# MAT 3373
# Methods of Machine Learning

# Chapter 4
# Clustering

P. Boily (uOttawa)

Fall – 2023

# Outline

4.1 – Clustering Overview (p.6)
- Unsupervised Learning (p.7)
- Clustering Framework (p.11)
- Philosophical Sidebar (p.23)

4.2 – $k-$Means Clustering (p.27)
- $k-$Means Algorithm (p.31)
- Example: Gapminder Dataset (p.37)

4.3 – Hierarchical Clustering (p.41)
- AGNES/DIANA Algorithms (p.42)
- Linkage Strategies (p.47)
- Example: Gapminder Dataset (p.50)

# Outline

4.4 – Density-Based Clustering (p.57)
- DBSCAN (p.60)
- Advantages and Limitations (p.73)
- Example: Gapminder Dataset (p.76)

4.5 – Clustering Evaluation (p.79)
- Clustering Assessment (p.80)
- Clustering Quality Measures (p.92)
- Internal Validation (p.95)
- Relative Validation (p.106)
- External Validation (p.116)
- Model Selection (p.121)

# Outline

4.6 – Spectral Clustering (p.123)
- Graphs and Cuts (p.127)
- Similarity, Degree, and Laplacian Matrices (p.135)
- SC Algorithm (p.139)
- Practical Details and Limitations (p.145)
- Example: Gapminder Dataset (p.147)

4.7 – Probability-Based Clustering (p.151)
- Mixture Models (p.152)
- Generative Process (p.156)
- Gaussian Mixture Models (p.159)
- EM Algorithm (p.162)
- Example: Gapminder Dataset (p.169)

# Outline

The Rest of the Clustering Picture (p.171)

References (p.172)

## Main References:

- *Data Understanding, Data Analysis, and Data Science*, chapter 22.
- *Data Understanding, Data Analysis, and Data Science*, chapter 19.

# 4 – Clustering

In Chapter 1 (*Machine Learning 101*), we provided a (basically) math-free general overview of machine learning.

In this chapter, we begin our mathematical treatment of **unsupervised learning**, with a focus on **clustering**, and on **related issues** and **challenges** (including **clustering validation**).

This chapter is independent of Chapters 2 (*Regression and Value Estimation*) and 3 (*Classification and Supervised Learning*), but could be studied in parallel with Chapter 5 (*Additional Topics*).

# 4.1 – Clustering Overview

**SL** methods are often formally presented as an offshoot of **regression analysis**, and their performance are **easy to evaluate**; they have been studied extensively and often form the **backbone** of ML training.

**Unsupervised learning (UL)** tasks are not usually presented with the same depth, primarily due to the **vagueness** which infect their core – some of the concepts are defined **ambiguously**; results validation can be **elusive**, and the **actionable** applications of the outcomes are not always **clear**.

The interest in UL methods and tasks (**clustering** and **segmentation**, **association rules mining**, **link profiling**, etc.) is increasing with the recent advances in **AI** and **ML** research.

# 4.1.1 – Unsupervised Learning

In SL, we differentiate a dataset's **response variables** $Y_1, \ldots, Y_m$ from its **predictor variables** $X_1, \ldots, X_p$. Variables can be **predictors** in one context and **responses** in another.

UL tasks forego the **responses** $\Longrightarrow$ **prediction** is off the table, but variables are not necessarily **removed from the dataset**.

The objective is to **identify** and **uncover interesting insights** about the dataset and the system that it represents, such as:

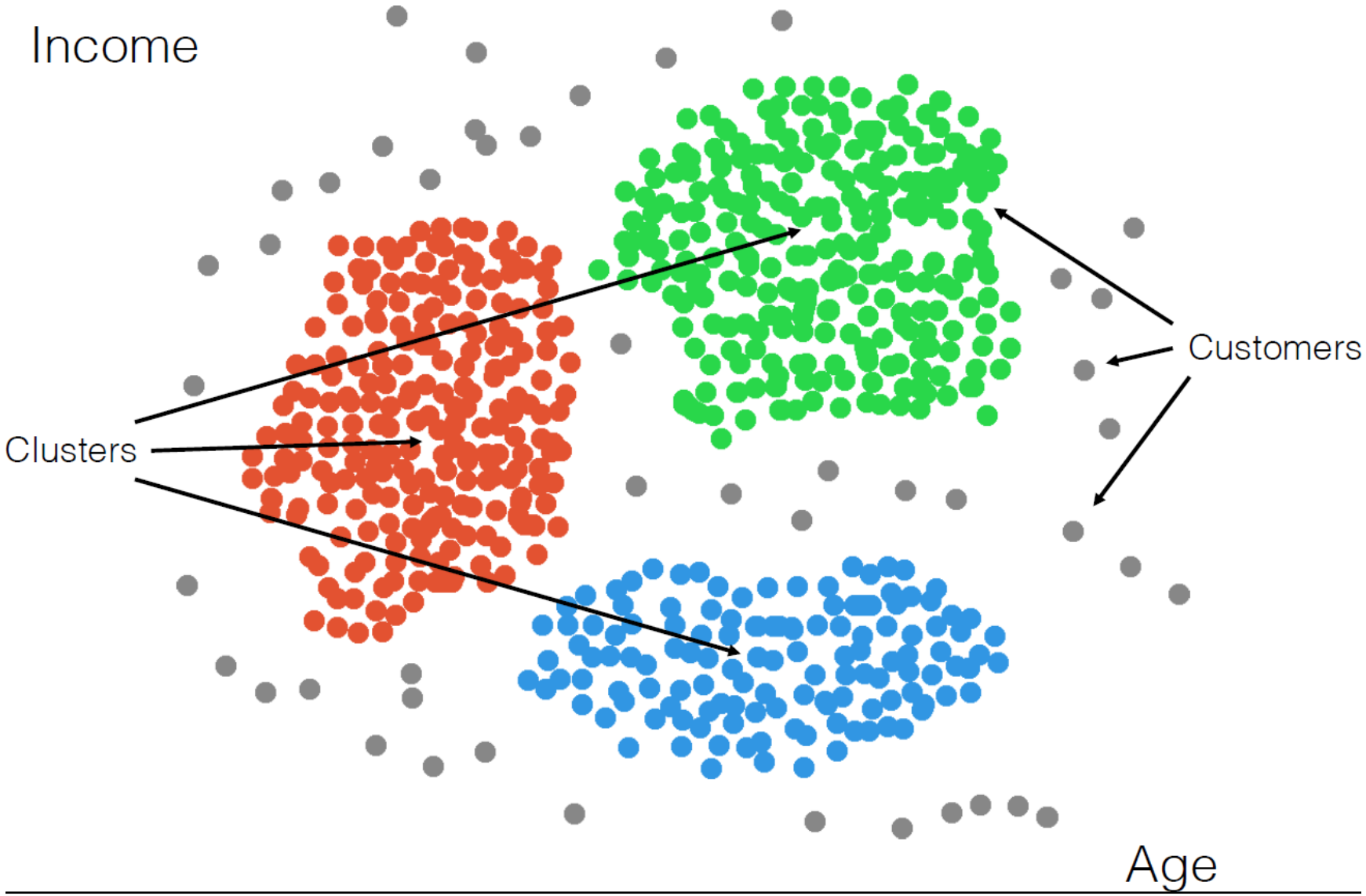
- informative ways of **visualizing** the dataset;
- highlighting **subgroups** among the dataset's variables or observations;
- finding **links** between variables.

We can make a variety of **quantitative statements** about a dataset, at the **univariate** level. For instance, we can

- compute **frequency counts** for the variables;
- identify **measures of centrality** (mean, mode, median), and
- measure the **dispersion** (range, standard deviation), among others.

At the **multivariate** level, the various options include $n-$**way tabulations**, **correlation analysis**, and **data visualization**, among others.

While these can provide insights in simple situations, datasets with a **large number of variables** or with **mixed types** (categorical and numerical) might not yield to such an approach. Instead, insights might come in the form of **aggregation** or **clustering** of **similar** observations.


Income
Customers
Clusters
Age

The **number** of latent groups is generally a true unknown of the problem.

The **subjectivity** of UL tasks may seem to be an insurmountable flaw – when looking for latent groups in a dataset, different analysts may:

- obtain a **different number** of groups, or
- assign **different observations** to each groups,

without one of them being necessarily "**wrong**"; but they may produce **sub-optimal** groups, as we will discuss shortly.

Still, clustering is a popular analytical task, in part because it is **much easier** to obtain **unlabeled data** than it is to obtain **labeled data** (a requirement of SL methods).

# 4.1.2 – Clustering Framework

**Clustering** consists of a large family of algorithms and methods used to discover **latent groups** in the datasets.

These are **natural groups** that exist but have not been **identified** or **labeled** as such (yet).

Clustering is a **subjective** analytical task; unlike classification and regression, clustering analysis does not have as "simple" a goal as **predicting a response** for a new observation based on **historical data patterns**.

Furthermore, there is no "**solution key**" against which to **validate** analysis results... rather frustrating, isn't it?

# Applications

- finding **subgroups** of breast cancer patients based on **gene expression measurements** or **socio-demographic characteristics** in order to better understand the disease and potential treatment side-effects;
- **grouping** products in an online shop based on **ratings** and **reviews** assigned by customers in order to make **product recommendations**;
- finding documents that apply to **search queries**, and finding **similar queries** to those entered by a user to increase the odds of finding the documents **they are really looking for**;
- identifying **population segments** to test **vaccination incentives**, etc.

# What is a Cluster?

A **cluster** is a **subset** of observations that are **similar**, according to some **measure of similarity**. Ideally, a cluster's observations should also be **dissimilar** to other clusters' observations.

Distance to points in own clusters (left, smaller is better) and to points in other clusters (right, larger is better).

Clusters do not necessarily need to be **disjoint** (**hard** clustering) – in some cases, it might be sufficient to quantify the **likelihood** or the **degree** to which an observation belongs to a cluster (**soft** clustering).

The choice of a **similarity/dissimilarity** measure is (for the most part) entirely **subjective**; there are contexts for which **proximity** could be used as a decent proxy for **similarity**, and others where it could not.

Even in the former case, a **distance measure** (metric) has to be selected; there are **infinitely many** choices..

Without **domain-specific considerations** (this requires thorough data and context understanding), the choice of measure might as well be **arbitrary**; but understanding the data and the context is still **no guarantee** that all reasonable analysts would agree on such a measure.

For instance, in any group of human beings, which of:

> age, ethnic background, gender, postal code, sexual orientation, linguistic abilities, mathematical skills, career, social class, political affiliation, operating system preferences, educational achievements, hockey club fandom, etc.

is responsible for separating its members into "**US**" vs. "**THEM**" groups? Is it some **combination**? Are the groups **fixed**? Is everybody in the "$\mathbf{US}_{\mathbf{age}}$" group also in the "$\mathbf{US}_{\mathbf{gender}}$" group?

We could bypass the problem by creating **more** groups; given an age group and gender, we could create the clusters: "same age group and gender" (**US**), "same age group, different gender" ($\mathbf{THEM}_1$), "different age group, same gender' ($\mathbf{THEM}_2$)', "different age group and gender" ($\mathbf{THEM}_3$).

And so on, and so on... but how many "**THEM**" groups are too many for analysts or human brains to process?

# Clustering Approaches

There are **tons** of clustering algorithms (100+), found in six main families:

- **partitional** ($k-$means and variants, CLARA, etc.);
- **hierarchical** (AGNES, DIANA, BIRCH, etc.);
- **density-based** (DBSCAN, DENCLUE, OPTICS, etc.);
- **connectivity-based** (DBSCAN (?), spectral and variants, etc.);
- **grid-based** (GRIDCLUS, STING and variants, etc.);
- **model-based** (mixture models, latent Dirichlet, expectation-maximization, etc.).

Some modifications are required when dealing with **Big Data**, for **high-dimensional data**, or for specific **types of datasets** (stream data, network data, categorical data, text and multimedia data, time series data, etc.).

**Ensemble methods**, which combine various clustering results, are useful.

# Distance, Similarity, and Dissimilarity

The choice of how to interpret and compute **similarity** between observations is up to the analysts, but it must take on:

- **large** values for **similar** objects, and
- **small** (or even **negative**) values for **dissimilar** objects.

Some **similarity** measures are derived from **distance (metrics)** functions $d : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}_0^+$, with special properties:

1. $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$;
2. $d(\mathbf{x}, \mathbf{y}) \geq 0$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$;
3. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$;
4. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$.

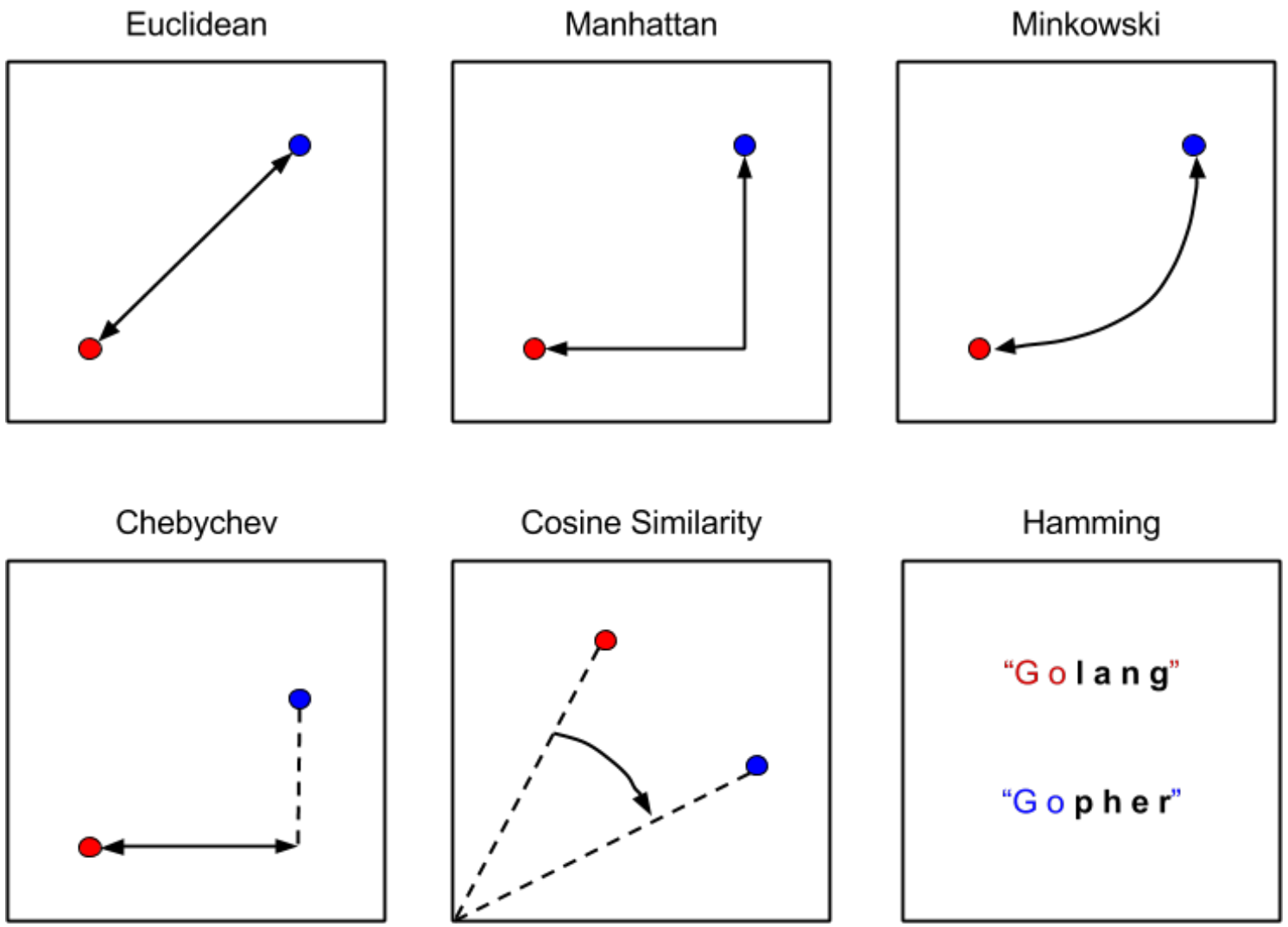
Commonly-used distances include the:

- **Euclidean** distance $d_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$;
- **Manhattan** distance $d_1(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1$;
- **supremum** distance $d_\infty(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_\infty$;
- more general **Minkowski** distance $d_p(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p$, for $p \geq 1$;
- and more esoteric distances (**Jaccard**, **Hamming**, **Canberra**, **cosine**, **mixed**, etc.).

Given a distance $d$, a common construction is to define the **associated similarity measures**

$$w = \ell - d, \quad w = \exp(-kd^2), \quad \text{or} \quad w = \frac{1}{1+d}.$$

Note that there are **similarities** that **cannot** be derived from **distances**, however.


Euclidean
Manhattan
Minkowski
Chebychev
Cosine Similarity
Hamming
“G o l a n g”
“G o p h e r”

# Data Transformations for Clustering

Prior to clustering, the data must be **scaled** and (potentially) **centered** so that none of the variables unduly influence the outcomes (e.g., so that **we do not have to compare apples with oranges**).

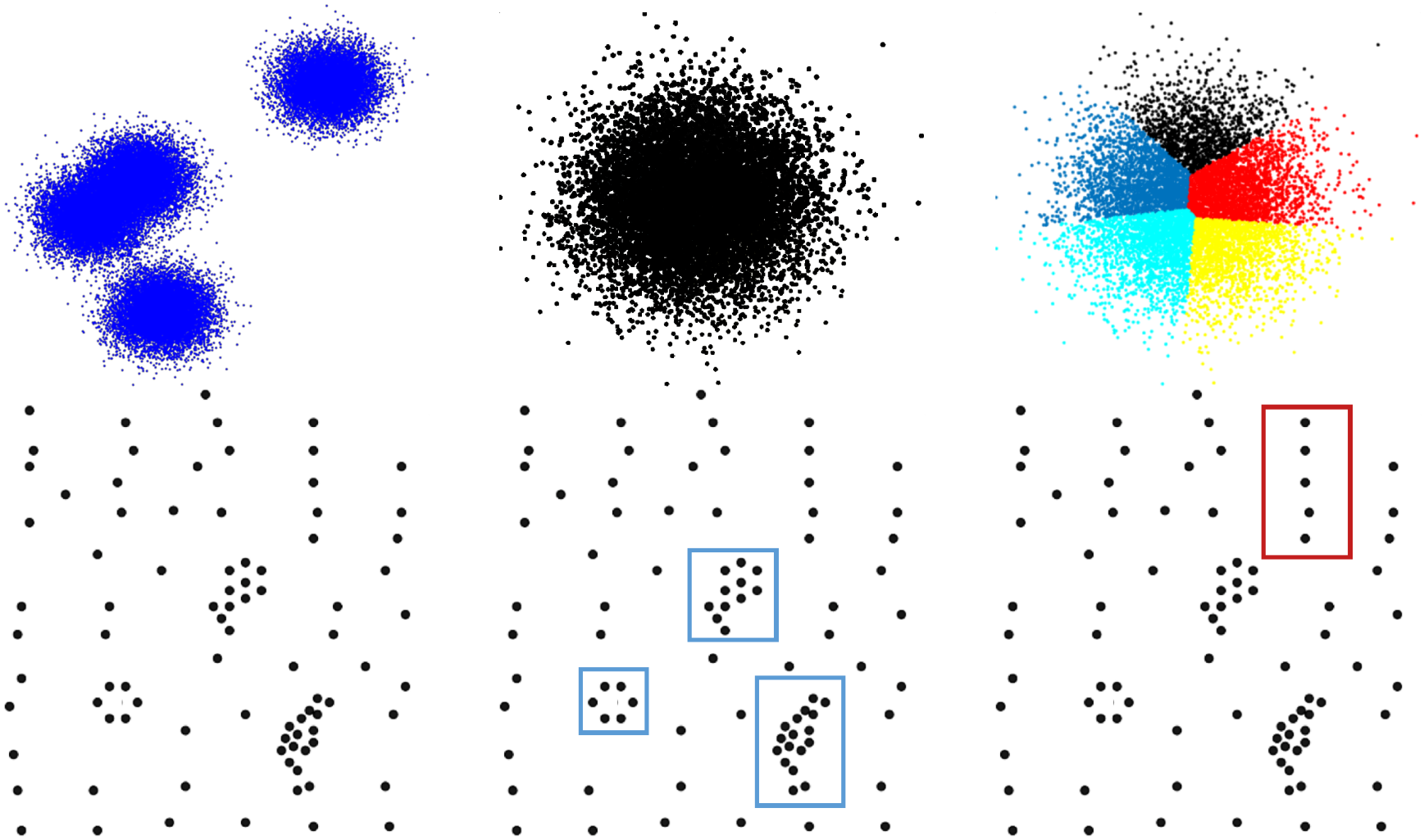
If age in years and height in cm are dataset variables, a 10-unit difference in age is likely to be **more significant** than a 10-unit difference in height.

Putting everything on a $(\min, \max)$ scale guarantees that **relative differences** play the important role.

There are other ways to scale the data (**standardization**, etc.) $\Longrightarrow$ the scaling approach can have an impact on the clustering results.

# Challenges

- In many instances, the underlying assumption is that **nearness of observations** (in whatever metric) is linked with **object similarity**, and that **large distances** are linked with **dissimilarity**;
- the **lack of a clear-cut definition** of what a cluster actually is makes it difficult to validate clustering results;
- various clustering algorithms are **non-deterministic**;
- the **number of clusters** cannot usually be known before the analysis;
- even when a cluster scheme has been accepted as valid, a **cluster description** might be difficult to come by;
- most methods will find clusters in the data **even if there are none**;
- once clusters have been found, it is tempting to try to "**explain**" them, but that is a job for **domain experts**.



# 4.1.3 – Philosophical Sidebar

In the context of general AI, clustering provides a basic way for **intelligences** to structure their experience of the world.

Clustering allow these machines to identify **object instances** and, on this basis, to identify or define **types of objects** by grouping together the object instances they have discovered.

We can then view creating **concepts** as the fundamental purpose of clustering; these concepts allow an **intelligent agent** to:

- work in **shorthand** when dealing with objects (easier to deal with 10 '**cats**' than with 10 **unique objects**);
- **make assumptions** about the object instances in a cluster associated with the concept (if an object is a '**cat**', then that object probably likes '**fish**').

If some "**ground truth**" exists about what **should** be **clustered together** and/or **counted as a concept**, then choosing one algorithm over another may lead to a "**better**/**worse**" reflection of the former.

What counts as a **ground truth**? We often assume that there is a **natural** and/or **objective** way in which objects are **properly grouped** in nature.

The fact that such a **ground truth** is not known by the clustering agent does not mean that it does not exist, or that it cannot be sought out (e.g., **scientific method**).

Even if the existence of **natural kinds** is rejected, some clustering results could still **achieve a stated goal** to a greater or lesser extent than others.

This does appear to be **more subjective**, in the sense that the **goal**, and the success of the outcome relative to the goal, are both defined by **individual(s)**, rather than being **independent** of them.

It is not unusual for people to create **contextual definitions** of what counts as '**good**': a "**good meal**" at a **campsite** is not the same as a "**good meal**" at a **four-star resort**; but this does not mean that there is no "**bad meal**", or that anything counts as a "**good meal**". Context matters.

Even if there were some **abstract**/**subjective** sense in which something was **common** to all "**good meals**", how could it be stated with any **precision**? And yet we all know a "**good meal**" when we see one.

It is worth trying to capture this **less than rigorous** approach within the context of ML, whose success which will depend on two considerations:

- the **chosen goal** it is intended to support (explicitly known and stated), and
- the **underlying structure** of the data itself (unlikely to be known in advance).

A **multitude** of clustering algorithms can be applied to problems, and for each of those, many **different parameter settings** exist.

Presumably, some will be more effective than others **depending on the objectives**: an outcome allowing customers to make their purchases most quickly might not be the one that leads them to spend the most money.

If the best way to increase sales is to have loose clusters where customers are forced to browse a certain amount, while being exposed to similar but interesting products to find what they want, it might be possible to select a clustering approach to generate clusters with this desirable property.

To eliminate the possibility that the problem is not linked with the chosen clustering procedure, one strategy is to use **multiple clustering techniques**, as well as **multiple parameter settings** for each technique.

If the issue remains, then we could conclude that it is likely that **there is no good clustering structure** in the data and by extension, in the **objects** being represented by the data.

