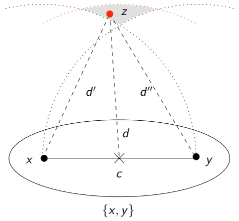
Mathematical Models and Applications

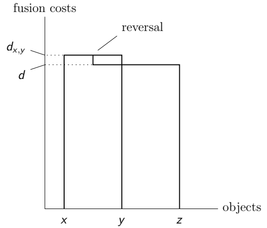
An introduction to Clustering Analysis

Pedro Cristiano Silva

Instituto Superior de Agronomia

2017-18





Outline

Main bibliography

Main reference An important part of this course material is based on the textbook

L P. Legendre, L. Legendre, *Numerical Ecology* (2003)

Other important references

C J. Cadima, *Multivariate Statistics*, course notes (2009/10)

E M. Ester, H.P. Kriegel, J. Sander, X. Xu, *Density Based algorithm for discovering clusters in large spatial databases with noise* (1996)

Ev B. S. Everitt, *Cluster Analysis* (1993)

M C. Manning, P. Raghavan, H. Shutze, *An Introduction to Information Retrieval* (2009)

R A.C Rencher, *Methods of Multivariate Analysis* (2002)

T Theodoridis, S. and Koutroumbas, K., Pattern Recognition (2009)

1. INTRODUCTION

What is clustering?

Oirginally developed by the biologists Robert Sokal and Peter Sneath in their seminal paper 'Principles of Numerical Taxonomy' in 1963, as method to classify organisms into species, given a set of proeminent features

A quest for discontinuities in data

It is a method of unsupervised classification

No priori information on the groups is assumed

Does not involve predicting

Domains of applications

- Life sciences
- Earth sciences
- Social sciences
- Economics
- . . .

Clustering

Clustering *is an operation of multidimensional analysis consisting in partitioning a collection of objects/variables/descriptors/etc, such that:*

- *Each object belongs to one and only one subset (cluster) of the partition*
- *Objects belonging to the same cluster are more similar (***internal cohesion***) than objects belonging to distinct clusters (***external separation***), given the variables considered*

The definition above is called **hard** *or* **crisp** *clustering.*
There is also a notion of **fuzzy** *clustering where the objects belong to a given set with a certain degree of membership*

Remark

Clustering always imposes some structure, even when no special structure or discontinuities are present! For instance, many clustering techniques tend to form globular clusters, e.g., with elliptical or spherical shapes

A more formal definition of clustering

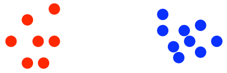
Given a collection of N objects, $X = \{x_1, \ldots, x_N\}$, one seeks a partition of X into K sets (clusters), i.e., a decomposition

$$X = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_K$$

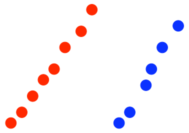
with $\mathcal{C}_i \neq \emptyset$ and $C_i \cap C_j = \emptyset$, $i \neq j$, maximizing (given the resemblance notion considered):

1. *the* **internal homogeneity or cluster cohesion**, *or equivalently, minimizing the* intra-cluster variability - *objects belonging to the same cluster should be very similar and share the same features*
2. *the* **external heterogeneity or cluster separation**, *i.e., the* between-cluster separation - *objects belonging to distinct clusters should be very dissimilar and have clear distinguished features*

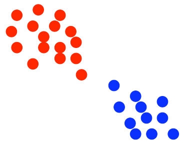
Examples


strong internal cohesion

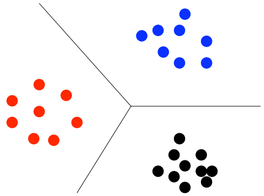
strong separation

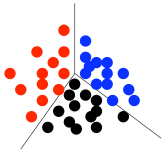

weak internal cohesion

strong separation


strong internal cohesion

weak separation


clear clustering structure


artificial clustering structure

Huge solution space...

The possible number of partitions of N elements into K clusters $(1 \leq K \leq N)$ equals

$$\xi(N,K) = \frac{1}{K!}\sum_{j=1}^{K}\binom{K}{j}(-1)^{K-j}j^N,$$

which is a huge number, known as Stirling of second kind, even for relatively small values of N and K, making impossible to to find the best partition by exhaustion

For instance, for the total number partitions of a set with 25 elements into 8 clusters gives

$$\xi(25,8) = 690223721118368580$$

The clustering analysis process

usually comprises the following steps:

- **Features selection**

  To choose the best features to encode as much as possible the information concerning the task, avoiding redundancy (i.e., highly correlated variables) but at the same time being parsimony

  Some questions arise: to normalize or not the variables? Types of features: nominal, ordinal, interval scaled and ratio-scaled, . . .

- **Clustering algorithm design/selection***:*

  What method and resemblance notions?

- **Cluster validation**

  internal*: how to assess the quality/stability of the clusters*

  external*: how the cluster results compares with results obtained using different clustering models or known information*

- **Interpretation of the results***:*

  Are the outcomes interpretable in the context of the problem? How to associate the most important variables/features to the groups

Cluster algorithm design/selection

A cluster model result depends on

- **the notion of distance/dissimilarity** *between individuals and clusters: should be adequate to the type of variables involved and to the type of results sought*
- **the clustering method***: should take into account the type of structure/shape of the clusters sought (rounded shape/arbitrary shape/. . .) and characteristics of the method itself (sensitivity to outliers/noise/ldots), computational issues (is scalable for large data sets), etc*

When two or more clustering models seem to be appropriate one should compare the outputs of such models to see what kind of patterns emerges out - **robust solutions**

2. (DIS)SIMILARITY MEASURES

Dissimilarity measures between individuals

A **dissimilarity measure** on a set X is a real function

$$d : X \times X \to \mathbb{R},$$

such that, for all $x, y, \in X$, we have

- $d(x, y) \geq 0$
- $d(x, y) = 0$
- $d(x, y) = d(y, x) \quad$ (*symmetric*)

If additionally, d verifies the *triangle inequality*

- $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in \mathbb{R}$,

d is called a **distance** or **metric**.

An important step in several multivariate analyses is to compute the dissimilarity/distance matrix of a given set of observations

Dissimilarity measures for quantitative data

Given $p \in]0, +\infty]$, the **Minkowski** or L_p**-norm** of an element $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ is defined as

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}},$$

The **Minkowski dissimilarity** between $x, y \in \mathbb{R}^d$ is defined as

$$D_p(x, y) = \|x - y\|_p$$

For $p \geq 1$ this dissimilarity is actually a distance

Important cases of Minkowski distance: $p = 1,\ 2,\ \infty$

- If $p = 2$ we get the usual **Euclidean metric** or ℓ_2-**distance**:

$$\begin{aligned} D_2(x, y) &= \sqrt{(x - y)^T (x - y)} \\ &= \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2} \end{aligned}$$

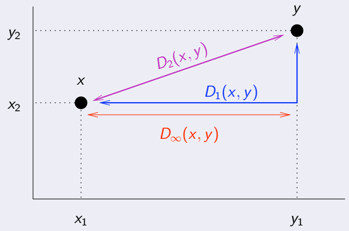
- If $p = 1$ we get the **Manhattan** city block or ℓ_1-**distance**:

$$D_1(x, y) = \sum_i |x_i - y_i|.$$

- If $p = \infty$ we get the **maximum metric** or Chebyshev or ℓ_∞-**distance**:

$$\begin{aligned} D_\infty(x, y) &= \lim_{p \to \infty} D_p(x, y) \\ &= \max_i \{|x_i - y_i|\}. \end{aligned}$$

Relation among the Minkowski distances



For all $x, y \in \mathbb{R}^2$ *we have*

$$D_1(x,y) \geq D_2(x,y) \geq D_\infty(x,y)$$

In general, if $1 \leq p < q \leq \infty$,

$$D_p(x,y) \geq D_q(x,y)$$

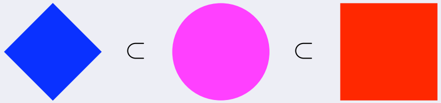
for all $x, y \in \mathbb{R}^N$

Relation among the Minkowski balls

For the 2-dimensional ball with norm 1, i,e, for the set of points lying at a distance inferior or equal to one from the origin,

$$B(0) = \{x \in \mathbb{R}^2 : \|x\| \leq 1\},$$

w.r.t. the Minkowski norms D_1, D_2 and D_∞, the relations among the metrics of the previous slide yield the following inclusions among the 1-balls for these norms,



As before, the property extends to the n-dimensional case

Important cases of Minkowski distance: $p = 1, 2, \infty$

- If $p = 2$ we get the usual **Euclidean metric** or ℓ_2-**distance**:

$$\begin{aligned} D_2(x, y) &= \sqrt{(x-y)^T(x-y)} \\ &= \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2 + \cdots + (x_d-y_d)^2} \end{aligned}$$

- If $p = 1$ we get the **Manhattan** city block or ℓ_1-**distance**:

$$D_1(x, y) = \sum_i |x_i - y_i|.$$

- If $p = \infty$ we get the **maximum metric** or Chebyshev or ℓ_∞-**distance**:

$$\begin{aligned} D_\infty(x, y) &= \lim_{p\to\infty} D_p(x, y) \\ &= \max_i\{|x_i - y_i|\}. \end{aligned}$$

The canberra distance

If x, y are N-dimensional vectors that only take positive values one can define the so-called **canberra** distance

$$d(x,y) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{x_i + y_i}$$

This distance is based relativizes the Manhattan distance in order to become more sensitive to small values

Graphical display of dissimilarity

R script (Script-1)

```
require(lattice)
x<-cbind(runif(10),rnorm(10))
d.1<-dist(x,method=‘‘manhattan’’,diag=FALSE,upper=FALSE,p=2)
d.1
as.matrix(d.1)
d.2<-dist(x,method=‘‘euclidean’’,diag=FALSE,upper=FALSE,p=2)
d.2
d.inf<-dist(x,method=‘‘maximum’’,diag=FALSE,upper=FALSE,p=2)
d.inf
d.can <-dist(x,method=‘‘canberra’’,diag=FALSE,upper=FALSE,p=2)
levelplot(as.matrix(d.1))
levelplot(as.matrix(d.2))
levelplot(as.matrix(d.infty))
```

Remark

x: *a numeric matrix, data frame or an object of class “dist” (usually arises as the outcome of* dist *function)*

method: *a distance measure to be used, among “euclidean”, “maximum”, “manhattan”, “canberra”, “binary” or “minkowski”*

Generalized euclidean distance

Let W *be* symmetric positive definite matrix *of order* n *and* $x, y \in \mathbb{R}^n$,

$$\|x\|_W = \sqrt{x^T W x}, \qquad D_W(x,y) = \|x - y\|_W,$$

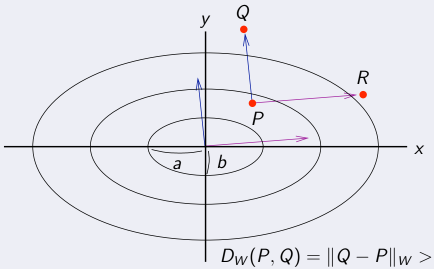
is the generalized norm defined by the inner produt $x^T W x$
Clearly, if W *equals the identity matrix we get the usual inner product and the usual Euclidean distance*

For instance, if $W = \begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix}$, *one gets*

$$\|x\|_W = \sqrt{\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2}}$$

Thus the points at distance 1 from the origin lie on a ellipse of semi-axes a *and* b. *In general a point* x *lis at distance* r *from the origin if it lies in the ellipse with semi-axes lengths* ra *and* rb.

Generalized euclidean distance



$D_W(P,Q) = \|Q - P\|_W > 2 > D_W(P,R) = \|R - P\|_W$

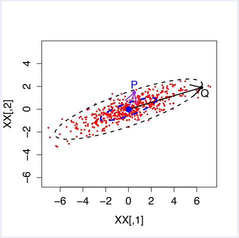
Important special cases

If $W = \Sigma^{-1}$ is the inverse of the **variance-covariance** *matrix of a set of observations X, D_W is called a* **Mahalanobis distance**

- *In the one dimensional case, $D_W(x, \mu) = \sqrt{\frac{(x-\mu)^2}{s^2}} = \frac{|x-\mu|}{s}$, which equals how many standard deviations x is away from μ*
- *For uncorrelated variables, W is the inverse of the diagonal matrix containing the variances and the points at distance 1 from the mean lie in an ellipsoid centred at the mean, with axes paralels to the coordinate axes and semi-axes lengths equal to the standard deviations*
- *In general, the points at Mahalanobis distance r from the mean, lie on a hyperellipsoid centred at the mean, with principal axes defined by a set of (orthogonal) eigenvectors of Σ^{-1} and the lengths of semi-axes are equal to the square roots of the eigenvalues multiplied by r. Thus, the Mahalanobis distances from the origin are 'smaller' along the directions of greater variability*
- *Used in supervised classifiers (like the minimum distance classifier) and to detect outliers*

Mahalanobis distance - example

The two points P and Q depicted in the figure



belong a data set with mean $\mu = \vec{0}$ and variance-covariance matrix Σ
Their mahalanobis distance equals ($W = \Sigma^{-1}$)

$$D_W(P,Q) = D_W(P-Q,\vec{0}) = \sqrt{(Q-P)^T\Sigma^{-1}(Q-P)} = 3.570066,$$

while their euclidean distances equals

$$D_2(P,Q) = D_2(P-Q,\vec{0}) = \sqrt{(Q-P)^T(Q-P)} = 5.907128$$

Some dissimilarity measures for binary data

$x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ binary vectors

a: nr components where both variables take value 1 (positive agreement)

b: nr of components where x take value 1 and y value 0 (disagreement)

c: nr of components where x take value 0 and y value 1 (disagreement)

d: nr of comp. where both variables take value 0 (negative agreement)

- **Simple matching** (counts double-zeroes, suitable if 0-1 represent equally valued attributes like male-female):

$$S(x,y) = \frac{a+d}{a+b+c+d} \qquad D(x,y) = 1 - S(x,y) = \frac{b+c}{a+b+c+d}$$

- **Jaccard coefficient** (does not count double zeroes, suitable if 0-1 represent unequal valued attributes, like species presences-absences):

$$J(x,y) = \frac{a}{a+b+c} \qquad D(x,y) = 1 - J(x,y) = \frac{b+c}{a+b+c}$$

Example

Assume that we have two binary variables representing the presence (1) or absence (0) at 16 spots of two species:

SpA=c(0,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0),
SpB=c(0,1,0,0,1,0,0,0,0,1,0,0,0,0,0,1)

How similar are the two species with regard to their distribution in the 16 spots ?

