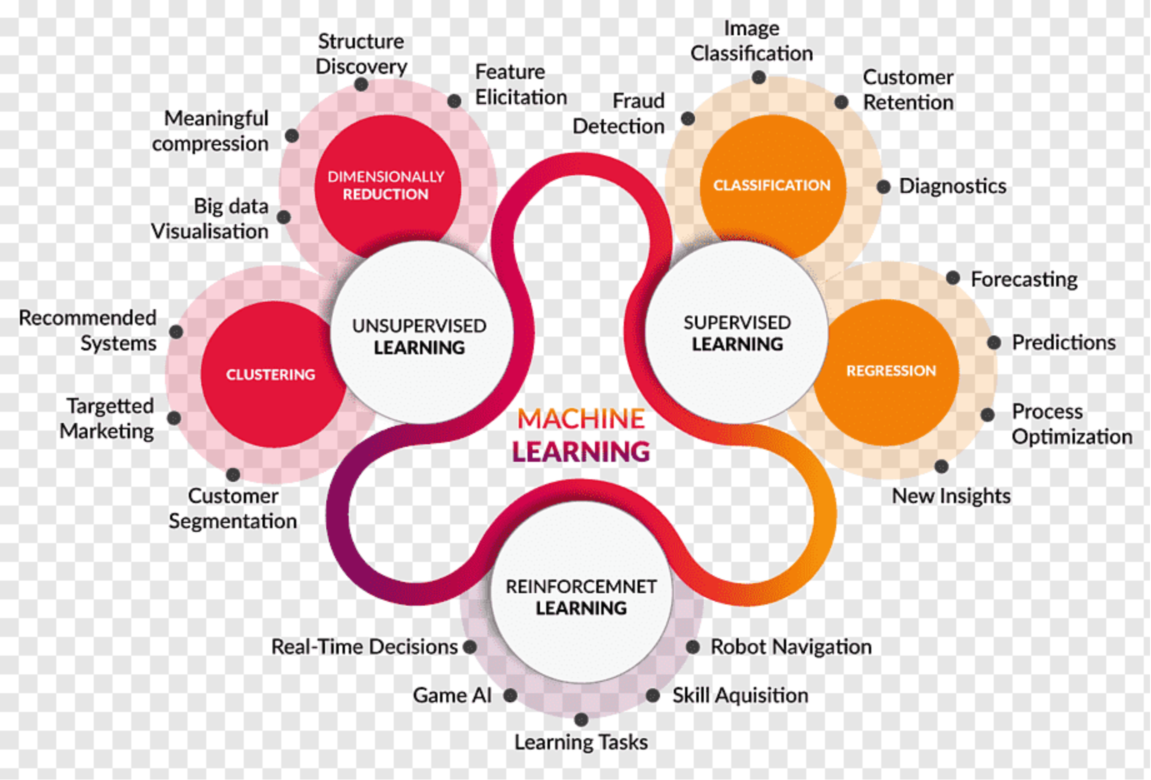
“Without big data analytics, companies are blind and deaf, wandering out onto the Web like deers on a freeway.”

Geoffrey Moore’s tweet (2012)

Statistical Learning

Supervised Learning

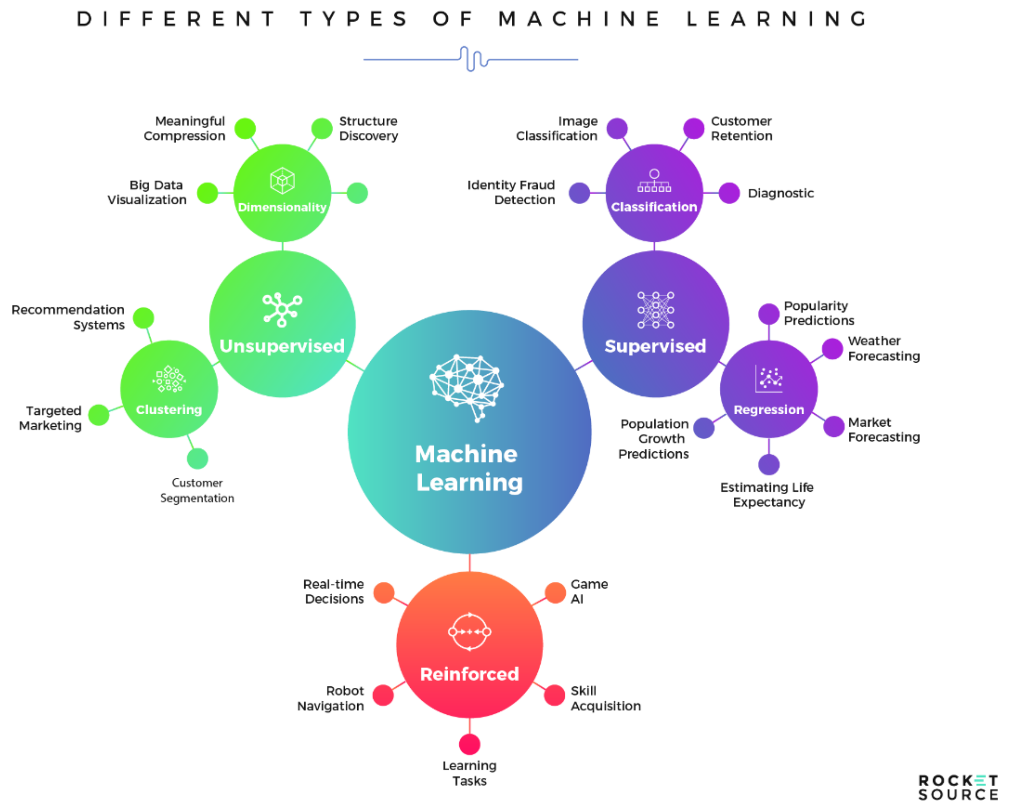
- Regression: linear, polynomial, spline
- Classification: logistic regression, k-nearest neighbors
- Decision trees (classification and regression trees)



Statistical Learning

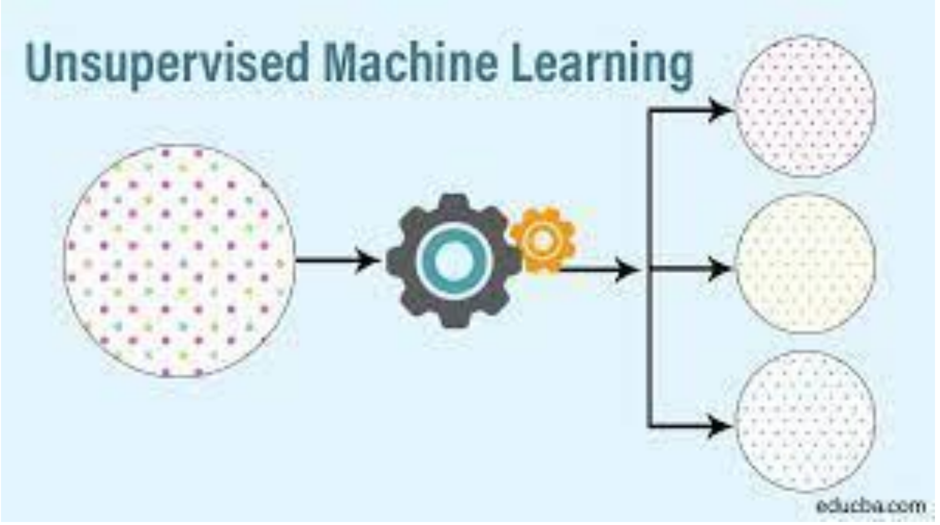
Unsupervised Learning

- Dimensionality reduction (e.g., Principal Component Analysis, Exploratory Factor Analysis)
- Cluster analysis



ROCKET SOURCE

Statistical Learning



Goals of Unsupervised Learning

- Consider a set of features X_1, X_2, …, X_p.
- Lack of outcome variable, Y.
- We are not interested in prediction, because we do not have an associated response variable Y .

The goal is to discover interesting things about the measurements:

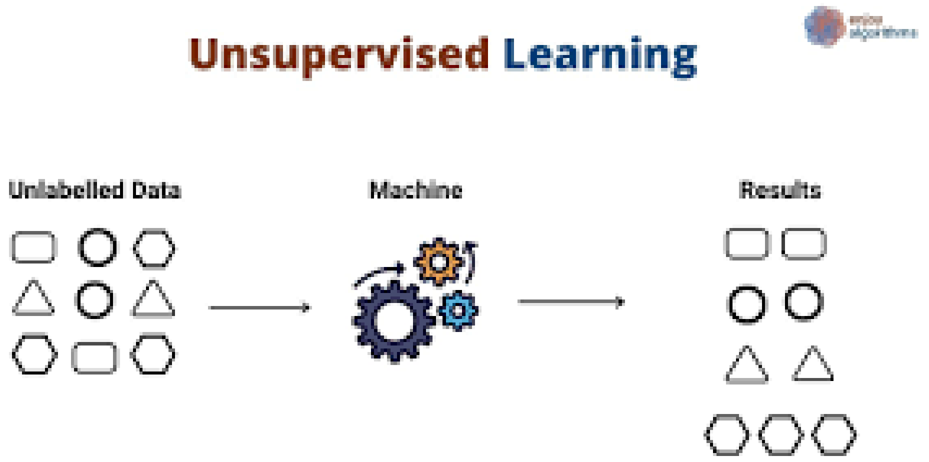
- Is there an informative way to visualize the data?
- Can we discover subgroups among the variables or among the observations?

Unsupervised Learning

Methods

We introduce various methods:

- Principal Components Analysis (PCA), a tool used for data visualization or data pre-processing before supervised techniques are applied;
- Exploratory Factor Analysis (EFA), a family of techniques to assess the relationship of constructs (concepts) in surveys;
- Clustering, a broad class of methods for discovering unknown subgroups in data.



Unsupervised Learning

The Challenge

Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.

Techniques for unsupervised learning are of growing importance in a number of fields:

- subgroups of breast cancer patients grouped by their gene expression measurements;
- groups of shoppers characterized by their browsing and purchase histories;
- movies grouped by the ratings assigned by movie viewers.

eg, Suominen et al. (2017) use unsupervised learning to create an overall view of patenting within the industry, and to forecast future trends.

(Suominen A., Toivanen, H., Seppänen, M. (2017). Firms' knowledge profiles: Mapping patent data with unsupervised learning, Technological Forecasting and Social Change)

Unsupervised Learning

In *unsupervised learning* we do not know the outcome groupings but attempt to discover them from structure in the data. For instance, we might explore a direct marketing campaign and ask:

“Are there groups that differ in how and when they respond to offers? If so, what are the characteristics of those groups?”

We use the term *clustering* for this approach.

Unsupervised Learning

Clustering

The cluster analysis is an important method belonging to the unsupervised learning techniques family;

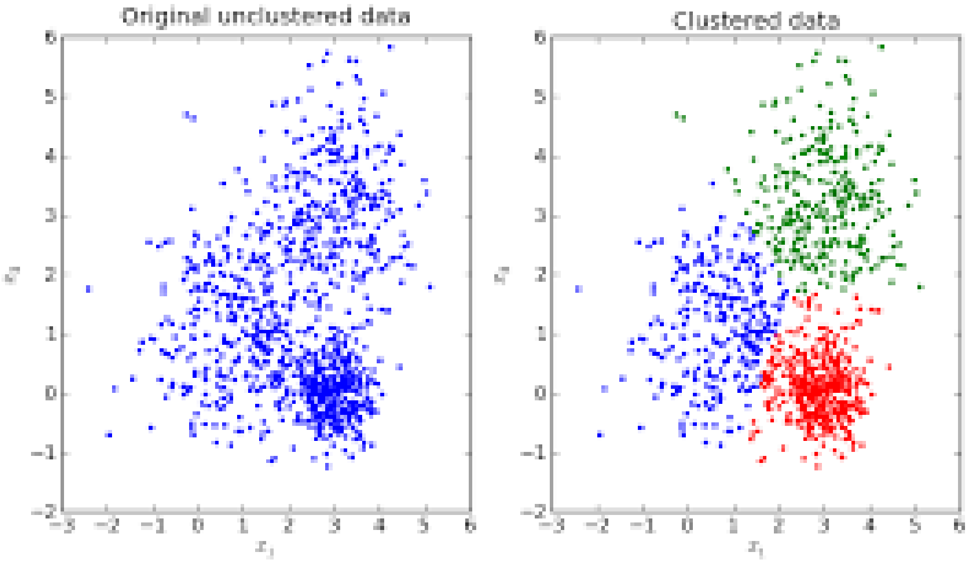
Clustering is one of the biggest topics in data science.

Clustering refers to a very broad set of techniques for finding homogeneous subgroups, or clusters, in a dataset.

When we cluster the observations of a dataset, we seek to partition them into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other.

Of course, to make this concrete, we must define what it means for two or more observations to be similar or different

The subtopic of text clustering is no exception.



Unsupervised Learning

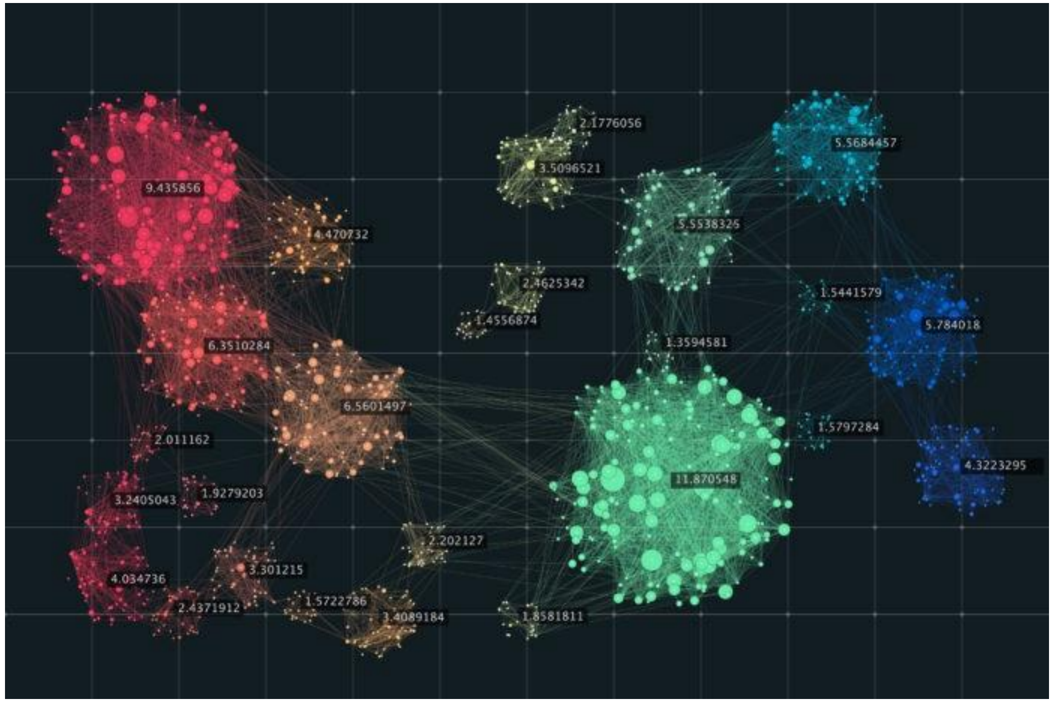
Goals of Clustering

The goal of cluster analysis is to ascertain, on the basis of the variables X_1, X_2 . . . , X_n, whether the observations fall into analysis relatively distinct groups.