# 4.2 – $k-$Means Clustering

**Potential clustering objective:** achieve minimal **within-cluster variation** – observations **within** a cluster should be **very similar** to one another, and the **total variation** over all clusters should be **small**.

Assume that there are $k$ **clusters** in the (scaled) dataset

$$\mathbf{X}_{n\times p} = \begin{bmatrix} \mathbf{x}_1 & | & \cdots & | & \mathbf{x}_n \end{bmatrix}^\top.$$

Let $C_1, \ldots, C_k$ denote the set of indices in each cluster, so that

$$\{1, \ldots, n\} = C_1 \sqcup \cdots \sqcup C_k \quad \textbf{(hard clustering)}.$$

**Notation:** $\mathbf{x}_i \in C_\ell \iff i \in C_\ell \implies$ observation $i$ lies in cluster $\ell$.

The **within-cluster variation** $\mathrm{WCV}(C_\ell)$ measures the degree to which the observations in $C_\ell$ **differ from one another**.

The approach is **partition-based**; we look for a **partition** $\{C_\ell^*\}_{\ell=1}^k$ s.t. the total $\mathrm{WCV}$ is **minimized**:

$$\{C_\ell^*\} = \arg\min_{\{C_\ell\}} \left\{ \sum_{\ell=1}^{k} \mathrm{WCV}(C_\ell) \right\}.$$

**First challenge:** there are numerous ways to define $\mathrm{WCV}(C_\ell)$ and they do not necessarily **lead to the same results**. Most definitions fall in line with expressions such as

$$\mathrm{WCV}(C_\ell) = \frac{1}{(|C_\ell| - g)^\mu} \sum_{i,j \in C_\ell} \mathrm{variation}(\mathbf{x}_i, \mathbf{x}_j), \quad \text{with } \mathrm{variation}(\mathbf{x}, \mathbf{x}) = 0.$$

**Common choices:**

$$\text{variation}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \sum_{m=1}^{p} (x_{i,m} - x_{j,m})^2$$

$$\text{variation}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{m=1}^{p} |x_{i,m} - x_{j,m}|;$$

often used because of the ease of **vectorizing the distance measurements**, not necessarily because they **make the most sense in context**.

In both cases, we expect $\text{WCV}(C)$ to be **small** if all observations $\mathbf{x}$ within a cluster $C$ are **near one another**.

The values of the parameter $\mu$ can be adjusted to influence the **cluster sizes**.

Traditionally, we use $\mu = 0$ or $\mu = 1$, and $g = 0$, and the **partition problem** reduces to

$$\{C_\ell^*\} = \arg\min_{\{C_\ell\}} \left\{ \sum_{\ell=1}^{k} \frac{1}{|C_\ell|^\mu} \sum_{i,j \in C_\ell} \text{variation}(\mathbf{x}_i, \mathbf{x}_j) \right\}.$$

As an optimization problem, obtaining $\{C_\ell^*\}_{\ell=1}^k$ is **NP−hard** due to the **combinatorial explosion of possible partitions** $\{C_\ell\}_{\ell=1}^k$ when $n$ is **large**.

Computing the number of such partitions cannot be done by **elementary means** (but it is easy to show that the number is bounded above by $n^k$).

So how do we find the **optimal** partition?

# 4.2.1 – $k-$Means Algorithm

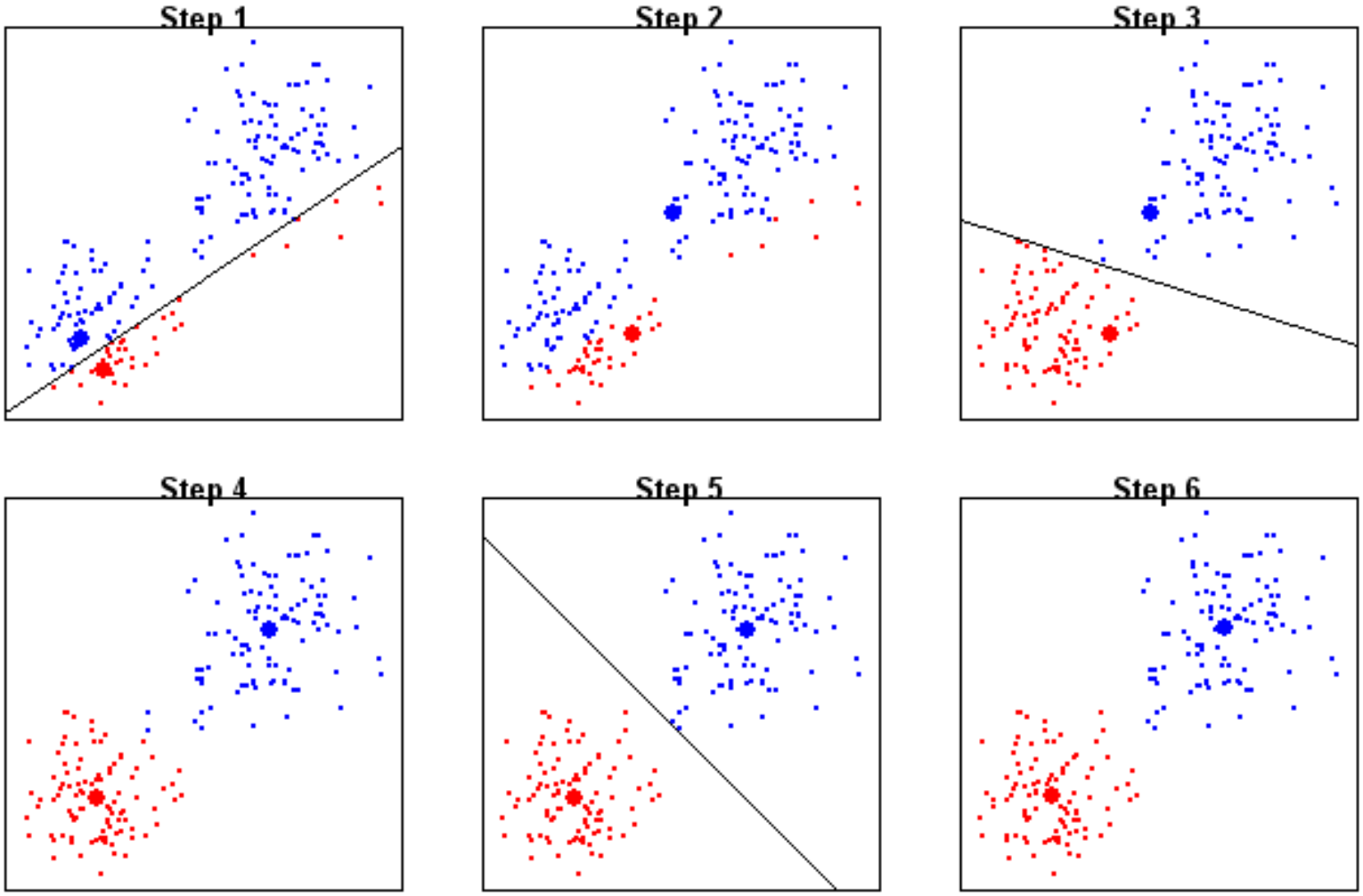
The $k-$**means solution** to the partition problem is (hopefully) "close" to the optimal one. It is obtained **without having to check** all partitions:

1. **randomly assign** a cluster number $\{1, \ldots, k\}$ to each observation in the dataset;

2. for each $C_\ell$, compute the cluster **centroid**;

3. assign each observation to the cluster whose **centroid** is **nearest** to the observation;

4. repeat steps 2-3 until the clusters are **stable**.


Step 1
Step 2
Step 3
Step 4
Step 5
Step 6

## Algorithm choices:

- the **number of clusters** $k$ in step 1;
- the **centroid computation measure** in step 2;
- the **distance metric** used in step 3.

The most common choice of **centroid** measure for numerical data is the means along each feature of the observations in each cluster ($k-$**means**).

Other centrality measures yield different methods (i.e., $k-$**medians**).

For categorical data, the algorithm becomes $k-$**modes**.

The **distance** used in step 3 is usually aligned with the **centroid measure** of step 2 and with the choice of a **variation function**: **Euclidean** for $k-$means, **Manhattan** for $k-$medians, **Hamming** for $k-$modes.

**Variant:** start by picking $k$ **centroids**, then continue with the algorithm.

Note that results depend very strongly on the **initial randomization** – a "poor" selection can yield **sub-optimal** results; $k-$**means++** selects the initial centroids so as to maximize the chance that they will be **well-spread in the dataset** $\Longrightarrow$ speeds up the **run-time**.

**Other variants:** process computations **in parallel** or for **data streams**.

$k$-means converges to a **stable cluster assignment**, but this may not be a **global minimizer** of the objective function; **different** initial conditions can find different **local minima** $\Longrightarrow$ different **clustering schemes** (**stable** vs. **unstable** clusters).

# Strengths

- **Elegant** and **easy to implement**, without having to compute the **pairwise distances**,
- **extremely** common (cannot think of a data analysis package without an implementation);
- **natural way** of **grouping** observations in many contexts;
- provides a first-pass **basic understanding of the data structure**.

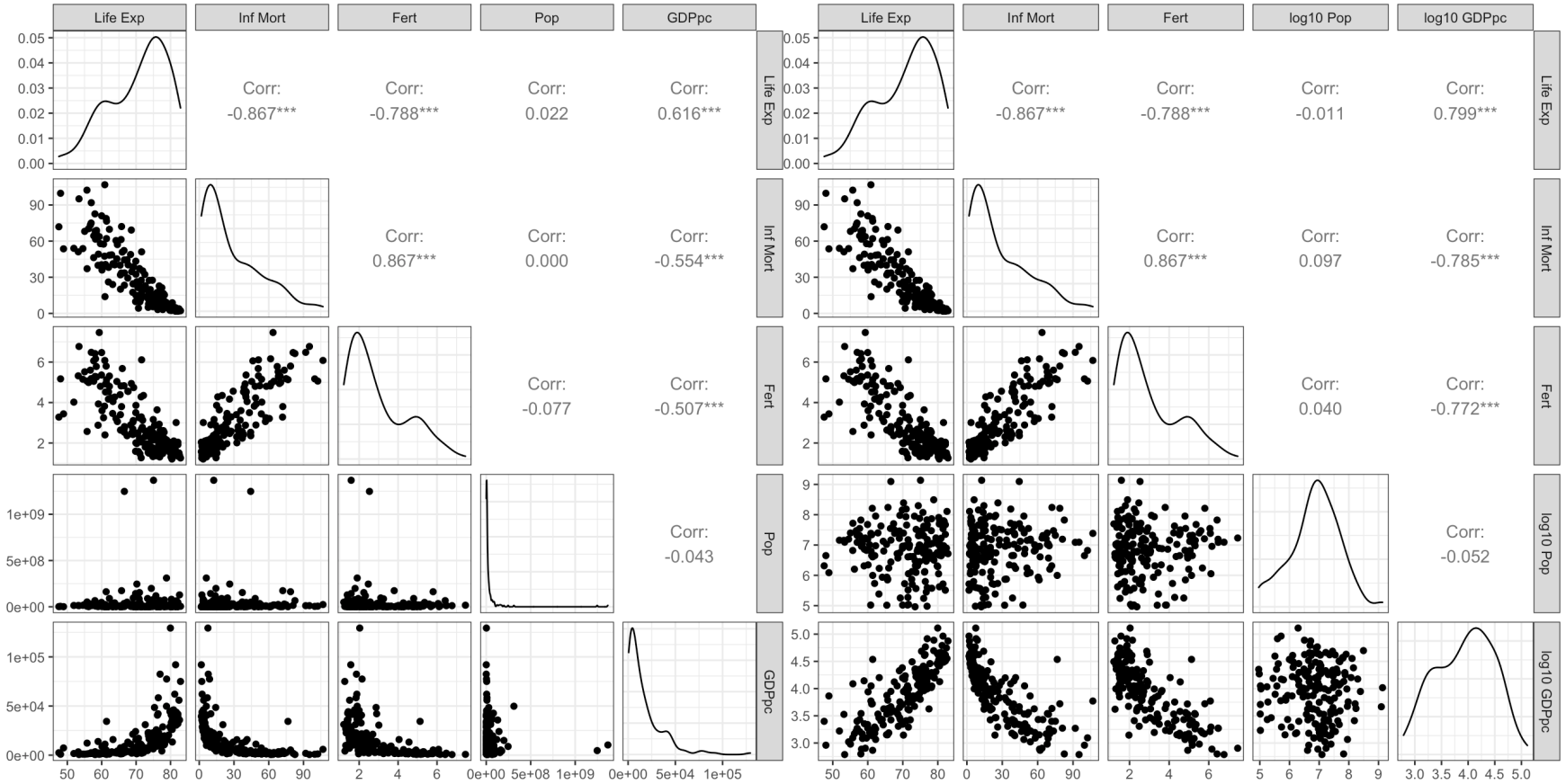
# Limitations

- Can only assign an instance to **one** cluster $\Longrightarrow$ risk of **overfitting** – a more robust solution: compute the **probability of belonging** to each cluster (perhaps based on the **distance to the centroids**?);
- $k-$means requires the "**true**" underlying clusters to be **gaussian** (**blob-shaped**) – it will fail to produce **useful clusters** if that assumption is not met in practice;
- it does not allow for **overlapping** or **hierarchical** groupings;

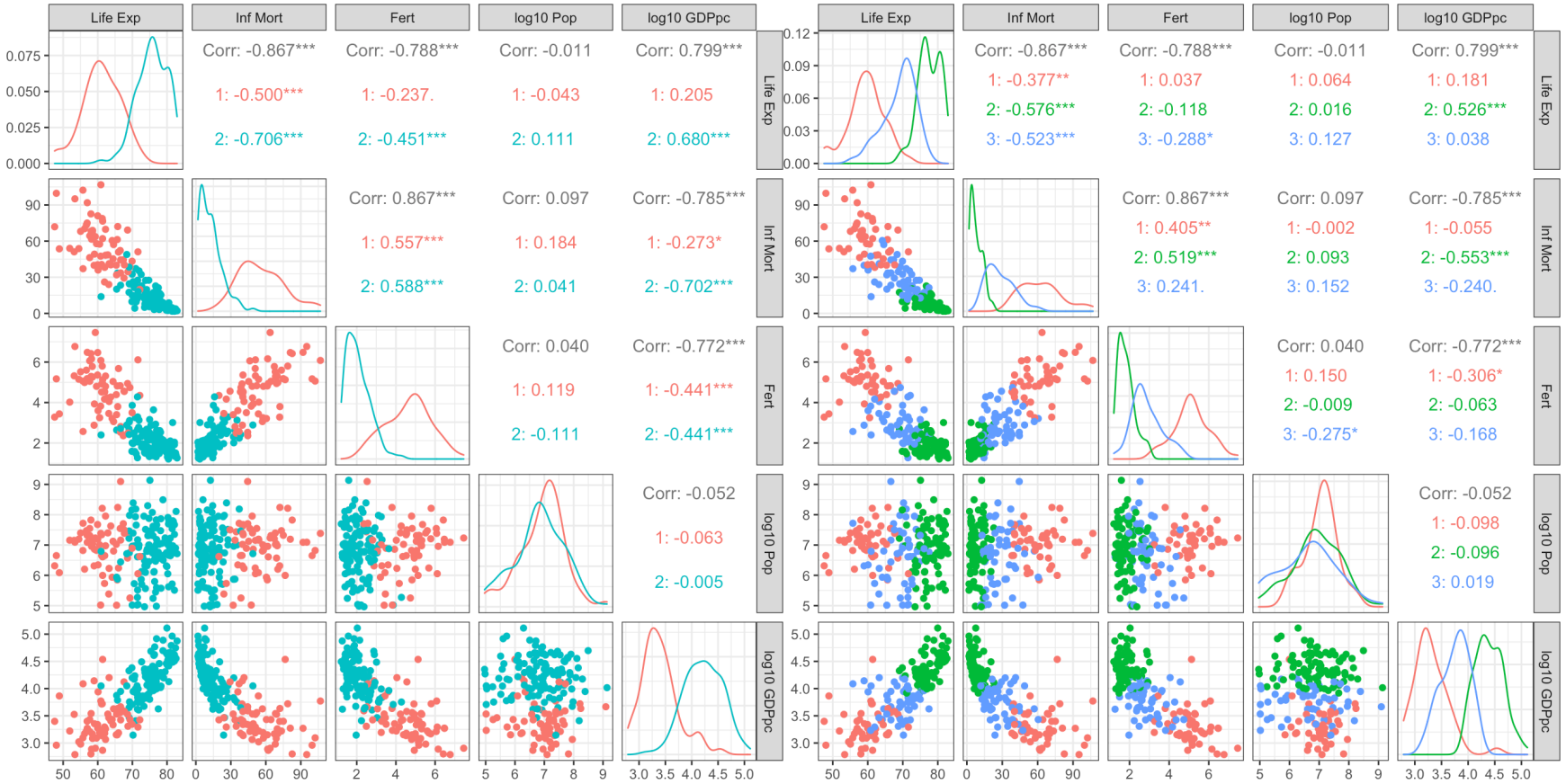
# 4.2.2 – Example: Gapminder Dataset

We work with a variant of `gapminder.csv` to illustrate $k-$means (DUDADS, 22.2.1, *$k$-Means and Variants* for code). The 2011 data contains observations on $n = 184$ countries, for:

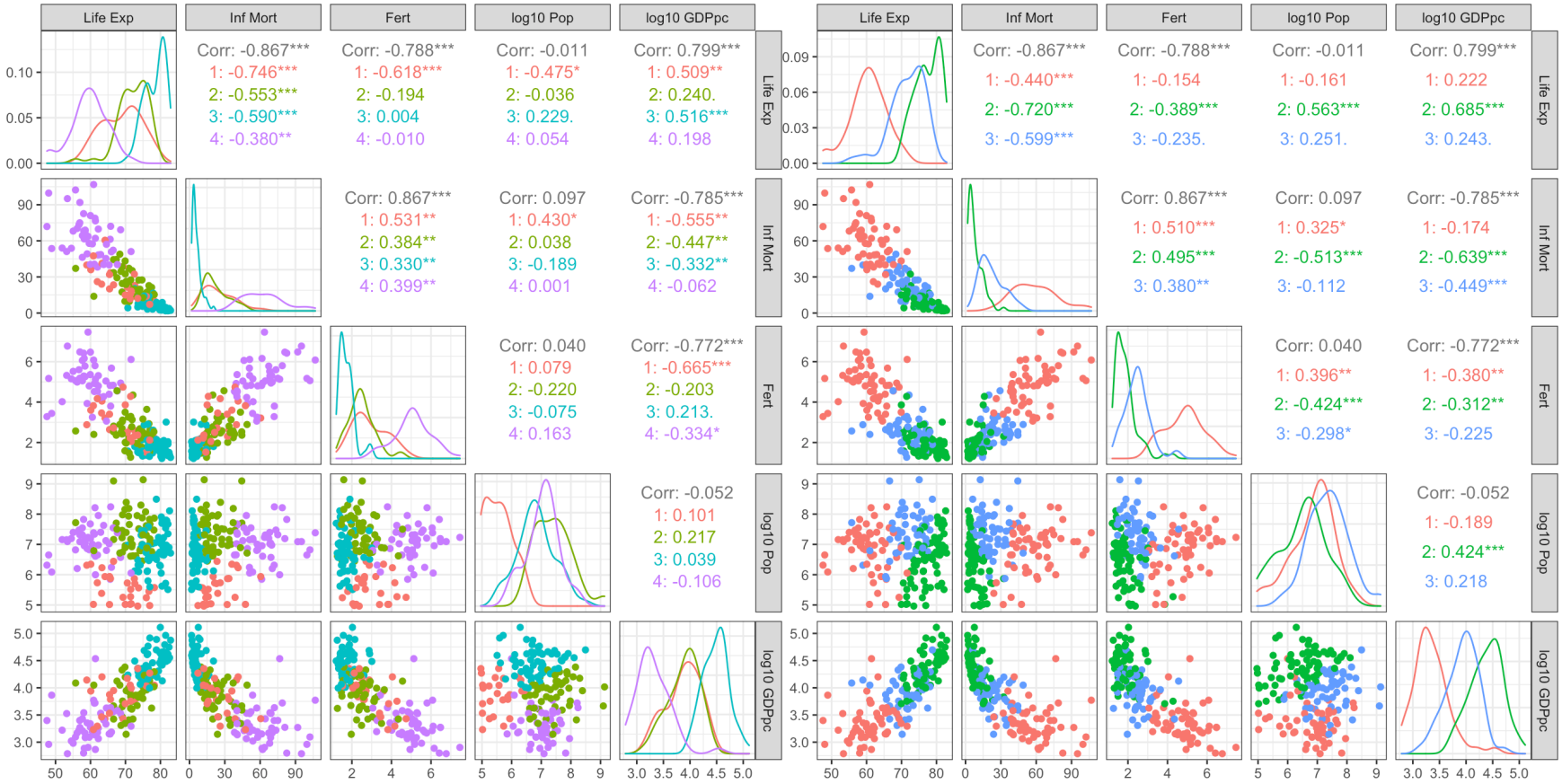
- **life expectancy** (in years);
- **infant mortality rate** (per 1000 births);
- **fertility rate** (in children per woman);
- **population** (we use the logarithm for clustering), and
- **GDP per capita** (same).




original data scaled data




2−means, Euclidean 3−means, Euclidean




4−means, Euclidean

3−means, Euclidean (new)

# 4.3 – Hierarchical Clustering

**Problem:** the results of **a single $k-$means run** don't indicate if the choice of $k$ was **optimal**; they might look good on a 2D representation of the data, but could they be **better**?

**Solution:** run the algorithm over a range of **"reasonable" values of $k$** and compare the outputs using some of the methods discussed in Section 4.5. But this process can be **memory-** (and **time-**)**extensive**.

**Hierarchical clustering** (HC) sidesteps this problem by building a **deterministic global clustering structure** (given parameters).

**Bonus:** if we want to use a different **number of clusters**, no need to re-run the clustering algorithm – simply **read off** the new clusters from the **global clustering structure**.