Using the simple matching one gets $S(SpA, SpB) = 10/16$, while using Jaccard coefficient one gets, $J(SpA, SpB) = 1/7$ (better)

Thus the Jaccard coefficient seems to be a more suitable similarity measure to determine homogeneous groups of species with respect to their distribution at a collection of spots

R script

```
# The R function dist with the method ‘‘binary’’ computes the
dissimilarity as $d(x,y) = 1 - S(x,y)$, where $S$ is the Jaccard coefficient
d = dist(cbind(x,y),method=‘‘binary’’,diag=FALSE,upper=FALSE,p=2)
# 6/7
```

Clustering of variables

1. An usual similarity notion between two variables x and y is Pearson's correlation
$$r = \frac{cov(x, y)}{s_x \, s_y}$$
This similarity can be transformed into a dissimilarity using the transformation $d = \sqrt{1 - r^2}$, which take values in the interval $[0, 1]$
2. Highly linearly correlated variables (positively or negatively) will have $d \approx 0$ while for uncorrelated variables $d \approx 1$
3. Alternatively, we can define $d = (1 - r)/2$. In this case the strengh of the linear relationship and the direction are both accounted
4. Each cluster consists of a set of variables highly correlated. This can be useful to detect redundancies and can give an idea of the number of principal dimensions of data
5. Actually, for each cluster we can define a synthetic variable called **latent variable**, which is a linear combination of the variables in the group, that minimizes the sum of the dissimilarities $(1 - r^2)$ with all these variables (a kind of centroid of the cluster of variables)

3. CLUSTERING METHODS

Clustering methods

Distance-based models rely only on pairwise dissimilarities between individuals

- **Hierarchical methods** - produce a *nested* structure of partitions. It does not require the number of clusters to be known *a priori*:
  - *Agglomerative clustering* (bottom-up strategy) - It starts from the partition consisting of one individual per cluster (singletons) until it aggregates all individual in the same cluster: single, complete, average, McQuitty, centroid, median, Ward, . . .
  - *Divisive clustering* (top-down strategy) - it proceeds in the opposite way: diana, . . .
- **Partitional methods** - produce *flat* (non-nested) partitions. Usually tries to maximize some intra-cluster homogeneity and inter-cluster heterogeneity criterion. It requires the number of clusters to be known *a priori*: K-means, K-medoids . . . ,

Clustering methods (cont.)

Other types of methods include:

- **Density-based clustering**: seek for high density regions of points (clusters) separated by low density of points (noise)
- **Model-based clustering** assumes that some model (or mixture of models) generates the data
- **Constrained-clustering**: accounts for other type of information, such as spatial relationships between individuals (for instance, contiguity relationships between cells in a map)
- **Fuzzy**: the same individual can belong to several classes with a certain degree of membership (probability)

Hierarchical agglomerative clustering

The most well-known HAC methods are:

1. Single-linkage or nearest-neighbor
2. Complete-linkage or furthest-neighbor
3. Average (UPGMA)
4. Weighted average or McQuitty (WPGMA)
5. Centroid (UPGMAC)
6. Median (WPGMC)
7. Ward or mininum-variance clustering

Agglomerative clustering algorithm based on proximity matrix

Algorithm

Input: *the proximity matrix containing the pairwise dissimilarities between N individuals $x_1, \ldots, x_N$*

1. *Starts with N clusters containing a single object each (singletons);*
2. *Merges the least dissimilar pair of clusters, i.e., the pair of clusters with smallest fusion cost into a new cluster and updates the proximity matrix (reducing its order by one);*
3. *Repeats step 2 until only one cluster remains ($N - 1$ steps).*

Output: *a sequence of length $N - 1$ encoding the merged clusters and their fusion costs*

Updating formula for HAC

During the aggregation process, the pairwise dissimilarity between the recently merged clusters $\mathcal{C}' \cup \mathcal{C}''$ and each one of the remaining clusters $\mathcal{C}$ is determined only in terms of the pairwise dissimilarities of $\mathcal{C}' \cup \mathcal{C}''$ and $\mathcal{C}$, i.e., in terms of the data from the previous proximity matrix

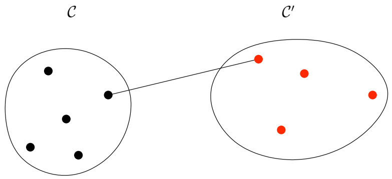
Thus, unlike other statistical methods the clustering analysis using a HAC does not require the knowledge of the original data set. Only the proximity matrix containing the pairwise distances is required!

The simplest HAC method

The most basic HAC algorithm is single-linkage

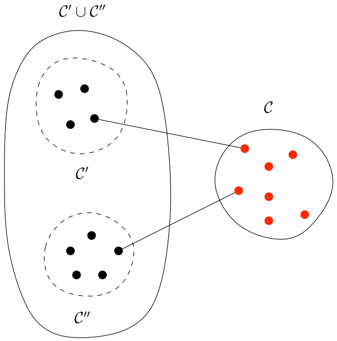
The fusion cost between two clusters $\mathcal{C}$ and $\mathcal{C}'$ is defined as the distance between the nearest pair of points, one in each cluster, i.e.,

$$D(\mathcal{C},\mathcal{C}') = \min_{x\in\mathcal{C},x'\in\mathcal{C}'} d(x,x')$$



Updating formula for single linkage

$$D(\mathcal{C}' \cup \mathcal{C}'', \mathcal{C}) = \min\{D(\mathcal{C}', \mathcal{C}), D(\mathcal{C}'', \mathcal{C})\}$$



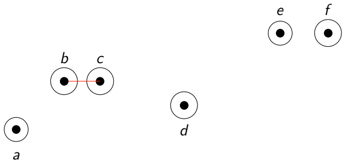
Dendrogram

The sequence of length $N - 1$ of the merged clusters and their fusion costs can be encoded in a special tree graph called **dendrogram**

Dendrograms *are tree-like diagrams made of branches that join terminal nodes (leaves).* Branches *represent clusters and the heights at which the branches are connected represent fusion costs.* Leaves *represent objects*
The lifetime *of a branch is the difference of fusion costs between the moments it appears and the moment it is aggregated*

Example of single-linkage clustering: step -1

$X = \{a, b, c, d, e, f\}$



PROXIMITY MATRIX

$$\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline b & 0.7 & & & & \\ c & 1.0 & \color{red}{0.3} & & & \\ d & 1.8 & .1.3 & 0.9 & & \\ e & 2.9 & 2.4 & 1.9 & 1.3 & \\ f & 3.4 & 2.8 & 2.4 & 1.7 & .5 \end{array}$$

At the initial step all clusters are singletons

Next step merges the clusters $\{b\}$ and $\{c\}$

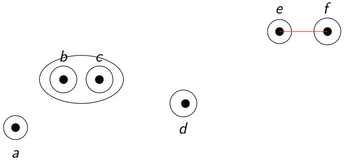
with fusion cost 0.3 (the least dissimilar pair) and in each dashed box

the minimum value is chosen, reducing the proximity matrix order by one,

and defining the dissimilarities between each one of the singletons and the new formed cluster $\{b, c\}$

Step -2

$X = \{a, b, c, d, e, f\}$



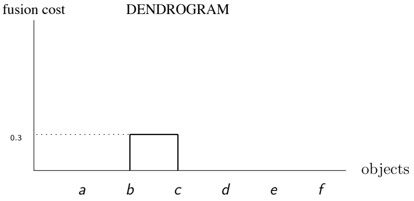
PROXIMITY MATRIX

	a	$\{b,c\}$	d	e
$\{b,c\}$	0.7			
d	1.8	0.9		
e	2.9	1.9	1.3	
f	3.4	2.4	1.7	0.5

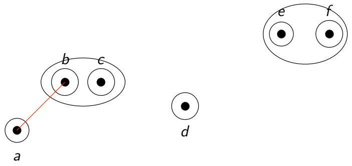
Next step merges the singletons $\{e\}$ and $\{f\}$

with fusion cost 0.5

DENDROGRAM



Step - 3

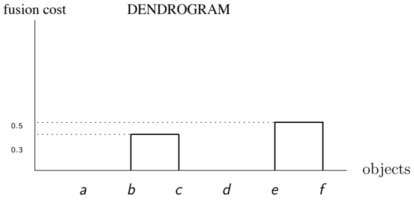


PROXIMITY MATRIX

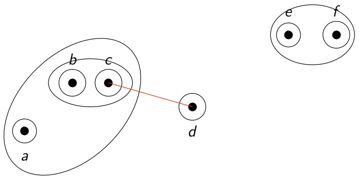
$$\begin{array}{c|ccc} & a & \{b,c\} & d \\ \hline \{b,c\} & \color{red}{0.7} & & \\ d & 1.8 & 0.9 & \\ \{e,f\} & 2.9 & 1.9 & 1.3 \end{array}$$

Next step merges the pair of clusters $\{a\}$ and $\{b, c\}$ with fusion cost 0.7

DENDROGRAM



Step - 4

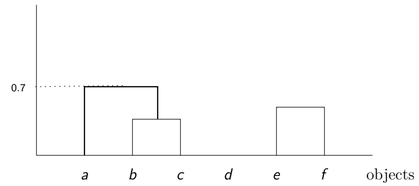


PROXIMITY MATRIX

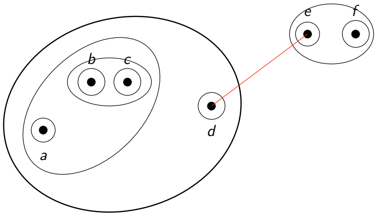
$$\begin{array}{c|cc} & \{a,b,c\} & d \\ \hline d & 0.9 & \\ \{e,f\} & 1.9 & 1.3 \end{array}$$

Next step merges the clusters $\{a, b, c\}$ and $\{d\}$ with fusion cost 0.91

DENDROGRAM



Step - 5

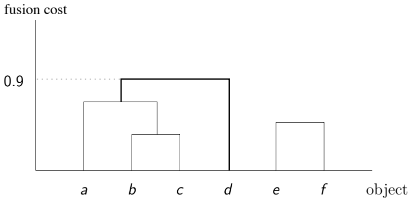


Next step is the final one and merges the clusters $\{a, b, c, d\}$ and $\{e, f\}$ with fusion cost 1.3

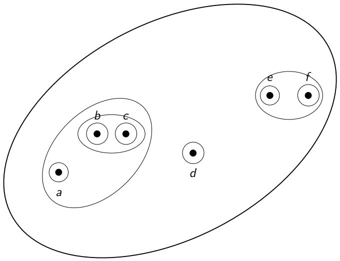
PROXIMITY MATRIX

$$\begin{array}{c|c} & \{a,b,c,d\} \\ \hline \{e,f\} & 1.3 \end{array}$$

DENDROGRAM

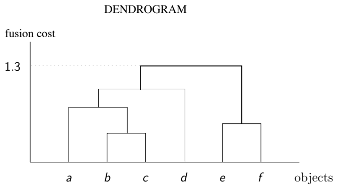


step - 6



PROXIMITY MATRIX

EMPTY



The R function hclust

It performs hierarchical aglomerative clustering using several aggregation criterion methods and admits an arbitrary dissimilarity matrix as input

input: a *dissimilarity matrix d* and the clustering *method* among the options, “ward”, “single”, “complete” (default), “average”, “mcquitty”, “median” or “centroid”.

value: the function returns an object of the class *hclust*, which consists of a list including, among others, the following elements:
merge: a $(n-1) \times 2$ matrix indicating the clusters being merged
heigth: the list of fusion costs

R script

```
hc<-hclust(d, method=‘‘complete’’, members=NULL)
```

`plot(hc)` *or* `plot(hc, hang=-1)` *to plot the dendrogram with all leaves at the same height*

Example

R script (SL-1)

```
X<-matrix(c(0,0,0.5,0.5,0.85,0.5,1.75,0.25,2.75,1,3.25,1),
nrow=6,byrow=TRUE)
d<-dist(X)
SL<-hclust(d, method="single")
SL$height
[1] 0.375 0.5 0.707 0.91 1.25
SL$merge
[,1] [,2]
[1,] -2 -3 (merges singletons {2} with {3})
[2,] -5 -6 ( merges singletons {5} with {6})
[3,] -1 1 (merges singleton {1} with cluster {2,3})
[4,] -4 3 (merges singleton {4} with cluster {1,2,3})
[5,] 2 4 (merges clusters {5,6} with cluster {1,2,3})
# The number with minus sign refers to a singleton ID,
# otherwise refers to the step number where the cluster was aggregated
plot(SL)
plot(SL, hang=-1)

# try with horiz=TRUE
```

Where to cut the dendrogram?