For example, in a market segmentation study we might observe multiple characteristics (variables) for potential customers, such as zip code, family income, and shopping habits. We might believe that the customers fall into different groups, such as big spenders versus low spenders; or

To individuate the clusters of innovative firms.

To boost the innovation (eg, OECD, Boosting Innovation - The Cluster Approach)



Unsupervised Learning

Cluster Analysis

Many application of clustering arises in marketing.

We may have access to a large number of measurements (e.g. household income, occupation, distance from nearest urban area, and so forth) for a large number of people.

Our goal is to perform market segmentation by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product.

Performing market segmentation amounts to clustering the people in the dataset. Stakeholders often view segmentation as discovering groups in the data in order to derive new insight about customers.

If the information about each customer’s spending patterns is not available, we can try to cluster the customers on the basis of the variables measured, in order to identify distinct groups of potential customers.

Identifying such groups can be of interest because it might be that the groups differ with respect to some property of interest, such as spending habits.

Unsupervised Learning

Clusters analysis in practise

Data Selection

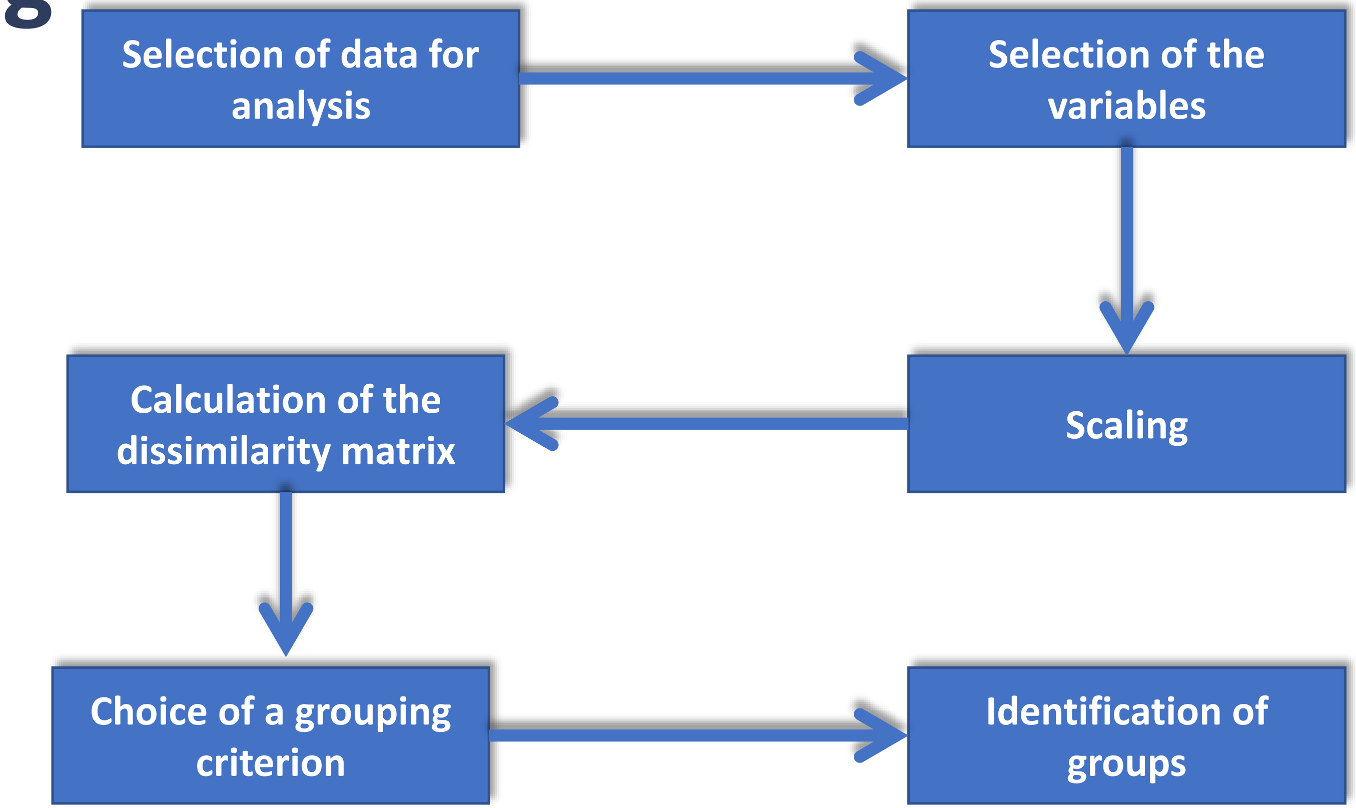
Variables Selection

Scaling Variables

Dissimilarity

Aggregation Criterion

Identification


Selection of data for analysis

Selection of the variables

Scaling

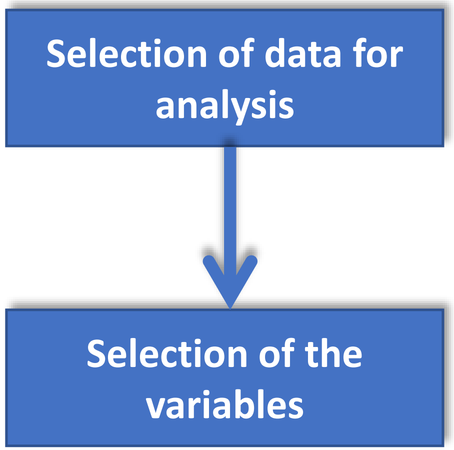
Calculation of the dissimilarity matrix

Choice of a grouping criterion

Identification of groups

Unsupervised Learning

Clusters analysis in practise


Selection of data for analysis

Selection of the variables

A topic we do not address is how to determine what data to use for clustering, the observed *basis variables* that go into the model. That is primarily a choice based on business need, strategy, and data availability.

Unsupervised Learning

Clusters analysis in practise

Why scale the data?

Rescaling the data

The value of distance measures is intimately related to the scale on which measurements are made. Therefore, variables are often scaled (i.e. standardized) before measuring the inter-observation dissimilarities. This is particularly recommended when variables are measured in different scales (e.g: kilograms, kilometers, centimeters, . . .); otherwise, the dissimilarity measures obtained will be severely affected.

The goal is to make the variables comparable. Generally variables are scaled to have i) standard deviation one and ii) mean zero.

We might also want to scale the data when the mean and/or the standard deviation of variables are largely different.

Unsupervised Learning

Clusters analysis in practise

Why scale the data?

Rescaling the data

Given a variable X_1 with mean μ_1 and standard deviation σ_1 its standardization is given by

$$Z_1 = \frac{X_1 - \mu_1}{\sigma_1}$$

The mean of Z_1 is null, that is equal to 0.

The variance (and the standard deviation) of Z_1 is equal to 1.

Unsupervised Learning

Clusters analysis in practise

Calculation of the dissimilarity matrix

We have an idea of the distance between pairs of observations, but how do we define the distance between two clusters or one observation and a cluster?

Actually, the idea of distance between a pair of observations needs to be extended to a pair of groups of observations.

This extension is achieved by developing the notion of linkage, which defines the distance between two groups of observations.

Unsupervised Learning

Clusters analysis in practise

Calculation of the dissimilarity matrix

The classification of observations into groups requires some methods for computing the **distance** or the (dis)**similarity** between each pair of observations. The result of this computation is known as a dissimilarity or **distance matrix.**

Properties of the distance

Generic distance d_{ij} between unit *i* and unit *j* has the following properties:

1. Non-negativity: $d_{ij} \geq 0$
2. Identity: $d_{ij} = 0$ if $i = j$
3. Simmetry: $d_{ij} = d_{ji}$
4. Triangular inequality: $d_{ij} \leq d_{is} + d_{sj}$

Unsupervised Learning

Calculation of the dissimilarity matrix

Clusters analysis in practise

There are many methods to compute the distance information. Here, we describe the common distance measures (providing R codes for computing and visualizing distances).

The best-known method is the Euclidean distance.

Unsupervised Learning

Calculation of the dissimilarity matrix

Distance matrix

Starting from *n* units whse distances have been computed, we can define the distance matrix of order *n x n* (*n* rows, *n* columns)

$$D = \begin{bmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & 0 & d_{23} & \dots & d_{2n} \\ d_{31} & d_{32} & 0 & \dots & \dots \\ \dots & \dots & \dots & 0 & \dots \\ d_{n1} & d_{n2} & \dots & \dots & 0 \end{bmatrix} \Longrightarrow D = \begin{bmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ d_{12} & 0 & d_{23} & \dots & d_{2n} \\ d_{13} & d_{23} & 0 & \dots & \dots \\ \dots & \dots & \dots & 0 & \dots \\ d_{1n} & d_{2n} & \dots & \dots & 0 \end{bmatrix}$$

Unsupervised Learning

Distance matrix –

How to compute distances?

Euclidean distance can be computed when the variables are quantitative.

Euclidean distance between unit *i* and unit *j* according to the variable X_1 is given by

$$d_{ij} = \sqrt{(X_{1i} - X_{1j})^2}$$

Euclidean distance between unit *i* and unit *j* according to the (standardized) variables X_1 and X_2 is given by

$$d_{ij} = \sqrt{(X_{1i} - X_{1j})^2 + (X_{2i} - X_{2j})^2}$$

Calculation of the dissimilarity matrix

$$d_{ij} = \sqrt{(X_{1i} - X_{1j})^2 + (X_{2i} - X_{2j})^2 + \cdots + (X_{pi} - X_{pj})^2}$$

$$d_{ij} = \sqrt{\sum_{h=1}^{p} (X_{hi} - X_{hj})^2}$$

Unsupervised Learning

Distance matrix –

How to compute distances?

Calculation of the dissimilarity matrix

Manhattan distance:

$$d_{man}(x, y) = \sum_{i=1}^{n} |(x_i - y_i)|$$

Other dissimilarity measures exist such as **correlation-based distances.**

The choice of distance measures is very important, as it has a strong influence on the clustering results. For most common clustering software, the default distance measure is the Euclidean distance

Unsupervised Learning

Distance matrix –

How to compute distances?

DATA: USArrests data (only a subset of the data will be used, by taking 15 random rows among the 50 rows in the data set. This is done by using the function *sample*()).

Next, we scale the data using the function *scale*():

```
# Subset of the data
set.seed(123)
ss <- sample(1:50, 15) # Take 15 random rows
df <- USArrests[ss, ] # Subset the 15 rows
df.scaled <- scale(df) # Standardize the variables
```

To compute Euclidean distance, you can use the R base *dist*() function, as follow:

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

Calculation of the dissimilarity matrix

```
round(as.matrix(dist.eucl)[1:3, 1:3], 1)
```

To make it easier to see the distance information generated by the *dist*() function, you can reformat the distance vector into a matrix using the *as.matrix*() function.

Unsupervised Learning

Distance matrix –

How to compute distances?

Calculation of the dissimilarity matrix

When we have *p* variables, some quantitative, some categorical, Gower distance is usually adopted.

Gower distance is given by

$$d_{ij} = \frac{\sum_{h=1}^{p} d_{ij,h}}{\sum_{h=1}^{p} \delta_{ij,h}}$$

$\delta_{ij,h}$ takes value 1 if the units i and j can be compared with respect to the *h*-th variable and 0 otherwise (if the phenomenon is absent in both units, e.g. no-no).

For quantitative variables

$$d_{ij,h} = \frac{|X_{hi} - X_{hj}|}{range(X_h)}$$

There are many R functions for computing distances between pairs of observations:
1. *dist*() R base function [*stats* package]: Accepts only numeric data as an input.
2. *get_dist*() function [*factoextra* package]: Accepts only numeric data as an input. Compared to the standard dist() function, it supports correlation-based distance measures including “pearson”, “kendall” and “spearman” methods.
3. *daisy()* function [*cluster* package]: Able to handle other variable types (e.g. nominal, ordinal, (a)symmetric binary). In that case, the Gower’s coefficient will be automatically used as the metric.

Unsupervised Learning

Distance matrix –

How to compute distances for mixed data?

```
library(cluster)
# Load data
data(flower)
head(flower, 3)
str(flower)
dd <- daisy(flower)
round(as.matrix(dd)[1:3, 1:3], 2)
```