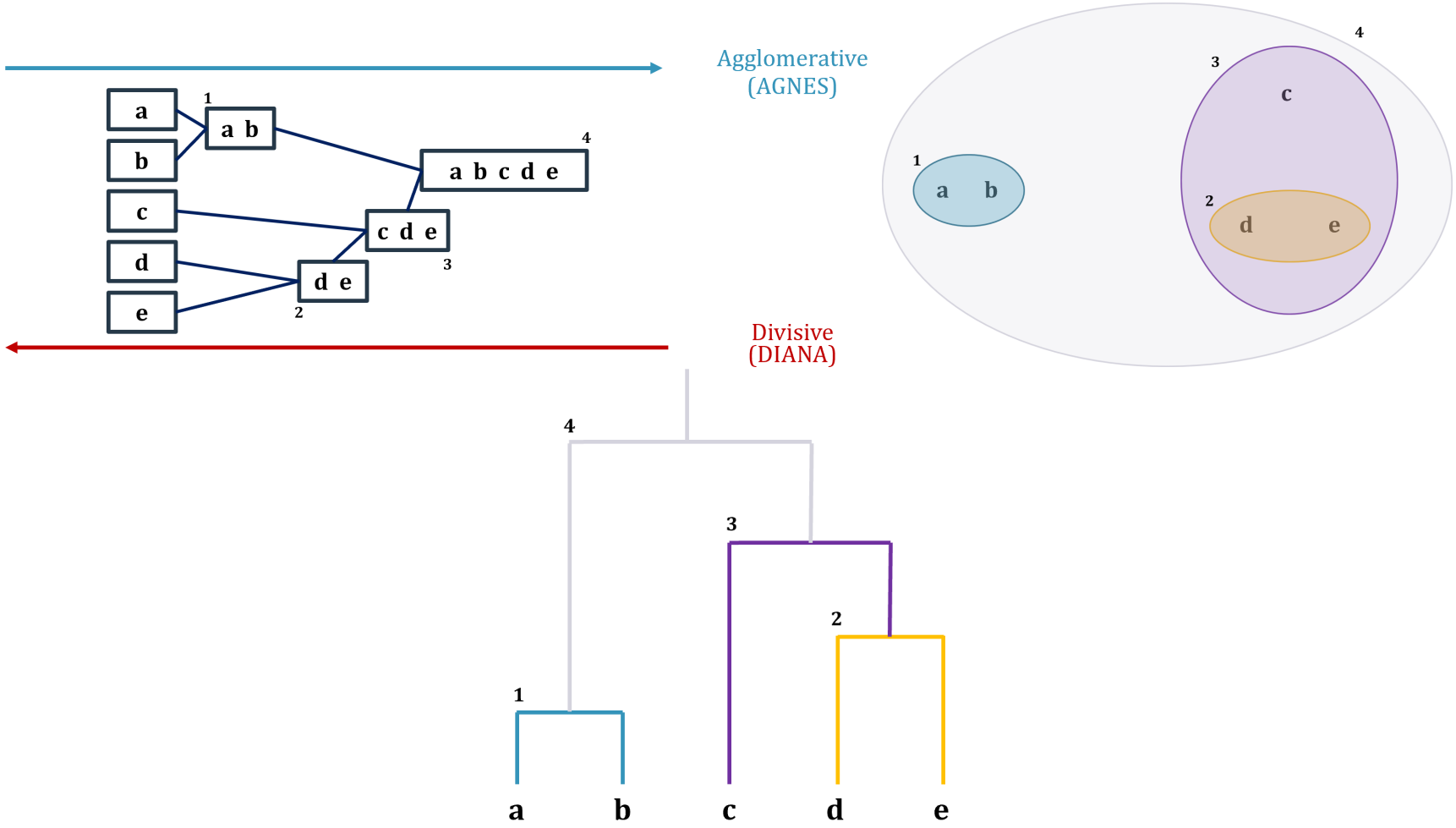
# 4.3.1 – AGNES/DIANA Algorithms

There are two main conceptual approaches:

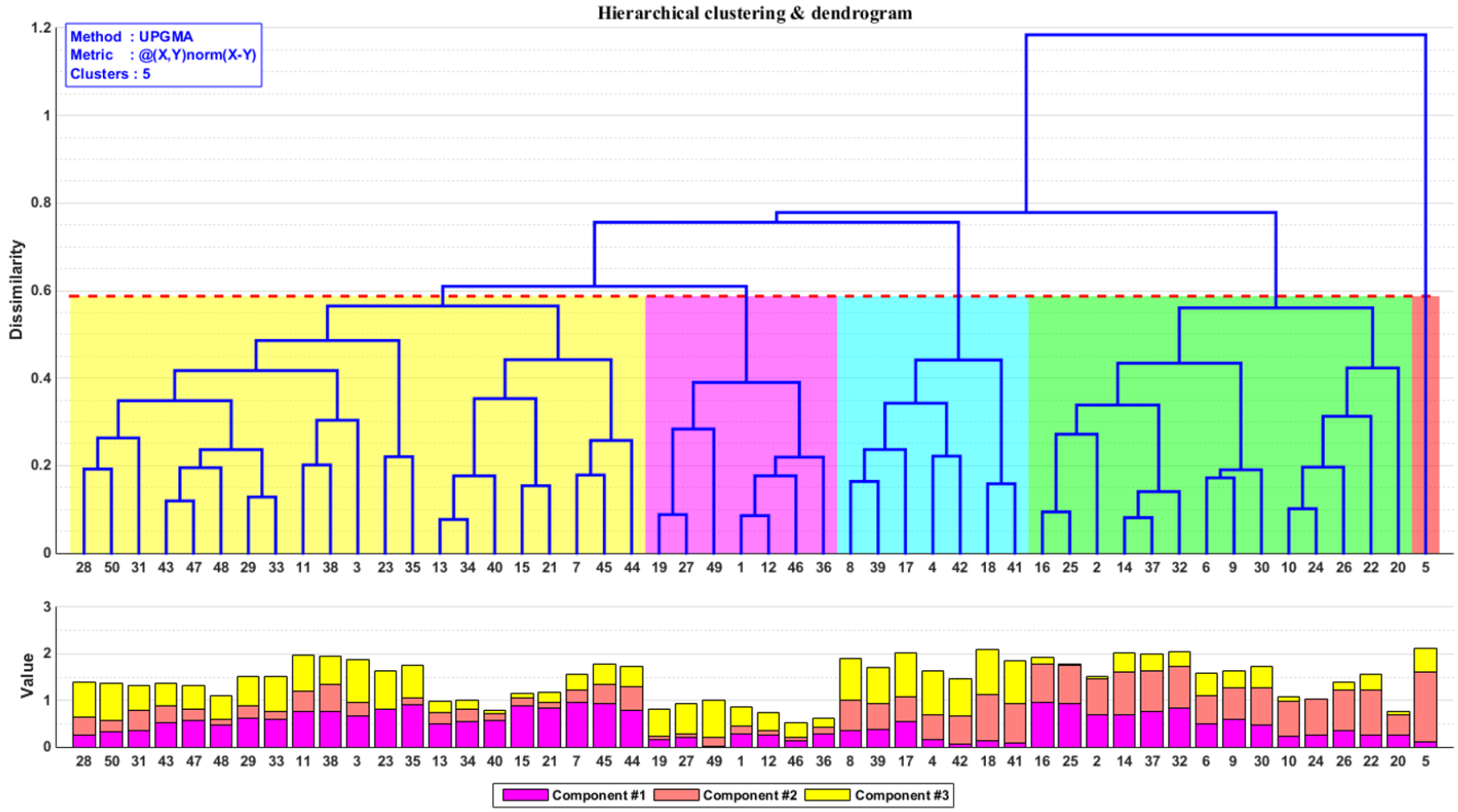
- **bottom-up/agglomerative** (**AGNES**) – initially, each observation sits in **its own separate cluster**, which are then **merged** (in **pairs**) as the hierarchy is **climbed**, leading (after the last **merge**) to a **large cluster containing all observations**;

- **top-down/divisive** (**DIANA**) – initially, all observations lie **in the same cluster**, which is **split** (and **re-split**) in **pairs** as the hierarchy is traversed **downward**, leading (after the last **split**) to each observation sitting in **its own separate cluster**.

In theory, the two are **equivalent**; in practice, we use **AGNES** over **DIANA**.




Conceptual representation of AGNES and DIANA.




HC dendrogram with 5 clusters [author unknown].

# Algorithm: AGNES

The **global clustering structure** is built as follows:

1. each observation is assigned to **its own cluster** (there are $n$ clusters, initially);
2. the two **clusters** that are the **least dissimilar** are merged into a **supra-cluster**;
3. repeat step 2 until **all of the observations** belong to a **single** large merged cluster (with $n$ observations).

Three decisions need to be made in order for the algorithm to run:

- the choice of a **linkage strategy** in steps 2 and 3;
- the **dissimilarity measure** used in step 2;
- the **dissimilarity threshold** required to "cut" the dendrogram into clusters.

In the previous example, the dataset is first split into $n = 50$ clusters.

Observations $13$ and $34$ are next found to be **most similar**, and merged into a **single cluster**: the $50$ observations are now grouped into $49$ clusters.

The next two observations which are most similar are $14$ and $37$, which are themselves merged, so that there are $48$ clusters at the next level.

The process continues (**how?**) until all observations are merged into a single cluster $\Longrightarrow$ **global clustering structure** $\Longrightarrow$ **clustering dendrogram**.

Cutting the **dendrogram** at a selected **dissimilarity level** $\Longrightarrow$ data clusters ($5$ in this example)

When dissimilarity threshold $\nearrow$ $\Longrightarrow$ number of clusters $\searrow$ (and *vice-versa*).

# 4.3.2 – Linkage Strategies

In the first AGNES stage, we compare all **pairs of observations** to find which 2 are **least dissimilar**; these are then **merged** into a cluster.

With $n$ observations, there are $1 + \cdots + (n-1) = \frac{(n-1)n}{2}$ such pairs.

In the second stage, we must also compare each of the **non-merged** observations with one **cluster** of (**two**) observations to find their dissimilarity; the other dissimilarities have **already been computed in the first stage**.

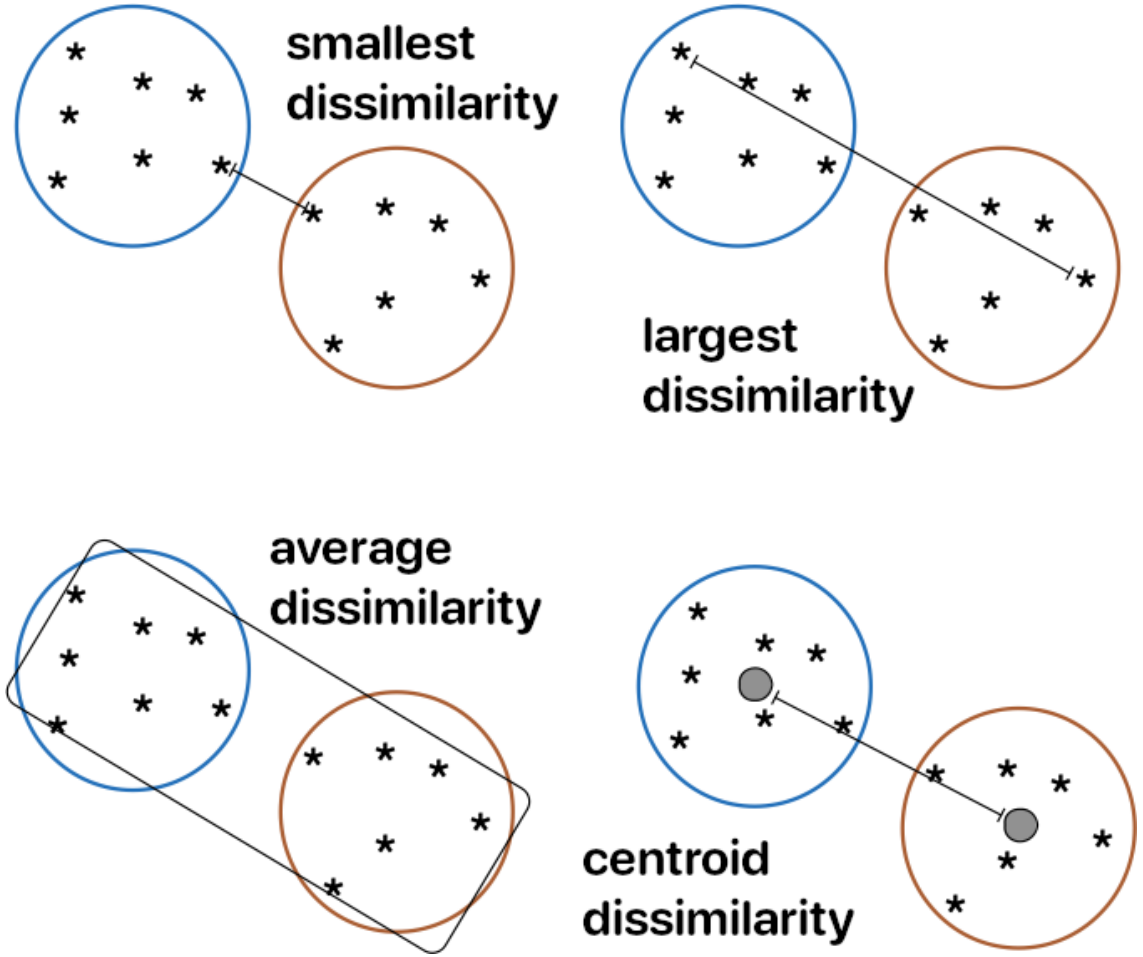
In subsequent stages, we will need to compare **pairs of clusters** with **any number** of observations for **overall similarity**.

Let $A$ and $B$ be clusters in the data, with $|A| = n_A$, $|B| = n_B$.

The **dissimilarity** between $A$ and $B$ can be computed in multiple ways:

- **complete linkage** – the **largest** dissimilarity among all pairwise dissimilarities between the observations in $A$ and those in $B$ ($n_A n_B$ comparisons);
- **single linkage** – the **smallest** dissimilarity among all pairwise dissimilarities as in complete linkage ($n_A n_B$ comparisons);
- **average linkage** – the **average** dissimilarity among all pairwise dissimilarities as in complete (or single) linkage ($n_A n_B$ comparisons);
- **centroid linkage** – the dissimilarity between the **centroids** of $A$ and $B$ (found using whatever method is appropriate for the context; only $1$ comparison, but...);
- **Ward's method** (and its variants) uses reasonable **objective function** which reflects domain knowledge;

The choice of a **linkage strategy** (and dissimilarity measure) affects not only how clusters are **compared** and **merged**, but also the **topology** (structure) of the **resulting dendrogram** (are clusters tight, loose, blob-like, etc.).




Conceptual notions of linkage, assuming Euclidean dissimilarity.

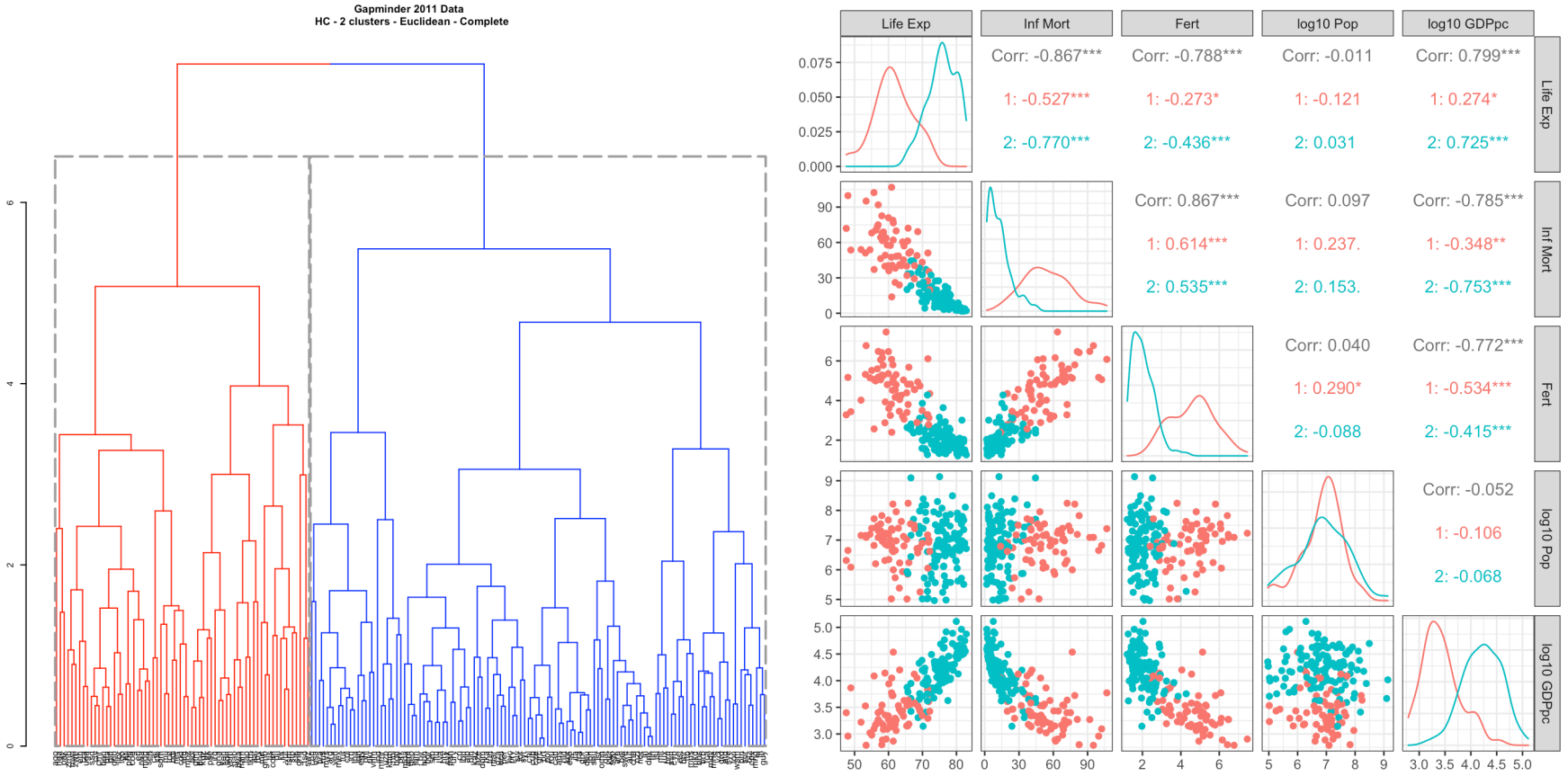
# 4.3.3 – Example: Gapminder Dataset

We work with the (scaled) 2011 Gapminder data, to illustrate HC, using **complete linkage** + **Euclidean dissimilarity**, and **Ward $D$ linkage** + **complete dissimilarity**, with $k = 2, 3, 4$ clusters (code: DUDADS, 22.2.2).

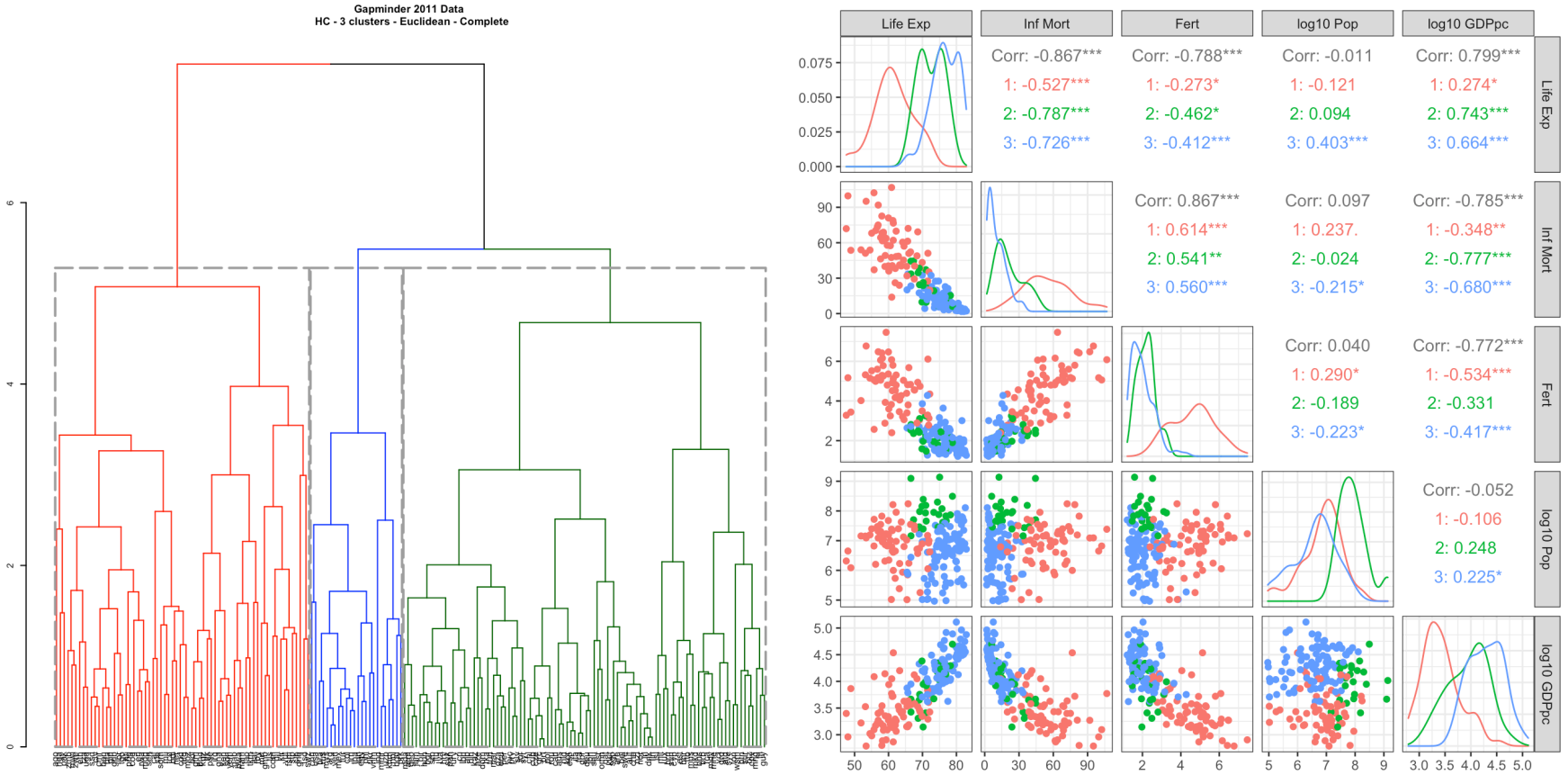
The breakdown for $k = 2, 3, 4$ clusters in the first approach is:

```
 1   2
66 118

 1  2  3
66 24 94

 1  2  3  4
35 24 94 31
```

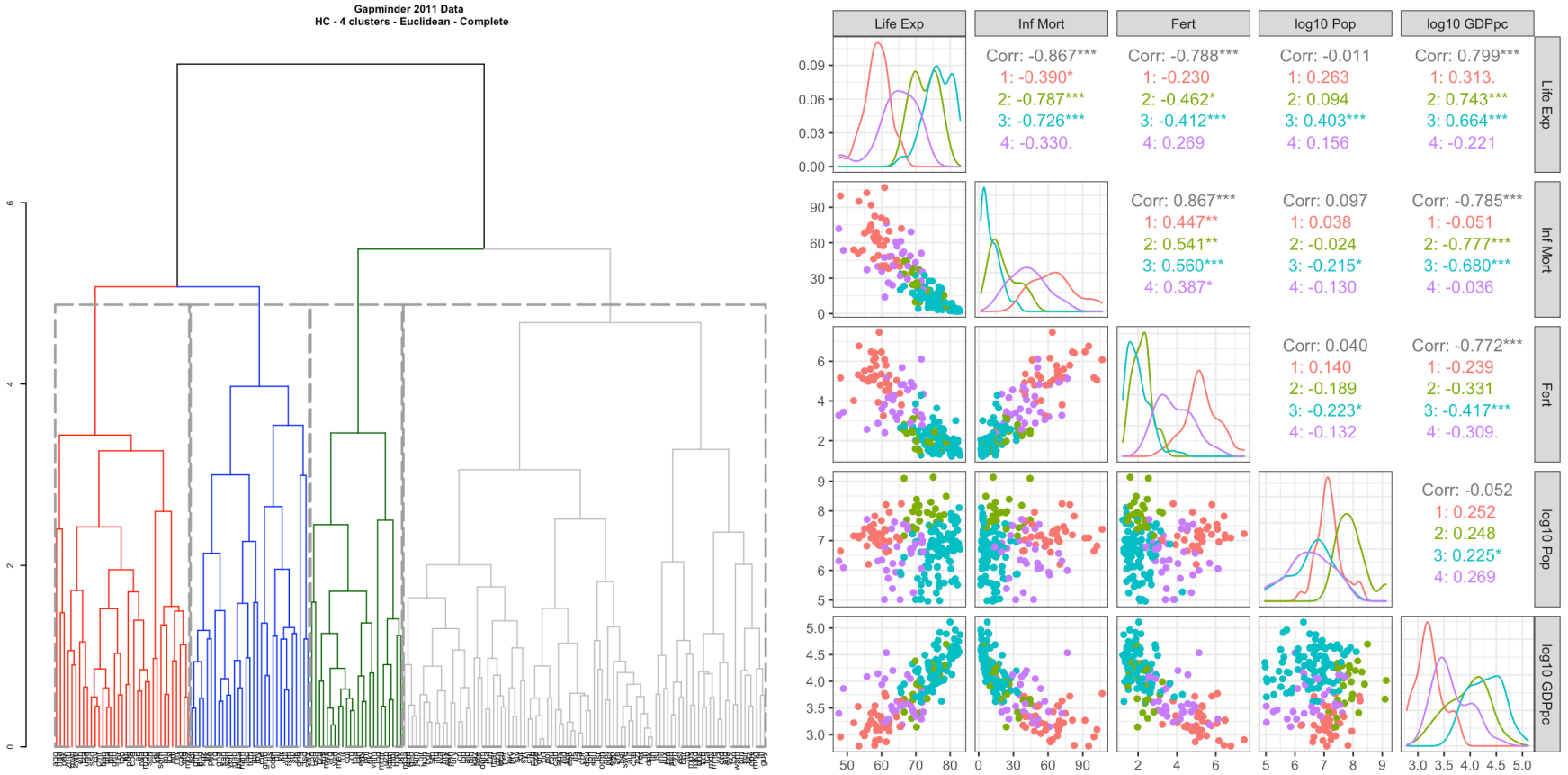
Note how the # of obs in each cluster follows a **hierarchical** structure: when we go from $k = 2$ to $k = 3$ clusters, the **new cluster** is a subset of one of the **old clusters** (and similarly from $k = 3$ to $k = 4$).