A cut in a dendrogram at a given height τ produces the (flat) partition into the clusters whose fusion cost is smaller than or equal to τ

Usually one seeks cuts in the dendrogram such that:

- **splits high consecutive height differences (high lifetimes)** *to get high inter-cluster heterogeneity*
- **as close to the leaves as possible** *to get high intra-class homogeneity*

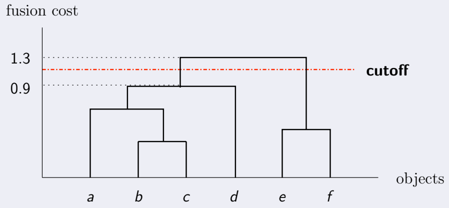
Some caution has to be applied regarding the decision where to cut the dendrogram (and what is the "best" number of clusters). With some methods (for instance, the Ward method), the dendrogram height distances tend to be higher when larger clusters are merged

Several internal validity indices can be used complementarily to estimate the optimal number of clusters

Example

For instance to obtain a partition into 2 clusters we have to cut the dendrogram at some height in the interval $]0.9, 1.3[$, *yielding the clusters* $\mathcal{C} = \{a, b, c, d\}$ *and* $\mathcal{C}' = \{e, f\}$

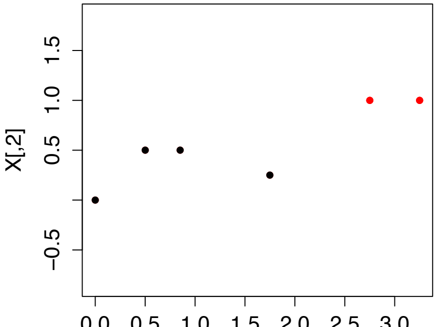
CUTTING THE DENDROGRAM



Cutting the dendrogram in R

R script (SL-1-cutree)

```
SL<-hclust(X,method="single")
part<-cutree(SL,2) # 2 clusters
# # or
part<-cutree(SL,h=1.1) # h is the height
part

plot(X,type="p",cex=0.8,pch=16, col=part,asp=TRUE)
```



Contraction of the attribute space

In single-linkage two clusters are merged at a fusion cost τ if and only if there is a pair of objects, one in each cluster, with pairwise distance inferior or equal than τ

As the cluster growths it becames more and more easier to incorporate new elements in a cluster, as if the distances between objects were getting smaller and smaller (actually, the distance of a point to any point in the cluster becames the distance to the nearest point of the cluster)

New individuals tend to aggregate to existing clusters, often producing elongated clusters (chain effect)

Cophenetic distance

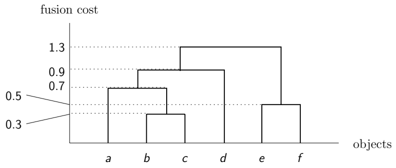
1. *Introduced by Sokal and Rohlf*
2. *It assess how well the dendrogramatic distance preserve the original distances*
3. *Measures the degree of fit of a classification with respect to the original data set*
4. *It is considered an internal validation criterion for evaluating the efficiency of various clustering techniques, particularly for hierarchical methods*

Cophenetic distance

Definition

The **cophenetic distance** *between two individuals x and y with respect to a given HAC is the merging cost at which x and y became members of the same cluster, during the course of hierachical clustering. Cophenetic distances are distances in the usual sense, i.e., they are dissimilarities that verify the triangle inequality, under the assumption of monotonicity*

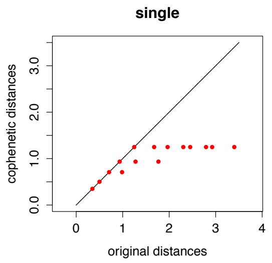
Any dendrogram can be uniquely represented by its matrix of cophenetic distances. This matrix can be used to compare distinct classifications



$$\left[\begin{array}{c|ccccc}
 & a & b & c & d & e \\
\hline
b & 0.7 & \cdot & \cdot & \cdot & \cdot \\
c & 0.7 & 0.3 & \cdot & \cdot & \cdot \\
d & 0.9 & 0.9 & 0.9 & \cdot & \cdot \\
e & 1.3 & 1.3 & 1.3 & 1.3 & \cdot \\
f & 1.3 & 1.3 & 1.3 & 1.3 & 0.5
\end{array}\right]$$

Shepard-like diagram

The Shepard-like diagram of the previous example shows that the cophenetic distances are smaller than the original ones, due to the *contraction of the space of attributes*.



Remark

Shepard diagrams *are scatter plots commonly used to visualize comparisons between distances*

Distortion measures - Cophenetic Correlation Coefficient

The correlation between the original and the cophenetic distances in a agglomerative clustering algorithm can be numerically assessed using several correlation coefficients.

The **cophenetic correlation coefficient** *(CPCC) is the Pearson's correlation between the original and the cophenetic distances (using half of the proximity matrix), i.e.,*

$$CPCC = \frac{cov(D,C)}{s_D s_C} = \frac{\sum_{i<j}(d_{ij}-\bar{d})(c_{ij}-\bar{c})}{\sqrt{\sum_{i<j}(d_{ij}-\bar{d})^2 \sum_{i<j}(c_{ij}-\bar{c})^2}}$$

where $C = (c_{ij})$ and $D = (d_{ij})$ are the vectors containing the original and cophenetic distances

CPCC has been used as an internal validation criterion for hiearchical clusterings A high CPCC usually means that the cophenetic distances are a good portray of the original distances. The cophenetic correlation usually ranges between 0.6 and 0.95. Cophenetic correlations above .75 are considered good.

Cophenetic Spearman Correlation Coefficient

Another distortion measure is **Cophenetic Spearman's rank order correlation coefficient** *(CSCC), which only depends on the ranks of the variables*

Spearman's rank order correlation coefficient can be applied to compare the cophenetic and original dissimilarities and is computed as r Pearson's correlation between the respective ranked variables $rk(C) = (c'_{ij})$ and $rk(D) = (d'_{ij})$ defined by the vectors of original and cophenetic distances,

$$CSCC = \frac{cov(rk(D), rk(C))}{s_{rk(D)} s_{rk(C)}} = \frac{\sum_{i<j}(d'_{ij} - \bar{d})(c'_{ij} - \bar{c}')}{\sqrt{\sum_{i<j}(d'_{ij} - \bar{d}')^2 \sum_{i<j}(c'_{ij} - \bar{c}')^2}}.$$

A Spearman's rank order correlation close to 1 signifies that we have a strong positive relationship between the ranks of original and cophenetic distances. In particular can detect non linear correlations between the cophenetic and the original distances. The higher the original distances the higher the cophenetic distances and vice-versa. If the rank order correlation is close to -1 the opposite behavior occurs

Shepard diagram and cophenetic correlations in R

R script (SL-3)

```
coph.SL<-cophenetic(SL) # an object of type dist

cor.coph<-cor(coph.SL,d)

cor.coph

sp.coph<-cor(rank(coph.SL),rank(d))

sp.coph

plot(seq(0,3.5,.01),seq(0,3.5,.01),pch=16,type="p",
cex=.2, main="single", asp=T, xlab="original distances",
ylab="cophenetic distances")

points(d,coph.SL,asp=1,xlim=c(0,5),ylim=c(0,5),pch=16,
col="red",type="p")
```

MST

A **graph** *is a set of nodes (also called vertices) connected by edges. A graph is called a* tree *if it does not contain closed paths (i.e., closed sequences of edges). In a graph we may have weights on its edges (or vertices)*

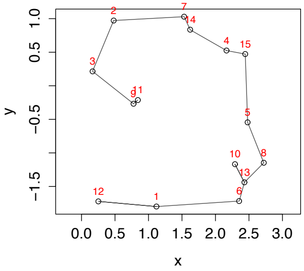
An minimum spanning tree (MST) of a set of points in a d-dimensional Euclidean space is a tree connecting all points such that the overall sum of the lengths of line segments (edges) connecting the nodes is minimal

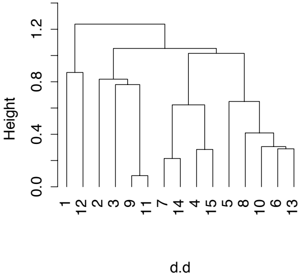
MST are not necessarily unique (if we have ties in the dissimilarity matrix)

In general MSTs can also be defined for abstract graphs with edges weights. MSTs can be efficiently computed using Kruskal or Prim algorithms

Single-Linkage and MST algorithm

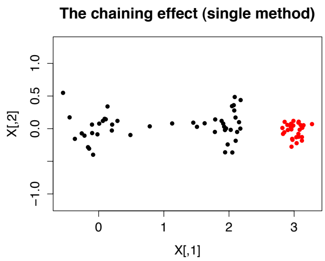
The hierarchical clustering of a cloud of points with the single linkage can be obtained in the following way: consider an MST connecting all points and sequencially aggregate the clusters whose pair of nearest neighbors correspond to vertices of the edges of the MST, ordered from the smallest to the largest length

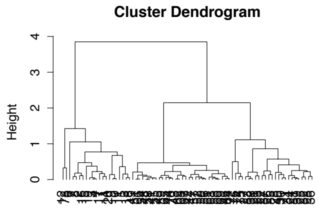




Chaining effect

The chaining effect is usually produced by the existence of a few intermediate points between clusters, giving rise to elongated clusters.

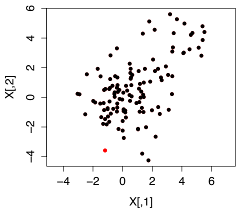


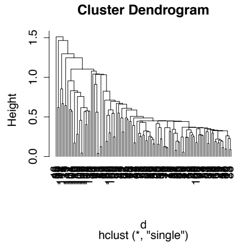


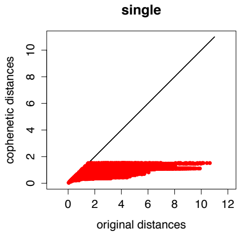
Chaining effect

In the next example the single linkage produced a 2-partition with an elongated cluster and a singleton (red dot). The chaining effect is clearly visible in the dendrogram where long chains of nearby points are aggregated

All points are aggregated at very lower costs: the maximum heights of the red bars in the shepard diagram is very low (below 2), while the original distances range from 0 to above 10







Single-linkage clustering - summary

Pros

- Can detect arbitrary cluster shapes
- Can be applied to large data sets, since it is computationally efficient - there are polynomial-time algorithms
- Invariant under monotonic transformations of the proximity matrix - only ordinal properties, i.e., rank orders, are important
- Emphasis clusters separation
- Insensitive to ties in the proximity matrix

Cons

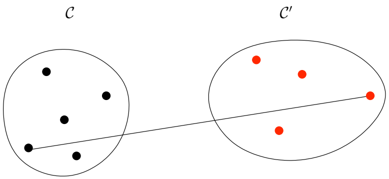
- Contracts the space of attributes
- Sensitive to observation errors and noise
- Suffers from the chain effect, producing elongated clusters, often originating very unbalanced clusters

Complete-linkage clustering model

The complete-linkage or farthest sorting is the opposite of nearest-neighbor clustering algorithm

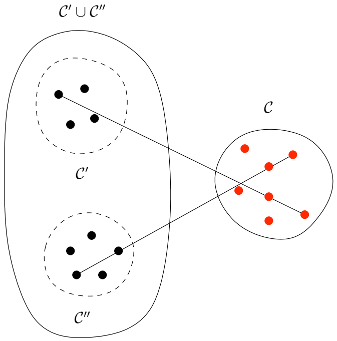
The fusion cost between two clusters $\mathcal{C}$ and $\mathcal{C}'$ in this method is defined as the distance between the farthest pair of points, one in each cluster, that is,

$$D(\mathcal{C}, \mathcal{C}') = \max_{x \in \mathcal{C}, x' \in \mathcal{C}'} d(x, x')$$



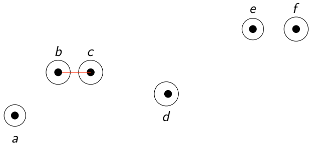
Updating formula for complete linkage

$$D(\mathcal{C}' \cup \mathcal{C}'', \mathcal{C}) = \max\{D(\mathcal{C}', \mathcal{C}), D(\mathcal{C}'', \mathcal{C})\}$$



Complete-linkage: step -1

$X = \{a, b, c, d, e, f\}$



First pair to be merged is $\{b, c\}$

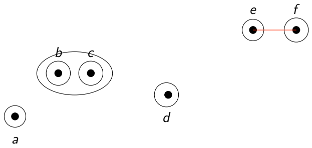
with fusion cost 0.3

PROXIMITY MATRIX

	a	b	c	d	e
b	0.7				
c	1.0	0.3			
d	1.8	1.3	0.9		
e	2.9	2.4	1.9	1.3	
f	3.4	2.8	2.4	1.7	.5