Unsupervised Learning

Visualizing distance matrices

```
library(factoextra)
fviz_dist(dist.eucl)
```

Unsupervised Learning

Clusters analysis in practise

Data Selection

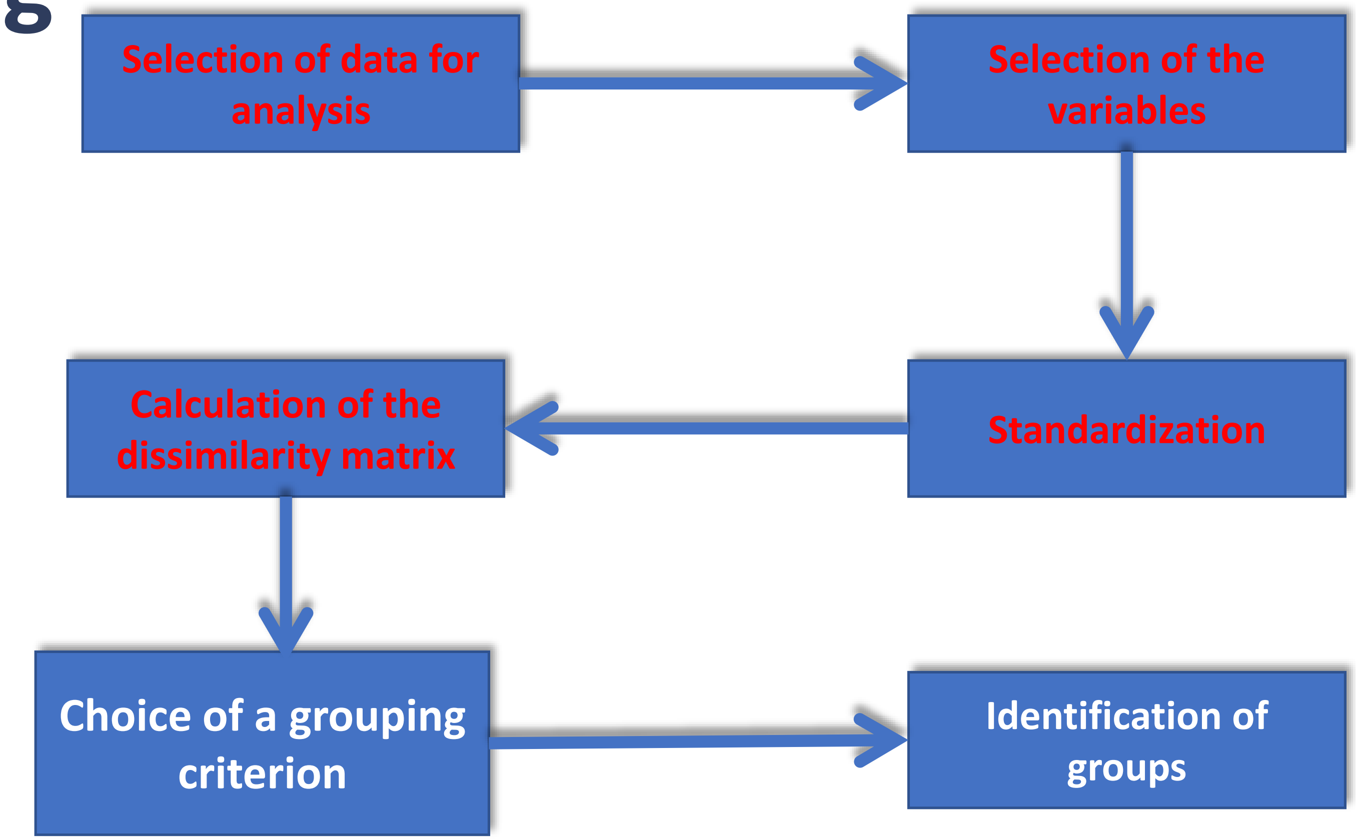
Variables Selection

Scaling Variables

Dissimilarity

Aggregation Criterion

Identification



Selection of data for analysis

Selection of the variables

Standardization

Calculation of the dissimilarity matrix

Choice of a grouping criterion

Identification of groups

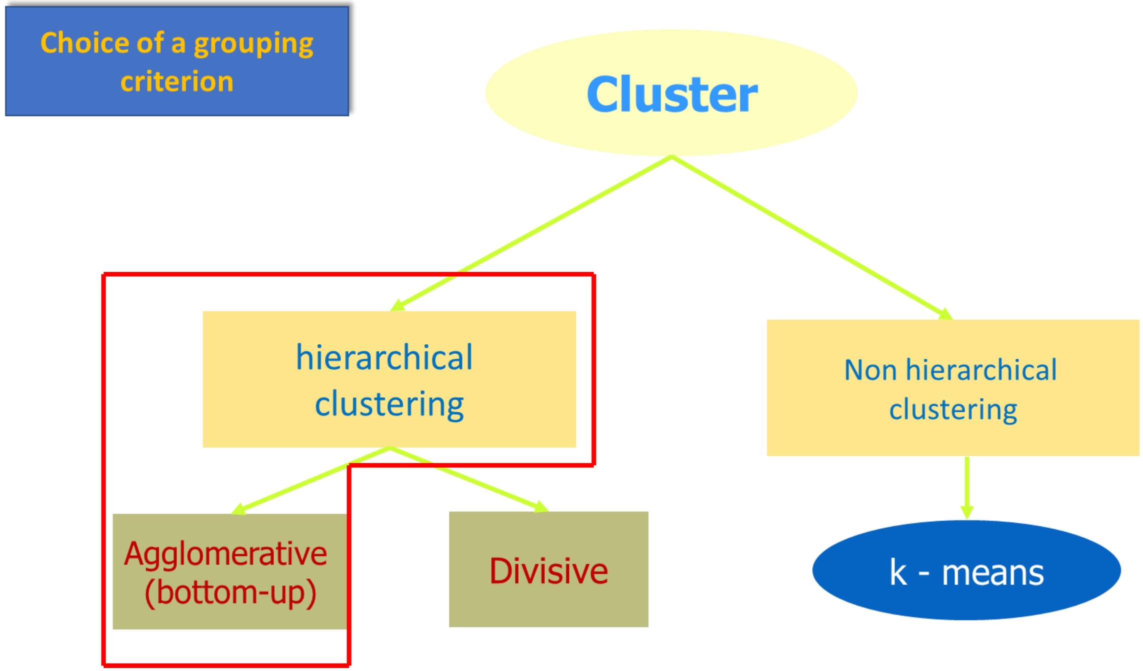
Unsupervised Learning

Choice of a grouping criterion

There exist a great number of clustering methods.

We focus on the two most popular clustering methods:

1. (agglomerative) hierarchical clustering
2. K-means clustering


Choice of a grouping criterion
Cluster
hierarchical clustering
Non hierarchical clustering
Agglomerative (bottom-up)
Divisive
k - means

Unsupervised Learning

Hierarchical clustering

Hierarchical clustering does not require to pre-specify the number of clusters *K*.

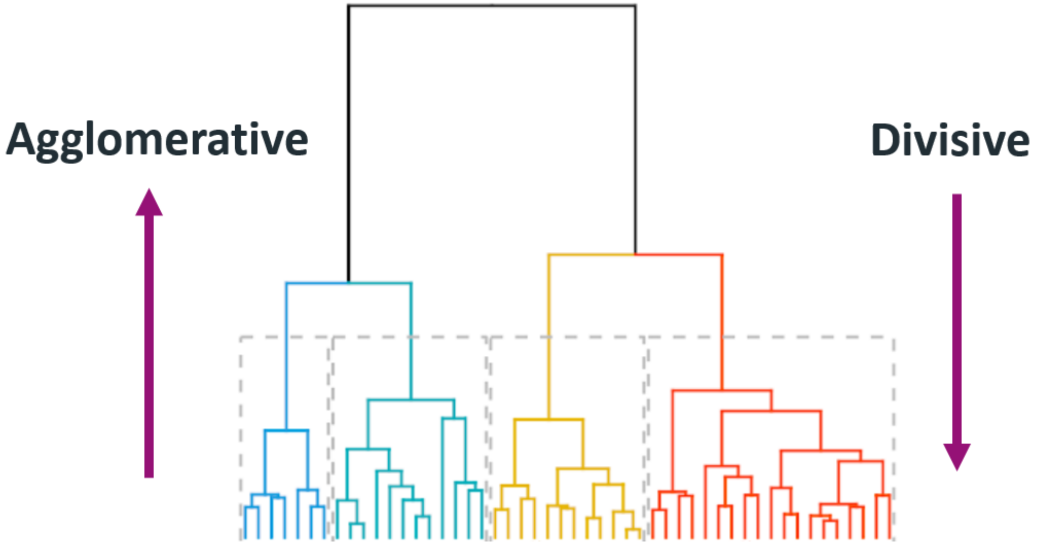
Hierarchical clustering results in an attractive tree-based representation of the observations, called dendrogram.

Observations can be subdivided into groups by cutting the dendrogram at a desired similarity level.

Let's describe the bottom-up or agglomerative clustering.

This is the most common type of hierarchical clustering.

Let's try to interpret a dendrogram and then show how hierarchical clustering is actually performed (how the dendrogram is built).



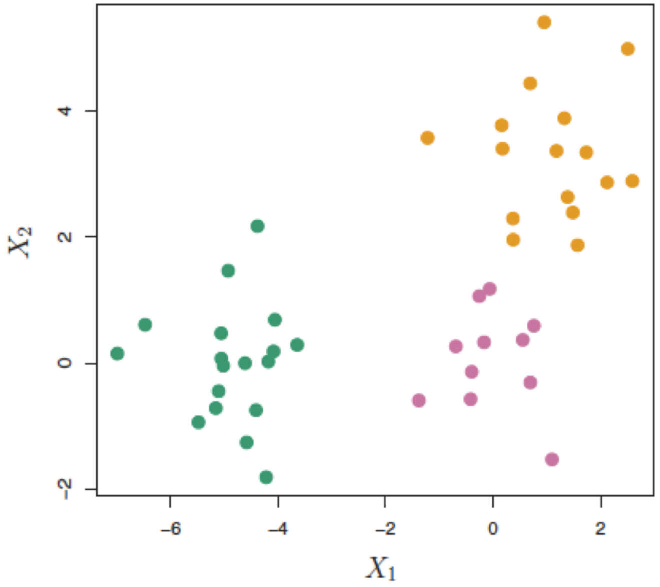
Unsupervised Learning

Hierarchical clustering

Let’s describe the bottom-up or agglomerative clustering.

This is the most common type of hierarchical clustering.

Let’s try to interpret a dendrogram and then show how hierarchical clustering is actually performed (how the dendrogram is built).



Unsupervised Learning

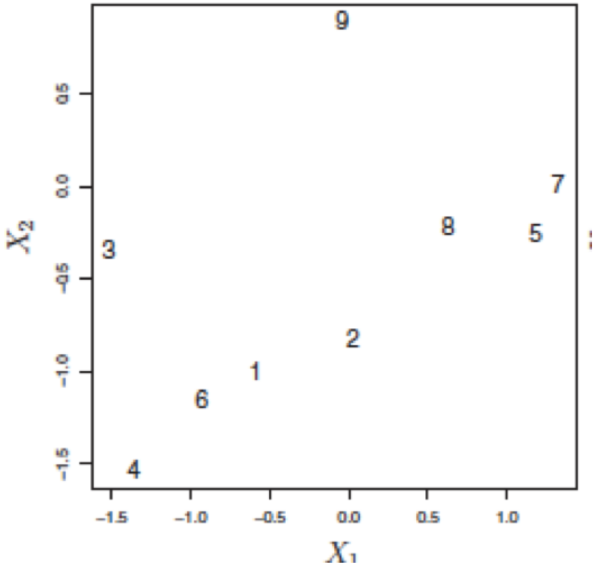
Hierarchical clustering

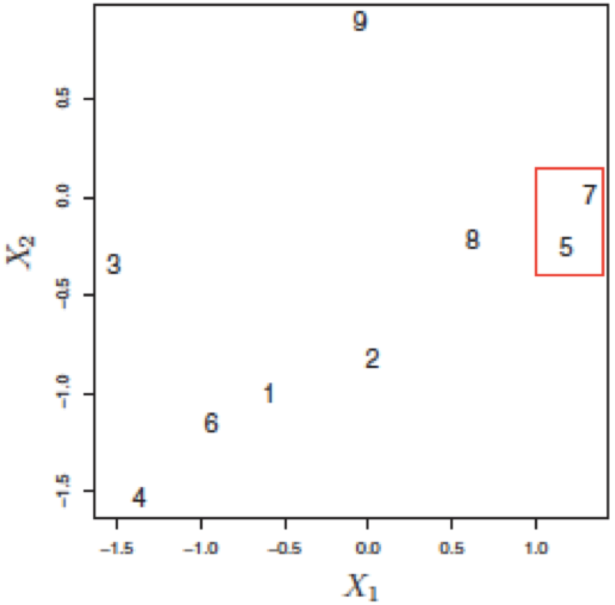
The hierarchical clustering dendrogram is obtained via an extremely simple algorithm.

At the beginning, each of the *n* observations are treated as its own cluster (*n* observations = *n* clusters).

Use distance between each pair of observations.

The two observations that are most similar to each other are then fused so that there now are *n* − 1 clusters.





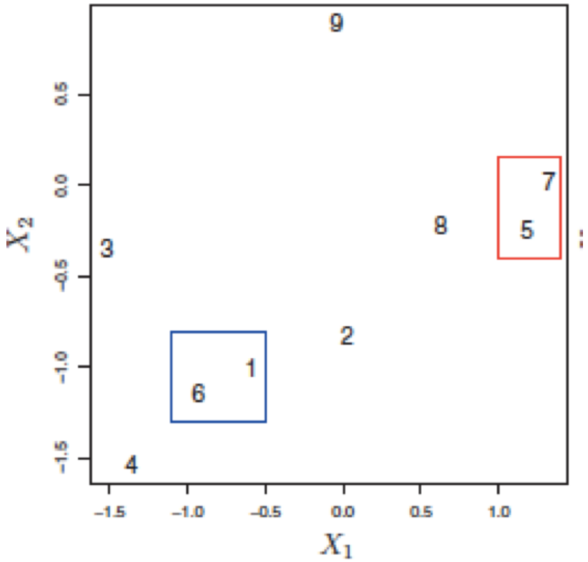
Unsupervised Learning

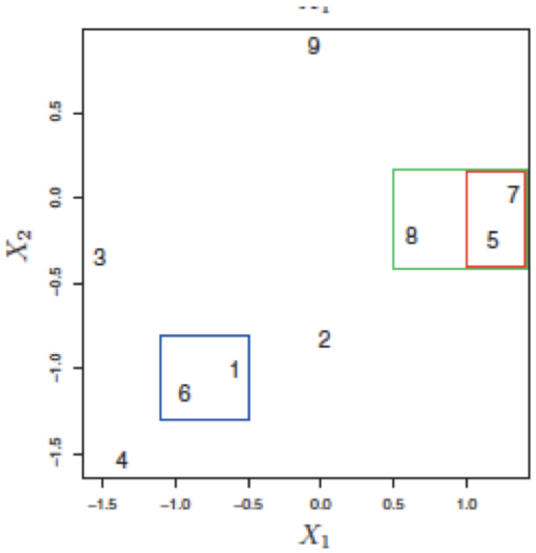
Hierarchical clustering

Next the two clusters that are most similar to each other are fused again, so that there now are $n - 2$ clusters.

And so on….

The algorithm proceeds in this fashion until all of the observations belong to one single cluster, and the dendrogram is complete.