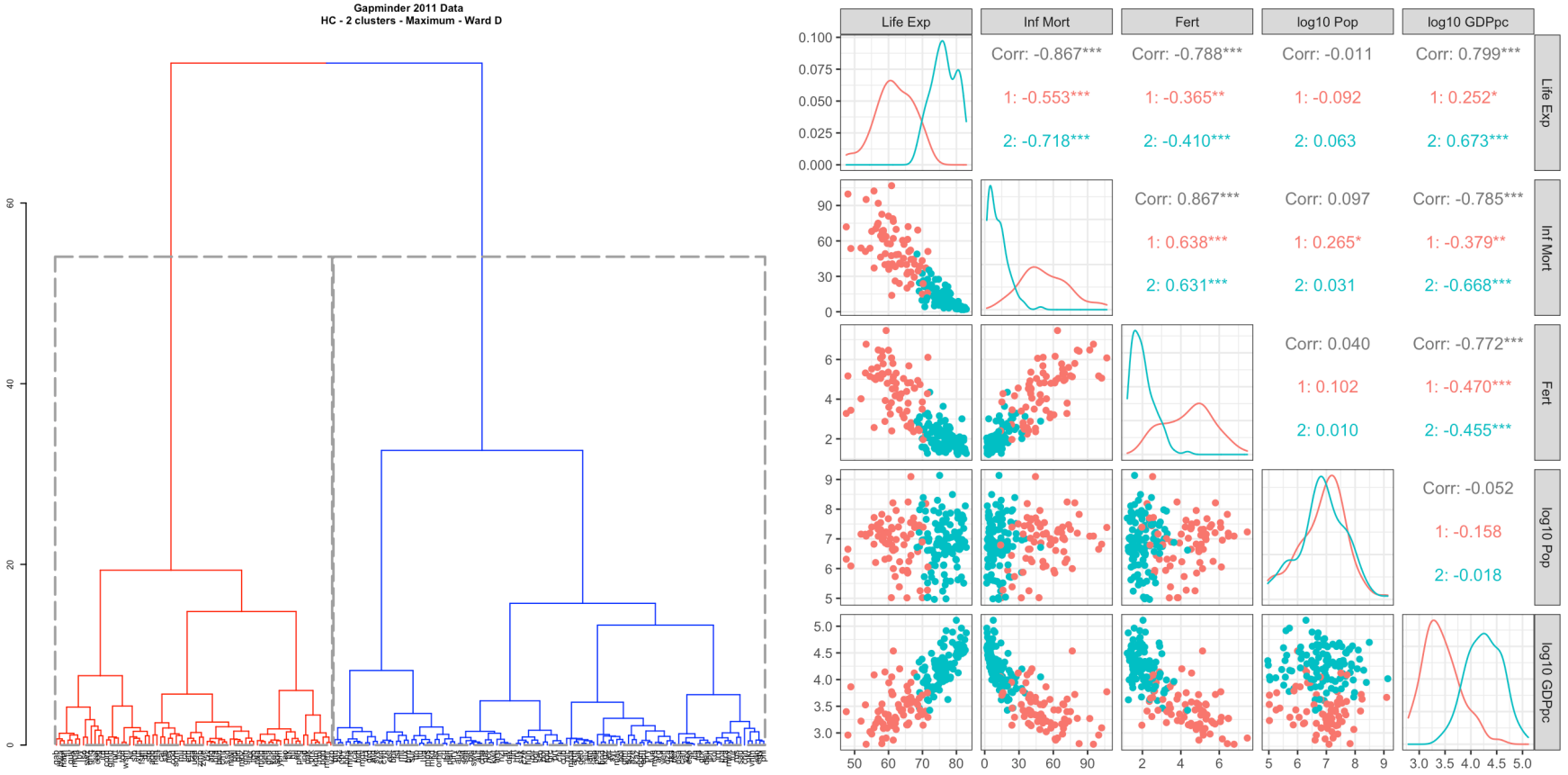

Realizations of hierarchical clustering (AGNES) on the 2011 Gapminder data: complete linkage, Euclidean dissimilarity for $k = 2$ clusters.



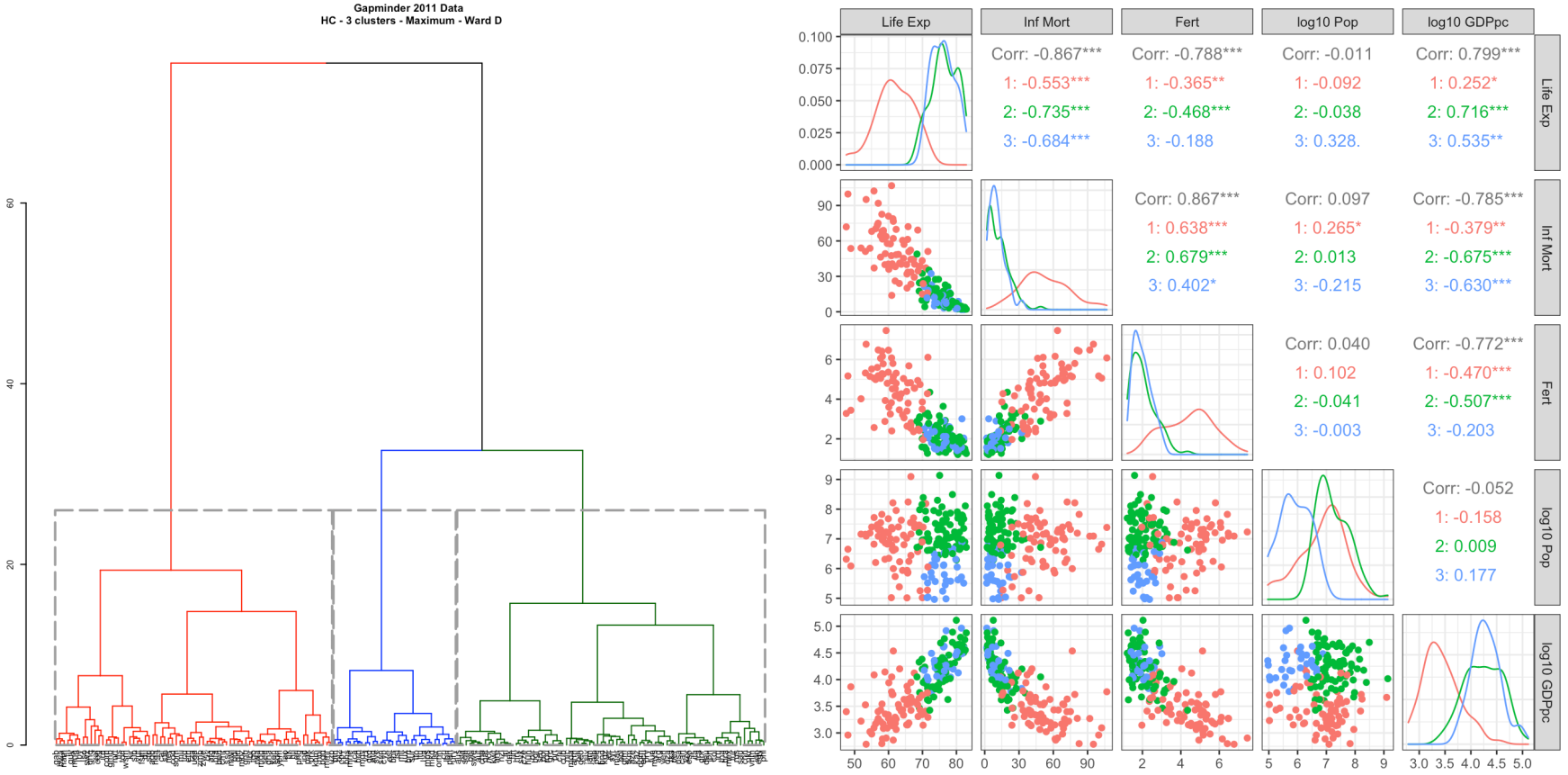

Realizations of hierarchical clustering (AGNES) on the 2011 Gapminder data: complete linkage, Euclidean dissimilarity for $k = 3$ clusters.



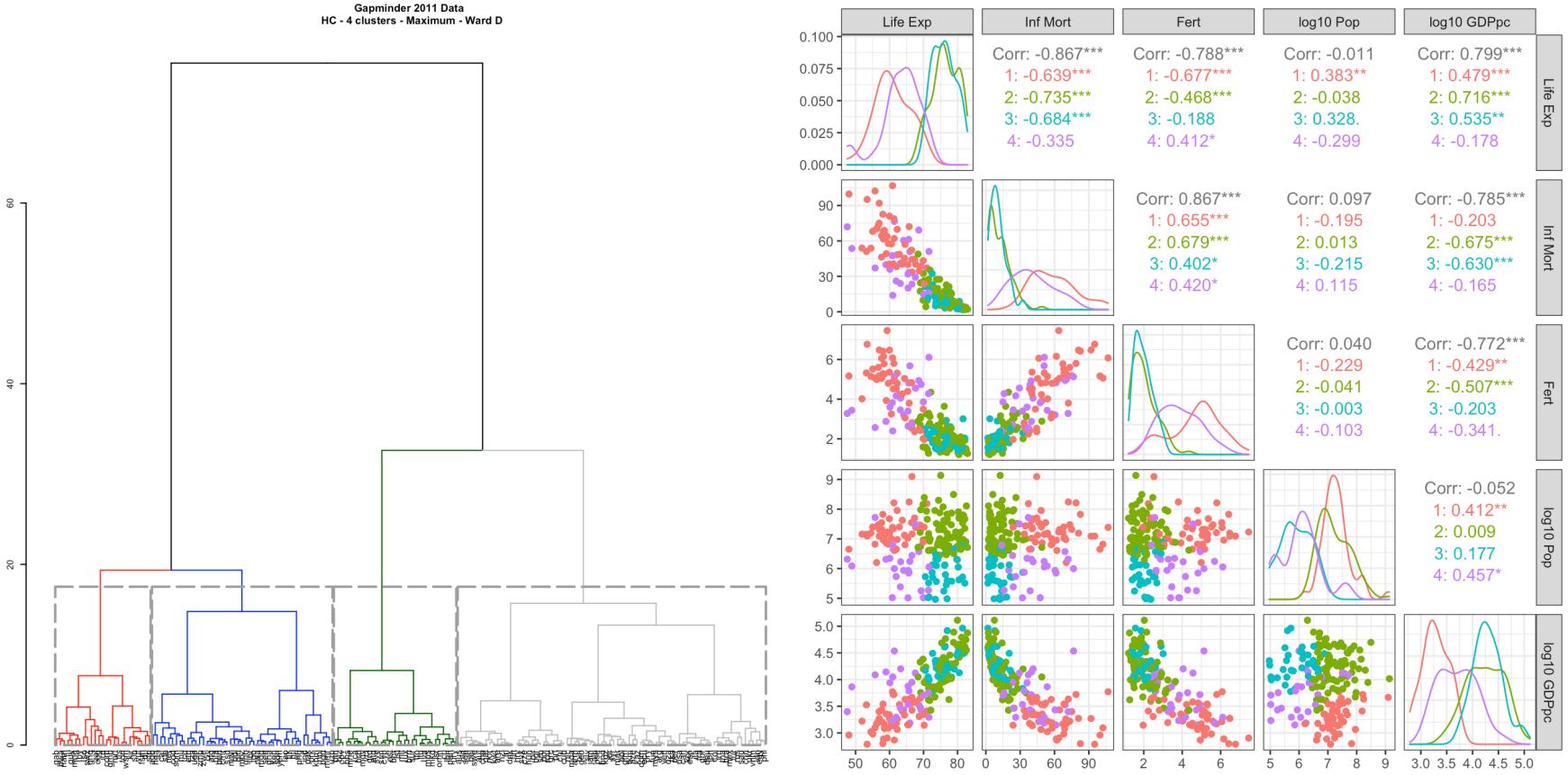

Realizations of hierarchical clustering (AGNES) on the 2011 Gapminder data: complete linkage, Euclidean dissimilarity for $k = 4$ clusters.




Realizations of hierarchical clustering (AGNES) on the 2011 Gapminder data: Ward $D$ linkage, maximum dissimilarity for $k = 2$ clusters.




Realizations of hierarchical clustering (AGNES) on the 2011 Gapminder data: Ward $D$ linkage, maximum dissimilarity for $k = 3$ clusters.




Realizations of hierarchical clustering (AGNES) on the 2011 Gapminder data: Ward $D$ linkage, maximum dissimilarity for $k = 4$ clusters.

# 4.4 – Density-Based Clustering

The assumptions of $k-$means imply that it works "best" when the underlying clusters are **Gaussian** (blobs). That is not always the **desired** outcome.

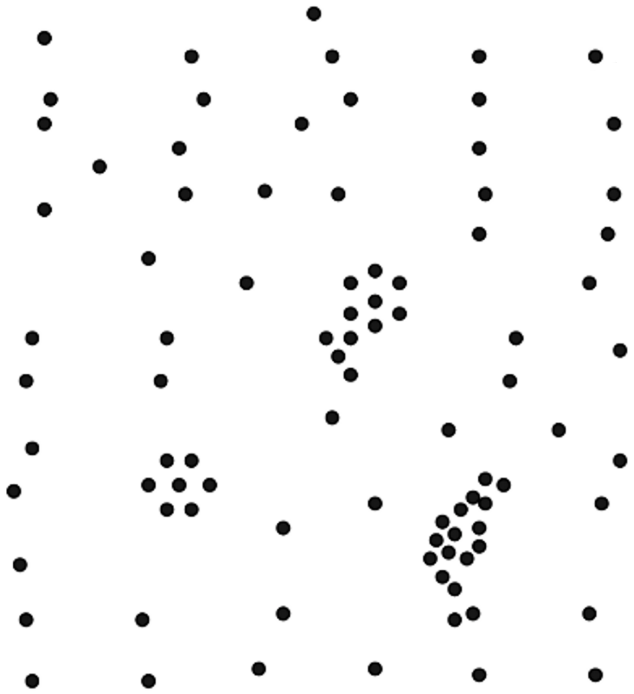
In **density-based clustering**, the **density** of observations and the **connectivity** of the clustering **network** (see Section 4.6) determine the **number** and **location** of clusters.

**Popular algorithms:** **DBSCAN**, **DENCLUE**, **OPTICS**, **CHAMELEON**, etc.

Once **density** has been defined in a **meaningful way**, the algorithms are straightforward to **implement** and **apply**.

# Density

How do we measure density? Intuitively, we can recognize areas of **low density** and **high density** in the (artificial) dataset below.



"Birds of a feather flock together"; it should not come as a surprise that areas of higher density may be viewed as **clusters** in the data.

In that context, if $\mathbf{\Psi} \subseteq \mathbb{R}^n$ is an $n-$dimensional **sub-manifold** of $\mathbb{R}^n$, we could define the **density** of $\mathbf{\Psi}$ around $\mathbf{x}$ by, say,

$$\text{density}_{\mathbf{\Psi}}(\mathbf{x}; d) = \lim_{\varepsilon \to 0^+} \frac{\text{Vol}_n(B_d(\mathbf{x}, \varepsilon) \cap \mathbf{\Psi})}{\text{Vol}_n(B_d(\mathbf{x}, \varepsilon))},$$

where the constituents of the definition are:

$$B_d(\mathbf{x}, \varepsilon) = \{\mathbf{y} \in \mathbb{R}^n \mid d(\mathbf{x}, \mathbf{y}) < \varepsilon\}$$

and

$$\text{Vol}_n(A) = n - \textbf{volume of } A \textbf{ in } \mathbf{R}^n.$$

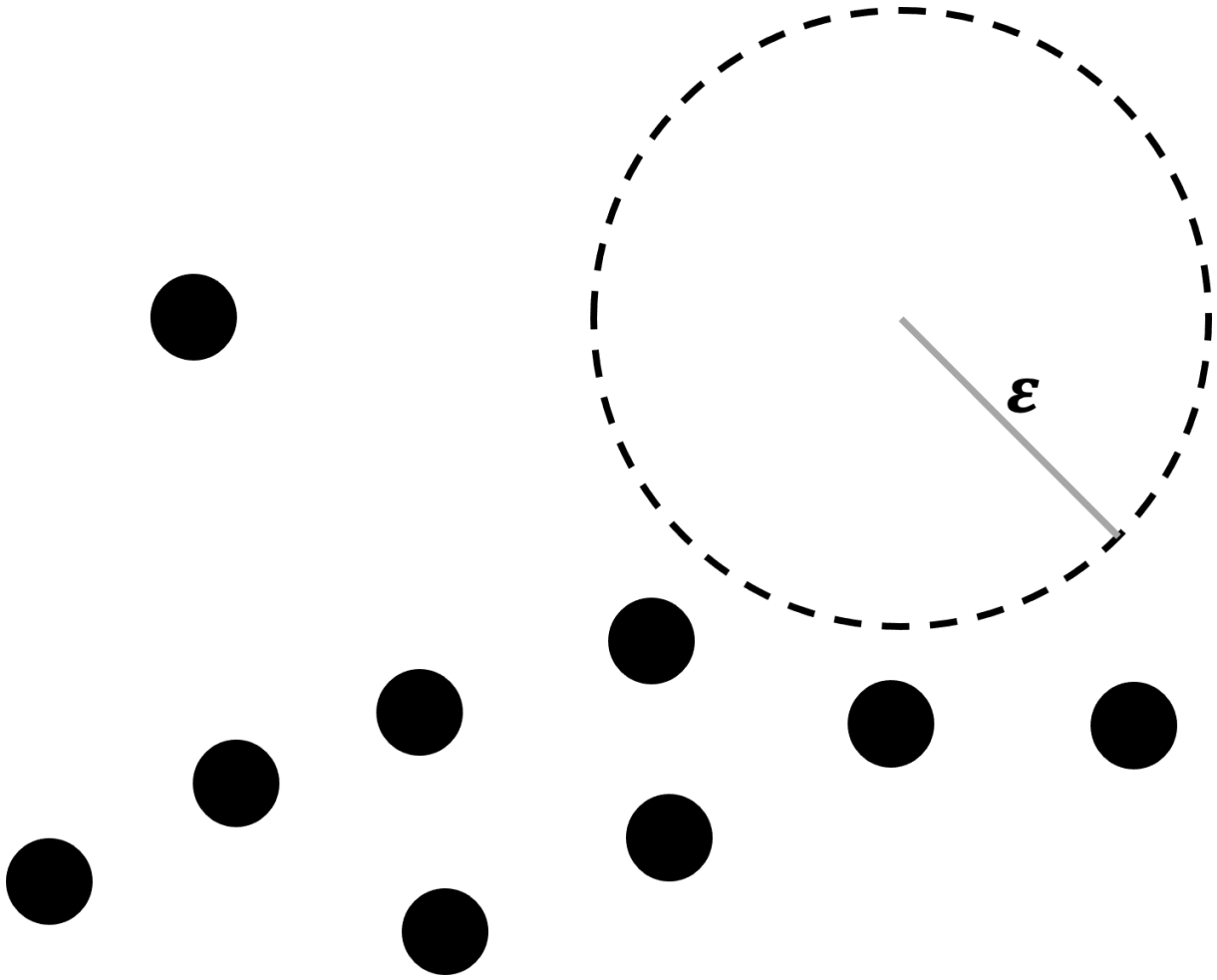
# 4.4.1 – DBSCAN

In practice, the dataset $\mathbf{X}$ is usually a **discrete subset** of $\mathbb{R}^n$, and the **limit definition** cannot be applied as is.

**Density-based spatial clustering of applications with noise** (DBSCAN) estimates the **density** at an observations $\mathbf{x} \in \mathbf{X}$ as follows by selecting a "**reasonable**" value of $\varepsilon^* > 0$ and setting

$$\text{density}_{\mathbf{X}}(\mathbf{x}; d) = |B_d(\mathbf{x}, \varepsilon^*) \cap \mathbf{X}|.$$

The outcome obviously depends on the choice of $\varepsilon^*$ and the **distance** $d$.


ε

DBSCAN also requires a **connectivity** parameter, the **minimum number of points** $minPts$ in

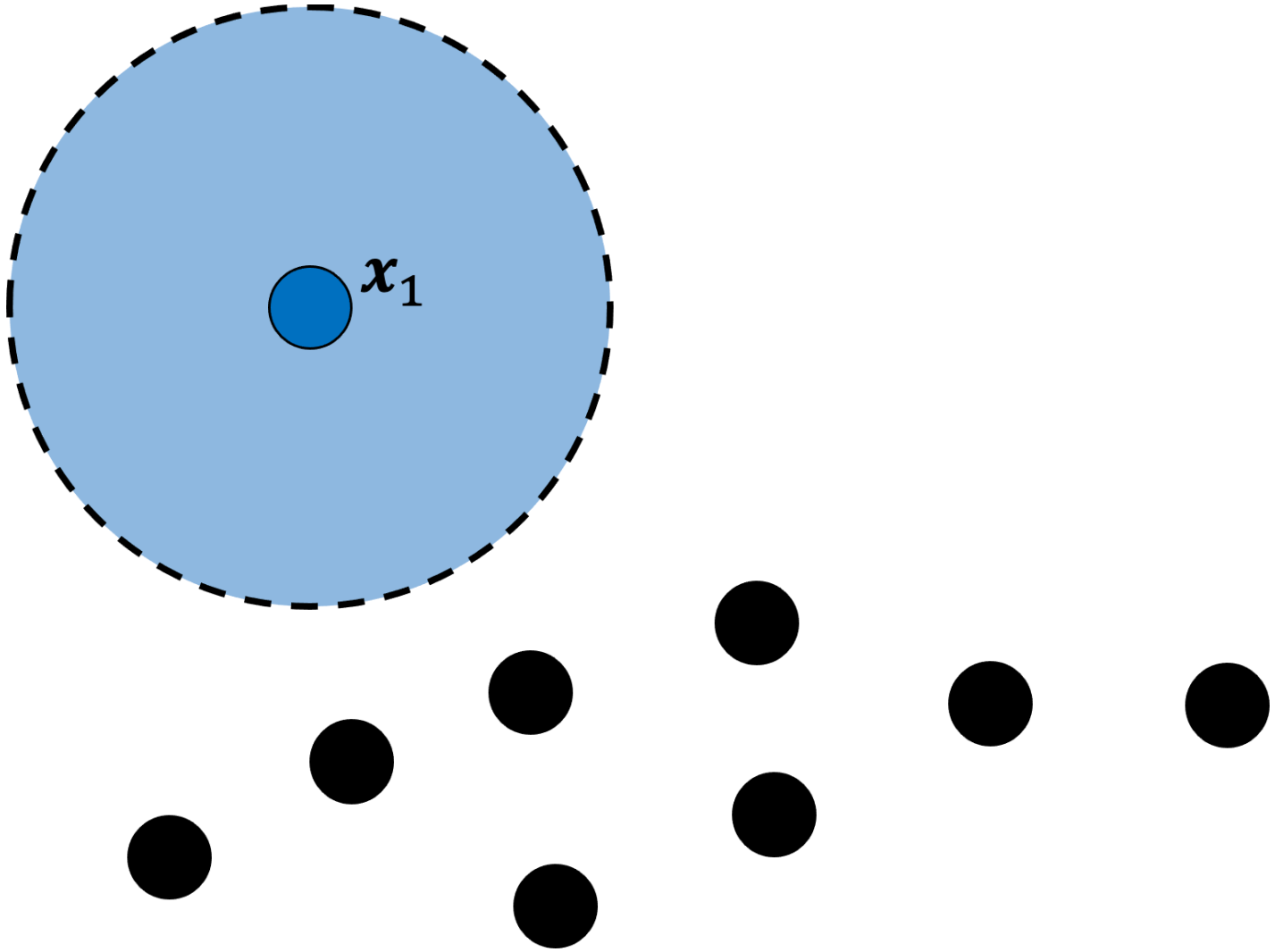
$$V_{\mathbf{x}} = B_d(\mathbf{x}, \varepsilon^*) \cap [\mathbf{X} \setminus \{\mathbf{x}\}]$$

(**excluding** $\mathbf{x}$).