Step -2

$X = \{a, b, c, d, e, f\}$

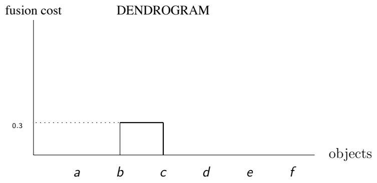


PROXIMITY MATRIX

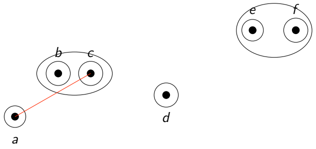
$$\begin{array}{c|cccc} & a & \{b,c\} & d & e \\ \hline \{b,c\} & 1.0 & & & \\ d & 1.8 & 1.3 & & \\ e & 2.9 & 2.4 & 1.3 & \\ f & 3.4 & 2.8 & 1.7 & 0.5 \end{array}$$

Next pair to be merged is $\{e\}$ and $\{f\}$

with fusion cost 0.5



Step - 3

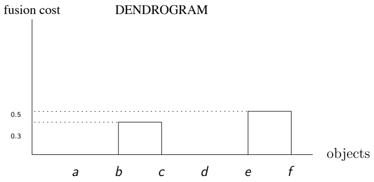


Next pair to be merged is $\{a\}$ and $\{b, c\}$

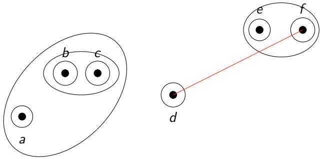
with fusion cost 1.0

PROXIMITY MATRIX

$$\begin{array}{r|ccc}
 & a & \{b,c\} & d \\
\hline
\{b,c\} & 1.0 & & \\
d & 1.8 & 1.3 & \\
\{e,f\} & 3.4 & 2.8 & 1.7
\end{array}$$



Step - 4



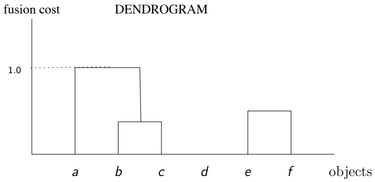
Next pair to be merged $\{d\}$ and $\{e, f\}$

with fusion cost 1.7

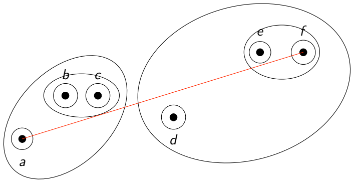
PROXIMITY MATRIX

$$\begin{array}{c|cc} & \{a,b,c\} & d \\ \hline d & 1.8 & \\ \{e,f\} & 3.4 & 1.7 \end{array}$$

DENDROGRAM

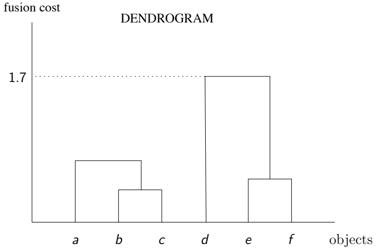


Step - 5



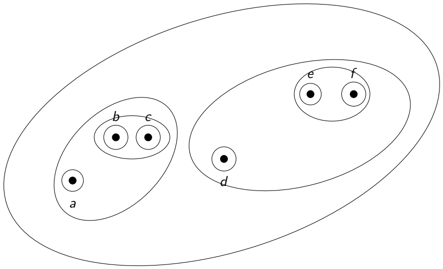
PROXIMITY MATRIX

$$\begin{array}{c|c} & \{a,b,c\} \\ \hline \{d,e,f\} & 3.4 \end{array}$$



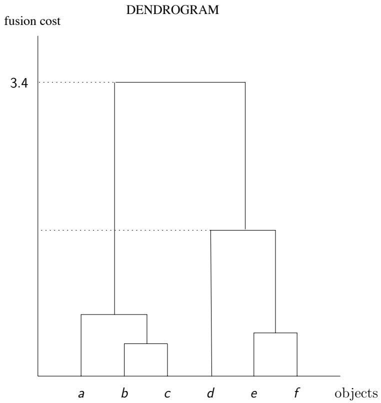
Next step merges (final) merges the pair of clusters $\{a, b, c\}$ and $\{d, e, f\}$ with fusion cost 3.4

step - 6



PROXIMITY MATRIX

EMPTY



Dilation of the attribute space

In complete-linkage two clusters are merged at a fusion cost τ if and only if all elements of one cluster are at a distance inferior to or equal than τ with respect to all elements of the other cluster.

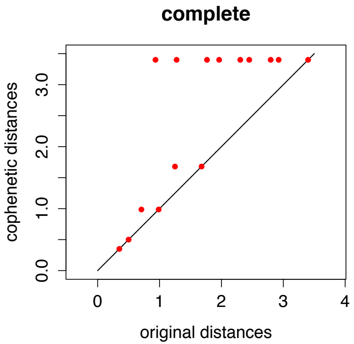
As the cluster growths it becames more and more harder to incorporate new elements in a cluster, as if the distances between objects were getting greater and greater. All aggregations tend to occur at small dissimilarities and consecutive dendrogram heights tend to become larger

New elements tend to form new clusters

This produces a "dilation effect" in the attribute space that can be observed comparing the original proximity matrix and the cophenetic matrix in a scatterplot shepard-like diagram

Shepard-like diagram and the cophenetic correlation

The Shepard-like diagram of the previous example shows that the cophenetic distances in complete-linakge clustering are greater than the original ones, due to the *dilation of the space of attributes*



The cophenetic and Spearman rank order correlation coefficients are, respectively, $r_c = 0.75$ and $s_c = 0.79$

Complete-linkage clustering - summary

Pros

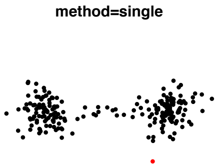
- Favors compactness - tend to form tight spherical clusters with small diameters
- Invariant under monotonic transformations of the proximity matrix - only the dissimilarity ranks are important.

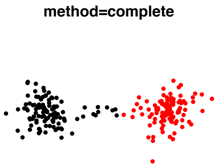
Cons

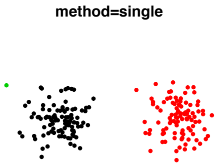
- Dilates the space of attributes
- The decision of aggregate two cluster only relies on one individual in each cluster
- Sensitive to outliers
- Cannot detect arbitrary cluster shapes

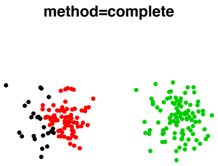
Noise and outliers: single vs complete methods

Next examples show that single clustering method is more sensitive to noise than complete, whereas the opposite occurs with outliers









Average clustering

In-between the contraction effect of single linkage method and the dilation effect of complete linkage method, we have the unweighted pair group method average *(UPGMA) method, simply known as* **average method**, *which conserves the metric properties of the attribute space*

This method usually presents the best cophenetic correlation coefficient among the three methods, but is not invariant under monotonic transformations of the proximity matrix

Remark

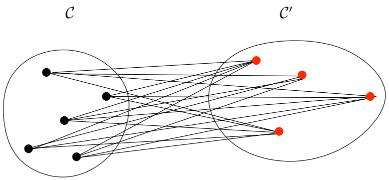
Average clustering methods sometimes refer to family methods, average (UPGMA), Mcquitty (WPGMA), centroid (UPGMC) and median (WPGMC)

Average fusion cost

The fusion cost between two clusters $\mathcal{C}$ and $\mathcal{C}'$ is defined as the arithmetic average of the distances connecting one point in $\mathcal{C}$ with one point in $\mathcal{C}'$,

$$D(\mathcal{C},\mathcal{C}') = \frac{\sum\limits_{x\in\mathcal{C},x'\in\mathcal{C}'} d(x,x')}{n\,n'},$$

where $n = |\mathcal{C}|$ and $n' = |\mathcal{C}'|$



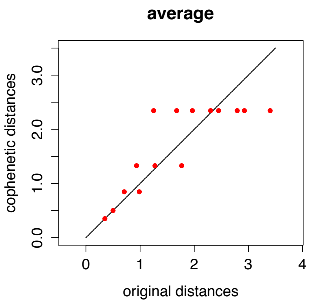
Updating formula

$$\boxed{D(\mathcal{C}' \cup \mathcal{C}'', \mathcal{C}) = \frac{n' D(\mathcal{C}', \mathcal{C}) + n'' D(\mathcal{C}'', \mathcal{C})}{n' + n''}}$$

where $n' = |\mathcal{C}'|$, $n'' = |\mathcal{C}''|$

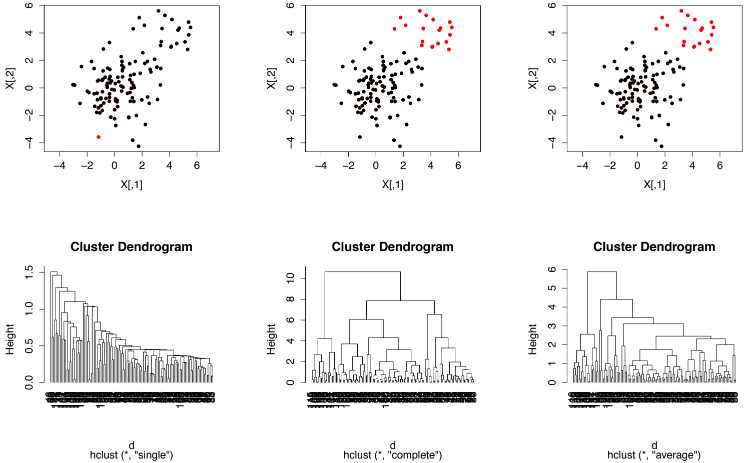
Shepard-like diagram and the cophenetic correlation

The Shepard-like diagram of the previous example shows that the cophenetic distances lie in both sides of the diagonal yielding a better portray of the original distances. This is also confirmed by the higher cophenetic correlation. This method conserves the metric in the space of attributes



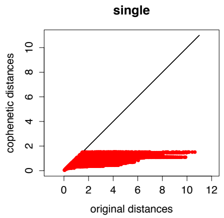
The Pearson and Spearman rank order cophenetic correlations are $CPCC = 0.83$ and $CSCC = 0.84$

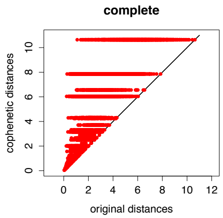
Comparison single-complete-average

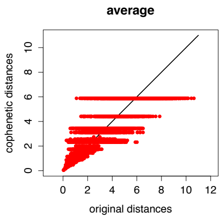


contracting dilating conserving

Comparison single-complete-average (cophenetic)







contracting

CPCC=0.69

dilating

CPCC=0.74

conserving

CPCC=0.82

Ultrametric property

Definition

We say that a distance D verify the **ultrametric property** *if it verifies the following stronger condition than triangle inequality: for all individuals x, y, z*

$$D(x,z) \leq \max(D(x,y), D(y,z))$$

The single-linkage, complete-linkage and average cophenetic distances verify the ultrametric property

Proposition

If the cophenetic distance verify the ultrametric property, the fusion costs are monotonically increasing as the clusters are being merged, i.e.,

$$D(\mathcal{C}' \cup \mathcal{C}'', \mathcal{C}) \geq D(\mathcal{C}', \mathcal{C}'')$$

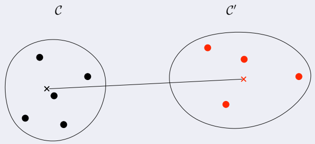
Remark

If this property fails the dendrogram may present crossovers (or, inversions), i.e., the fusion cost may decrease!

Centroid clustering model

In this method, also known as UPGMC (unweighted pair group method centroid) the **clusters are represented by their centroids** *and the fusion cost between two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ is defined as the distance between the respective centroids*

$$D(\mathcal{C}_i, \mathcal{C}_j) = \left\| \frac{1}{|\mathcal{C}_i|} \sum_{x_i \in \mathcal{C}_i} x_i - \frac{1}{|\mathcal{C}_j|} \sum_{x_j \in \mathcal{C}_j} x_j \right\|$$

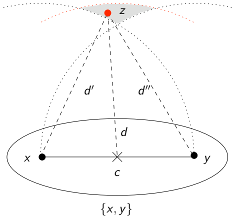


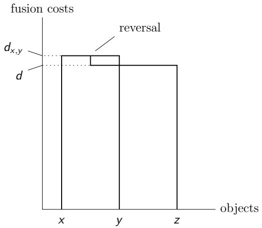
The centroid of the merged group will be $m_{ij} = \dfrac{n_i m_i + n_j m_j}{n_i + n_j}$

Centroid clustering model - crossovers

In the centroid method the cophenetic distances may not verify the ultrametric property, giving rise to non-monotonic fusion distances with **crossovers** *(alo called* **inversions***) in the dendrogram*

All circles have radii equal to the distance between x and y, $d_{x,y}$.