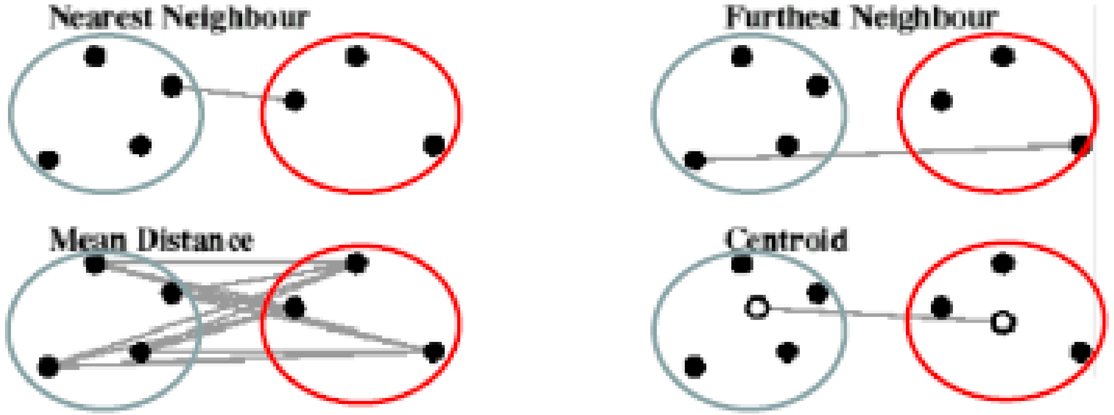
Unsupervised Learning

Hierarchical clustering

Linkage Functions

There are different possible linkages:

1. Single linkage (or nearest neighbour)
2. Complete linkage (or furthest neighbour)
3. Average linkage (mean distance)
4. Centroid linkage



Unsupervised Learning

Hierarchical clustering

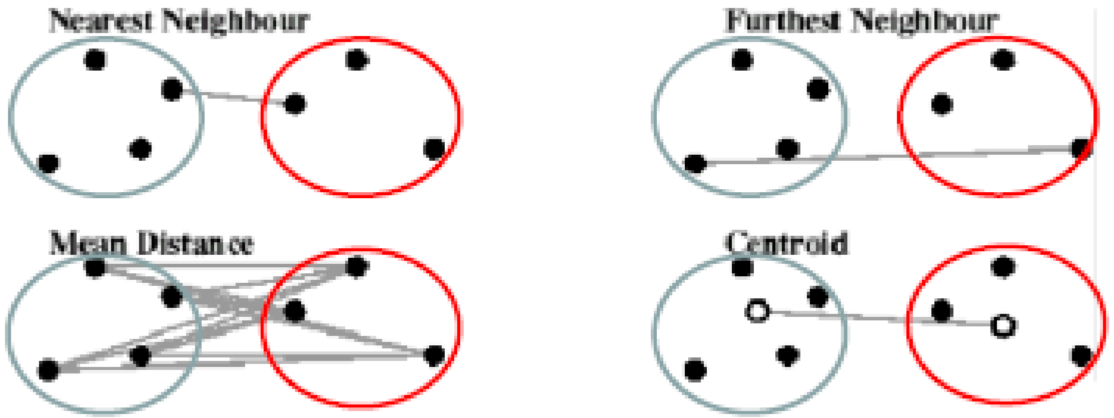
Linkage Functions

Maximum or *complete linkage*: The distance between two clusters is defined as the maximum value of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. It tends to produce more compact clusters.

Minimum or *single linkage*: The distance between two clusters is defined as the minimum value of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. It tends to produce long, "loose" clusters.

Mean or *average linkage*: The distance between two clusters is defined as the average distance between the elements in cluster 1 and the elements in cluster 2.

Centroid linkage: The distance between two clusters is defined as the distance between the centroid for cluster 1 (a mean vector of length p variables) and the centroid for cluster 2.



Unsupervised Learning

Agglomerative Hierarchical Clustering

Choice of a grouping criterion

DATA: USArrests data

```
set.seed(123)
ss <- sample(1:50, 15) # Take 15 random rows
df <- USArrests[ss, ] # Subset the 15 rows
df.scaled <- scale(df) # Standardize the variables
dist.eucl <- dist(df.scaled, method = "euclidean")
res.hc<- hclust(d=dist.eucl, method=“single”)
```

Unsupervised Learning

Dendrogram

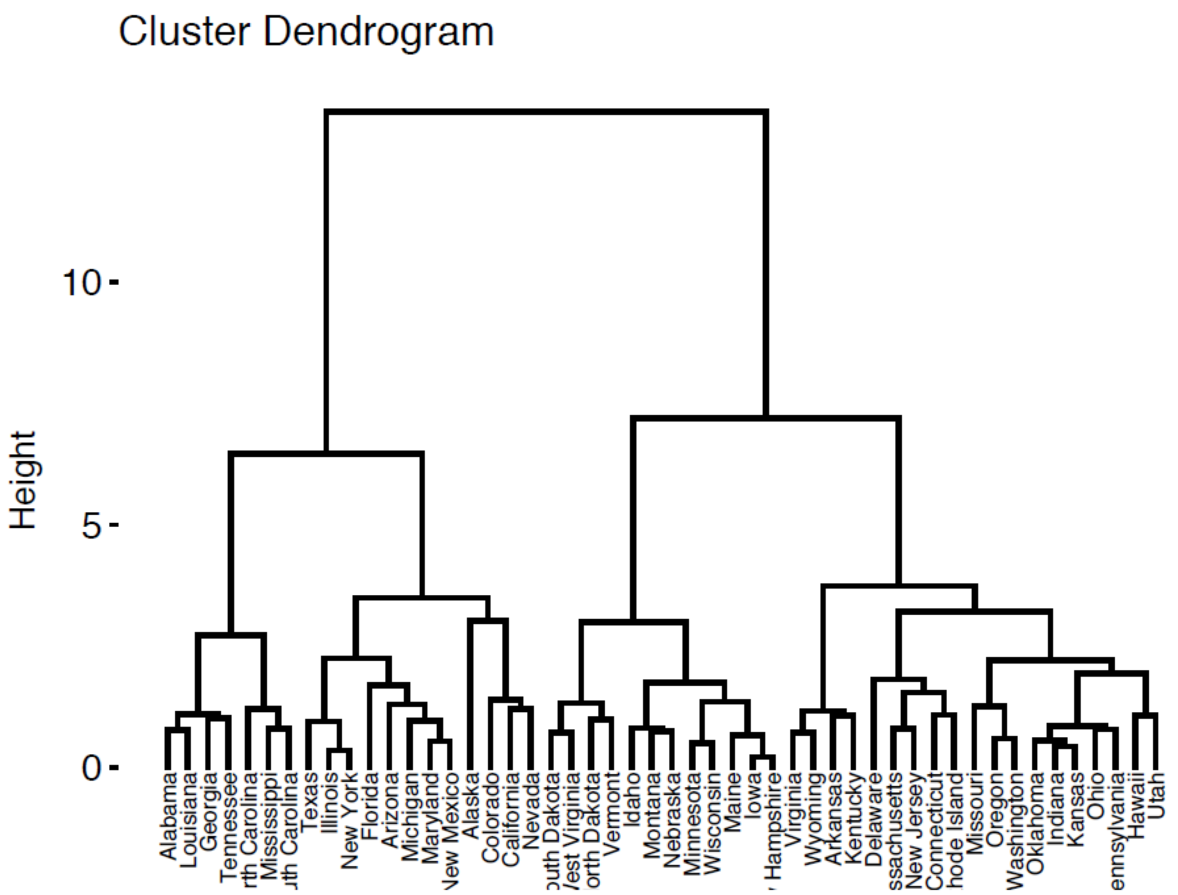
It corresponds to the graphical representation of the hierarchical tree generated by the function *hclust*()

In R:

# cex: label size

library("factoextra")

fviz_dend(res.hc, cex = 0.5)

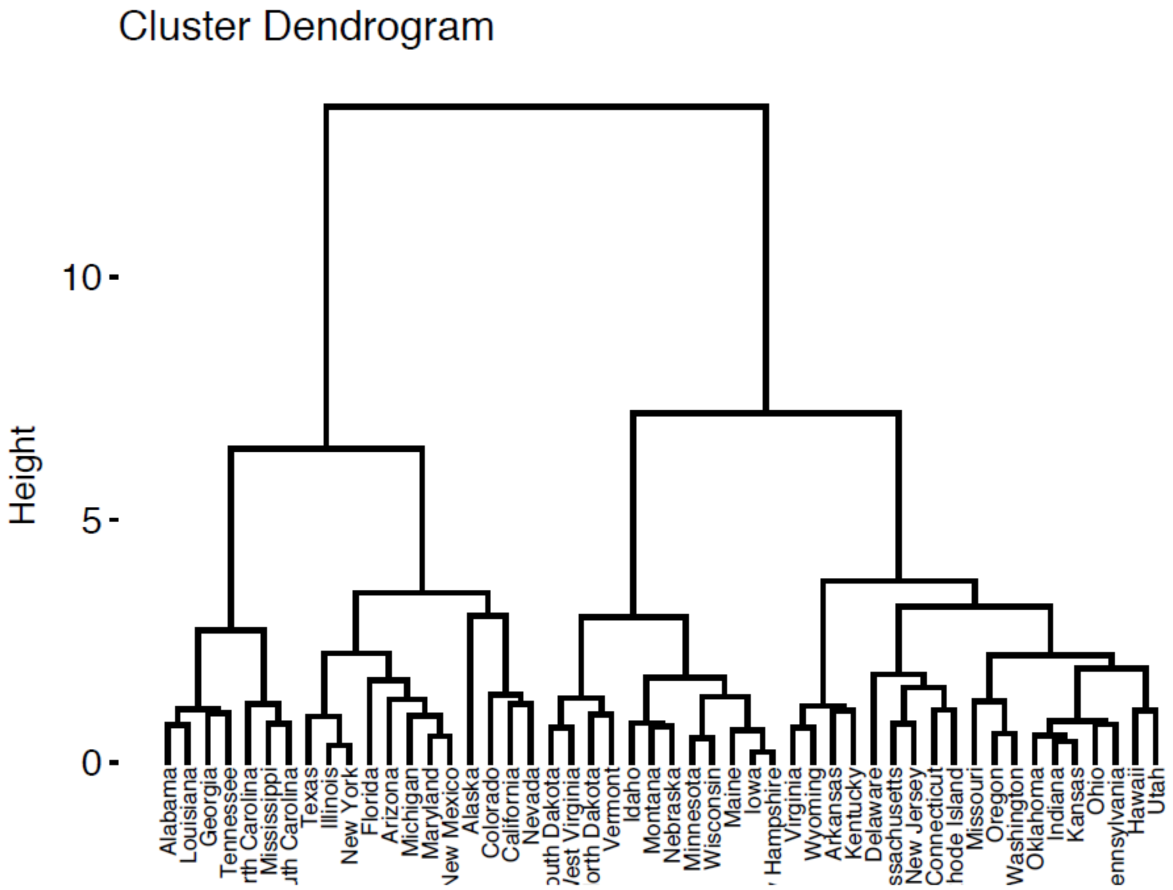


Unsupervised Learning

Dendrogram

Each leaf corresponds to one object. As we move up the tree, objects that are similar to each other are combined into branches, which are themselves fused at a higher height.

The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the *cophenetic distance* between the two objects



Unsupervised Learning

Clusters analysis in practise

Data Selection

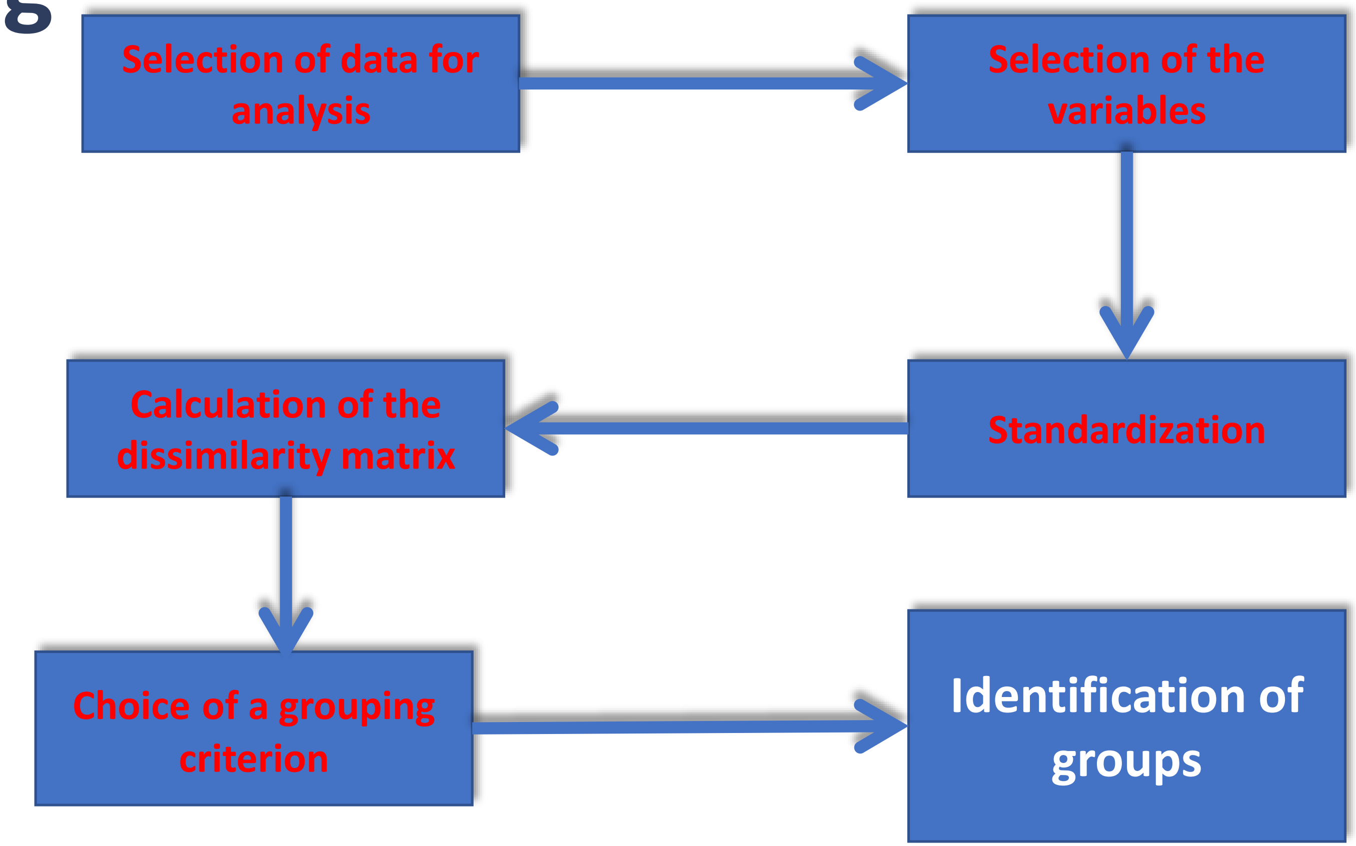
Variables Selection

Scaling Variables

Dissimilarity

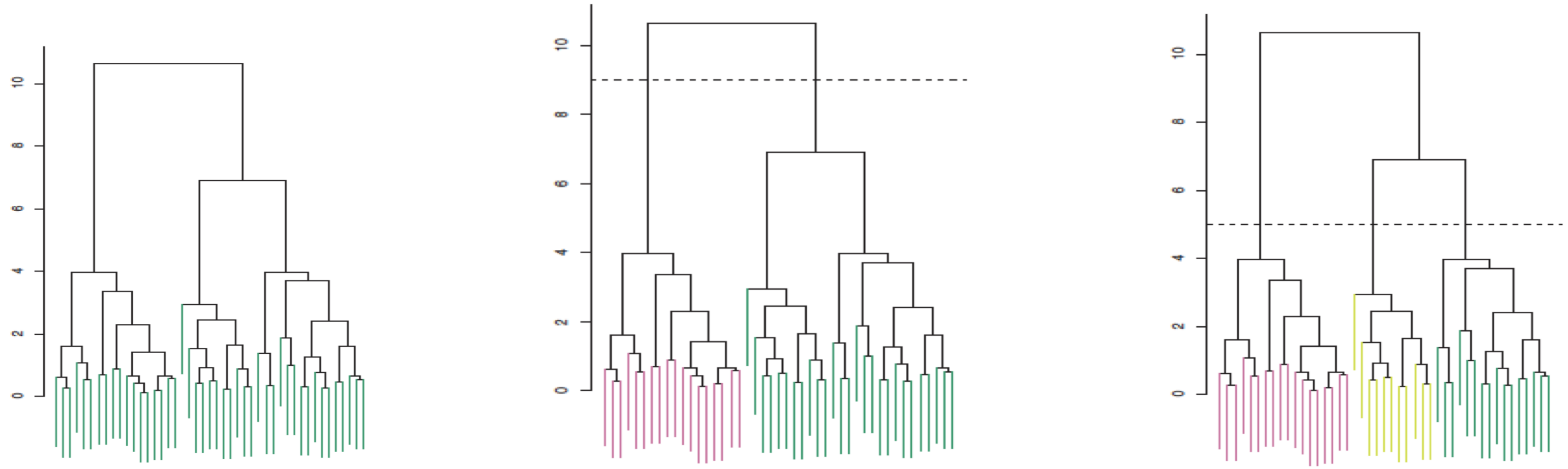
Aggregation Criterion

Identification


Selection of data for analysis
Selection of the variables
Standardization
Calculation of the dissimilarity matrix
Choice of a grouping criterion
Identification of groups