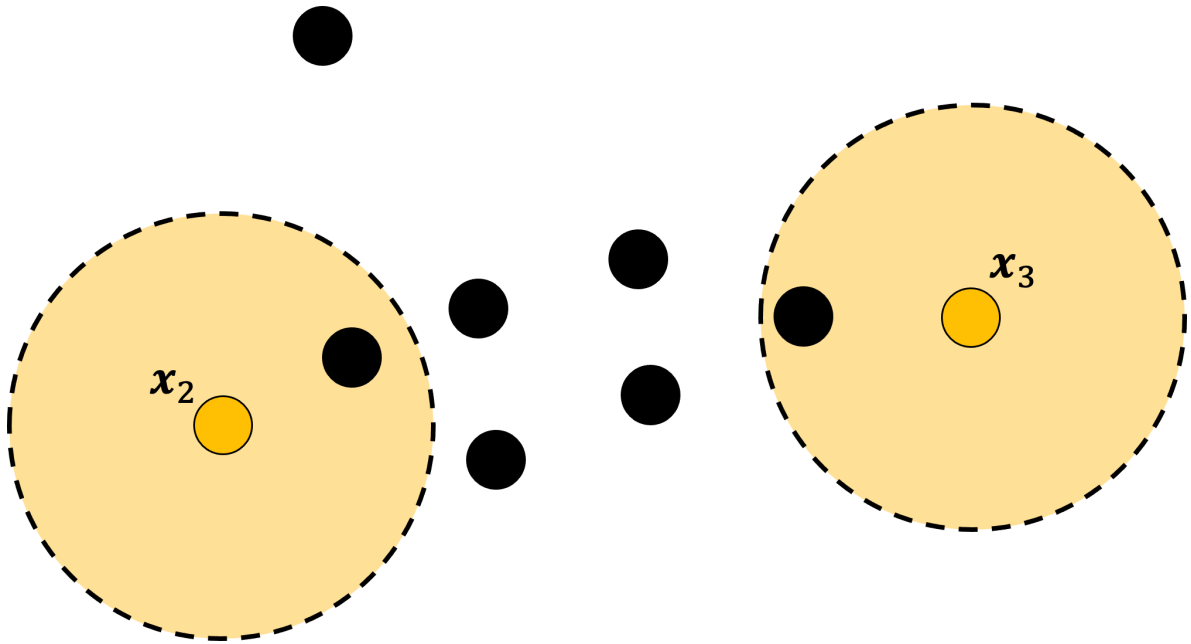
If $|V_{\mathbf{x}}| \geq minPts$, the observations in $V_{\mathbf{x}}$ are said to be **within reach of** (or **reachable from**) $\mathbf{x}$.

For a given choice of $d$, $\varepsilon^*$, and $minPts$, there are thus three **types of observations** in $\mathbf{X}$:

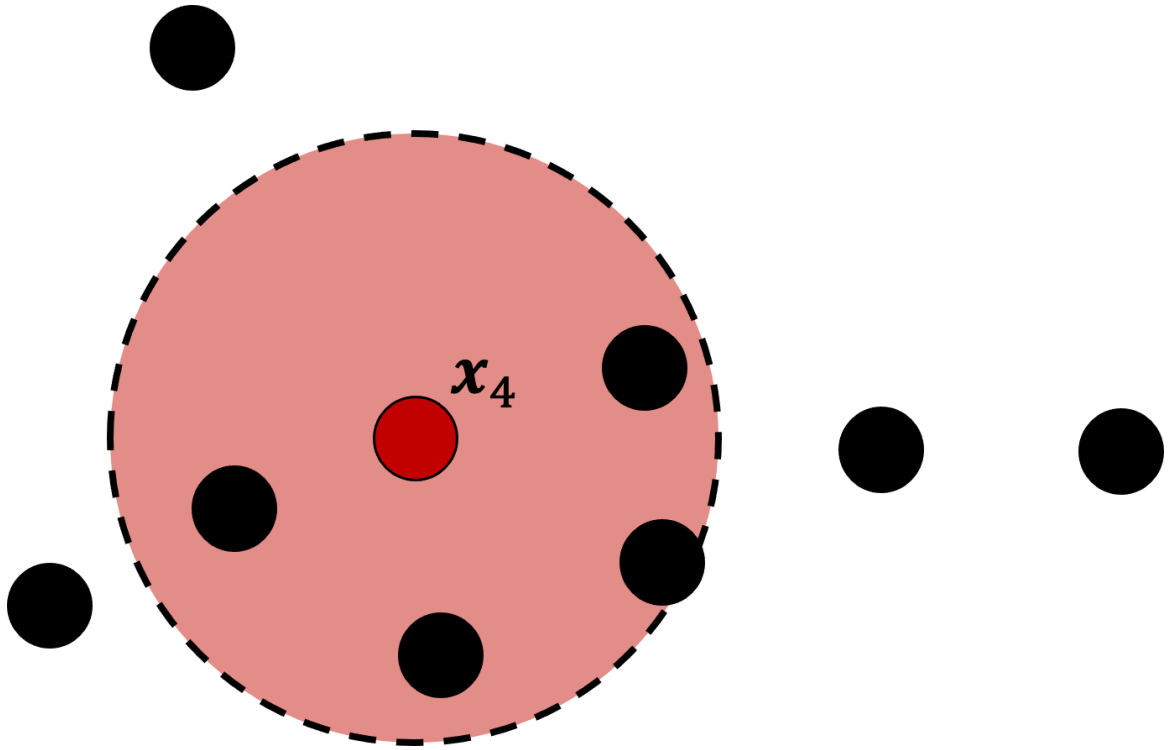
- **outliers** are observations that are not within reach of any of the other observations, such as $\mathbf{x}_1$ below:


$x_1$

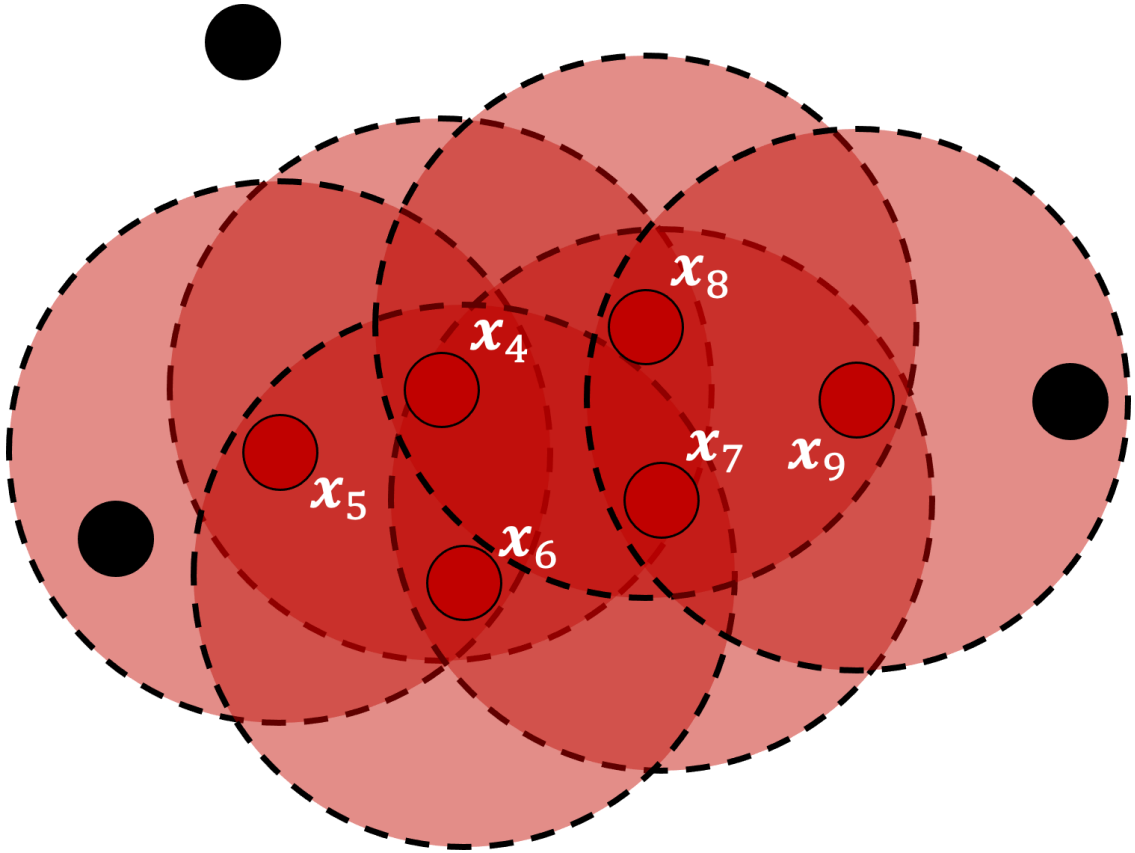
- **reachable** (**non-core**) **observations** are observations that are within reach of fewer than $minPts$ other observations, such as $\mathbf{x}_2$ and $\mathbf{x}_3$ below (with $minPts = 3$):



- **core observations** are within reach of at least $minPts$ other observations, such as $\mathbf{x}_4$ below (with $minPts = 3$):



There are other **core** points: $\mathbf{x}_5$, $\mathbf{x}_6$, $\mathbf{x}_7$, $\mathbf{x}_8$, and $\mathbf{x}_9$.



Reachability is **not a symmetric relation**: no observation can be reached from a **non-core point**, but such a point may be reachable.

We can build a **symmetric relation** on **non-outlying** observations, however:

$$\mathbf{p}, \mathbf{q} \in \mathbf{X} \setminus \{\text{outliers}(\mathbf{X})\}$$

are said to be **density-connected** for $\varepsilon^* > 0$ and $d$ if there is an observation $\mathbf{o} \in \mathbf{X}$ such that $\mathbf{p}, \mathbf{q} \in V_{\mathbf{o}}$, with $|V_{\mathbf{o}}| \geq minPts$.

The same $\mathbf{p}, \mathbf{q}$ are **density-connected in a path** if either they are **density-connected** or if there is a sequence of observations

$$\mathbf{p} = \mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{k-1}, \mathbf{r}_k = \mathbf{q}$$

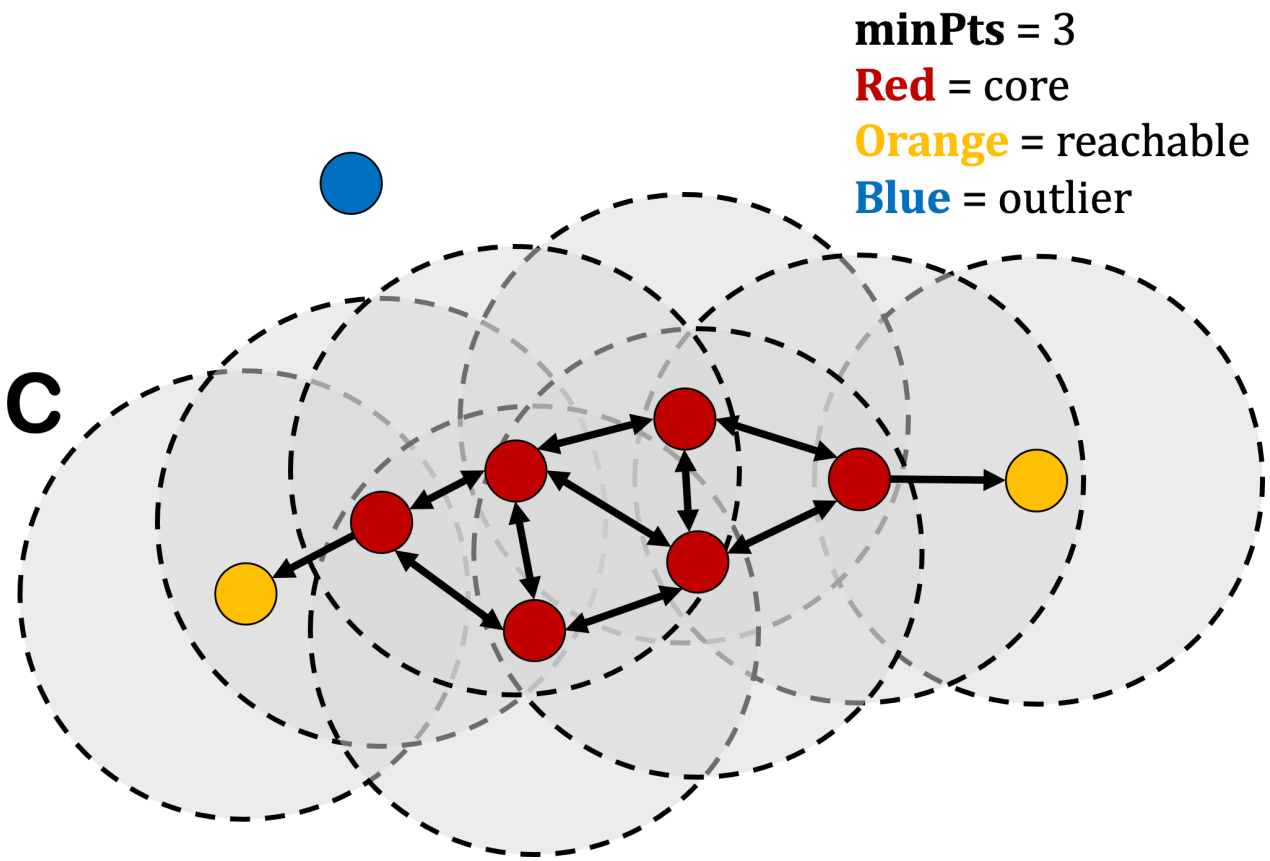
such that $\mathbf{r}_{i-1}, \mathbf{r}_i$ is **density-connected** for all $i = 1, \ldots, k$.

That this is a relation on $\mathbf{X} \setminus \{\text{outliers}(\mathbf{X})\}$ is clear:

- it is **reflexive** as every $\mathbf{x} \in \mathbf{X} \setminus \{\text{outliers}(\mathbf{X})\}$ is either **reachable** or a **core** observation, so that $\exists \mathbf{o_x} \in \mathbf{X}$ with $\mathbf{x} \in V_{\mathbf{o_x}}$ and $|V_{\mathbf{o_x}}| \geq minPts$, and so $\mathbf{x}$ is **density-connected** to itself;

- it is **symmetric** (using the reverse sequence) and **transitive** (using the concatenation of sequences), by construction.

Essentially, DBSCAN clusters are composed of observations **density-connected in a path**.




Density path connection in a DBSCAN cluster $C$: arrows represent density-connection: each orange observation is within reach of a red one, but no observation can be reached from the orange points.

# Algorithm

1. Select an observation **at random** not yet assigned to a cluster, from the list of not previously selected observations;
2. determine the selected observation's type (outlier, non-core, core);
3. if an outlier or a non-core points, assign to the **noise cluster**;
4. otherwise, build the **network of density-connected paths**;
5. assign all observations in the network to a **stand-alone cluster**;
6. repeat steps 1 to 5 until all points have assigned to a cluster.

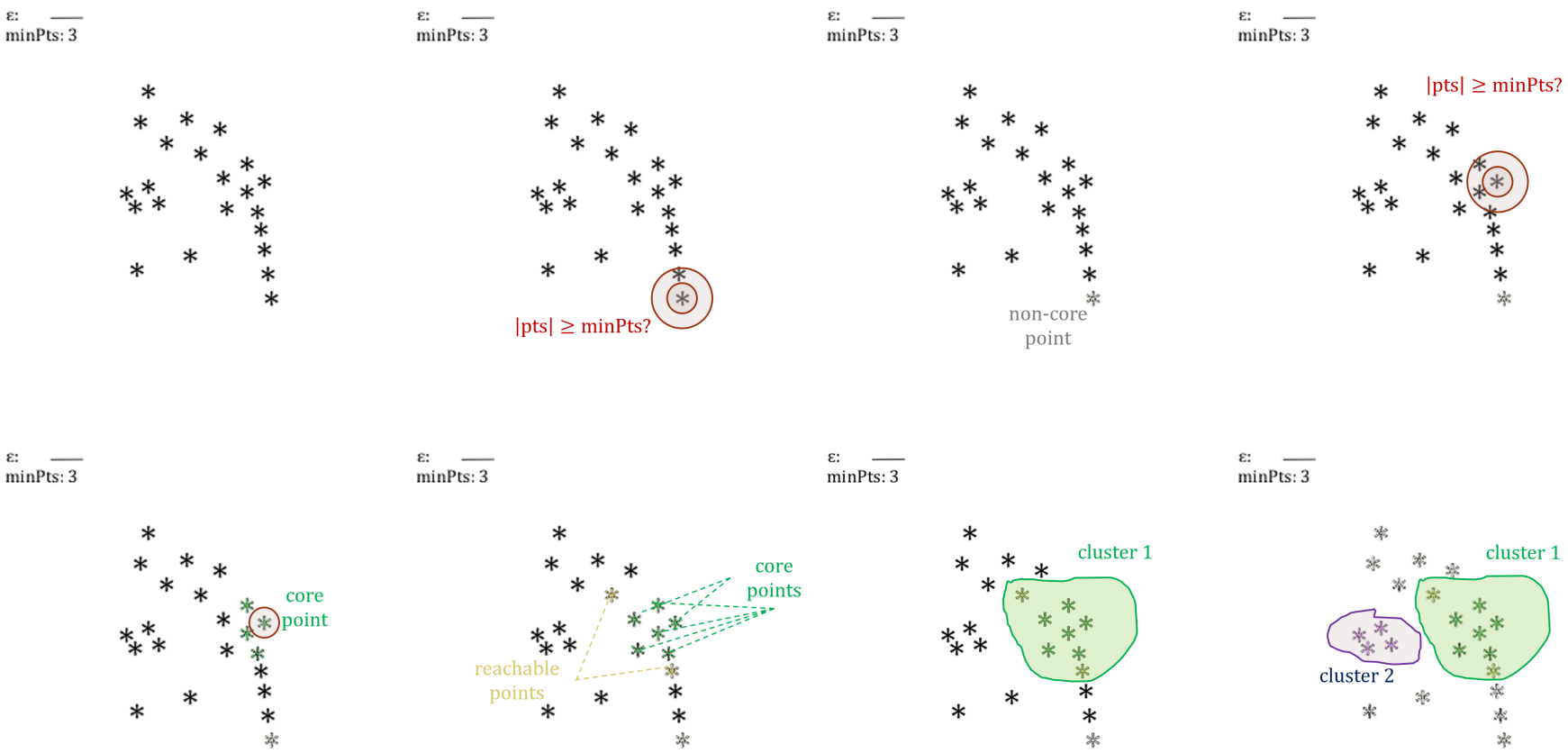



Illustration of DBSCAN on an artificial dataset.

DBSCAN clusters must contain **at least one core point**, by definition.

Small groups of observations that are not **density-connected to any core points** are assigned to the **noise** cluster.

The observations in the noise cluster are typically identified as **outliers** $\Longrightarrow$ DBSCAN is a reasonable UL approach for **anomaly detection**.

A **non-core** point that has been assigned to the **noise** cluster may end up being assigned to a **stand-alone** cluster at a **later stage** $\Longrightarrow$ the opposite **cannot occur**.

It is possible for two clusters to **share** non-core points $\Longrightarrow$ they are randomly assigned to a cluster; consequently, some clusters may end up containing **fewer than** $minPts$ observations.

# 4.4.2 – Advantages and Limitations

The main **advantages** of DBSCAN are that:

- there is no need to specify the **number** of clusters to find in the data;
- clusters of **arbitrary shapes** can be discovered;
- observations that are "noisy"/outlying are not forced into a cluster;
- the clusters are **robust** with respect to outliers, and
- it only requires **two** parameters ($\varepsilon^* > 0$ and $minPts$) to run, which can be set by **domain experts** if the data is well understood.

**Suggestions:** use $minPts \geq p + 1$, with larger values being preferable for **noisy** datasets, or $minPts \geq 2p$ for **large** datasets or sets with **duplicates**.

If $\varepsilon^* > 0$ is too **small**, a **large** portion of the observations will be **outliers**; if it is too **large**, a **large** proportion will be found in a **single cluster**. Small values are preferable... **but how small is too small**?

Parameter and distance choices impact the DBSCAN results; pick $d$ **before** $\varepsilon^*$ to avoid **data dredging** and "begging the question".

**Question:** given that DBSCAN can handle **globular** and **non-globular** clusters, why not always use it?

**Answer:** **computational efficiency**. For a dataset with $n$ observations, $k-$means algorithm has order $O(nk)$; DBSCAN has order $O(n \log n)$ $\Longrightarrow$ when $n \nearrow$, the DBSCAN runtime $\nearrow$ **faster** than the $k-$means runtime.

DBSCAN cluster density is assumed to be **constant**.

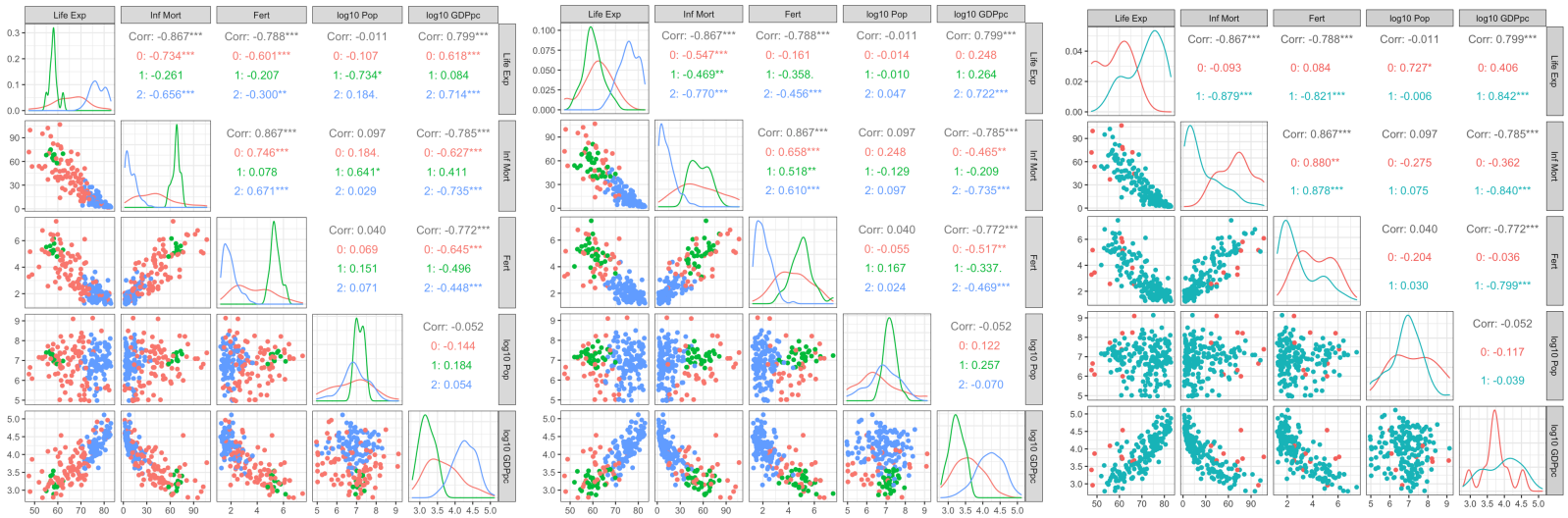
Are there **two clusters** in the image? A **loose** one (bottom/left) and a **tight** one (top/right), as well as some **outliers** around the **tight** cluster.