Since z (red point) lie in the grey area, ouside the black circles, $d_{x,y} < d', d''$. Hence x and y are the first pair of objects to be merged. Since z lie inside the red circle centred at the centroid c of x and y,

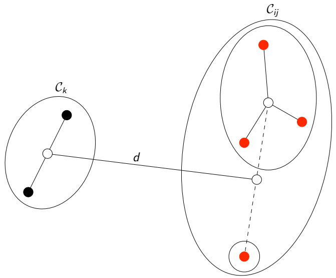
$$D(\{x,y\},z) = d_{c,z} < d_{x,y} = D(\{x\},\{y\})$$

Median clustering model (WPGMC)

In the centroid clustering if two clusters have very different sizes the centroid of the merged cluster tend to be close or even inside the largest cluster. The median clustering is a variant designed to correct this distortion effect

Updating formula: the distance between a cluster $\mathcal{C}_k$ and the cluster $\mathcal{C}_{ij} = \mathcal{C}_i \cup \mathcal{C}_j$, is given by the distance of the centroid of $\mathcal{C}_k$ to the median point of the centroids of $\mathcal{C}_i$ and $\mathcal{C}_j$, i.e.,

$$D(\mathcal{C}_{ij}, \mathcal{C}_k) = \left\| \frac{m_i + m_j}{2} - m_k \right\|$$



Huygens theorem

Let X be data set consisting of N objects, $x_1, \dots, x_N$, in p quantitative variables and mean $\bar{x}$. Assume moreover that X is partitioned into K clusters

$$X = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_K$$

Let m_k be the centroid of $\mathcal{C}_k$ and n_k the number of its elements

By Huygens theorem

$$\boxed{SSQ_t = SSQ_b + SSQ_w}$$

where,

$$SSQ_t = \sum_{i=1}^{N} \|x_i - \bar{x}\|^2 \textit{ (total inertia)}$$

$$SSQ_b = \sum_{k=1}^{K} n_k \|m_k - \bar{x}\|^2 \textit{ (between-clusters inertia)}$$

$$SSQ_w = \sum_{k=1}^{K} \sum_{x \in \mathcal{C}_k} \|x - m_k\|^2 \textit{ (total within-clusters inertia)}$$

Ward's method

In Ward's method **each cluster is represented by its centroid**

At beginning all clusters have a single element and thus,

$$SSQ_t = SSQ_b, \qquad SSQ_w = 0$$

Note that SSQ_b correspond to the sum squared pairwise distances

Ward's method seeks **to maximize the between clusters inertia** *SSQ_b as the groups are being clustered, which is equivalent, by Huygens theorem,* **to minimize the total within-group inertia** *SSQ_w (information loss for replacing the elemens by their centroid)*

Increase in the sum of within-cluster inertia

In each step, Ward's method seeks to merge the pair of clusters $\mathcal{C}_i$, $\mathcal{C}_j$ producing the smallest increase in the total within-cluster inertia

Since

$$SSQ_w = \sum_{k=1}^{K} e_k^2,$$

where e_k^2 is the within group inertia of cluster k,

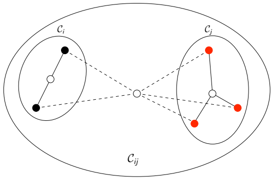
$$e_k^2 = \sum_{x \in \mathcal{C}_k} \|x - m_k\|^2 = \frac{\sum_{x,y \in \mathcal{C}_k} \|x - y\|^2}{2n_k}$$

When two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ are merged into a cluster $\mathcal{C}_{ij}$, the increase in SSQ_w reduces to the following ΔSSQ_w statistic:

$$\Delta_{i,j} SSQ_w = e_{ij}^2 - e_i^2 - e_j^2$$

since all other within-group inertias are not affected

Updating formula



It can be proved that

$$\Delta_{ij} SSQ_w = \frac{n_i n_j}{n_i + n_j} \|m_i - m_j\|^2,$$

In particular $\Delta_{ij} SSQ_w$ only depends on squared pairwise dissimilarities between the cluster centroids m_i, m_j and their sizes n_i and n_j

One can use the above formula to determine the next pair to be clustered but requires the original coordinates of the points (raw data). There is an alternative formula that only requires the (squared) pairwise distances. However the matrix can be huge for large data sets

Ward's method - dendrogram height scale

Depending on implementation details the dendrogram height scale produced by Ward's method can de defined in terms of:

- *Squared fusion distances*
- *Non-squared fusion distances (usefull, e.g., to compute cophenetic correlation coefficients)*
- SSQ_w *statistic*
- ΔSSQ_w
- *Proportion of (retained) variance:* $\frac{SSQ_b}{SSQ_t}$

Exercise

Perform Ward's clustering on the data set $X = \{1, 2, 4, 8\}$ *using the update formula of the previous slide with squared fusion distances and* ΔSSQ_w *statistic as dendrogram height scales*

Ward's clustering method - summary

Pros

- Tend to create hyperspherical shape clusters, with approximately the same number of elements each (balanced)
- Conservative (some authors consider it contracting although less than single-linkage method)
- No crossovers
- Is regarded by some authors as the companion hierarchical method to use with correspondence analysis (CA) since it shares the same variance criterion

Cons

- Computational intensive
- Cannot detect arbitrary cluster shapes
- Sensitive to outliers since it uses centroids

Application to the Iris flower data set

Exercise

Considere o conjunto dos dados dos lírios

1. *Efectue uma análise classificatória hierárquica usando os métodos do vizinho mais próximo (single), do vizinho mais distante (complete), das distâncias médias (average), do centroide e do inércia mínima (ward), com as distâncias euclideana, Manhattan, do máximo e de Canberra.*
2. *Represente os respetivos dendrogramas. Quantos grupos os dendrogramas sugerem?*
3. *Efetue o corte na árvore de modo a obter os grupos sugeridos na alínea anterior em cada um dos casos e compare a classificação obtida com a classificação verdadeira. Comente*
4. *Calcule os coeficientes de correlação de Pearson e Spearman. Comente.*

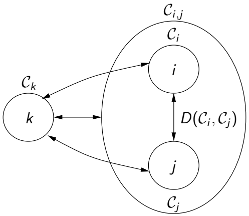
Lance-Williams updating formula

All HAC methods considered so far can be implemented using a general formula to update the cluster dissimilarities after a merge step in terms of the dissimilarities prior to that fusion

Given clusters $\mathcal{C}_i$, $\mathcal{C}_j$ and $\mathcal{C}_k$, denote by $\mathcal{C}_{ij}$ the cluster obtained merging clusters $\mathcal{C}_i$, $\mathcal{C}_j$ and set

$$\boxed{D(\mathcal{C}_{ij},\mathcal{C}_k) = \alpha_i D(\mathcal{C}_i,\mathcal{C}_k) + \alpha_j D(\mathcal{C}_j,\mathcal{C}_k) + \beta D(\mathcal{C}_i,\mathcal{C}_j) + \gamma |D(\mathcal{C}_i,\mathcal{C}_k) - D(\mathcal{C}_j,\mathcal{C}_k)|}$$

where α_i, α_j, β and γ are convenient parameters and $D(\cdot,\cdot)$ refer the distances below (see the chart in the next slide):



Lance-Williams chart

	α_i	α_j	β	γ	dissimilarity matrix	effect on metric	reversals
single	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	d_{ij}	very contracting	NO
complete	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	d_{ij}	very dilating	NO
average (UPGMA)	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0	d_{ij}	conserving	NO
McQuitty (WPGMA)	$\frac{1}{2}$	$\frac{1}{2}$	0	0	d_{ij}	conserving	NO
centroid (UPGMC)	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$\frac{-n_i n_j}{(n_i+n_j)^2}$	0	d_{ij}^2	conserving	can occur
median (WPGMC)	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0	d_{ij}	conserving	can occur
Ward	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	$-\frac{n_k}{n_i+n_j+n_k}$	0	d_{ij}^2	contracting	NO

E.g.: Ward clustering via LW updating formula

The LW updating formula for Ward's method is given by

$$\boxed{D^2(\mathcal{C}_i \cup \mathcal{C}_j, \mathcal{C}_k) = \frac{(n_i + n_k) \cdot D^2(\mathcal{C}_i, \mathcal{C}_k) + (n_j + n_k) \cdot D^2(\mathcal{C}_j, \mathcal{C}_k) - n_k \cdot D^2(\mathcal{C}_i, \mathcal{C}_j)}{n_i + n_j + n_k}}$$

where $n_i = |\mathcal{C}_i|$, $n_j = |\mathcal{C}_j|$ and $n_k = |\mathcal{C}_k|$
This expression returns the increase in the SSQ_w statistic when clusters $\mathcal{C}_i \cup \mathcal{C}_j$ and $\mathcal{C}_k$ are merged, depending only on the clusters involved

Let us exemplify on the data set $X = \{a, b, c, d\} = \{1, 2, 4, 8\}$
The dissimilarity and squared dissimilarity matrices are, respectively,

$$\left[\begin{array}{c|ccc} D & a & b & c \\ \hline b & 1 & & \\ c & 3 & 2 & \\ d & 7 & 6 & 4 \end{array}\right], \qquad \left[\begin{array}{c|ccc} D^2 & a & b & c \\ \hline b & 1 & & \\ c & 9 & 4 & \\ d & 49 & 36 & 16 \end{array}\right]$$

The minimum of the squared distances is attained for $D^2(a, b)$ so the first pair to be clustered will be $a \cup b$ with squared fusion cost 1

Ward clustering using LW updating formula (cont.)

$$\begin{aligned} D^2(a \cup b, c) &= \frac{2\,D^2(a,c) + 2\,D^2(b,c) - D^2(a,b)}{3} \\ &= \frac{2 \cdot 9 + 2 \cdot 4 - 1}{3} = \frac{25}{3} \end{aligned}$$

and

$$\begin{aligned} D^2(a \cup b, d) &= \frac{2\,D^2(a,d) + 2\,D^2(b,d) - D^2(a,b)}{3} \\ &= \frac{2 \cdot 49 + 2 \cdot 36 - 1}{3} = \frac{169}{3} \end{aligned}$$

$D^2(c,d)$ is not affected. Thus the new squared dissimilarity matrix is

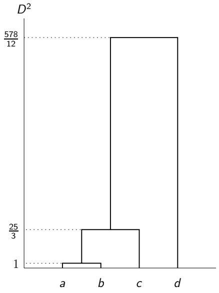
$$\left[\begin{array}{c|cc} D^2 & a \cup b & c \\ \hline c & \frac{25}{3} & \\ d & \frac{169}{3} & 16 \end{array}\right]$$

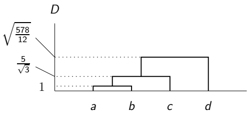
The minimum of the squared distances is attained for $D^2(a \cup b, c)$ so the next pair to be clustered will be $(a \cup b) \cup c$ with squared fusion cost $\frac{25}{3}$

Ward clustering using LW updating formula (concl.)

$$\begin{aligned} D^2((a \cup b) \cup c, d) &= \frac{3\, D^2(a \cup b, d) + 2\, D^2(c, d) - D^2(a \cup b, c)}{4} \\ &= \frac{3 \cdot \frac{169}{3} + 2 \cdot 16 - \frac{25}{3}}{4} = \frac{578}{12} \end{aligned}$$

The dendrogram can be presented either using squared or not squared fusion costs. Its topology however does not change





Computing using the R function *hclust*

Remark

Most old R packages compute the so-called **Ward1** *method although what is usually called Ward's method (from original Ward's paper), consists on a slightly distinct implementation, called* **Ward.D2** *method*
In these old packages to compute Ward.D2 method it is enough apply Ward to the squared dissimilarity matrix and then take square root of the dendrogram heights
In newer packages it is enough to choose the option ward.D2

R script

```
X<-c(1,2,4,8)
d<-dist(X) # (euclidean) distance matrix
h.ward<-hclust(d,method="ward.D2")
h.ward$height
plot(h.ward, hang=-1)
```

Monotonicity and Crossover

We say that a clustering method satisfies the **monotonicity** condition if whenever two clusters C_i and C_j are merged into a cluster C_s we have

$$D(C_k, C_s) \geq D(C_i, C_j) \qquad \forall k \neq i, j, s$$

This implies that the dendrogram cannot have crossovers

Proposition

If in the Lance-Williams updating formula the parameters α_i, α_j are nonnegative, $\alpha_i + \alpha_j + \beta \geq 1$, and either $\gamma \geq 0$ or $\max\{-\alpha_i, \alpha_j\} \leq \gamma \leq 0$, the corresponding clustering method satisfies the monotonicity condition

From the Lance-Williams table we deduce that the single, complete, average, McQuitty and Ward methods are in conditions of the proposition and therefore satisfy the monotonicity condition. Therefore their dendrograms cannot have crossovers

HAC - summary

Pros

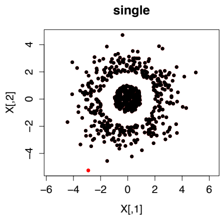
- The dendrogram provides "taxonomical information" on the clusters
- The number of clusters does not need to be defined *a priori*
- Many methods rely on a proximity matrix allowing almost any kind of resemblance notion

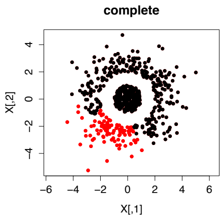
Cons

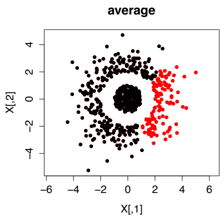
- **The aggregation of a point in a group at a given step cannot be revised, even if the point is misplaced in that group**
- Computationally demanding for large data sets since keeps track of a square matrix of order n (number of individuals): time and space complexity of most algorithms are not better than $O(n^2 \log(n))$
- Dendrogram difficult to visualize and interpret for large data sets
- Most HAC algorithms are greedy and produce suboptimal solutions

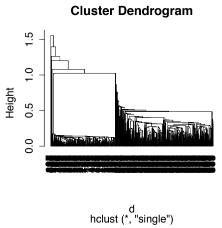
Average and Ward (specially when groups have similar sizes) are often considered among the best overall HAC methods

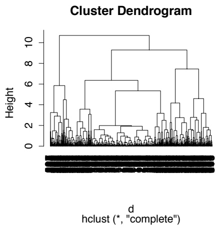
Assessment of an example with 2 clusters