Unsupervised Learning

Cut the dendrogram

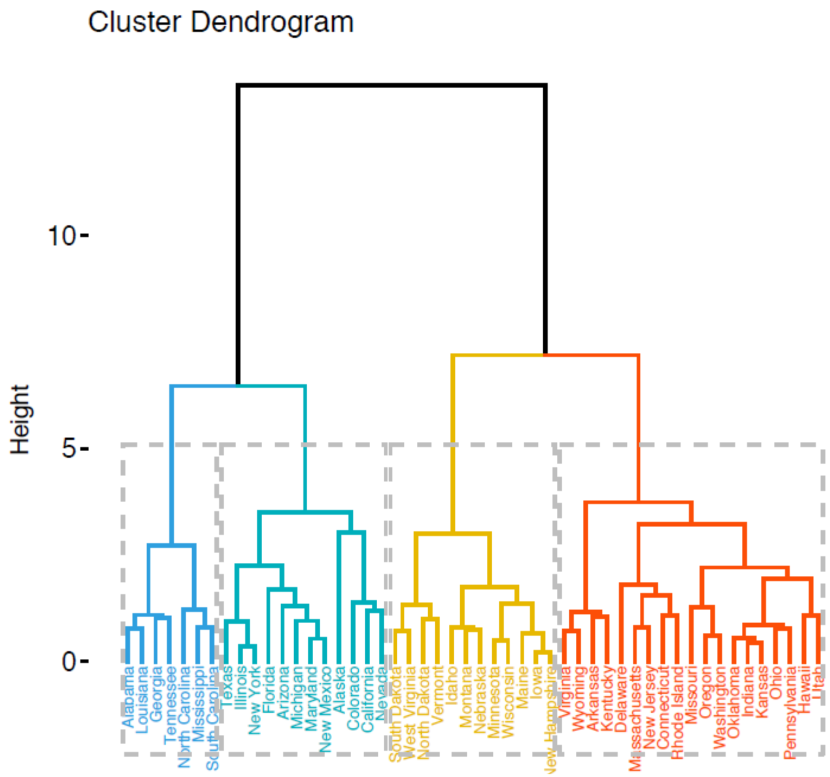


Unsupervised Learning

Cut the dendrogram

We can cut the hierarchical tree at a given height in order to partition data into clusters.

```
# Cut in 4 groups and color by groups
fviz_dend(res.hc, k = 4, # Cut in four groups
cex = 0.5, # label size
k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
color_labels_by_k = TRUE, # color labels by groups
rect = TRUE # Add rectangle around groups
)
```

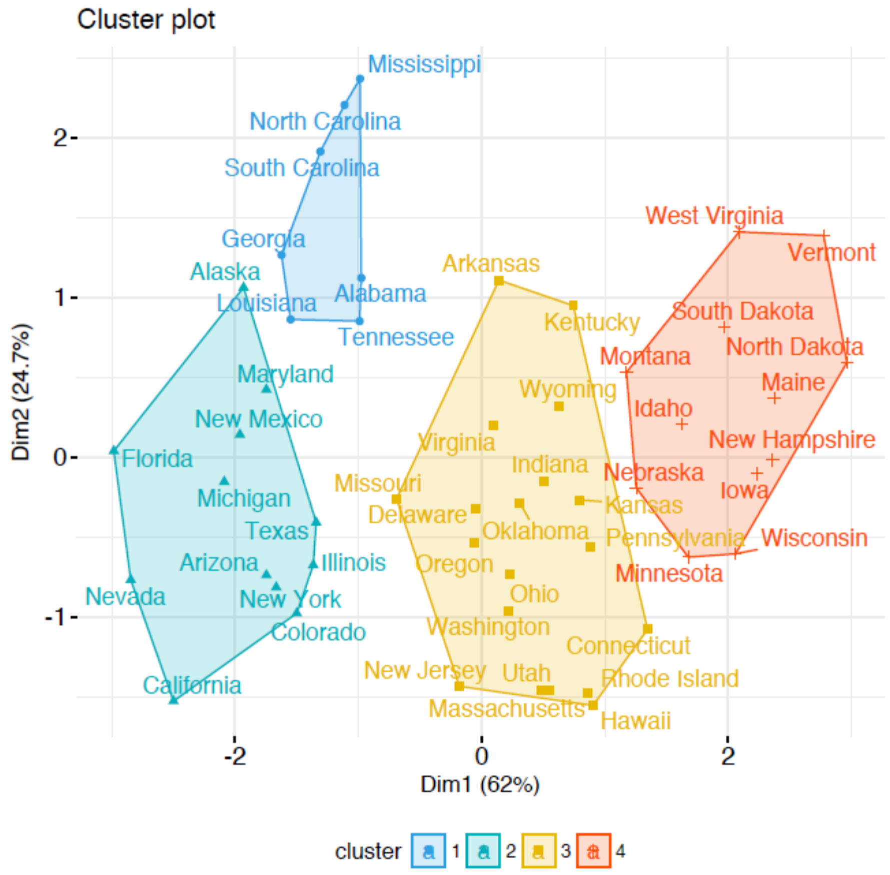


Unsupervised Learning

Cut the dendrogram

Using the function *fviz_cluster*() [in *factoextra*], we can also visualize the result in a scatter plot. Observations are represented by points in the plot, using principal components. A frame is drawn around each cluster.

```
# Cut tree into 4 groups
grp <- cutree(res.hc, k = 4)
head(grp, n = 4)
# Get the names for the members of cluster 1
rownames(df)[grp == 1]
fviz_cluster(list(data = df, cluster = grp),
palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
ellipse.type = "convex", # Concentration ellipse
repel = TRUE, # Avoid label overplotting (slow)
show.clust.cent = FALSE, ggtheme = theme_minimal())
```



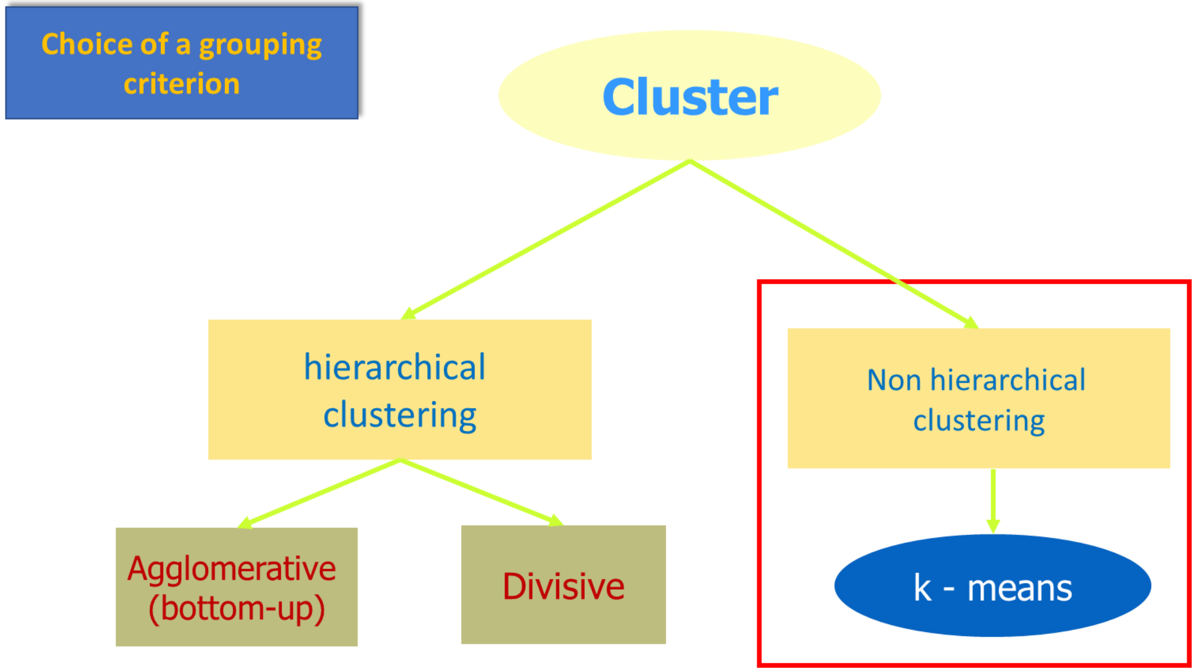
Unsupervised Learning

Choice of a grouping criterion

There exist a great number of clustering methods

We focus on the two most popular clustering methods:

1. (agglomerative) hierarchical clustering
2. **K-means clustering**


Choice of a grouping criterion
Cluster
hierarchical clustering
Non hierarchical clustering
Agglomerative (bottom-up)
Divisive
k - means

Unsupervised Learning

Non-Hierarchical clustering

k-means

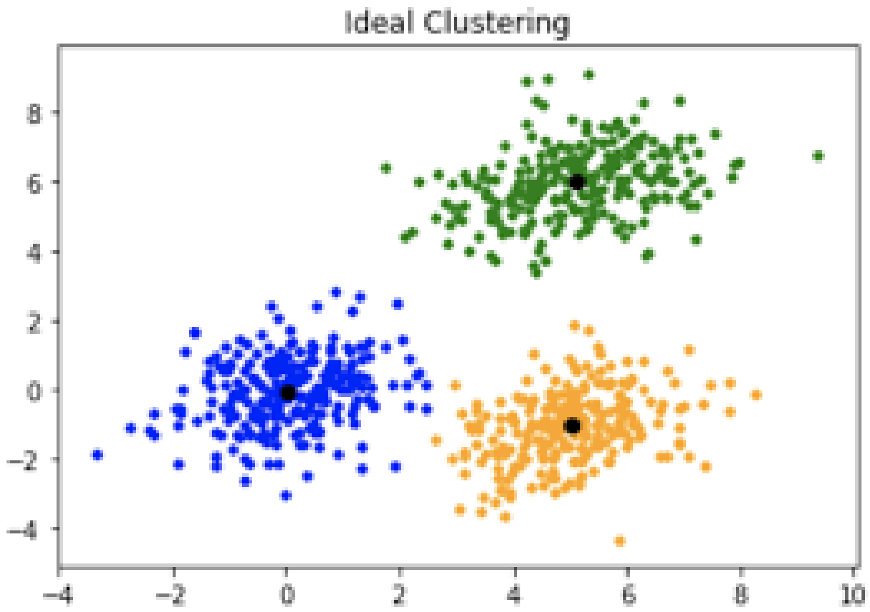
We will now look at the most famous vector-based clustering algorithm out there: k-means.

What k-means does is returning a cluster assignment to one of k possible clusters for each object.

K-means clustering is a simple approach for partitioning a data set into K distinct, non-overlapping clusters.

To perform K-means clustering, we must first specify the desired number of clusters K; then the K-means algorithm will assign each observation to exactly one of the K clusters.

If the *i*-th observation is in the *k*-th cluster, then $i \in C_k$.



Unsupervised Learning

Non-Hierarchical clustering

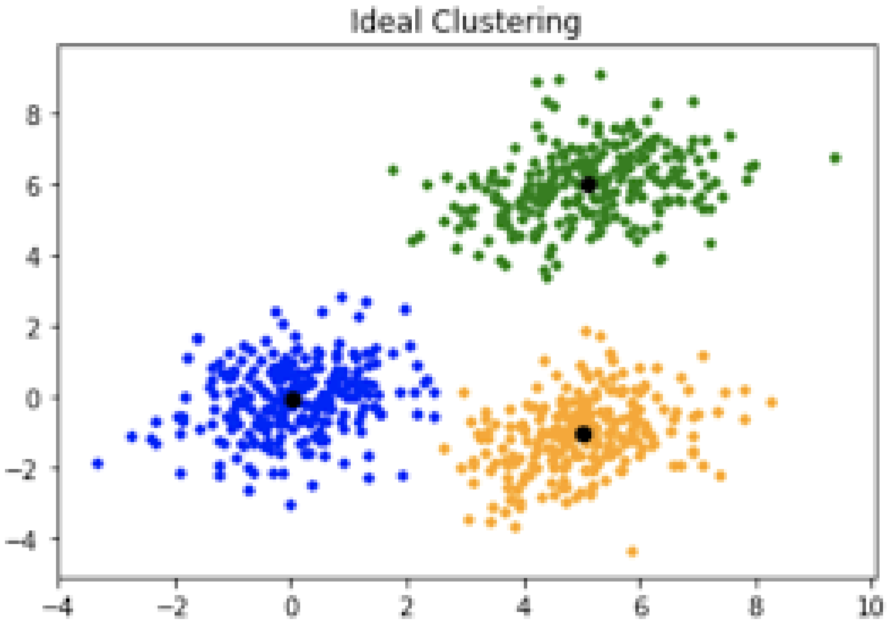
k-means

The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible.

The within-cluster variation for cluster C_k is a measure of the amount by which the observations within a cluster differ from each other.

The total within-cluster variation, summed over all K clusters, has to be as small as possible.