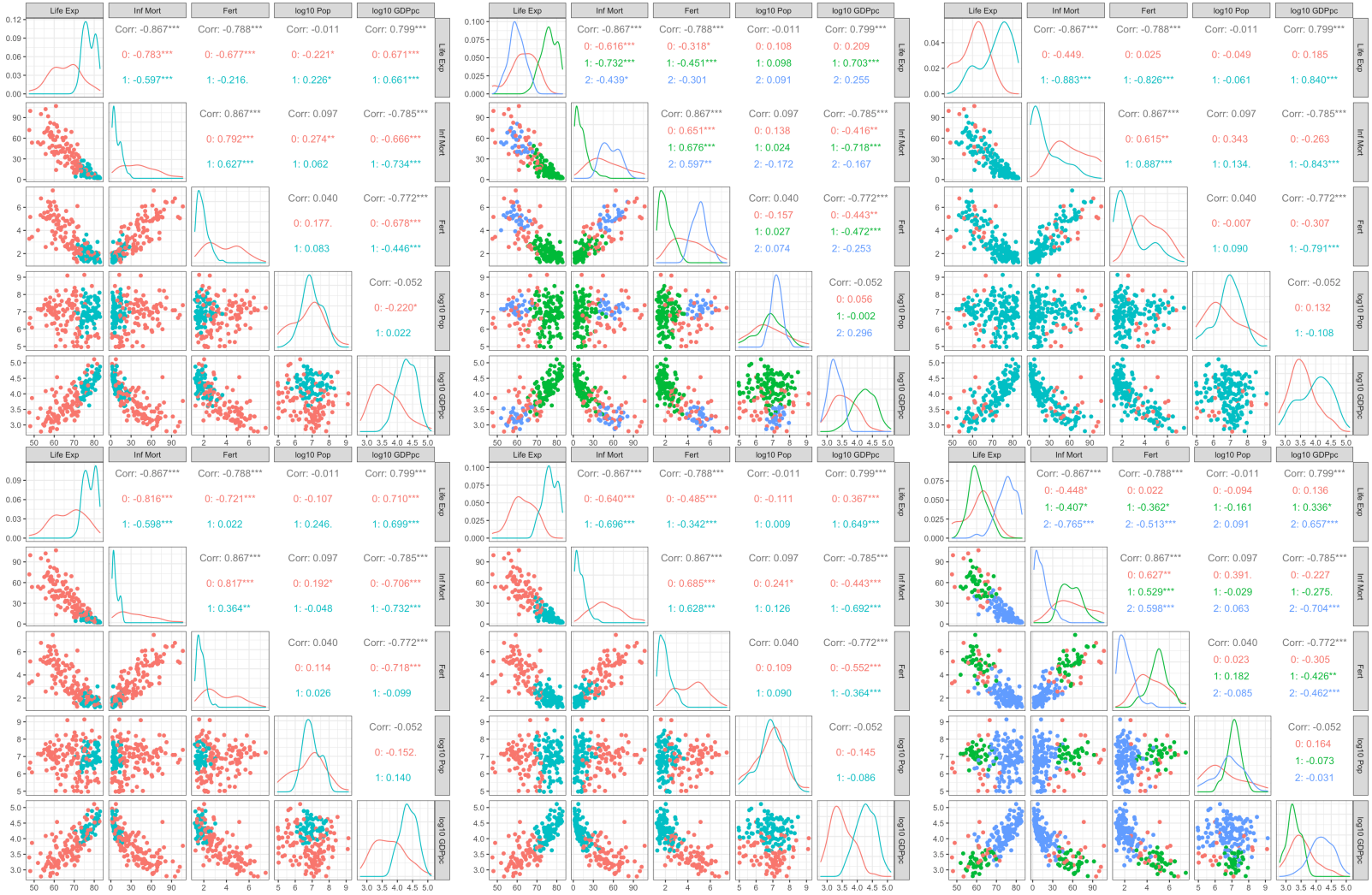
DBSCAN cannot discover this structure: either it finds **no outliers**, or it only finds the one **tight cluster**.

# 4.4.3 – Example: Gapminder Dataset

We re-visit the (scaled) 2011 Gapminder dataset, using DBSCAN with Euclidean dissimilarity (code: DUDADS, 22.4.1), with

$$\left[\varepsilon^* \in \{0.75, 1, 1.25\} \textbf{ (columns)}\right] \times \left[minPts \in \{6, 10, 15\} \textbf{ (rows)}\right].$$




Life Exp
Inf Mort
Fert
log10 Pop
log10 GDPpc
Corr: -0.867***
Corr: -0.788***
Corr: -0.011
Corr: 0.799***
0: -0.783***
0: -0.677***
0: -0.221*
0: 0.671***
1: -0.597***
1: -0.216.
1: 0.226*
1: 0.661***
Corr: 0.867***
Corr: 0.097
Corr: -0.785***
0: 0.792***
0: 0.274**
0: -0.666***
1: 0.627***
1: 0.062
1: -0.734***
Corr: 0.040
Corr: -0.772***
0: 0.177.
0: -0.678***
1: 0.083
1: -0.446***
Corr: -0.052
0: -0.220*
1: 0.022
Corr: -0.867***
Corr: -0.788***
Corr: -0.011
Corr: 0.799***
0: -0.616***
0: -0.318*
0: 0.108
0: 0.209
1: -0.732***
1: -0.451***
1: 0.098
1: 0.703***
2: -0.439*
2: -0.301
2: 0.091
2: 0.255
Corr: 0.867***
Corr: 0.097
Corr: -0.785***
0: 0.651***
0: 0.138
0: -0.416**
1: 0.676***
1: 0.024
1: -0.718***
2: 0.597**
2: -0.172
2: -0.167
Corr: 0.040
Corr: -0.772***
0: -0.157
0: -0.443**
1: 0.027
1: -0.472***
2: 0.074
2: -0.253
Corr: -0.052
0: 0.056
1: -0.002
2: 0.296
Corr: -0.867***
Corr: -0.788***
Corr: -0.011
Corr: 0.799***
0: -0.449.
0: 0.025
0: -0.049
0: 0.185
1: -0.883***
1: -0.826***
1: -0.061
1: 0.840***
Corr: 0.867***
Corr: 0.097
Corr: -0.785***
0: 0.615**
0: 0.343
0: -0.263
1: 0.887***
1: 0.134.
1: -0.843***
Corr: 0.040
Corr: -0.772***
0: -0.007
0: -0.307
1: 0.090
1: -0.791***
Corr: -0.052
0: 0.132
1: -0.108
Corr: -0.867***
Corr: -0.788***
Corr: -0.011
Corr: 0.799***
0: -0.816***
0: -0.721***
0: -0.107
0: 0.710***
1: -0.598***
1: 0.022
1: 0.246.
1: 0.699***
Corr: 0.867***
Corr: 0.097
Corr: -0.785***
0: 0.817***
0: 0.192*
0: -0.706***
1: 0.364**
1: -0.048
1: -0.732***
Corr: 0.040
Corr: -0.772***
0: 0.114
0: -0.718***
1: 0.026
1: -0.099
Corr: -0.052
0: -0.152.
1: 0.140
Corr: -0.867***
Corr: -0.788***
Corr: -0.011
Corr: 0.799***
0: -0.640***
0: -0.485***
0: -0.111
0: 0.367***
1: -0.696***
1: -0.342***
1: 0.009
1: 0.649***
Corr: 0.867***
Corr: 0.097
Corr: -0.785***
0: 0.685***
0: 0.241*
0: -0.443***
1: 0.628***
1: 0.126
1: -0.692***
Corr: 0.040
Corr: -0.772***
0: 0.109
0: -0.552***
1: 0.090
1: -0.364***
Corr: -0.052
0: -0.145
1: -0.086
Corr: -0.867***
Corr: -0.788***
Corr: -0.011
Corr: 0.799***
0: -0.448*
0: 0.022
0: -0.094
0: 0.136
1: -0.407*
1: -0.362*
1: -0.161
1: 0.336*
2: -0.765***
2: -0.513***
2: 0.091
2: 0.657***
Corr: 0.867***
Corr: 0.097
Corr: -0.785***
0: 0.627**
0: 0.391.
0: -0.227
1: 0.529***
1: -0.029
1: -0.275.
2: 0.598***
2: 0.063
2: -0.704***
Corr: 0.040
Corr: -0.772***
0: 0.023
0: -0.305
1: 0.182
1: -0.426**
2: -0.085
2: -0.462***
Corr: -0.052
0: 0.164
1: -0.073
2: -0.031

The noisy observations are shown in red: two immediate insights are that the number of **outlying** observations $\searrow$ as $\varepsilon^* \nearrow$, and $\nearrow$ as $minPts \nearrow$.

DBSCAN can produce clusters which do not stretch our notions of clusters: problematic observations are explained away as **outliers**.

The various runs find either **noisy** observations, and $1$ or $2$ **stand-alone** clusters (could change with different parameter values).

In the realization with $\varepsilon^* = 1$ and $minPts = 6$, we have:

| | **noise** | **cluster 1** | **cluster 2** |
|---|---|---|---|
| outlier | 34 | – | – |
| reachable | – | 10 | 17 |
| core | – | 20 | 103 |
| **total** | **34** | **30** | **120** |

# 4.5 – Clustering Evaluation

Clustering algorithms attempt to separate the data into **natural groups**, using different conceptual approaches (each with their advantages):

- $k-$means tries to minimize **within-cluster variation**;
- HC builds a **global clustering structure**;
- DBSCAN uses the associated **clustering network** and **reachability**, etc.

**Main take-away**: they may yield **different clustering outcomes** depending on the **analytical choices** made along the way $\Longrightarrow$ UL is **difficult**!

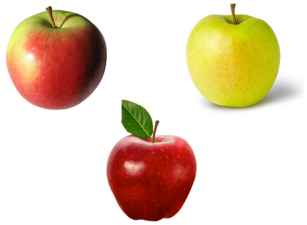
**Clustering hallmark:** whenever a new approach overcomes a difficulty, it does so **at the cost of introducing holes in previously sound aspects**.

We cannot speak of the "**best**" clustering approach – instead, we try to identify which of several schemes is "**best-suited**" **for a given task**.

# 4.5.1 – Clustering Assessment

Clustering **groups** objects based on their overall **similarity** to each other.

**Avoid:** focusing on **a few** attributes **only**, especially for **visual** or "**eyeball**" clustering – **all** the attributes must "**come along for the ride**".



- They are all **apples**; they all have **stems**; they are of **similar sizes**.
- Two of them are **(mostly) red**; two of them have **no stem leaf**; two of them are **"tapered" at the bottom**.

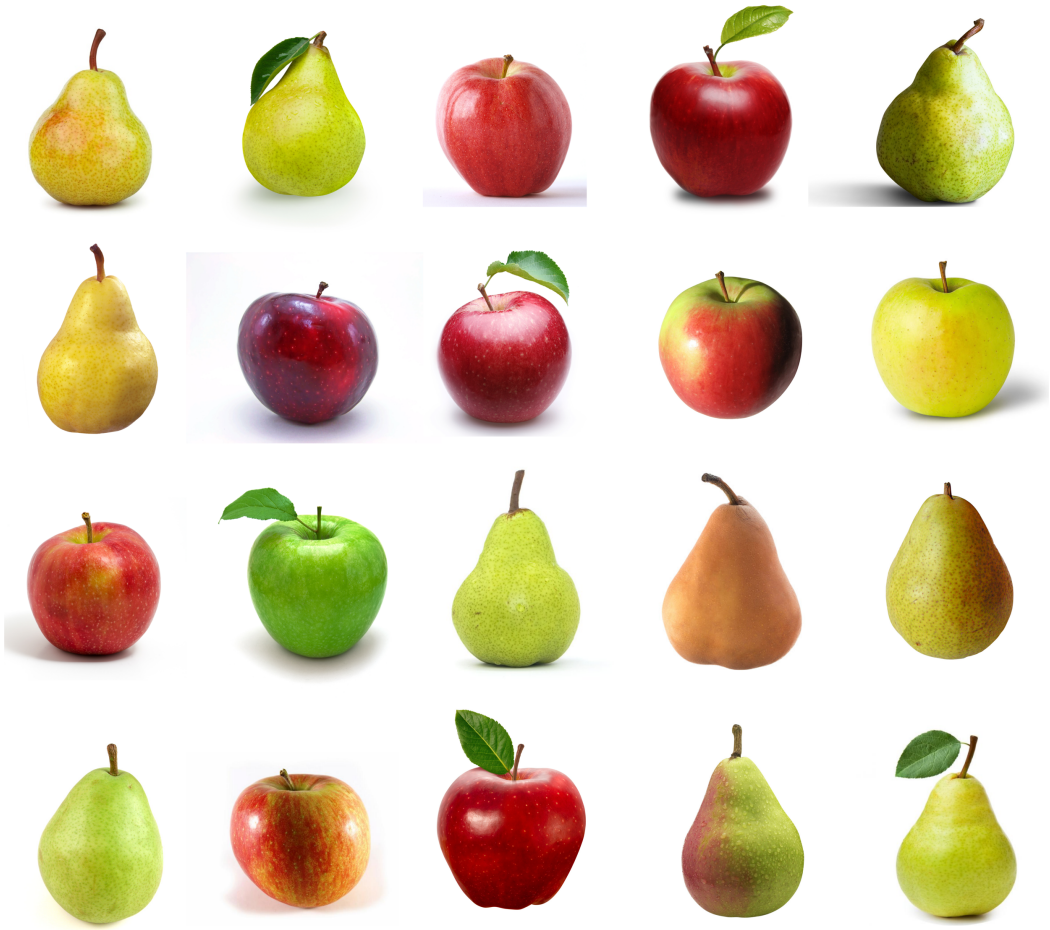
- What is the **same** about these objects?
- What is **different**?
- Do they belong to the same **group**?

In a way, each is **unlike** the other 2 (and so **like** another one?).

# Fruit Image Dataset

In order to appreciate the **challenges** presented by clustering validation, it will be helpful to relate the concepts to something tangible. We will explore some of these notions through an artificial dataset of 20 fruit images:

- are there **right** or **wrong** clusterings of this dataset?
- are there multiple possible **'natural' clusterings**?
- could **different** clusterings be used for **different** tasks?
- will some clusterings be of **(objectively) higher quality** than others?



Toy dataset used to illustrate the key concepts of clustering validation

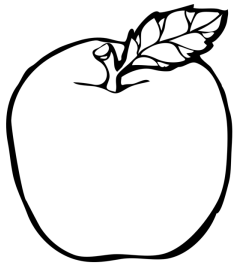
# Key Notions

Fundamentally, clustering relies on the notion of **representativeness**.

Ideally, the **essence** of a cluster's instances (observations) should be **faithfully captured** by the cluster **concept** (**examplar**, **representative**).

Each cluster **concept** could also be used to **differentiate** the clusters.

These are **concepts** for "**apples**":



"The fleshy, usually rounded red, yellow, or green edible pome fruit of a usually cultivated tree (genus Malus) of the rose family." [Merriam-Webster]

This is not *all* that an apple **is**, but most humans who have **seen** or **eaten** an apple **at some point** will have no trouble recognizing what the concepts **allude to**, even if our corresponding mental images differ.

A cluster concept is a **generalized representation** – it capture "**something**" of the cluster's instances.

**Question:** for each cluster, can we clearly identify a concept capturing its **essence**? If so, is the entire scheme then a **good** one?

In DS/ML, we use **signature vectors** to represent **instance properties**.

Each vector element represents an instance **attribute**; the element's value is the **measured value** of the corresponding object's property (for instance, the **colour** of the apple).

The apple below could be described by the **signature vector**:

$(12, 9.12, \text{tapered}, \text{golden delicious})$

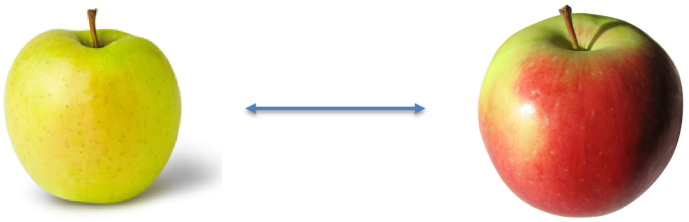
The components are the instance's:

- **colour** (ordinal)
- **height** (continuous)
- **shape** (categorical)
- **variety** (categorical)

**Important consideration:** does the **signature** provide a **sufficient description** of the associated object or is it **too crude** to be of use? (hard to determine **before** the analysis)

Signature vectors are used to compare objects (**instance-to-instance relationships**) $\Longrightarrow$ measure of **similarity** between instances.

Clustering comparisons require a similarity measure **across all dimensions**.

We could compare the two objects  by comparing their signature vectors:

$$\mathbf{v}_1 = (12, 9.12, \text{tapered}, \text{golden delicious})$$
$$\mathbf{v}_2 = (2, 10.43, \text{spherical}, \text{macintosh})$$

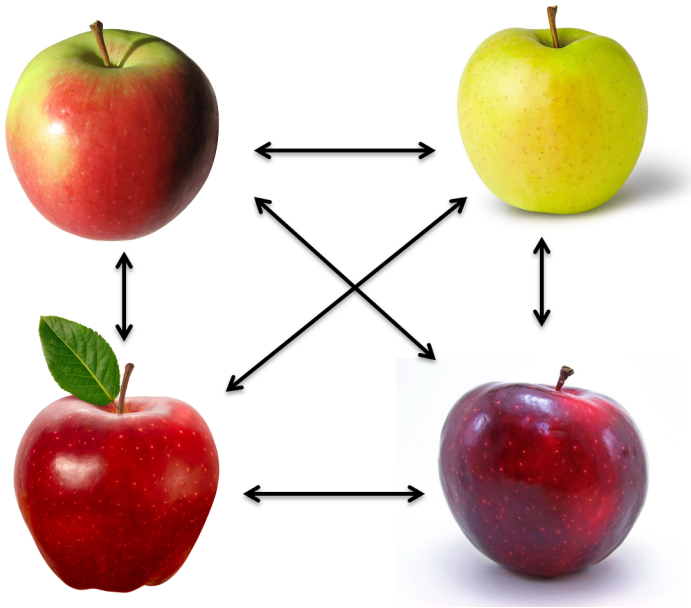
*via* a **similarity measure** $w(\mathbf{v}_1, \mathbf{v}_2)$.

**Different** similarity measures may yield **different** results: in some cases showing them to be **similar**, in others to be **dissimilar**.

**Distances** are often used to define how **dissimilar** two objects are.

It is possible to convert categorical attributes to numeric ones, or to define **mixed distances/similarity** measures.

In the **clustering framework**, we are often interested in **all pairwise** similarities between objects, not just in the similarity between two objects; pairs may be interesting **in relation to other pairs**.



In a dataset with $4$ objects, say, we might require the computation of $6$ pairwise similarities.

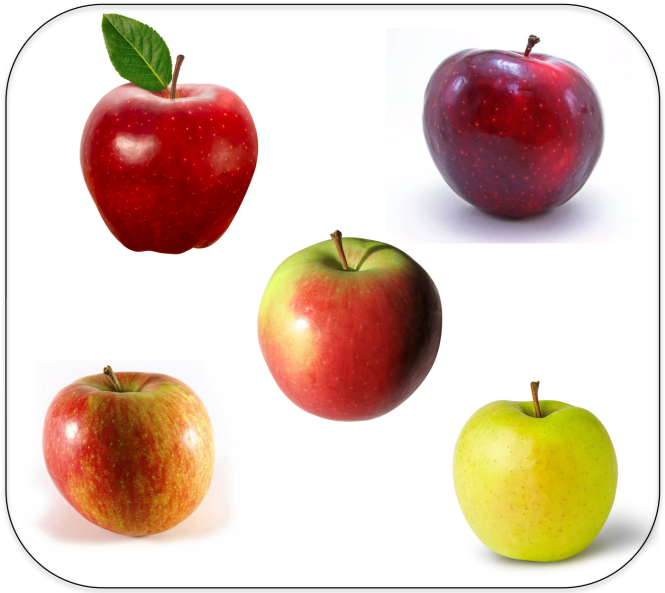
As with objects, **clusters** also have **properties**.

These could include:

- the **number of instances** in a cluster;
- **similarity stats** across instances within a cluster (**min**, **max**, **avg**, **med**, **sd**, **mad**, **var**, etc.);
- the **cluster representative**, which may be an actual instance, or an amalgamation of multiple instances (**exemplar**).

**How many instances** are there in the cluster to the right, for instance? What pair of observations is **most similar**? **Least similar**?

What are the **similarity statistics**?
Which instance is **most representative**?



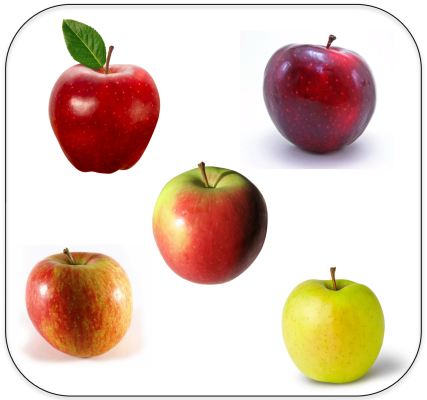
**Cluster-to-instance** relationships can also be defined.

A **specific** instance can compare to:

- a cluster **representative**;
- **specific instances** in a cluster (**most similar** instance, **most distant** instance, etc.)

$\Longrightarrow$ allows **membership** questions

- is the green apple **similar** to the cluster below?
- does it **belong** in the cluster, or is it most likely to **belong in another** cluster?
- or perhaps to **no cluster in particular**?



We might be interested in **cluster-to-cluster** relationships, where we compare **cluster-level** properties:

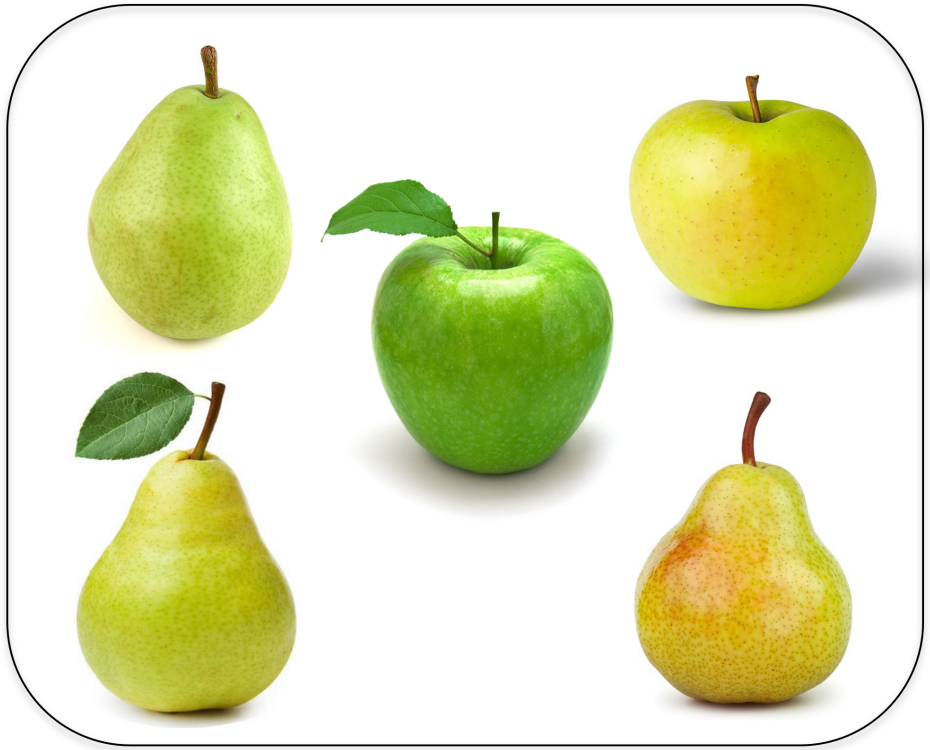
- **number** of instances;
- **within-cluster** similarities;
- cluster **representatives**;

We can also add **between-cluster** similarities to determine if the instances are **notably different** from one cluster to the next.

$\Longrightarrow$ allows **validity** questions:

- are the two clusters below **significantly different**?
- should they be joined into a **mega-cluster**?
- does it make sense to have them as **separate clusters** in the dataset?

How would we qualify the clustering outcome below, as it relates to **colour**, **height**, and **shape**? Could there be clusterings of **higher quality**? Of **lower quality**? How could this be **quantified**?



# 4.5.2 – Clustering Quality Measures

**Cluster-** and **instance-** comparisons can be combined in many different ways to generate **clustering validation functions**.

**Central cluster validation question:** what can these tell us about the **quality** of a particular clustering outcome relative to:

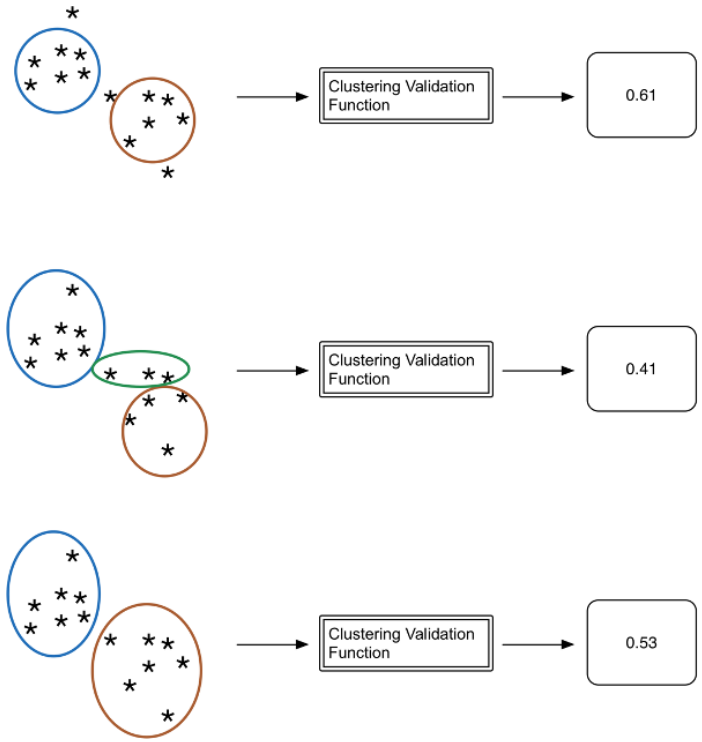
- some objective criteria about "good" clustering schemes (**internal validation**);
- another clustering option (**relative validation**);
- external information (**external validation**)?