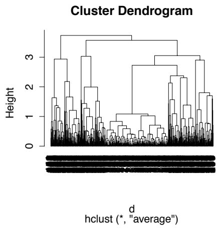




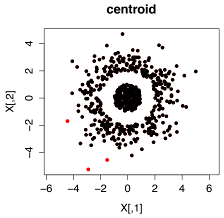


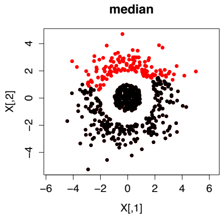


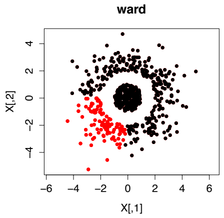


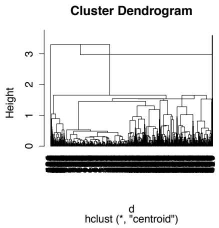


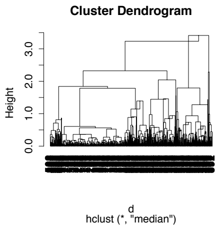
Examples (cont.) where inversions are clearly visible

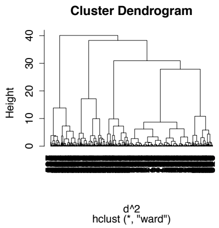




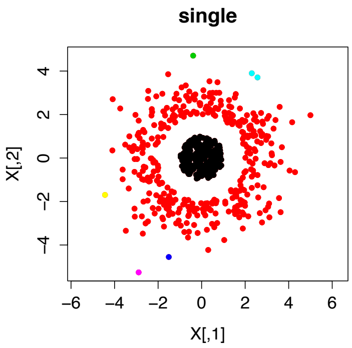


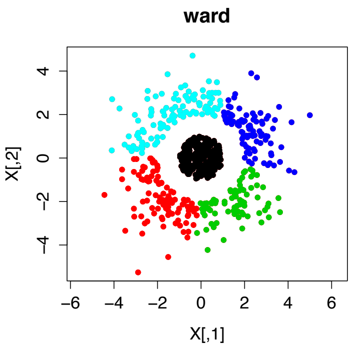






Same example with 7 clusters





A short remark on divisive hierarchical methods

It starts from a cluster consisting of all objects and successively splits every cluster until all clusters have only one element (singletons). In each step the splits can be done according to the value of a unique variable **monothetic** *(all objects of the same cluster must agree w.r.t. the that variable) or w.r.t. several variables* **polythetic**

Divisive clustering algorithms can be computationally demanding. In R can be performed using the function `diana` *of cluster package*

Example

R script

```
require(datasets)
require(cluster)
data(iris)
head(iris)
dist.iris<-dist(iris[-5])
iris.diana<-diana(d)
pltree(iris.diana)
```

Nonhierarchical clustering

To find a single partition into K clusters of a set of N objects in a p dimensional space

Two types of criteria are commonly found:

- **Global criterion** *such as to represent each cluster by a* type-object *(e.g., centroid, medoid) and to assign each object to the nearest* type-object, *optimizing some global criterion of internal homogeneity and external heterogeneity, such as, minimizing the within cluster inertia*

  Usually requires a prior estimate of the number of clusters

  Examples: K-means and K-medoids (PAM) algorithms

- **Local criterion** *such as to seek for higher density regions in data. May require to set some parameters*

  Example: DBSCAN

K-means

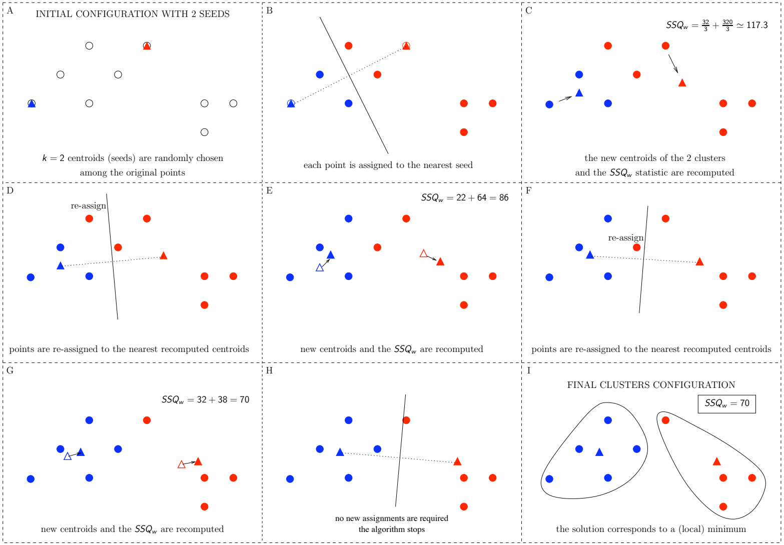
Shares the same global criterion with Ward's method: to minimize the total within-cluster sum of squares (SSQ_w) of a set of points partitioned into K clusters in a d-dimensional space

Algorithm (Lloyd)

1. *Starts with K randomly chosen initial* **seeds** *representing initial candidates to centroids;*
2. *Assigns each object to the nearest centroid*
3. *Recomputes the centroids of the K groups and use them as new seeds*
4. *Repeat through steps 2-4 until the algorithm converges, i.e., until no new assigns of points to clusters occur*

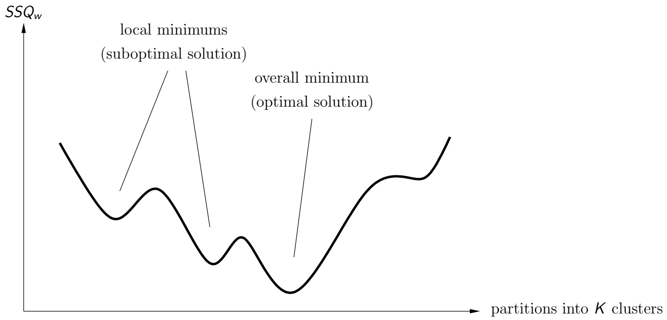
One can proved that the cost SSQ_w monotonically decreases during the course of the algorithm converging to a (possibly local) optimum

K-means algorithm


A
INITIAL CONFIGURATION WITH 2 SEEDS
$k = 2$ centroids (seeds) are randomly chosen among the original points
B
each point is assigned to the nearest seed
C
$SSQ_w = \frac{32}{3} + \frac{320}{3} \simeq 117.3$
the new centroids of the 2 clusters and the SSQ_w statistic are recomputed
D
re-assign
points are re-assigned to the nearest recomputed centroids
E
$SSQ_w = 22 + 64 = 86$
new centroids and the SSQ_w are recomputed
F
re-assign
points are re-assigned to the nearest recomputed centroids
G
$SSQ_w = 32 + 38 = 70$
new centroids and the SSQ_w are recomputed
H
no new assignments are required
the algorithm stops
I
FINAL CLUSTERS CONFIGURATION
$SSQ_w = 70$
the solution corresponds to a (local) minimum

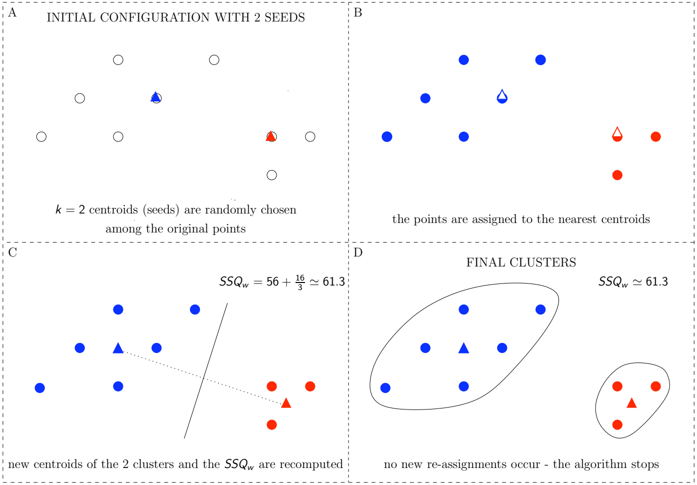
K-means: local minimum problem

The clustering solution can be highly depend on the choice of the initial position of the centroids (seeds), and may converge to a **local** *minimum*



Example

The solution found by the K-means algorithm in the previous example is not a global minimum. Actually, with new seeds the algorithm can converge to a solution that improves (i.e., lowers) the SSQ_w statistic


A
INITIAL CONFIGURATION WITH 2 SEEDS
$k = 2$ centroids (seeds) are randomly chosen among the original points
B
the points are assigned to the nearest centroids
C
$SSQ_w = 56 + \frac{16}{3} \simeq 61.3$
new centroids of the 2 clusters and the SSQ_w are recomputed
D
FINAL CLUSTERS
$SSQ_w \simeq 61.3$
no new re-assignments occur - the algorithm stops

Possible strategies to find the overall minimum?

- *To repeat the algorithm several times with randomized configurations of K seed points and keep the configuration giving the smallest SSQ_w value of the within-cluster inertia*
- *To provide initial configuration of the K seed points close to the final solution relying on some real hypothesis*
- *To provide an initial configuration of seed points issued from some hierarchical aggregation method (e.g., Ward, average), for instance, their clusters centroids*

K-means in the plane and the Voronoi diagram

Given a set of N points in the plane,

$$\{c_1, \ldots, c_N\}$$

the Voronoi diagram is defined as the partition of the plane into K convex regions, called **Voronoi cells**,

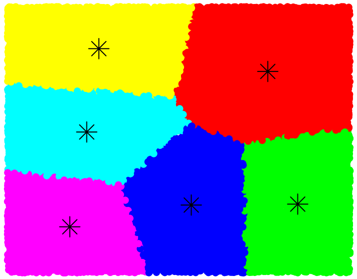
$$R_1, \ldots, R_K$$

such that each cell R_i consists of the points of the that are closest to c_i

In each step of K-means algorithm the clusters correspond to set of points of X belonging to the Voronoi cells defined be the K centroids $c_1, \ldots, c_K$, which is called Lloyd's algorithm or Voronoi iteration

The Voronoi partition and its centroids

This partition was originated applying K-means algorithm to a uniform distribution of points in the plane



K-means: summary

- *The optimizing function SSQ_w is always monotonic decreasing, i.e., the intra-group inertia decreases in each step, converging to some (possibly local) optimum*
- *The number of iterations required to converge to an optimum is usually small (usually ≈ 10 iterations are enough)*
- *Finding an optimal solution is NP-hard. Actually the time complexity is $O(n^{dK+1} \ln d)$, where K denotes the number of clusters, d the dimension and N the number of points)*
- *Tend to form convex clusters. In particular cannot detect arbitrary cluster shapes*
- *Nearby points can end in distinct classes. Groups can end empty*
- *Sensitive to noise and outliers*
- *Requires some geometric notion of center/centroid. In particular cannot be applied to categorical data. Assumes the euclidean distance*

Computing K-means with R

The K-means clustering can be performed using the R function

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

x: *numeric matrix of data*

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

nstart: *if centers is a number, how many random sets should be chosen (repeat)*

Returns a list with components:

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

centers: *A matrix of cluster centers.*

totss: *The total sum of squares SSQ*

withinss: *Vector of within-cluster sum of squares, one component per cluster*

tot.withinss: *Total within-cluster sum of squares, i.e., sum(withinss)* SSQ_w

betweenss: *The between-cluster sum of squares, i.e. totss-tot.withinss* SSQ_b

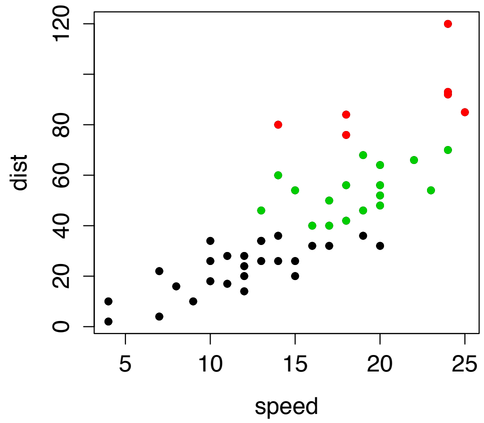
size: *The number of points in each cluster*

Example

R script

```
require(datasets)

data(cars)

?cars

head(cars)

cars.cl<-kmeans(cars, 3, nstart=100)
# 3 centers randomly chosen repeated 100 times

cars.cl

plot(cars,type=‘‘p’’,pch=16,cex=.5)

for(i in 1:50){points(cars[i,1],
cars[i,2],col=cars.cl$cluster[i], pch=16,type=‘‘p’’)}
```

Clustering result



Optimal number of clusters in K-means?