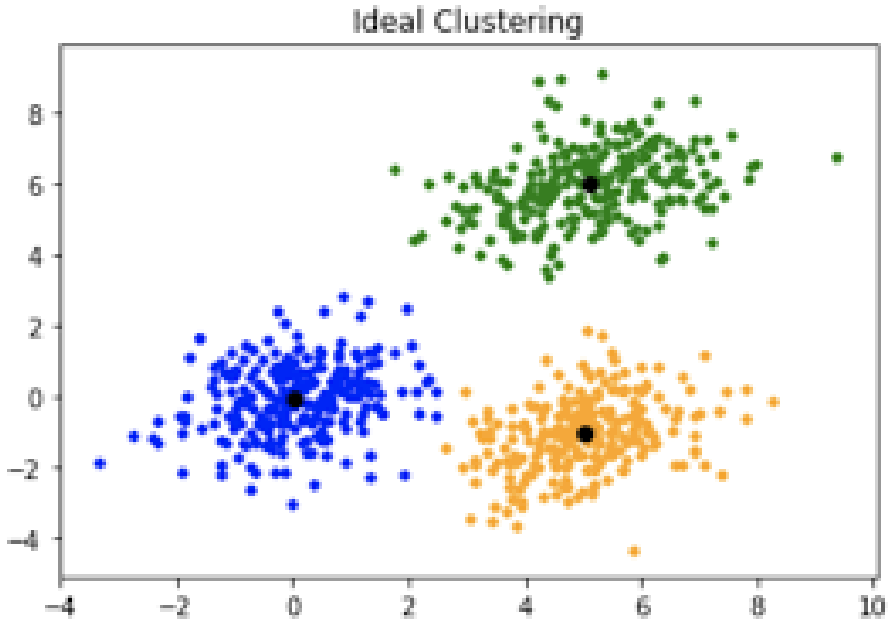
So, we need to define the within-cluster variation. There are many possible ways to define this concept, but the most common choice involves squared Euclidean distance.



Unsupervised Learning

Non-Hierarchical clustering

k-means



The within-cluster variation for the *k*-th cluster is the sum of all of the pairwise squared Euclidean distances between the observations in the cluster, divided by the total number of observations in the cluster.

Objective: to minimize the total within-cluster variation.

The use of Euclidean distances implies that:

1. standardization has to be carried out in presence of differente scale of measurements;

2. categorical variables are not allowed

Unsupervised Learning

Non-Hierarchical clustering

k-means

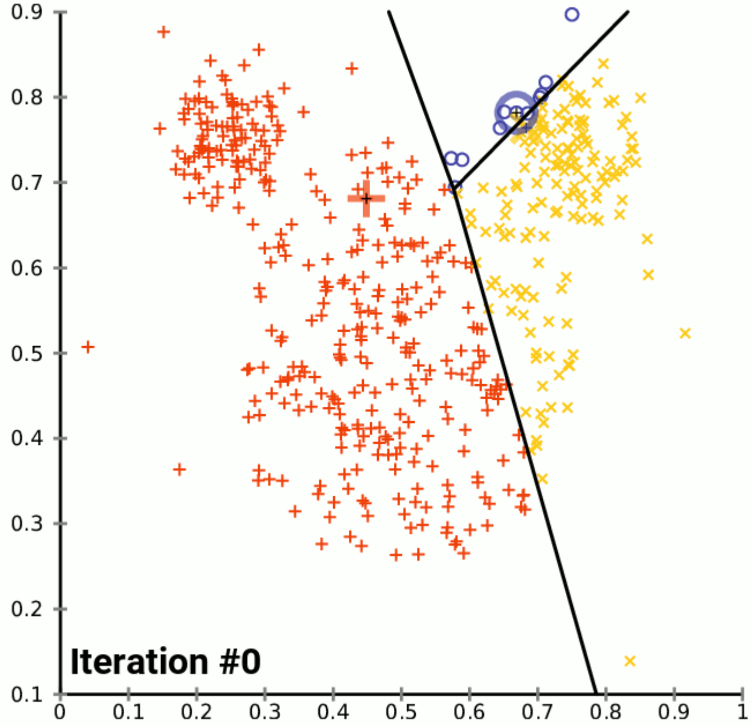
The algorithm starts randomly assigning a cluster, from 1 to K, to each of the observations.

For each of the K clusters, the cluster centroid is computed.

Then, each observation is assigned to the cluster whose centroid is closest.

This means that as the algorithm is run, the clustering obtained will continually improve

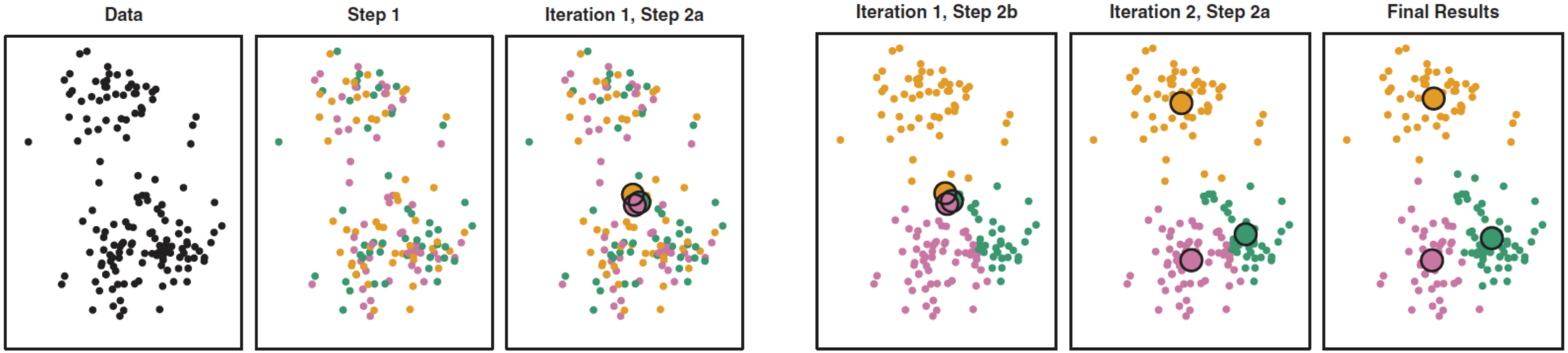
When the result no longer changes, we have reached the final clustering.



Unsupervised Learning

Non-Hierarchical clustering

k-means



Unsupervised Learning

Non-Hierarchical clustering

k-means in R

```
data("USArrests") # Loading the data set
df <- scale(USArrests) # Scaling the data
# View the firt 3 rows of the data
head(df, n = 3)
```

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

```
kmeans(x, centers, iter.max = 10, nstart = 1)
```

```
library(animation)

set.seed(534)

kmeans.ani(df)
```

Unsupervised Learning

Non-Hierarchical clustering

k-means in R

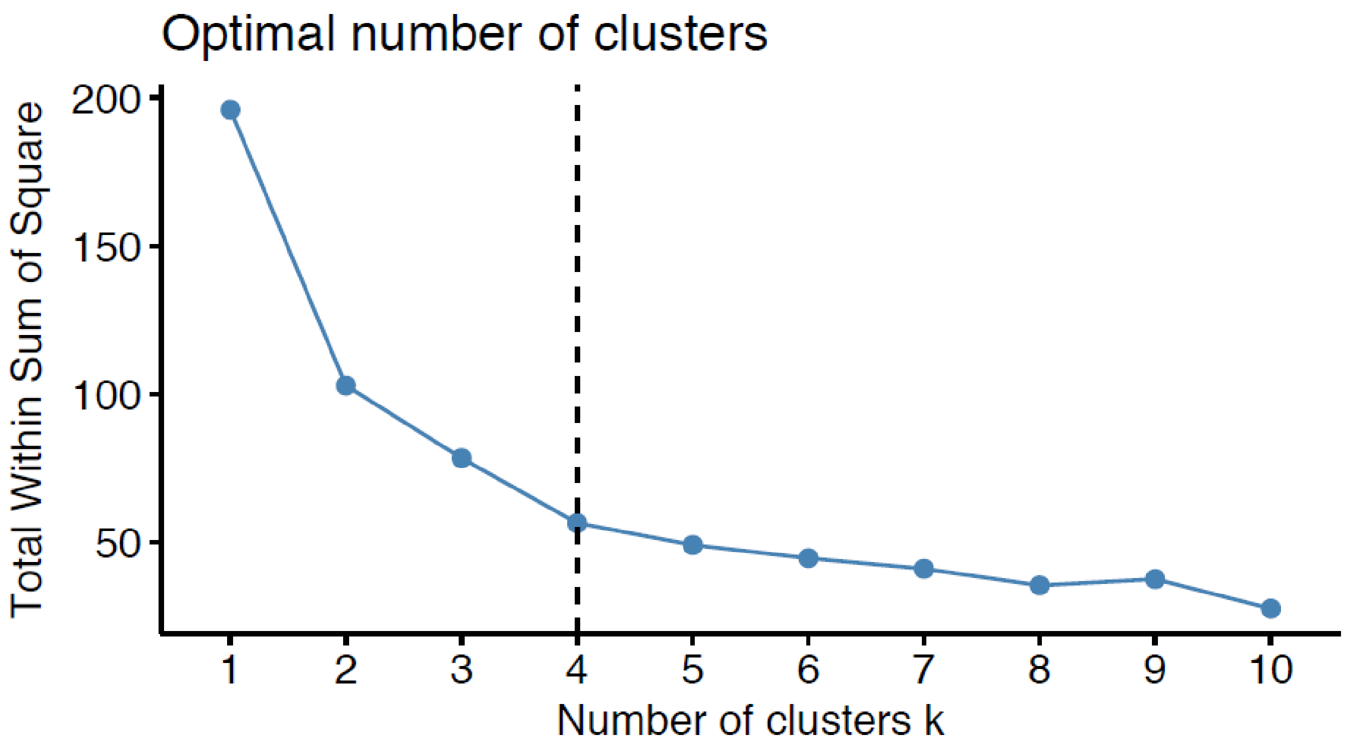
The k-means clustering requires the users to specify the number of clusters to be generated.
One fundamental question is:
How to choose the right number of expected clusters (k)?
Different methods exist.
Here, we provide a simple solution. The idea is to compute k-means clustering using different values of clusters k. Next, the wss (within sum of square) is drawn according to the number of clusters. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

Unsupervised Learning

Non-Hierarchical clustering

k-means in R

```
library(factoextra)
fviz_nbclust(df, kmeans, method = "wss") +
geom_vline(xintercept = 4, linetype = 2)
```



Unsupervised Learning

Non-Hierarchical clustering

k-means in R

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*.

```
# Compute k-means with k = 4
set.seed(123)
km.res <- kmeans(df, 4, nstart = 25)
# Print the results
print(km.res)
```

Unsupervised Learning

Non-Hierarchical clustering

k-means advantages and disanvantages

1. It assumes prior knowledge of the data and requires the analyst to choose the appropriate number of cluster (k) in advance.
2. The final results obtained is sensitive to the initial random selection of cluster centers. Why is this a problem? Because, for every different run of the algorithm on the same data set, you may choose different set of initial centers. This may lead to different clustering results on different runs of the algorithm.
3. It’s sensitive to outliers.
4. If you rearrange your data, it’s very possible that you’ll get a different solution every time you change the ordering of your data.

Unsupervised Learning

Non-Hierarchical clustering

k-means advantages and disanvantages

1. Solution to issue 1: Compute k-means for a range of k values, for example by varying k between 2 and 10. Then, choose the best k by comparing the clustering results obtained for the different k values.
2. Solution to issue 2: Compute K-means algorithm several times with different initial cluster centers. The run with the lowest total within-cluster sum of square is selected as the final clustering solution.
3. To avoid distortions caused by excessive outliers, it’s possible to use PAM algorithm, which is less sensitive to outliers.

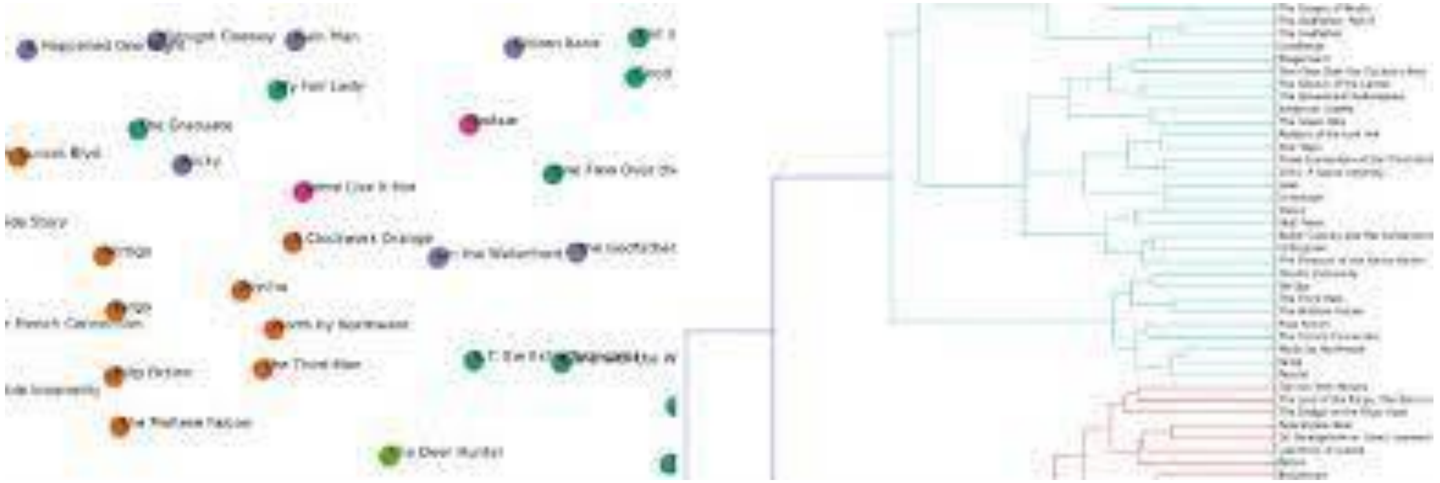
Text clustering

Word clustering

Text clustering is the process of grouping a set of unlabeled texts so that those in the same cluster are more similar than those in other groups.

Text clustering algorithms examine the text to see if there are any natural clusters (groups) in the data.

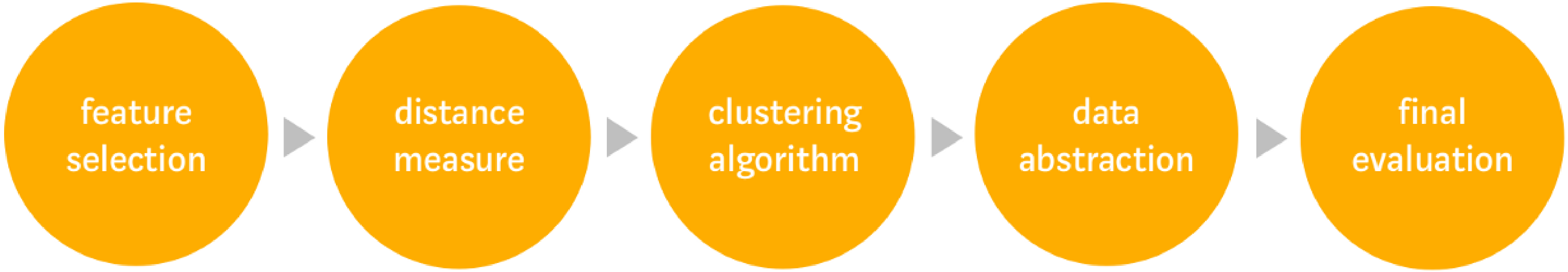
Process of clustering text is often messy and marked by many unsuccessful trials. However, if you tried to draw it in an idealized, linear manner, it might look like this:



.