In general, clustering involves two main activities:

- **creating/building** the clusters, and
- **assessing their quality**, individually and as a whole.

From a practical perspective, clustering requires two functions:

1. one which assigns each instance to a **cluster** (or prob.);
2. one which assigns each clustering scheme to a **cluster quality measurement**.




- **top** – two clusters are found (with outliers), and the quality of the clustering is assessed as $0.61$
- **middle** – three clusters are found (no outliers), with quality assessment at $0.41$
- **bottom** – two clusters are found (no outliers), with quality assessment at $0.53$

It will come as no surprise that a large number of **clustering validation functions** exist in practice.

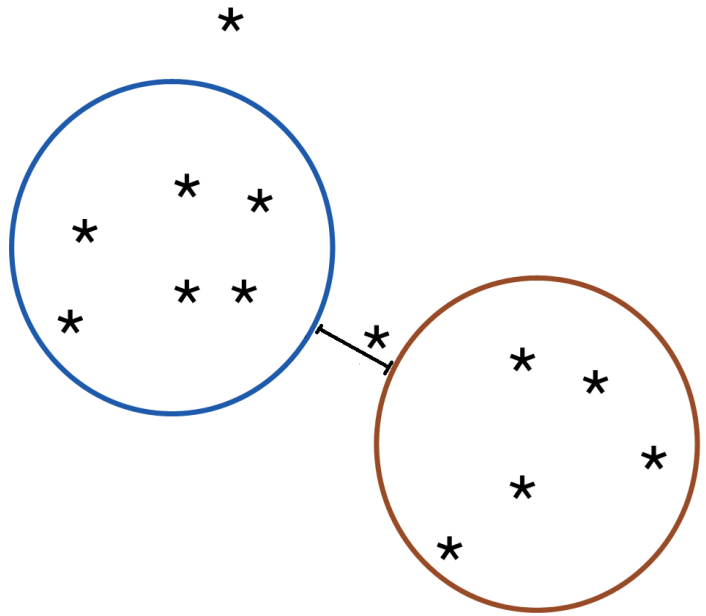
But they are all built out of the **basic measures** relating to instance or cluster properties we have already reviewed:

- **instance properties**;
- **cluster properties**;
- **instance-to-instance relationship properties**;
- **cluster-to-instance relationship properties**, and
- **cluster-to-cluster relationship properties**.

# 4.5.3 – Internal Validation

Clustering **quality** depends on **context**. But if there is no context? **Internal validation** tries to measure cluster quality **objectively** (**without context**).

We may chose to only use the resulting clusters' **properties**.

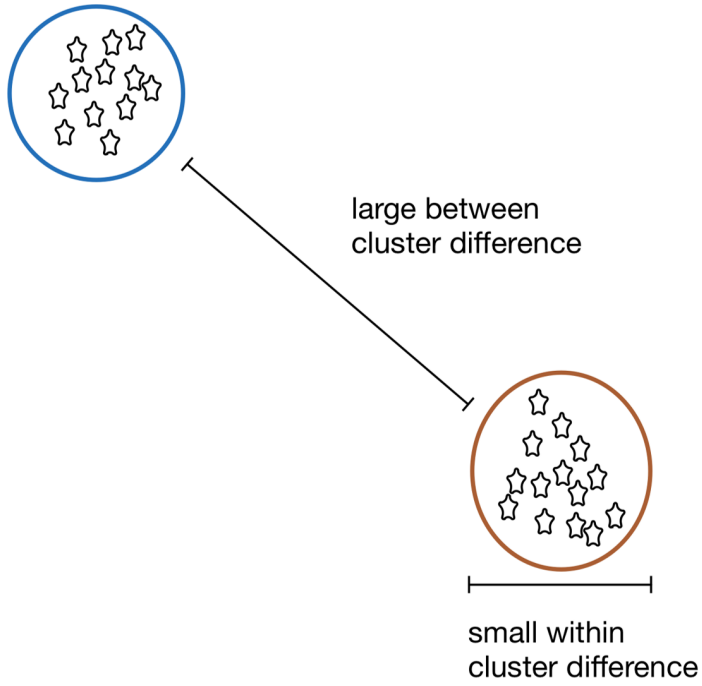


If the **average between-cluster distance** is large, say, there is some evidence that the resulting clusters provide a good **segmentation** of the data into **natural groups**,

Or we might validate cluster quality by tempering the **average between-cluster distance** vs. the **average within-cluster distance** between the instances.

$\Longrightarrow$ This rewards "**tight**" and "**isolated**" clusters, as opposed to simply "**isolated**" ones.

Easy, right?




There are multiple ways of including both **between-cluster** and **within-cluster** variances in a **cluster quality metric** (CQM).

The broad objectives of clustering are **universal**: instances **within** a cluster should be **similar**; instances in **different** clusters should be **dissimilar**.

**Problem:** there are **many ways** for clusters to **deviate** from this **ideal**. How do we weigh the "**good**" aspects (high **within-cluster similarity**, say) relative to the "**bad**" ones (low **between-cluster separation**, say)?

Other internal properties and relationships can also be used and combined in various ways – there are 25+ CQMs, including:

- **Davies-Bouldin**
- **Dunn**
- **Silhouette**
- **WSS**

It is useful to have access to so many CQMs, but the **No-Free Lunch theorems** apply: none of them is **universally superior** to any of the others.

# Within Sum of Squares

Let $\mathcal{C} = \{C_1, \ldots, C_K\}$ be the $K$ clusters obtained from a dataset $\mathbf{X}$ *via* some algorithm $\mathcal{A}$. Denote the **centroid** (or some other central representative) of cluster $C_k$ by $\mathbf{c}_k$. The **(within) sum of error** for $\mathcal{C}$ is

$$\text{WSE} = \sum_{k=1}^{K} \sum_{\mathbf{x} \in C_k} \text{dissimilarity}(\mathbf{x}, \mathbf{c}_k).$$

The dissimilarity is often selected to be the **square of the Euclidean distance**, but that choice is **arbitrary** – it would make more sense to use a dissimilarity related to $\mathcal{A}$'s **similarity measure**.

WSE $\searrow$ when the number of clusters $K$ $\nearrow$, and **optimality** is obtained at **points of diminishing returns** (use **(within) average error** instead?)

# Davies-Bouldin Index

Using the notation from the previous slide, let

$$s_k = \underset{\mathbf{x} \in C_k}{\text{Avg}} \left\{ \text{dissimilarity}(\mathbf{x}, \mathbf{c}_k) \right\}, \quad k = 1, \ldots, K.$$

The **Davies-Bouldin index** is defined as

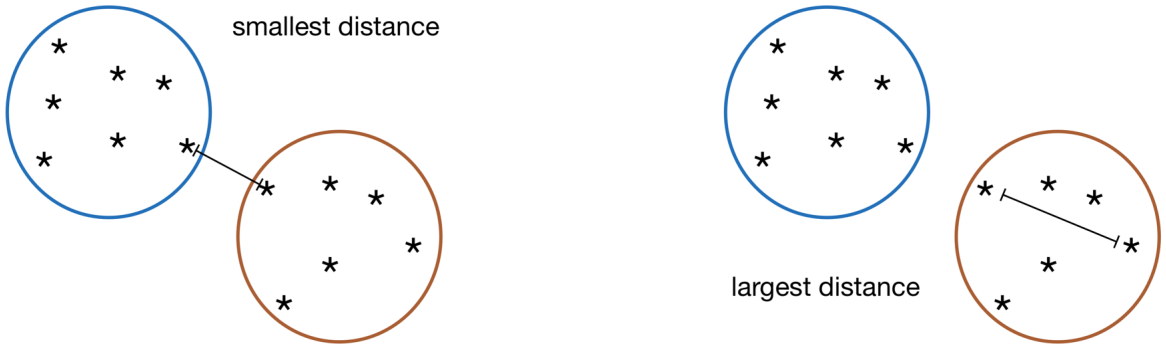
$$\text{DBI} = \frac{1}{K} \sum_{k=1}^{K} \max_{\ell \neq k} \left\{ \frac{s_k + s_\ell}{\text{dissimilarity}(\mathbf{c}_k, \mathbf{c}_\ell)} \right\}.$$

If the clusters $\{C_k\}$ are **tight** and **dissimilar to one another**, we expect the numerators $s_k + s_\ell$ to be **small** and the denominators $\text{dissimilarity}(\mathbf{c}_k, \mathbf{c}_\ell)$ to be **large**, so that the DBI would be **small**.

# Dunn's Index

With clusters that are **loose** or **somewhat similar to one another**, we expect $\mathrm{DBI}$ to be **large**.

**Dunn's index** is another way to assess **separation** and **tightness**; it is the ratio of the **smallest between-cluster dissimilarity** (for pairs of observations **not in the same cluster**) to the **largest cluster diameter** (**within-cluster dissimilarity**).



If the clusters $\{C_k\}$ are **tight** and **dissimilar to one another**, we expect the **smallest between-cluster dissimilarity** to be **large** and the **largest cluster diameter** to be **small**, leasing to a **large** Dunn ratio.

Conversely, with clusters that are **loose** or **somewhat similar to one another**, the Dunn ratio will be **small**.

As is the case with $\mathrm{WSE}$ and $\mathrm{DBI}$, the choice of the dissimilarity (or distance) measure leads to different variants of the Dunn index, but all of them take **non-negative values**.

**Denominator variants:**

- mean dissimilarity **between pairs of observations**;
- mean dissimilarity to the **cluster centroid**.

# Silhouette Metric

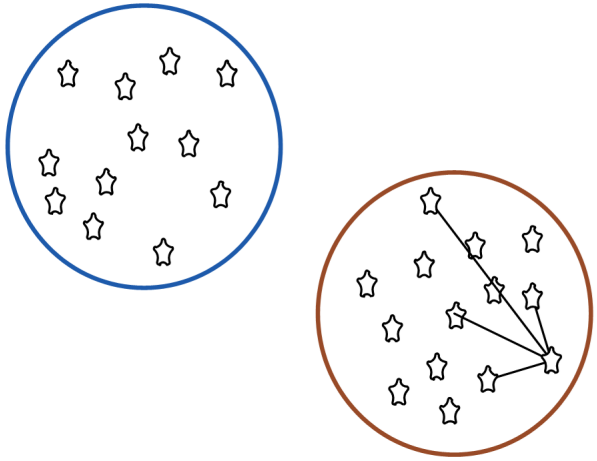
The previous CQM compute a validation metric for the **entire clustering outcome**; the **silhouette metric** instead assigns a score **to each observation**, and aggregate them at the **cluster** and **dataset** levels.

For a dissimilarity $d$ and a point $\mathbf{x}$ in a cluster $C$, set

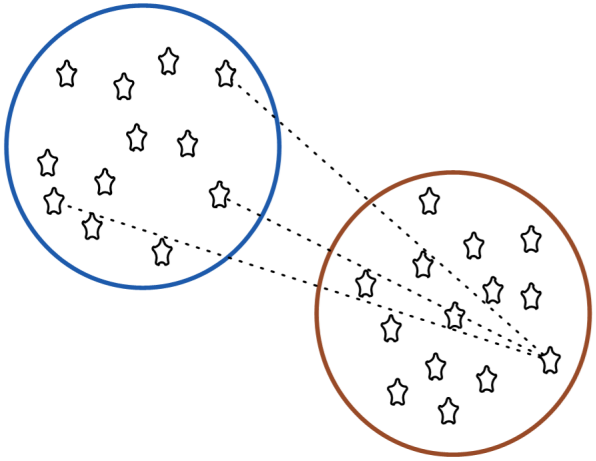
$$b(\mathbf{x}) = \min_{C' \neq C} \left\{ \operatorname*{Avg}_{\mathbf{y} \in C'} \{d(\mathbf{x}, \mathbf{y})\} \right\}, \quad a(\mathbf{x}) = \operatorname*{Avg}_{\substack{\mathbf{w} \in C \\ \mathbf{w} \neq \mathbf{x}}} \{d(\mathbf{x}, \mathbf{w})\}.$$

**Small** values of $a(\mathbf{x})$ imply that $\mathbf{x}$ is **similar** to the instances in **its** cluster; **large** values of $b(\mathbf{x})$ imply that it is **dissimilar** to instances in **other** clusters.

average distance to points in own cluster (low is good)



average distance to points in neighbouring cluster (high is good)



The silhouette metric at $\mathbf{x}$ is

$$\text{silhouette}(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\{a(\mathbf{x}), b(\mathbf{x})\}} = \begin{cases} 1 - a(\mathbf{x})/b(\mathbf{x}) & \text{if } a(\mathbf{x}) < b(\mathbf{x}) \\ 0 & \text{if } a(\mathbf{x}) = b(\mathbf{x}) \\ b(\mathbf{x})/a(\mathbf{x}) - 1 & \text{if } a(\mathbf{x}) > b(\mathbf{x}) \end{cases}$$

We have $-1 \leq \text{silhouette}(\mathbf{x}) \leq 1$ for all $\mathbf{x}$.

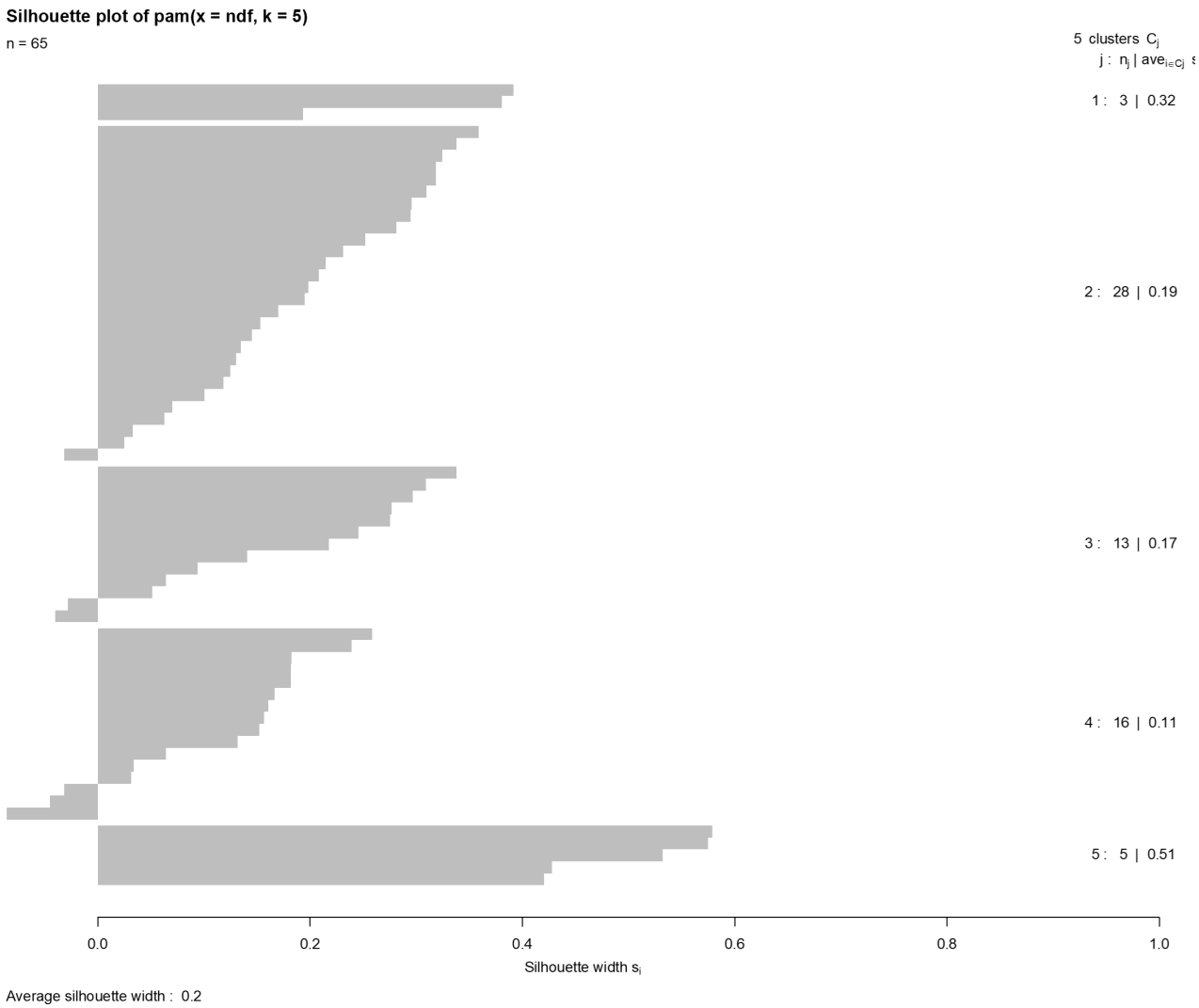
$\text{silhouette}(\mathbf{x})$ **large** $(> 0) \implies a(\mathbf{x})/b(\mathbf{x})$ is **small** $\implies \mathbf{x}$ is **correctly assigned** to cluster $C$ (and *vice-versa*).

The **silhouette diagram** of a clustering outcome displays the silhouette scores **for each observation**, and averages the scores **in each cluster**.

The **mean** over **all** observations and the **# of instances** that have been **mis-assigned** provide an indication of the **appropriateness** of the clusters.

In the next example, $65$ observations are separated into $5$ clusters: $6$ observations are **poorly assigned** (negative silhouette scores) and the **average silhouette score** over the entire dataset is $0.2 \implies$ the clustering is more **successful** than **unsuccessful**, overall.

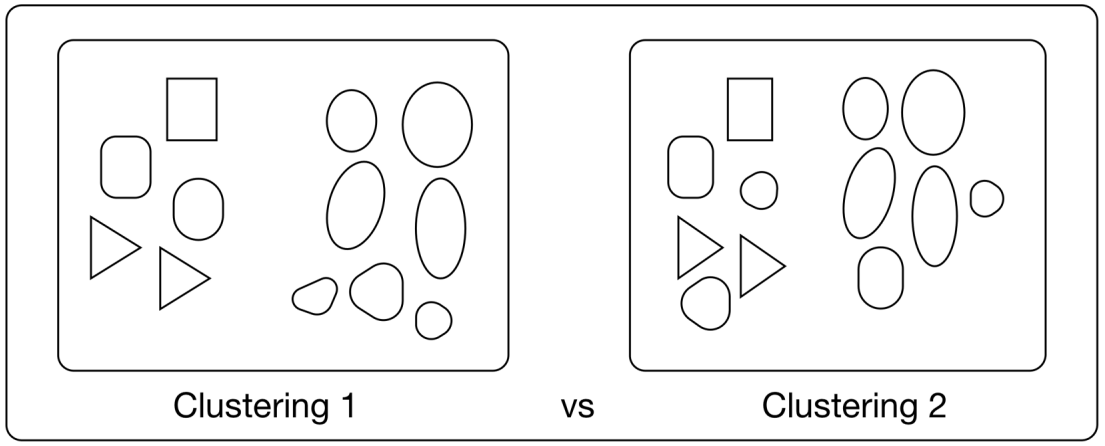
Is $0.2$ good or bad? Depends on **context** $\implies$ contrast with other CQM.


Silhouette plot of pam(x = ndf, k = 5)
n = 65
5 clusters $C_j$
j : $n_j$ | $ave_{i \in Cj}$ s
1 : 3 | 0.32
2 : 28 | 0.19
3 : 13 | 0.17
4 : 16 | 0.11
5 : 5 | 0.51
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width $s_i$
Average silhouette width : 0.2

# 4.5.4 – Relative Validation

A **single validation measure** for a **single clustering outcome** is not always **useful**. Could the results be **better**?

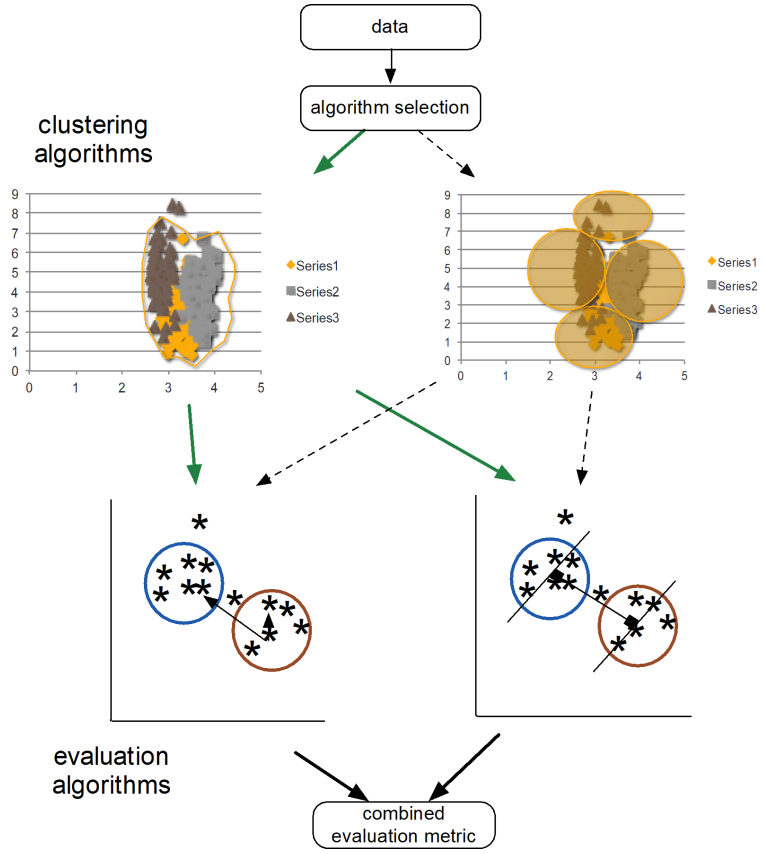
**One approach:** compare clustering outcomes across **multiple runs** with different **parameter values** and **initialization states**.

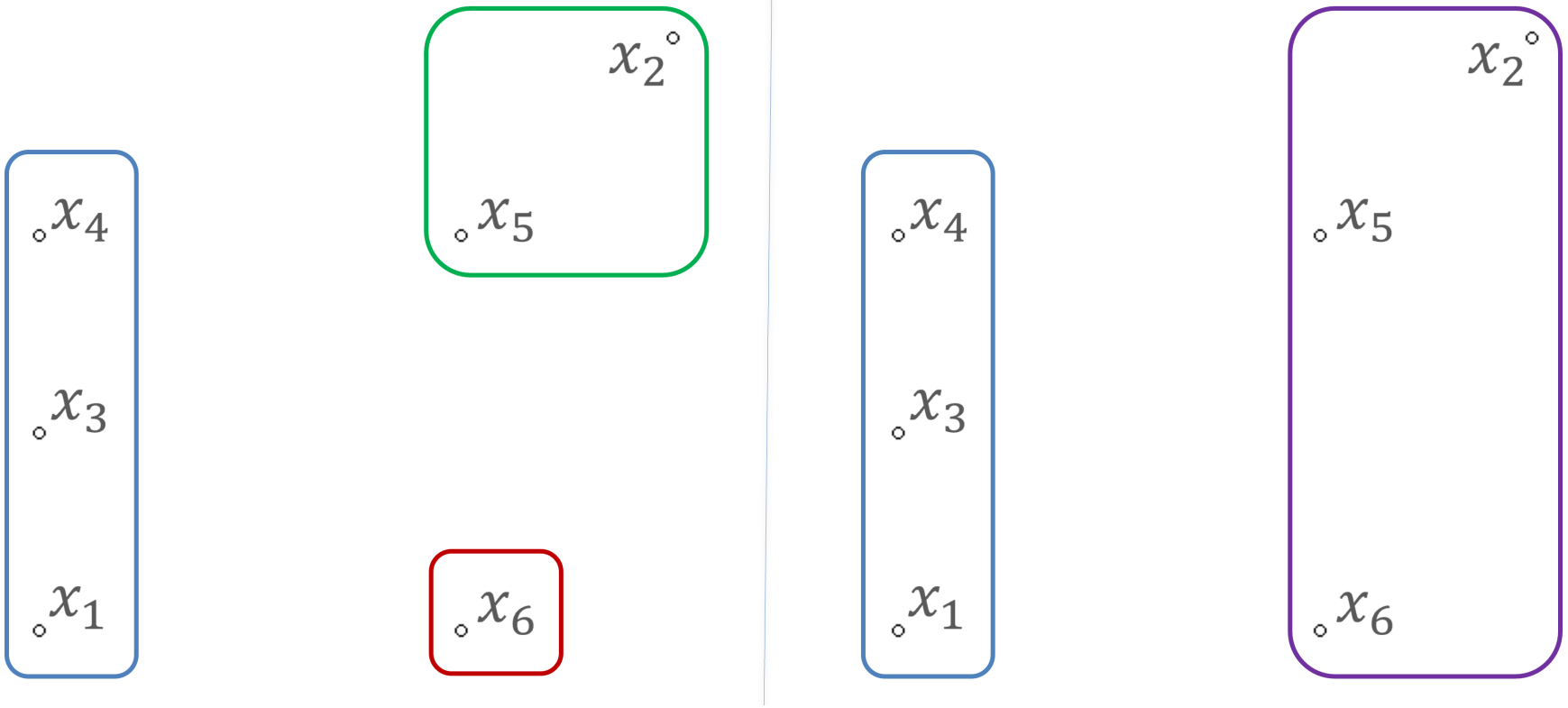


When the outcomes of different clustering algorithms on the same dataset are "**similar**", we have evidence in favour of the resulting clusters being **efficient**, **natural**, or **valid** (in some sense).