This is one of the most difficult parts in the clustering analysis
No definitive answer can usually be given

Over 30 indices were proposed to estimate the number of clusters as well to assess the internal cluster quality (goodness-of-fit). Some of the most well-known indices include:

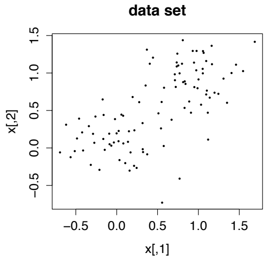
- *SSQ_w statistic*
- *Calinski-Harabasz index*
- *Silhouette coefficient*
- *Davies-Boudin*
- *Duhn index*

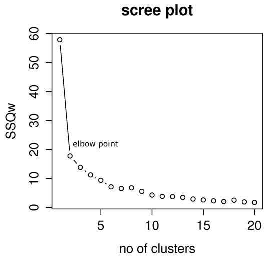
Remark

Several cluster validity indices can be computed with the R function `cluster.stats` *of* `fpc` *package or using the* `clustCrit` *or* `NbClust` `packages`

Scree plot of SSQ_w statistic

A simple method consists in analysing the variation of SSQ_w statistic, or equivalently, the percentage of explained variance SSQ_b/SSQ_t, against the number of clusters in a scree plot. This statistic is usually monotonically decreasing as the number of cluster increases. An elbow point in this plot indicating a high decrease in the SSQ_w statistic such that increasing the number of clusters only marginally improves (i.e., lowers) this statistics, may provide a good estimate for the number of clusters





Calinski-Harabaz index

Consider a set of N observations partitioned into K clusters. The **Calinski-Harabaz index** *is defined as*

$$CH(K) = \frac{N-K}{K-1} \cdot \frac{SSQ_b}{SSQ_w}$$

To maximize $CH(K)$ is equivalent to maximize SSQ_b(i.e., to maximize cluster separation) and to minimize SSQ_w (i.e., to maximize cluster cohesion). The optimal number of clusters is estimated as the number yielding the **largest** *value for $CH(K)$*

Several studies suggest that Calinski-Harabaz index is one of the internal validation indices yielding the best results
Can be computed using the R function `calinhara` *of the package* `fpc`

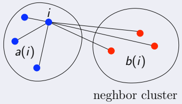
Corresponds to F-value of the one-way ANOVA *with K factors and is also known as the* **variance ratio criterion** *(VRC)*

Silhouette coefficient

- *For each observation i compute average dissimilarity $a(i)$ between i and the remaining points in its cluster*
- *For each one of the other clusters compute the average dissimilarity from i to the points of that cluster and take the minimum $b(i)$ of these average dissimilarities*
- *Silhouette coefficient of observation i is then defined as*

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

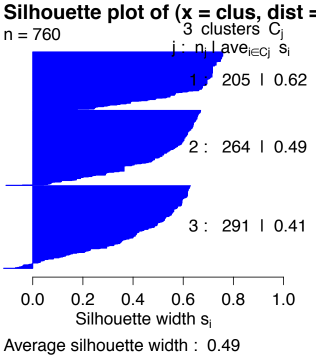
The cluster for which the minimum $b(i)$ is attained, i.e., the cluster with lowest average dissimilarity w.r.t to observation i, is called the **neighbor cluster** *to i*

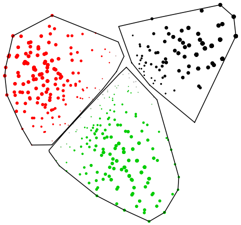


Interpretation of silhouette coefficients

1. *Silhouette coefficients range between -1 and 1.*
2. *Observations with silhouette coefficients* **close to one are very well classified**
3. *Observations with silhouette coefficients* **close to zero probably lie between clusters**
4. *Observations with* **negative silhouette coefficientes are probably misplaced in their clusters**

Silhouette plot





In the figure on the right the dot sizes are proportional to their silhouette coefficients. Larger dots lie in core regions of the clusters whereas smaller dots lie in border regions or between clusters

Average silhouette width

The **average silhouette width** *(ASW) is defined as the average of the silhouette coefficients for all observations*

- *It assess both* **cluster cohesion** *and* **cluster separation**
- *It increases with cluster separation (higher $b(i) - a(i)$ differences) and cluster tightness (smaller $a(i)$ and $b(i)$ values)*

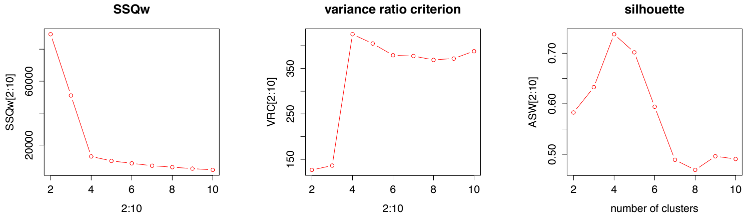
Range of ASW

- *between 0.71 and 1.0: a* **strong structure** *has been found*
- *between 0.5 and 0.7: a* **reasonabble strucuture** *has been found*
- *between 0.26 and 0.5: the* **structure is weak** *and can be artificial*
- *below 0.25:* **no substantial structure** *has been found*

The optimal number of clusters can be estimated maximizing the ASW

Number of clusters?

Applying the criteria SSQ_W statistic, VRC and ASW to the Ruspini data, a popular dataset in clustering analysis, all criteria agree on 4 clusters

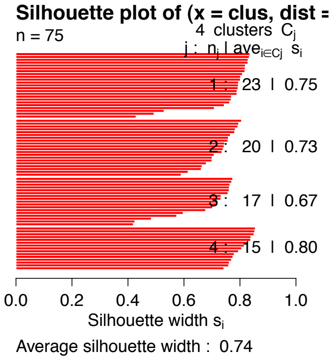


Remark

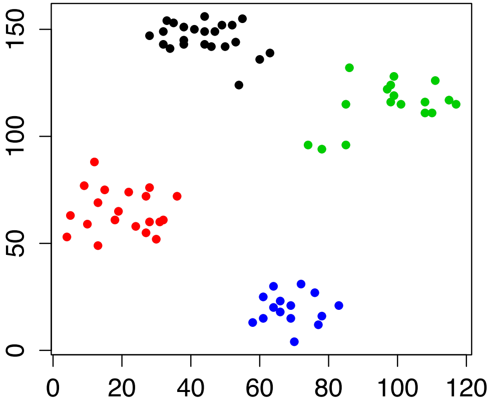
A monotone increase in the VCR criterion over the number of clusters k indicates that no group structure is present. A monotone decrease suggests hierarchical relationships

(Internal) cluster validity

The average of the silhouette widths of the previous example is close to .75 suggesting that a strong clustering structure was found in Ruspini data. Since all silhouette coefficients are above .4 no points are misplaced in their clusters



Ruspini plot into 4 clusters using the K-means algorithm



K-medoids clustering

- *The* **optimization problem** *is to find K points in the data set, called* **medoids***, such that the sum of distances of each point to the nearest medoid is minimal*
- *A* **medoid** *also called* **exemplar** *or* **centrotype** *is an element of the set whose average distance to every other element in the set is minimal, i.e., the most centrally located point in the set*
- *The algorithm uses medoids to represent clusters instead of centroids, since are less sensitive to outliers (why?)*
- *Can work with more general distance measures*
- *The most well known algorithm for K-medoids is the* greedy *algorithm PAM (Partitioning Around Medoids). May fail to converge to the overall optimum*
- *For large sets, there are the more efficient algorithm CLARA (Clustering LARge Applications), which applies PAM to samples, and CLARANS (Randomized CLARA)*
- *To handle categorical data there is the variant K-mode algorithm*

Partition around medoids (PAM) algorithm

Algorithm

1. *Start with a set M of K initial medoids randomly chosen, m_k, $k = 1, \dots, K$;*
2. *Set $O = X \setminus M$ containing the non-medoid points, o_j, $j = 1, \dots, \ell$;*
3. *For every pair (k, j) compute the* total swapping cost *of replacing the medoid m_k by the non-medoid o_j,*

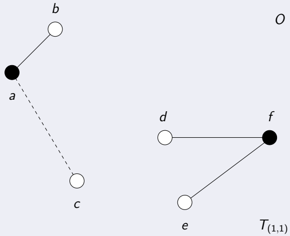
   $$T_{k,j} = \sum_{i \neq j} T_{k,j,i}$$

   where $T_{k,j,i}$ is the swapping cost accounted for object $o_i \neq o_j$, and selects the pair (k, j) that minimizes $T_{k,j}$
4. *If $T_{k,j} < 0$ perform the swap, update the sets M and O and return to step 3;*
5. *If $T_{k,j} \geq 0$, the objective cannot be improved and the algorithm stops*

Example of a swapping step

$M = \{a, f\} = \{m_1, m_2\}$

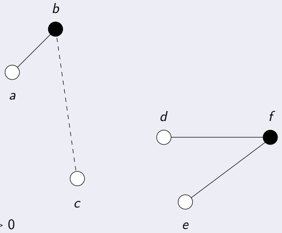
$O = \{b, c, d, e\} = \{o_1, o_2, o_3, o4\}$



$a \in M \leftrightarrow b \in O$

$(1, 1)$

$\dashrightarrow$



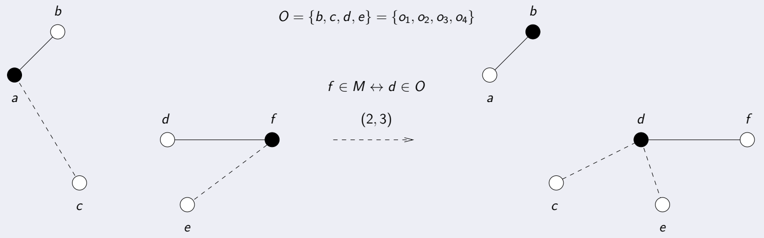
$T_{(1,1)} = T_{(1,1,2)} = d(c, b) - d(c, a) > 0$

Total swapping cost is positive - does not improve the objective function (sum of the edges lengths)

Example of a swapping step (cont.)

$M = \{a, f\} = \{m_1, m_2\}$

$O = \{b, c, d, e\} = \{o_1, o_2, o_3, o_4\}$



$T_{(d,f)} = T_{(d,f,c)} + T_{(d,f,e)} = d(c,d) - d(c,a) + d(e,b) - d(e,f) < 0$

Total swapping cost is negative - it improves the objective function

It can be seen that no other swap can improve this result. The sets M and O are then updated to $M = \{b, d\}$ and $O = \{a, c, e, f\}$ Continuing to perform the swaps, no further improvement occur. The algorithm stops yielding the clusters

$$\mathcal{C}_1 = \{a, b\}, \qquad \mathcal{C}_2 = \{c, d, e, f\}$$

Computing *K*-medoids with R

The partition around medoids clustering algorithm can be performed using the R function of the `cluster` *package*
(it accepts metrics distinct from the euclidean metric)

pam(x, k, diss = inherits(x, "dist"), metric = "euclidean", ...)

x: *data matrix or dataframe or a dissimilarity matrix or object*

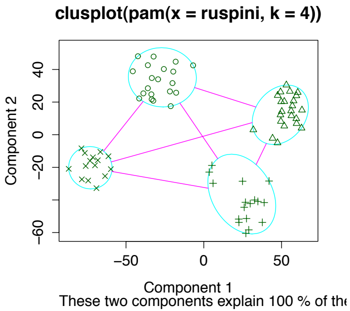
k: *is the number of clusters*

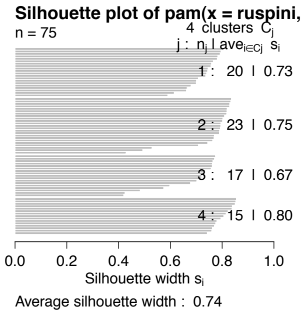
...
Returns an object of the class pam *consisting of a list with several components*

R script

```
require(datasets)
data(ruspini)
clus.PAM<-pam(ruspini,4) # perform the clustering with 4 clusters
clusplot(clus.PAM) # displays the clusters and the silhouette plot
cls<-clus.PAM$cluster # vector of classes for all elements
mdd<-clus.PAM$medoids # vector of coordinates of the medoids of the
final 4 clusters
```

Ruspini data 4-partition using PAM

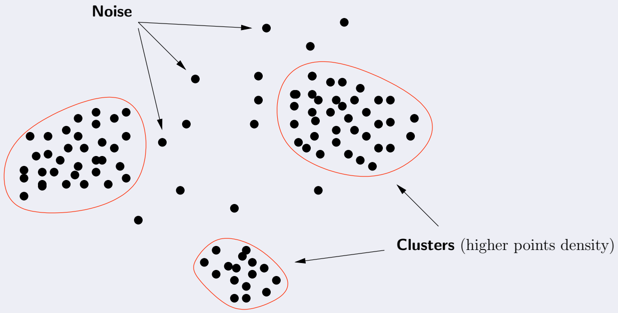




The R `clustplot` *function displays the projection of the centered variables onto the principal factorial plane of the PCA, allowing to visualize high dimensional data*

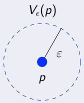
Density-based clustering: DBSCAN

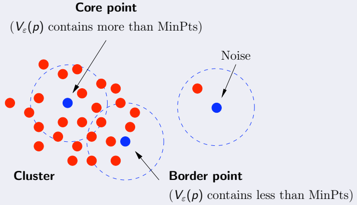
The DBSCAN algorithm implements the very natural idea that clusters are regions with high density of points separated by regions with low density of points (noise)



How to define a region with a minimum density of points?

- *Density is the number of points per area unit. A naïve idea could be to define a cluster $\mathcal{C}$ as the region where each point p has an neighborhood of radius ε, $V_\varepsilon(p)$ with at least a pre-defined minimum number of points MinPts of $\mathcal{C}$*
- *The definition works for* **core points** *but may fail for* **border points**, *where the number of points in the ε-neighborhood can be considerable lower. A border point in a cluster should however possess at least one core point in its ε-neighborhood*
- *Points in low density regions with no core points in their ε-vicinity are considered* **noise** *w.r.t.* $(\varepsilon, \mathrm{MinPts})$

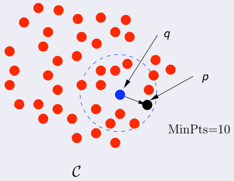



Core point
($V_\varepsilon(p)$ contains more than MinPts)
Noise
Cluster
Border point
($V_\varepsilon(p)$ contains less than MinPts)

Directly density reachable relation

We say that a point p is **directly density reachable** *from a point q w.r.t. $(\varepsilon, \mathrm{MinPts})$ and we denote by $p \leftarrow q$, if*

- $p \in V_\epsilon(q)$
- $|V_\epsilon(q)| \geq \mathrm{MinPts}$



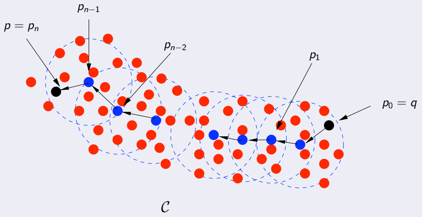
Remark

The relation directly density reachable is not symmetric (in general). Why? Give an example.