Text clustering

Word clustering



feature selection

Which features will we look at?

distance measure

What does it mean that features are similar?

clustering algorithm

Which method and which result do we want?

data abstraction

Which meanings do the obtained clusters have?

final evaluation

How useful are the obtained results?

Text clustering

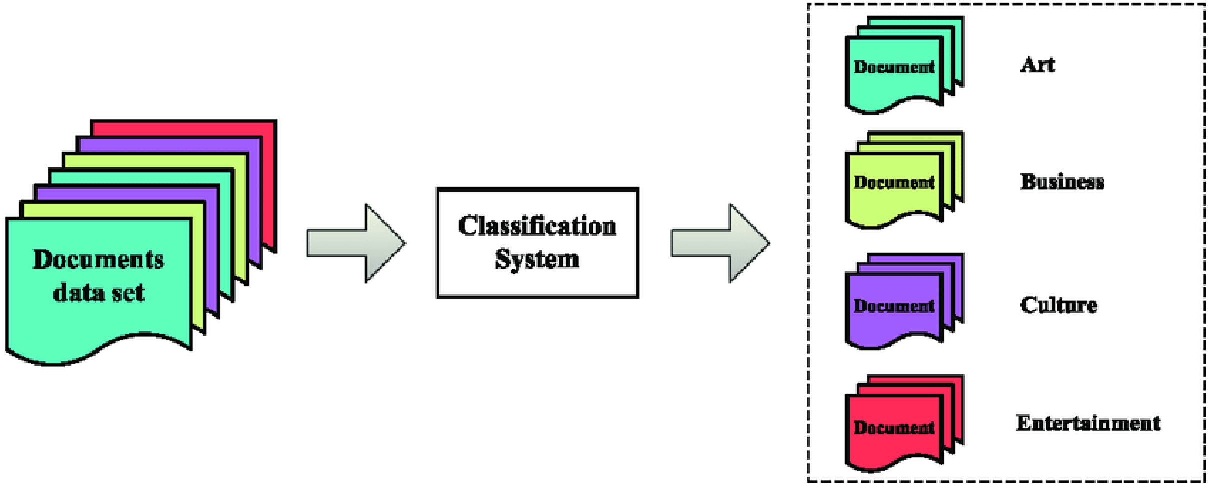
Word clustering

In the next few paragraphs, we will look at clustering methods for words. Let’s look at the following set of them:

$$\{Aardvark, on, under, Zebra\}$$

To us it immediately becomes apparent which words belong together. There should obviously be one cluster with animals containing the words Aardvark and Zebra and one with adverbs containing on and under. But is it equally obvious for a computer?

.



Text clustering

Word clustering

When talking about words with similar meaning, you often read about the distributional hypothesis in linguistics. This hypothesis states that words bearing a similar meaning will appear between similar word contexts. You could say “The box is on the shelf.”, but also “The box is under the shelf.” and still produce a meaningful sentence. On and under are interchangeable up to a certain extent.

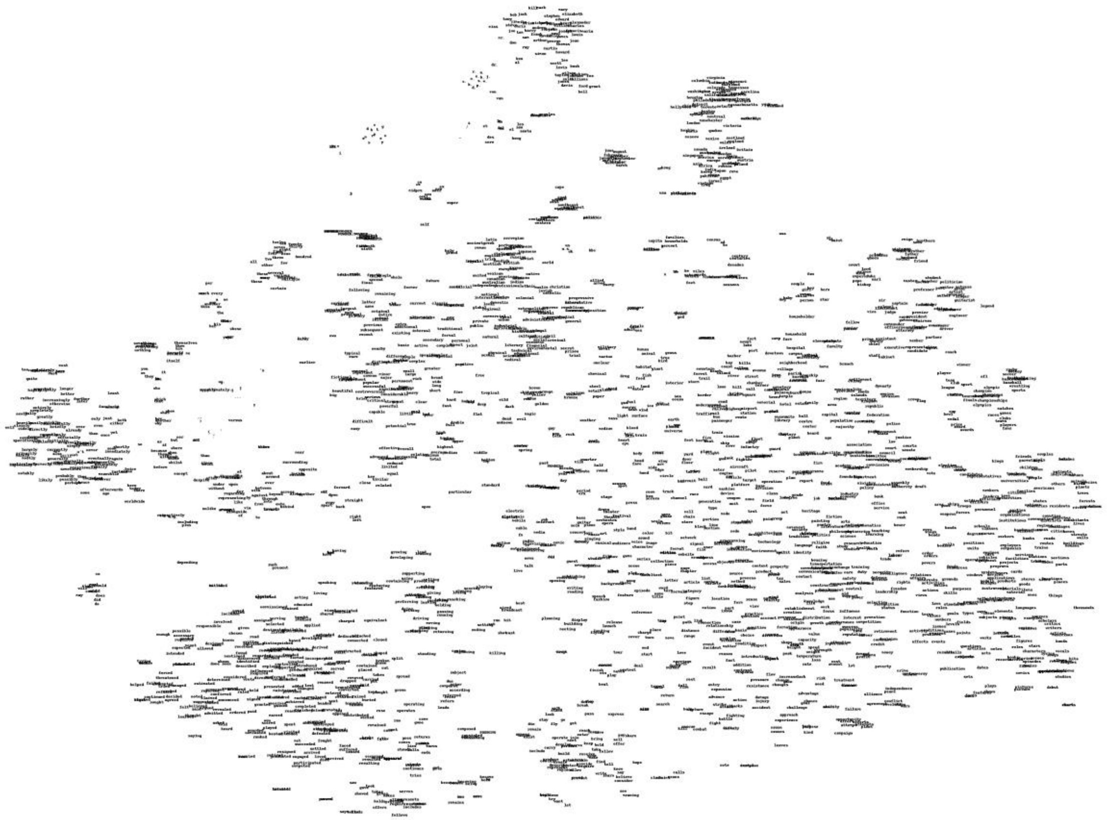
This hypothesis is utilized when creating word embeddings. Word embeddings map each word of a vocabulary onto a n-dimensional vector space. Words that have similar contexts will appear roughly in the same area of the vector space. One of these embeddings was developed by Weston, Ratle & Collobert in 2008. You can see an interesting segment of the word vectors (reduced to two dimensions with t-SNE) here:

Text clustering

Word clustering

Notice how neatly months, names and locations are grouped together. This will come in handy for clustering them in the next step. To learn more about how exactly word embeddings are created and the interesting properties they have, take a look at this Medium article by Hunter Heidenreich

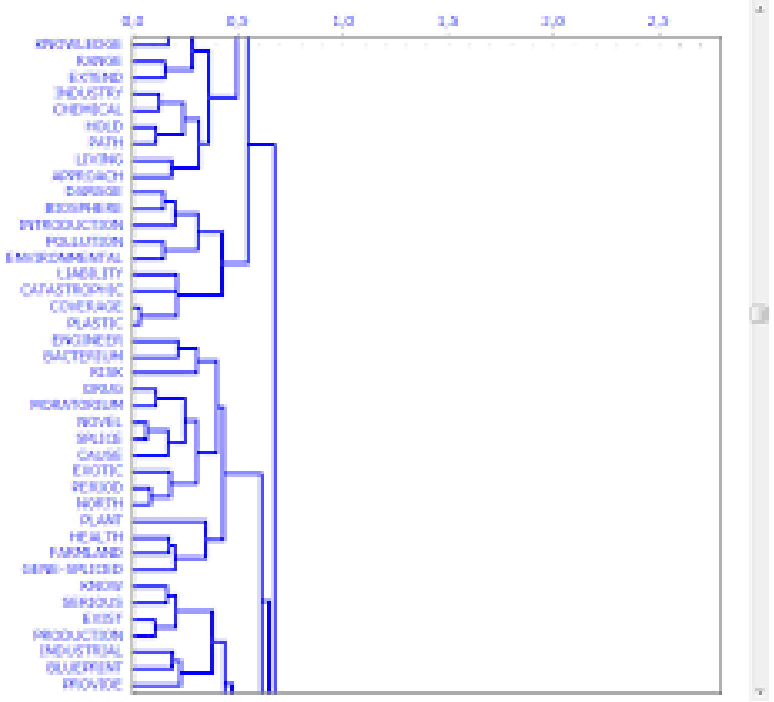
. It also includes information about more advanced word embeddings like word2vec.



Text clustering

Word clustering is used to identify groups of words used together, based on frequency distance. It is a size reduction technique. It helps to group words into related clusters.

The word clusters are visualized with dendrograms.



Statistical Learning

Unsupervised Learning

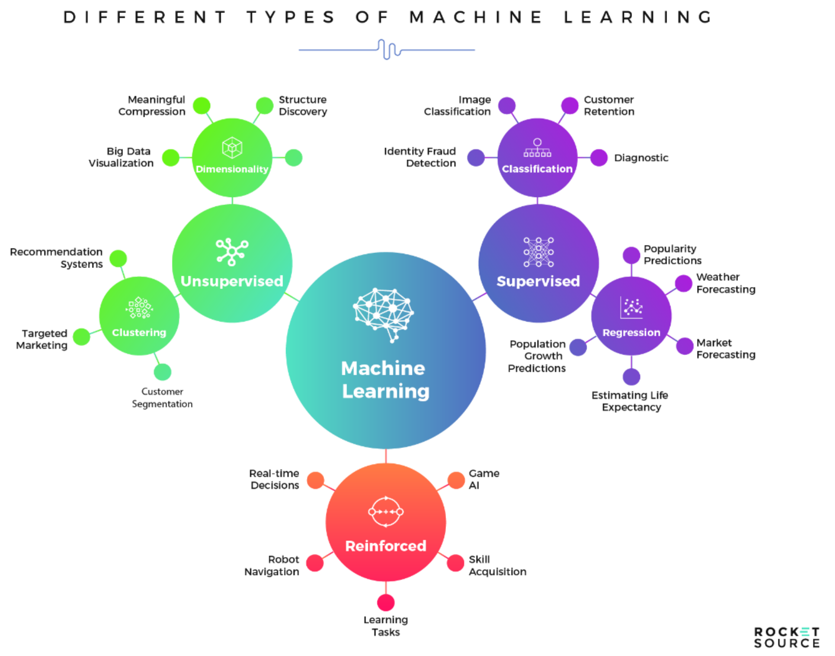
Dimensionality reduction (e.g., principal component analysis)

Cluster analysis

PCA vs Clustering

Clustering looks for homogeneous subgroups among the observations.

PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance.



Unsupervised Learning

Principal Component Analysis

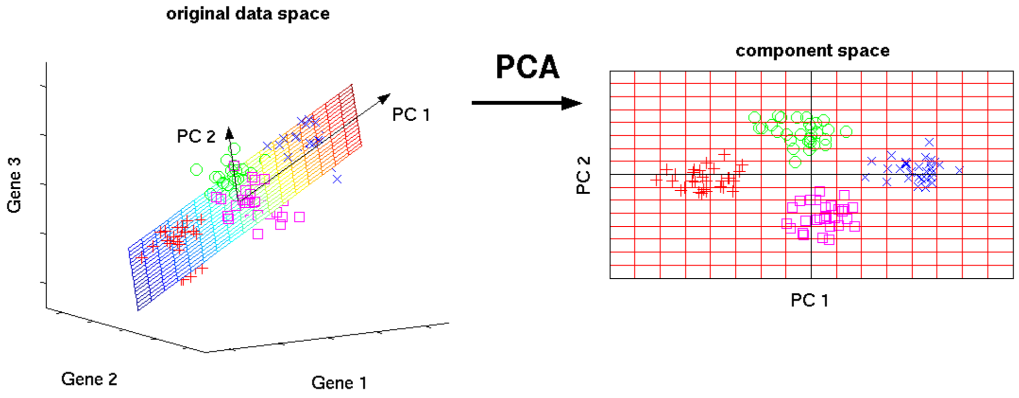
One of the most important objectives of the principal components analysis (PCA) is the reduction of the dimensionality of the data.

In practice, when p, the number of variables, is very high, we can opt for a reduction in the size of the dataset.

The idea is to transform the p original variables into k new variables ($k = p$).

PCA allows one to summarize this set with a smaller number of representative variables that collectively explain most of the variability in the original set

PCA is an unsupervised approach, since it involves only a set of features X_1, X_2 , . . . , X_p, and no associated response Y.



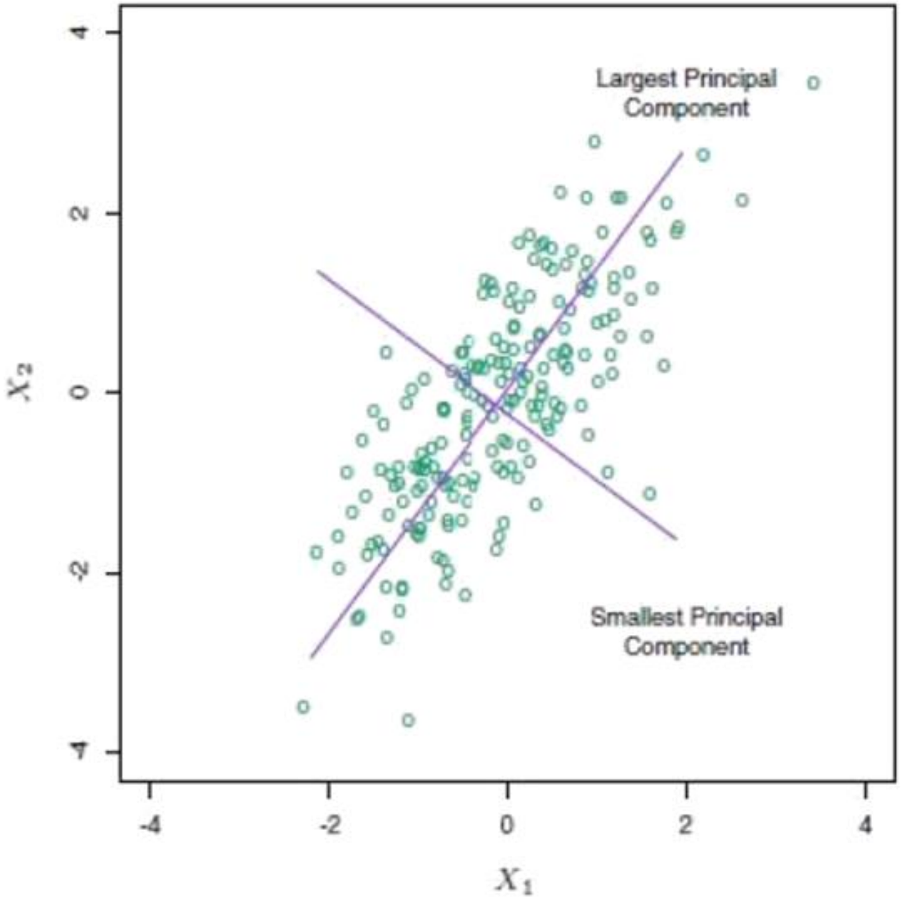
Unsupervised Learning

Principal Component Analysis

The new *k* variables are obtained as linear combinations of the original variables and are called principal components.

The linear combinations of the original variables are uncorrelated.

The *k* principal components are ordered according to the variability explained (the first explains the maximum variability, etc.)



Unsupervised Learning

Principal Component Analysis. Why?

Suppose that we wish to visualize *n* observations with measurements on a set of *p* variables, $X_1, X_2, \ldots, X_p$, as part of an exploratory data analysis.

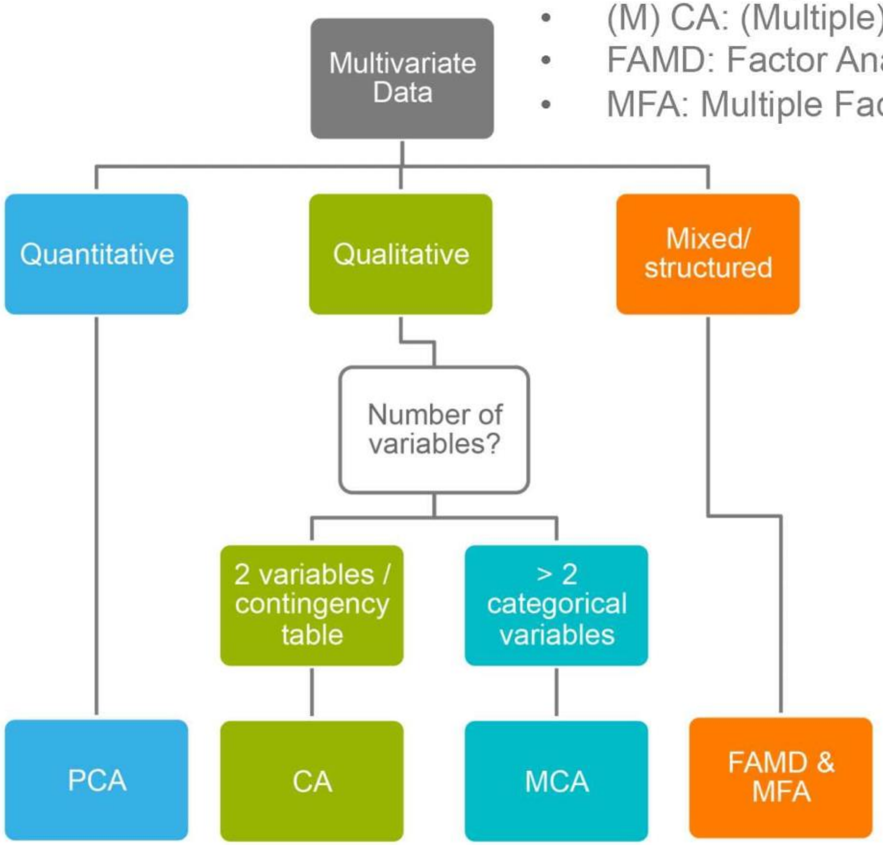
We would like to find a low-dimensional representation of the data that captures as much of the information as possible.

If we can obtain a two-dimensional representation of the data that captures most of the information, then we can plot the observations in this low-dimensional space.

PCA provides a tool to do just this.

Similarly to the cluster analysis, it is necessary to standardize the data.

- PCA: Principal Component Analysis
- (M) CA: (Multiple) Correspondence Analysis
- FAMD: Factor Analysis of Mixed Data
- MFA: Multiple Factor Analysis



Unsupervised Learning

Principal Component Analysis. Why?

Consider the data matrix *X* of standardized data with *n* rows and *p* columns.

The starting point is the correlation matrix S=Cor(X)

The S matrix has unit values on the main diagonal and the correlations outside the main diagonal and is a symmetric matrix

For example with *p* = 3,

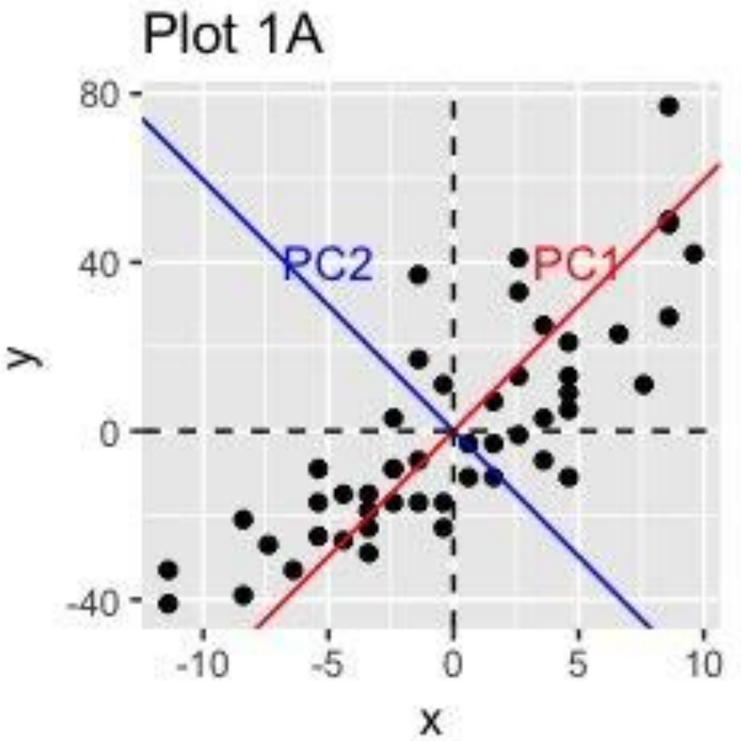
$$S = \begin{bmatrix} 1 & cor(X_1, X_2) & cor(X_1, X_3) \\ cor(X_1, X_2) & 1 & cor(X_2, X_3) \\ cor(X_1, X_3) & cor(X_2, X_3) & 1 \end{bmatrix}$$

Unsupervised Learning

Principal Component Analysis.

In the Plot 1A, the data are represented in the X-Y coordinate system. The dimension reduction is achieved by identifying the principal directions, called principal components, in which the data varies.

PCA assumes that the directions with the largest variances are the most "important" (i.e, the most principal).



Unsupervised Learning

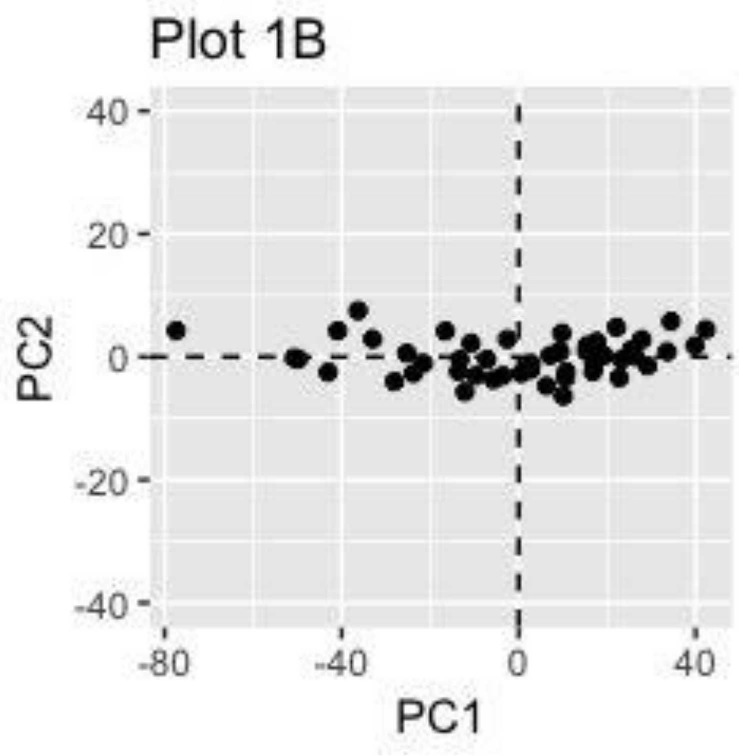
Principal Component Analysis.

In the figure 1A, the *PC1 axis* is the **first principal direction** along which the samples show the largest variation. The *PC2 axis* is the **second most important direction** and it is **orthogonal** to the PC1 axis.

The dimensionality of our two-dimensional data can be reduced to a single dimension by projecting each sample onto the first principal component (Plot 1B).

Technically speaking, the amount of variance retained by each principal component is measured by the so-called **eigenvalue**.

Note that, the PCA method is particularly useful when the variables within the data set are highly correlated. Correlation indicates that there is redundancy in the data. Due to this redundancy, PCA can be used to reduce the original variables into a smaller number of new variables (= **principal components**) explaining most of the variance in the original variables.



Unsupervised Learning

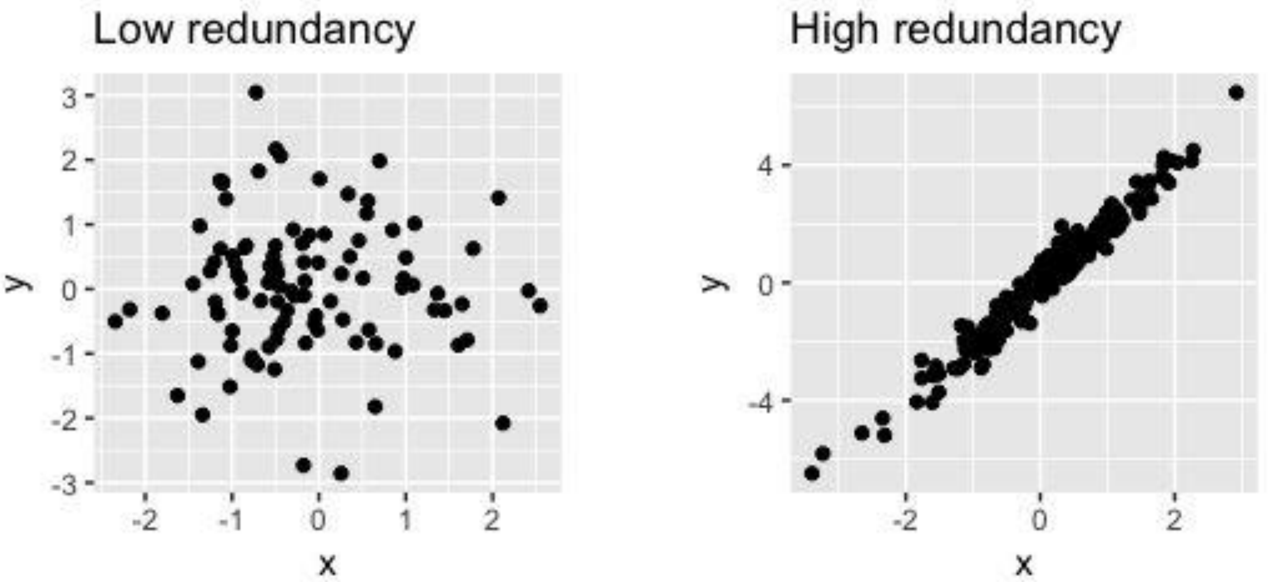
Principal Component Analysis.

Taken together, the main purpose of principal component analysis is to:

identify hidden pattern in a data set,

reduce the dimensionality of the data by **removing the noise** and **redundancy** in the data,

identify correlated variables



Unsupervised Learning

Interpretation of Principal Components.

One of the difficult task associated to the application of the technique of the principal components is their interpretation.

Since each principal component is a linear combination of the observed variables they do not have a clear unit of measurement.

Let us define the overall importance and the relative importance of the principal components.

The proportion of variance explained (PVE) by component *i* is

$$\frac{Var(y_i)}{p}$$

The overall importance of the first *k* principal components is given by

$$\frac{\sum_{i=1}^{k} Var(y_i)}{p}$$

It represents the share of variability explained by the components and therefore the statistical information explained by the principal components.

It is a measure of overall importance of the first *k* principal components.

Unsupervised Learning

Interpretation of Principal Components. Relative Importance

The degree of relative importance of each principal component is given by the linear correlation between a principal component and an original variable,

$$Cor\left(Y_j, X_i\right)$$

The sign and the value of this correlation indicate the sign and the value of the link between the *j*-th principal component and the *i*-th original variable.

This facilitates the interpretation of each principal component by referring it mainly to the variables with which it has a high association (high correlation in absolute value).

Unsupervised Learning

Number of the principal components.

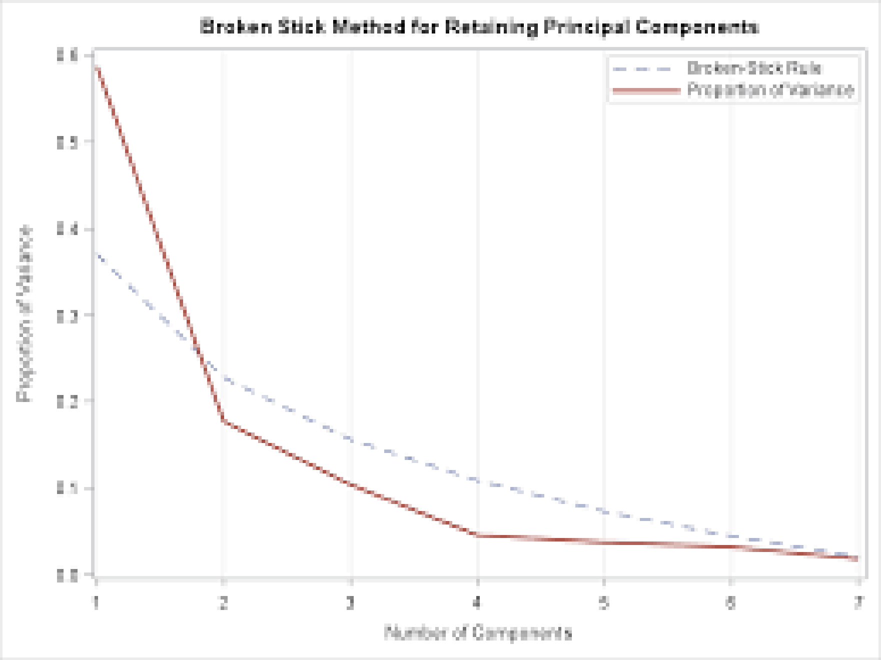
A critical point of the method is the choice of the number of principal components, that is the choice of *k*.

There is no unique criterion

One of the most commonly used is to consider many principal components until a certain percentage of explained variability defined a priori is reached, for example 60% or 80%.

Alternatively, it is possible to use the graphical tool (scree-plot) which presents the number of components on the x-axis and the corresponding variance on the y-axis.

Therefore, we can choose that number of components in correspondence with a clear reduction in the variance.



MASTER MEIM 2021-2022

THANKS FOR YOUR ATTENTION AND REMENBER THAT “Without big data analytics, companies are blind and deaf, wandering out onto the Web like deers on a freeway»!