Density reachable relation

We say that a point p is **density reachable** *from a point q w.r.t. $(\varepsilon, \mathrm{MinPts})$ if there is a sequence $q = p_0, p_1, \ldots, p_n = p$ s.t. each p_{i+1} is directly reachable from p_i, i.e.,*

$$p = p_n \leftarrow p_{n-1} \leftarrow p_{n-2} \leftarrow \cdots \leftarrow p_1 \leftarrow p_0 = q$$

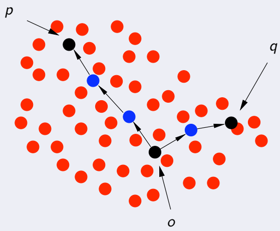


Remark

This definition extends the previous one. Moreover, it is not symmetric and may fail if both points p and q lie at the boundary a cluster

Density connected cluster

We say that points p and q are **density connected** *w.r.t.* $(\varepsilon, \mathrm{MinPts})$ *if there is a point o such that p and q are both density reachable from o*



$\mathcal{C}$

Clusters and noise

Definition (cluster)

A **cluster** *w.r.t.* $(\varepsilon, \mathrm{MinPts})$ *is a region* $\mathcal{C}$ *of pairwise density connect points that cannot be enlarged with new points (maximal)*

Remark

It can be proved that a cluster $\mathcal{C}$ *w.r.t.* $(\varepsilon, \mathrm{MinPts})$ *consists exactly of the set of points that can be density-reachable from any of its core points, i.e., from any of its points* $q \in \mathcal{C}$ *such that* $|V_\varepsilon(q)| \geq \mathrm{MinPts}$. *Each one of this core points can be considered as a seed defining the cluster.*
Points of the cluster where $|V_\varepsilon(q)| < \mathrm{MinPts}$ *are called* **border points**

Definition (noise)

Let $\mathcal{C}_1, \ldots, \mathcal{C}_k$ *be the clusters of a set* X *w.r.t.* $(\varepsilon, \mathrm{MinPts})$. *Points not belonging to any cluster are called* **noise** *or* **outliers**, *i.e.,* **noise points** *belong to the set*

$$X \setminus (\mathcal{C}_1 \cup \cdots \cup \mathcal{C}_k)$$

The algorithm

Algorithm (DBSCAN)

Assume $(\varepsilon, \mathrm{MinPts})$ *given*

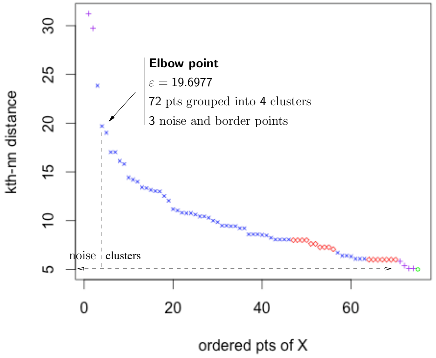
1. *Consider a point* p *in* X*;*
2. *If* p *has not been visited and* p *is a core point, i.e.,* $|V_\varepsilon(p)| \geq \mathrm{MinPts}$, *determine the cluster formed by all points density reachable from* p*;*
3. *Return to step 1 until all points of* X *have been visited;*
4. *Points that are not density reachable from any core point are considered noise points*

The R function `dbscan` *of package* `fpc` *implements the DBSCAN algorithm*

A simple estrategy to estimate ε and MinPts

- *Choose* $\mathrm{MinPts} \geq \dim X + 1$. *The default for the R* `dbscan` *function is* $\mathrm{MinPts} = 5$. *For two dimensional databases* $\mathrm{MinPts} = 4$ *is usually appropriated.*
- *For each element* $x \in X$ *determine the* k*-th nearest neighbor* y *of* x*, with* $k = \mathrm{MinPts}$. *Let* $\varepsilon_x = d(x, y)$ *be the distance from* x *to* y. *Then the* ε_x*-neighborhood of* x *will have at least* $k = \mathrm{MinPts}$ *elemens, with equality for most* x*;*
- *Plot the elements by decreasing values of* ε_x *called* sorted k-th nearest neighbor distance function. *This graph gives an idea of the distribution of densities on the dataset. If the dataset has clusters with similar densities the graph should exhibit a steady decrease for cluster points (specially core points), while for noise points, having their* k*-th nearest neighbor considerable farther may produce an abrupt change.*
  An elbow point in this plot usually provides a good candidate for the parameter ε. *Points on the left of this elbow point will (usually) correspond to mostly to noise points and border points;*
- *If several points are candidates, smaller choices of the* ε *parameter yield more tighter clusters although with more unclassified points (i.e. noise points)*

The sorted *k*-th distance function for the Ruspini example

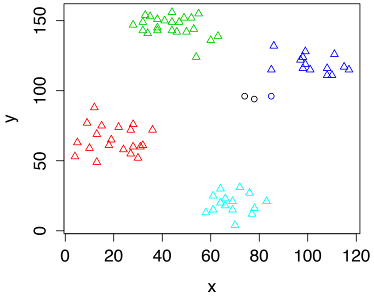


ε values in the y-axis corresponding to blue, red, purple and green points yielding 4, 5, 3 and 1 clusters respectively. Try values for the ε parameter in each one these cases and interpret the results

The partition obtained with DBSCAN algorithm

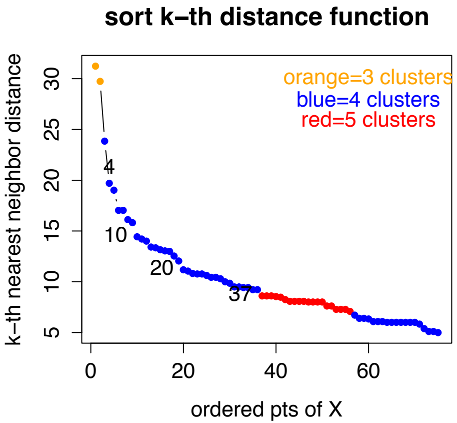
`dbs.ruspini<-dbscan(ruspini, MinPts=5,` $\varepsilon = 19.69$`)`

```
plot(dbs.ruspini,ruspini)
```



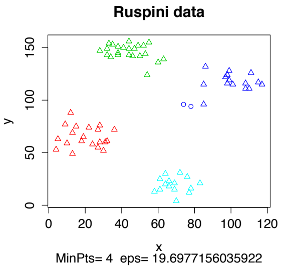
(core points are displayed as colored triangles, border points as colored circles and noise points as black circles)

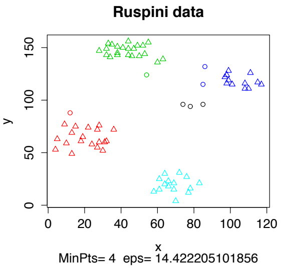
Ruspini example revisited with $\mathrm{MinPts}=4$ and 4 ε values

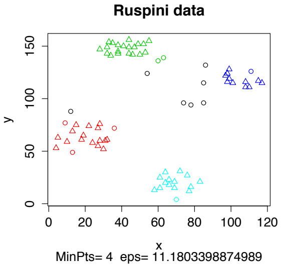


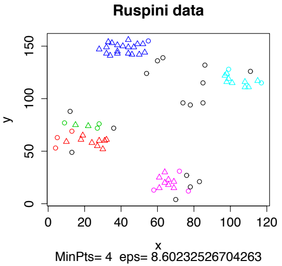
The points marked with labels, 4, 10, 20 and 37 correspond to $\varepsilon = 19.69$, $\varepsilon = 14.42$ (elbow point), $\varepsilon = 11.18$ and $\varepsilon = 8.60$

Clustering outputs corresponding to the 4 ε values









Summary of density-based clustering

Pros:

- *Discover clusters with arbitrary shapes*
- *Good efficiency in large datasets ($o(n \log n)$ time complexity)*
- *Handles noise points and is robust to outliers*
- *The number of clusters is not required* a priori

Cons:

- *The optimal parameters ε and MinPts have to be estimated*
- *Can be difficult or impossible to find a good combination of parameters $(\varepsilon, \mathrm{MinPts})$ when the data set contains regions of very distinct density*

Non-hierarchical clustering - summary

Pros

- **Can reallocate an individual that was misplaced in its cluster**
- Computationally efficient (K-means)
- Can improve the objective function obtained with some hierarchical methods (e.g., K-means vs Ward)

Cons

- The number of clusters (or some other parameters) has to be estimated *a priori*
- No taxonomic type of relationship between clusters is obtained and no dendrogram is produced
- Some methods work only with “geometric” data and may require the euclidean distance

4. COMPARING PARTITIONS

Motivation

Reasons to compare the extent to which two partitions coincide:

- *Several clustering analyses of the same data can be done using distinct meaningful combinations of clustering methods and resemblance notions;*
- *Clustering analyses having a high degree of agreement may suggest that the common clustering structure produced by these methods is robust;*
- *If the clustering structure is known* a priori *and it is important to assess how well the clustering method was able to reproduce this structure;*

It is very difficult to match each cluster of a partition with the correct cluster of the other partition (if not impossible)

The usual way is to compute the number of pairs of individuals that both clustering methods agree to assign in the same/distinct class

Rand index

Assume that N individuals are classified by two distinct clustering methods. The total number of pairs of individuals is $\binom{N}{2}$

A: number of pairs classified in the same class in both partitions

D: number of pairs classified in distinct classes in both partitions

B: number of pairs classified in the same [distinct] class for the first [second] partition

C: number of pairs classified in the distinct [same] class for the first [second] partition

The Rand index is a simple concordance index used as external validition index and is defined as

$$RI = \frac{A+D}{\binom{N}{2}} = \frac{A+D}{A+B+C+D}$$

where A+D is number of agreements for both partitions
It ranges from 0 (total disagreement) *to 1* (total agreement)

Rand index

To each partition of a set of N individuals, $x_1, \dots, x_N$ we associate a binary vector of length $\binom{N}{2}$, where the component corresponding to pair (i, j) is equal 1 if x_i and x_j are assigned in the same class and 0 otherwise
Then the Rand index of the two partitions is nothing more than the simple matching index between the binary vectors associated to these partitions

Note that the number of groups in each partition can be distinct

Rand index: example

$X = \{a, b, c, d, e, f, g\}$

Partition 1: a b e | c | d f *Partition 2: a c | b d| e f*

$$\left[\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline b & 1 & \cdot & \cdot & \cdot & \cdot \\ c & 0 & 0 & \cdot & \cdot & \cdot \\ d & 0 & 0 & 0 & \cdot & \cdot \\ e & 1 & 1 & 0 & 0 & \cdot \\ f & 0 & 0 & 0 & 1 & 0 \end{array}\right] \qquad \left[\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline b & 0 & \cdot & \cdot & \cdot & \cdot \\ c & 1 & 0 & \cdot & \cdot & \cdot \\ d & 0 & 1 & 0 & \cdot & \cdot \\ e & 0 & 0 & 0 & 0 & \cdot \\ f & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Contigency table between partition 1 and partition 2:

	1	0	
1	0	4	4
0	3	8	11
	3	12	15

$$\Rightarrow \qquad RI = \frac{0+8}{15}$$

Correction for chance: adjusted Rand index

The Rand index present some issues:

- *The expected value of Rand index between random partitions with the same number of elements in each class is not constant (e.g. equal to 0)*
- *It is highly depend on the number of clusters. For instance, even if the clusterings are independent, the index will converge to 1 as the number of clusters increase*

To correct these issues the so-called **adjusted Rand index** *as proposed*

$$ARI = \frac{RI - Expected\ RI}{Max - Expected\ RI} = \frac{\binom{N}{2}(A+D) - U}{\binom{N}{2}^2 - U}$$

where $U = (A+B)(A+C) + (C+D)(B+D)$*, assuming the generalized hypergeometric distribution as null hypothesis (keeping the clusters sizes) Gives value 0 for independent clusterings and 1 for identical clusterings. May give negative values indicating quite low agreement. More difficult to interpret than the more simple Rand index*

Computing the adjusted Rand index in R

To compute the adjusted Rand index of the two partitions in 3 classes,

$$\mathcal{P}_1 : \quad a\ b\ e \mid c \mid d\ f \qquad\qquad \mathcal{P}_1 : \quad a\ c \mid b\ d \mid e\ f,$$

we encoded these partitions as vectors

$$(1,1,2,3,1,3), \qquad (1,2,1,2,3,3),$$

representing the classes of the elements a, b, c, d, e, f

R script

```
require(mclust)
adjustedRandIndex(c(1,1,2,3,1,3),c(1,2,1,2,3,3))
-0.2962963
# 2 randoms vectors of length 100 consisting
# of elements in 1,...,10
p1<-sample(1:10,100,replace=TRUE)
p2<-sample(1:10,100,replace=TRUE)
adjustedRandIndex(p1,p2) # should be approximately zero!
```