For **two** schemes, we can compute the "**correlation**" between them.

Consider a dataset with 6 instances, clustered in two different ways ($\mathcal{A}, \mathcal{B}$, with $|\mathcal{A}| = 3, |\mathcal{B}| = 2$), as on the next page. How similar are $\mathcal{A}$ and $\mathcal{B}$?






Intuitively, we expect the similarity between $\mathcal{A}$ and $\mathcal{B}$ to be **high**, since the two outcomes are **not that different** from one another $\Longrightarrow$ in $\mathcal{B}$, the **two smallest clusters** of $\mathcal{A}$ have been **merged** into a **single, larger** cluster.

# This can be represented in matrix form:

| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 1 | | | | | |
| P2 | 0 | 1 | | | | |
| P3 | 1 | 0 | 1 | | | |
| P4 | 1 | 0 | 1 | 1 | | |
| P5 | 0 | 1 | 0 | 0 | 1 | |
| P6 | 0 | 0 | 0 | 0 | 0 | 1 |

| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 1 | | | | | |
| P2 | 0 | 1 | | | | |
| P3 | 1 | 0 | 1 | | | |
| P4 | 1 | 0 | 1 | 1 | | |
| P5 | 0 | 1 | 0 | 0 | 1 | |
| P6 | 0 | 1 | 0 | 0 | 1 | 1 |

We can quantify how the 2 schemes differ using the **(adjusted) Rand index** (see also: Jaccard, and Gamma indices).

Let $\mathcal{A} = \{A_1, \ldots, A_k\}$ and $\mathcal{B} = \{B_1, \ldots, B_\ell\}$ be two clusterings of a dataset with $n$ instances; there are thus

$$\binom{n}{2} = \frac{n(n-1)}{2} = \mathrm{ss} + \mathrm{dd} + \mathrm{sd} + \mathrm{ds}$$

**pairs** of observations in the data, where we denote the number of pairs of observations that are in:

- the **same** cluster in $\mathcal{A}$ **and** the **same** cluster in $\mathcal{B}$ by $\mathrm{ss}$,
- **different** clusters in $\mathcal{A}$ **and different** clusters in $\mathcal{B}$ by $\mathrm{dd}$;
- the **same** cluster in $\mathcal{A}$ **but different** clusters in $\mathcal{B}$ by $\mathrm{sd}$, and
- **different** clusters in $\mathcal{A}$ **but** the **same** cluster in $\mathcal{B}$ by $\mathrm{ds}$.

The **Rand index** of $\mathcal{A}$ and $\mathcal{B}$ as

$$\mathrm{RI}(\mathcal{A},\mathcal{B}) = \frac{\mathrm{ss}+\mathrm{dd}}{\mathrm{ss}+\mathrm{dd}+\mathrm{sd}+\mathrm{ds}}.$$

**Large** $\mathrm{RI} \implies$ **similarity** of clustering outcomes (⚠: $\mathrm{RI} \not\equiv$ accuracy)

The **adjusted Rand index** relies on the **contingency table** of $\mathcal{A}$ and $\mathcal{B}$:

| | $B_1$ | $\cdots$ | $B_\ell$ | **Total** |
|---|---|---|---|---|
| $A_1$ | $n_{1,1}$ | $\cdots$ | $n_{1,\ell}$ | $\mu_1$ |
| $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $A_k$ | $n_{k,1}$ | $\cdots$ | $n_{k,\ell}$ | $\mu_k$ |
| **Total** | $\nu_1$ | $\cdots$ | $\nu_\ell$ | $n$ |

$n_{i,j} = |A_i \cap B_j|$ is the # of instances found in **both** $A_i \in \mathcal{A}, B_j \in \mathcal{B}$.

The adjusted Rand index ($\in [-1, 1]$) is given by

$$\begin{aligned}\mathrm{ARI}(\mathcal{A}, \mathcal{B}) &= \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{\mu_i}{2} \sum_j \binom{\nu_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{\mu_i}{2} + \sum_j \binom{\nu_j}{2}\right] - \left[\sum_i \binom{\mu_i}{2} \sum_j \binom{\nu_j}{2}\right] \Big/ \binom{n}{2}} \\ &= \frac{2(\mathrm{ss} \cdot \mathrm{dd} - \mathrm{sd} \cdot \mathrm{ds})}{(\mathrm{ss} + \mathrm{sd})(\mathrm{ds} + \mathrm{dd}) + (\mathrm{ss} + \mathrm{ds})(\mathrm{sd} + \mathrm{dd})}.\end{aligned}$$

In the example, there are $n = 6$ observations and $6 \cdot 5/2 = 15$ pairs of observations in the data, and we have:

- $\mathrm{ss} = 4$ ($x_1, x_3$; $x_1, x_4$; $x_3, x_4$; $x_2, x_5$)
- $\mathrm{ds} = 2$ ($x_2, x_6$; $x_5, x_6$)
- $\mathrm{sd} = 0$ (**none**)
- $\mathrm{dd} = 9$ (**all remaining pairs**)

$\mathrm{RI}(\mathcal{A}, \mathcal{B}) = \frac{4+9}{4+9+0+2} = \frac{13}{15} = 0.87$

$\mathrm{ARI}(\mathcal{A}, \mathcal{B}) = \frac{2(4 \cdot 9 - 0 \cdot 2)}{(4+0)(2+9)+(4+2)(0+9)} = 0.73$

$\Longrightarrow$ **high-ish** similarity between $\mathcal{A}$ and $\mathcal{B}$

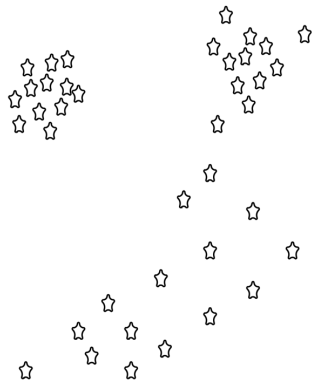
Cluster **stability** can also be used to "**validate**" the choice of **algorithm**. Assume that a clustering **algorithm** $\mathcal{G}$ has been applied to data, yielding a clustering scheme $\mathcal{C} = \{C_1, \ldots, C_K\}$.

We can view a dataset $\mathbf{X}$ as a **realization** of a (potentially unknown) underlying **data generating mechanism**; a **different realization** $\mathbf{X}'$ (collection of **new** data?) may yield a **distinct clustering scheme** $\mathcal{C}'$.

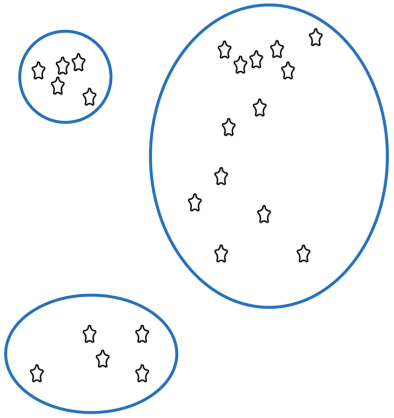
How **stable** is the clustering **outcome** of $\mathcal{G}$, relative to the **realization** $\mathbf{X}$?

We can get a sense for the underlying stability by sampling **multiple row subsets** from $\mathbf{X}$ (or **columns**, or sampling **columns** from **sampled rows**).

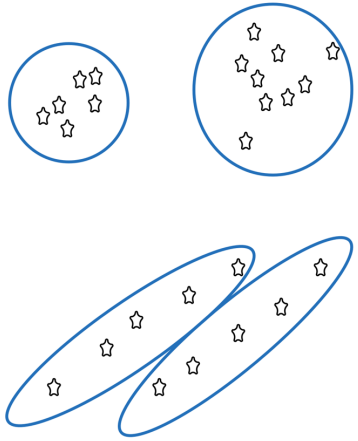
Assume then that we have $\ell$ subsets $\mathbf{X}_1, \ldots, \mathbf{X}_\ell \subseteq \mathbf{X}$, on which we apply the **algorithm** $\mathcal{G}$, with **parameters** $\mathcal{P}$, yielding the corresponding clustering schemes $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$.



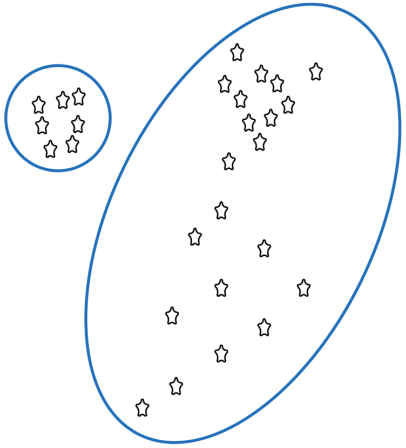
Full dataset



Sample 1 clustering



Sample 2 clustering



Sample 3 clustering

Let $\mathcal{W}$ be the **similarity matrix** for the various clustering schemes:

$$\mathcal{W} = \begin{pmatrix} w(\mathcal{C}_1, \mathcal{C}_1) & \cdots & (\mathcal{C}_1, \mathcal{C}_\ell) \\ \vdots & \ddots & \vdots \\ w(\mathcal{C}_\ell, \mathcal{C}_1) & \cdots & (\mathcal{C}_\ell, \mathcal{C}_\ell) \end{pmatrix};$$

this $\mathcal{W}$ my form the basis of a **meta-clustering scheme** in which $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ play the role of instances, using a **graph-based meta-clustering method** such as **DBSCAN**.

If the clustering results obtained from $\mathbf{X}$ by applying $\mathcal{G}$ are **stable**, we would expect the **meta-clustering results** to yield a **single meta-cluster**.

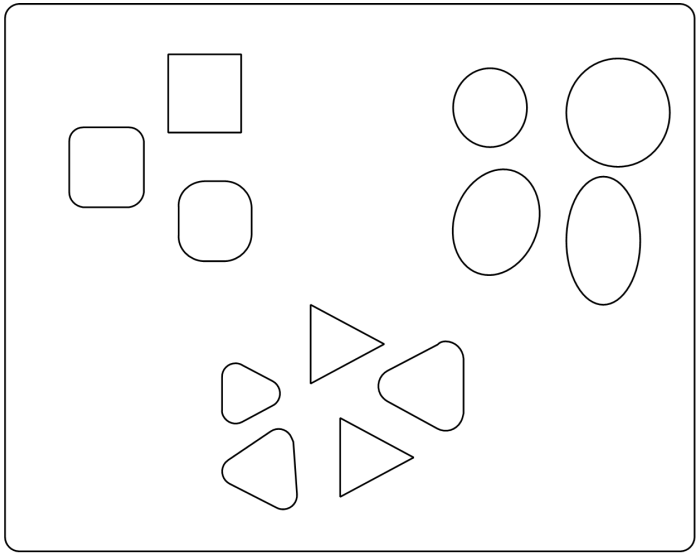
If the opposite holds, this supports the position that $\mathcal{G}$ does not produce **stable** clusters in $\mathbf{X}$, although it does not necessarily implying **instability** (could be an artefact related to the **meta-clustering method choices**).

# 4.5.5 – External Validation

In everyday applications, we gauge clustering results against some (usually implicit) **external expectation**: we cannot help but bring in **outside information** to evaluate the clusters (the **"correct" groups**).

In the example to the right, we might be interested in finding **natural groups** in a dataset of objects.

We might hold the **pre-conceived notion** that the **natural groups** are linked to the underlying **shape** (square, triangle, circle).

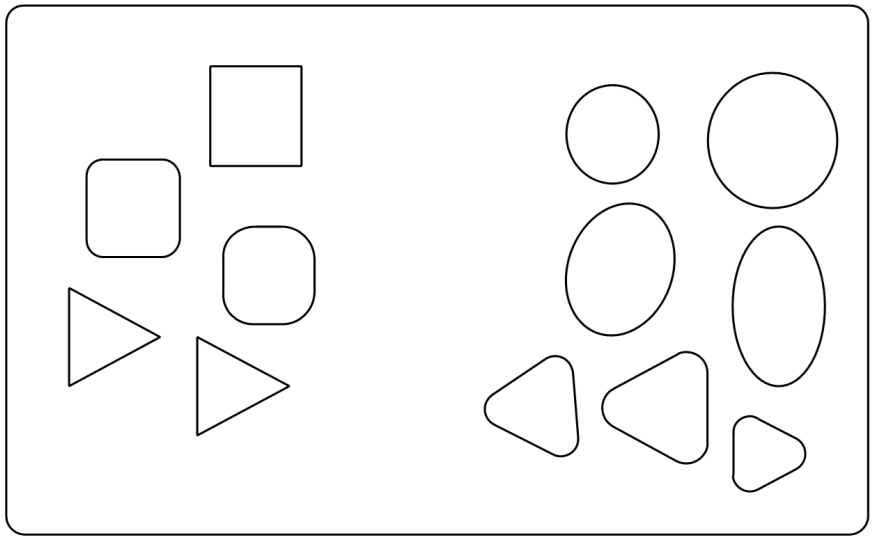


Natural Groupings

When the outcome agrees with the **(external) notion**, we usually feel that the clustering was **successful**.

If the outcome seems to pick up something about the **sharpness of the shapes' vertices**, say, we might conclude that the algorithm does not do a good job of capturing the **essential nature of the objects**.

Alternatively, such a situation might make us revisit our **pre-conceived notions** about the dataset and its **natural groups**.



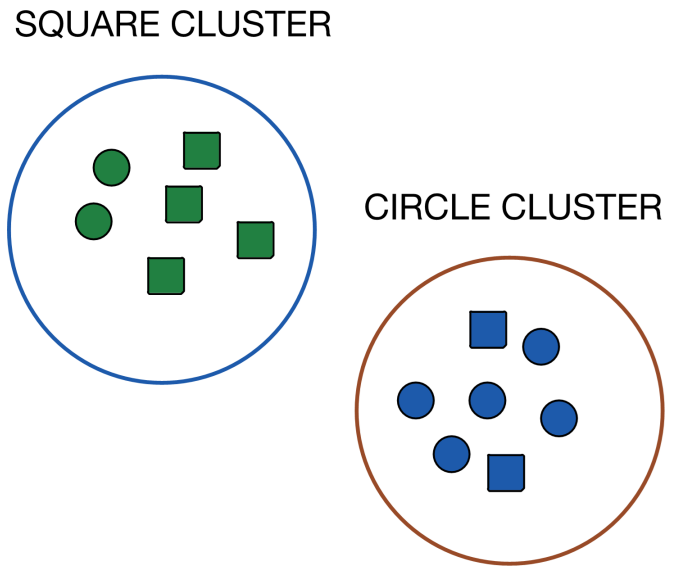
Clustering Results

This strategy is often used to **build confidence** in the clustering approach, based on **preliminary** or **sample results** ... but it rests on a **huge assumption** (which often conflicts with the UL framework): that the **natural groups** in the data are **identifiable**, explicitly or implicitly.

Due to the conceptual similarity to **classification** ( ⚠ : not SL), **external validation** measures resemble **classification performance** measures.

If an **external classification** variable exists, the **purity metric** assign a **label** to a cluster $C$ according to the **most frequent classes** of the instances in $C$, and

$$\text{purity}(C) = \frac{\text{\# correctly assigned pts in } C}{|C|}$$



The **clustering purity scores** for $\mathcal{C} = \{C_1, \ldots, C_K\}$ are:

$$\text{average purity}(\mathcal{C}) = \frac{1}{K} \sum_{\ell=1}^{K} \text{purity}(C_\ell)$$

$$\text{weighted purity}(\mathcal{C}) = \frac{1}{n} \sum_{\ell=1}^{K} |C_\ell| \cdot \text{purity}(C_\ell).$$

In the example, the **purity** scores at the **cluster** and **clustering** levels are:

$$\text{purity}(C_\square) = \frac{2}{3}, \quad \text{purity}(C_\bigcirc) = \frac{5}{7};$$

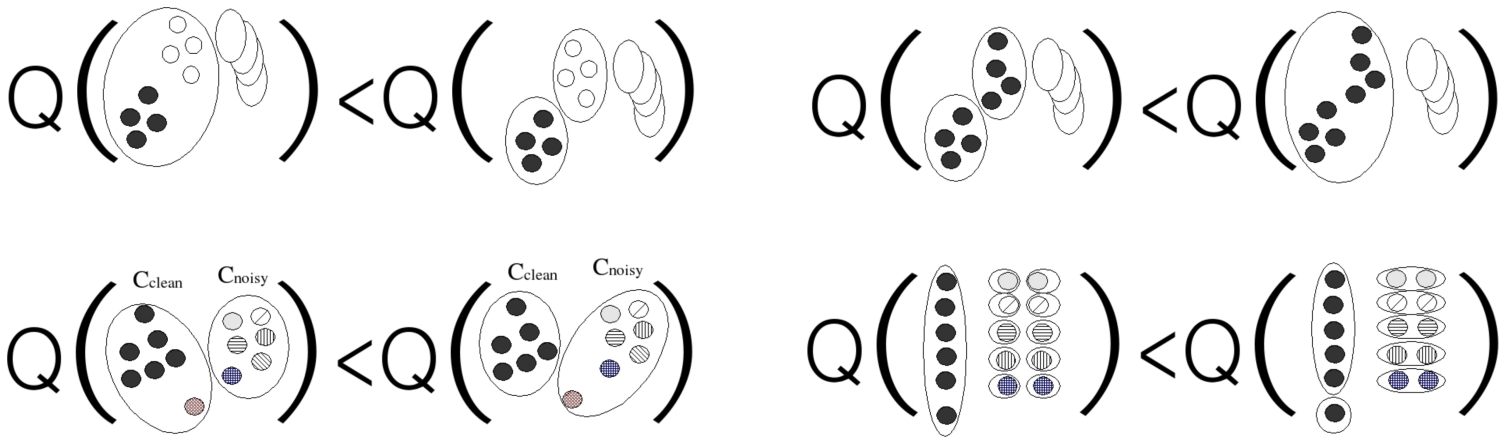
$$\text{average purity}(\mathcal{C}) = \frac{1}{2}\left(\frac{2}{3} + \frac{5}{7}\right) = 69.0\%$$

$$\text{weighted purity}(\mathcal{C}) = \frac{1}{6+7}\left(6 \cdot \frac{2}{3} + 7 \cdot \frac{5}{7}\right) = 69.2\%.$$

A **useful** external quality metric $Q$ should take advantage of the **natural classes** (if aligned with the **clustering results**), and take into account:

- cluster **homogeneity** (top left),
- **completeness**, (top right),
- **noisy and outlying data** (bottom left), and
- **size vs. quantity** considerations (bottom right)

The preferred behaviour is shown on the **right**.



# 4.5.6 – Model Selection

**Question:** how do we pick the **# of clusters** (and other **parameters**, including the choice of **algorithm**) for "**optimal**" results?

**Common approach:** consider **all outcomes** of various **parameter runs** and **replicates** for a given algorithm, and select the **parameter values** that optimize **a set of QCMs**.

**Optimizers** depends on each QCM's properties (**max**, **min**, **"knees"**, etc.).

But the parameter values that **optimize** a QCM may not work for others;

- **when they do** $\Longrightarrow$ support the existence of **natural groups**
- **when they don't** $\Longrightarrow$ **smaller collection of models** to pick from
- **either way** $\Longrightarrow$ some of the **arbitrariness** of clustering is removed

This approach can also be used to compare **different algorithms** – we search for the model that best describes the **natural groups** in the data among all results, according to **some metric(s)** (see previous).

**Other approaches:** **quadratic cost**, **stability assessment**, etc. Model assessment and selection is a popular **research topic**.

The various types of validation methods do not always **agree** – this is par for the course for UL, if **demoralizing at times** – this can be leveraged to extract information about the data's **underlying structure** and its **aspects**.

**Tip:** avoid using a **single** assessment method; it is preferable to seek "signals of agreement" across a **variety of strategies**.

**Final note:** clustering results may **just** be 'ok' . . . but that is ok too! We can study the situation and decide what is **important** and what can safely **be ignored** – as always, **a lot depends on the context**.

# 4.6 – Spectral Clustering

A large proportion of clustering algorithms are either **compactness**-based or **connectivity**-based.

**Compactness methods:** ($k-$means, etc.)

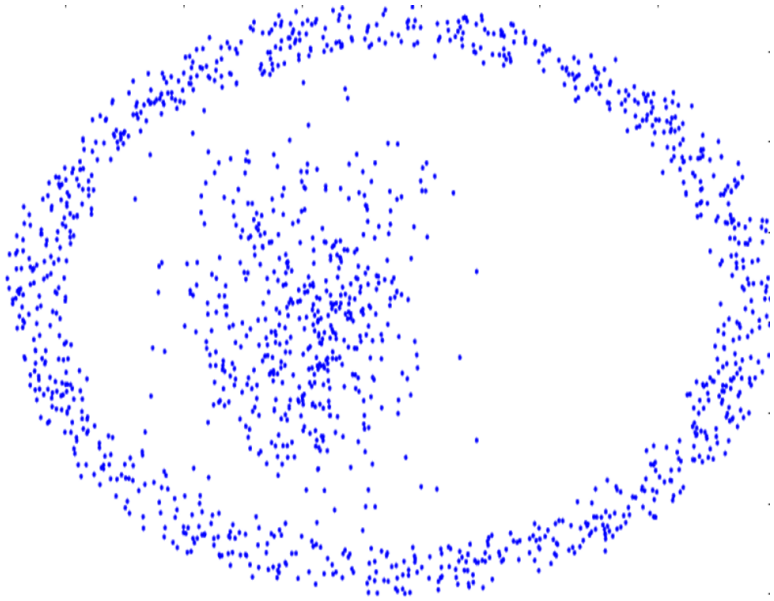
- usually variants of $k$NN methods;
- most effective when there are **distinct "clumps"** in the data;
- we can make assumptions about the distribution of the clusters **ahead of time**;
- struggle to achieve **meaningful results** when groups are not **linearly separable**.

**Little** knowledge of the dataset + **faulty** assumptions about the distributions of clusters $\implies$ **invalid** clustering schemes

**Connectivity methods:** (DBSCAN (?), spectral clustering, etc.)

- focus on dividing observations into groups based on their **similarity graphs**;
- observations quite different in their **features** (which would then be **differentiated** by compactness methods) may still end up in the **same cluster** if a **chain** of **sufficiently similar** observations links them.

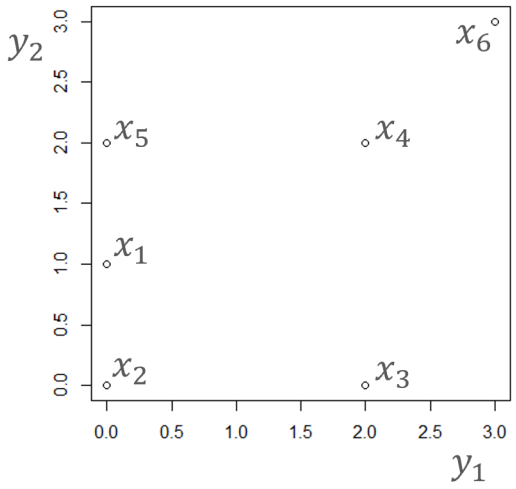
They require **few** initial assumptions, but may end up being hard to justify **mathematically**; validity may only be determined ***post hoc***.



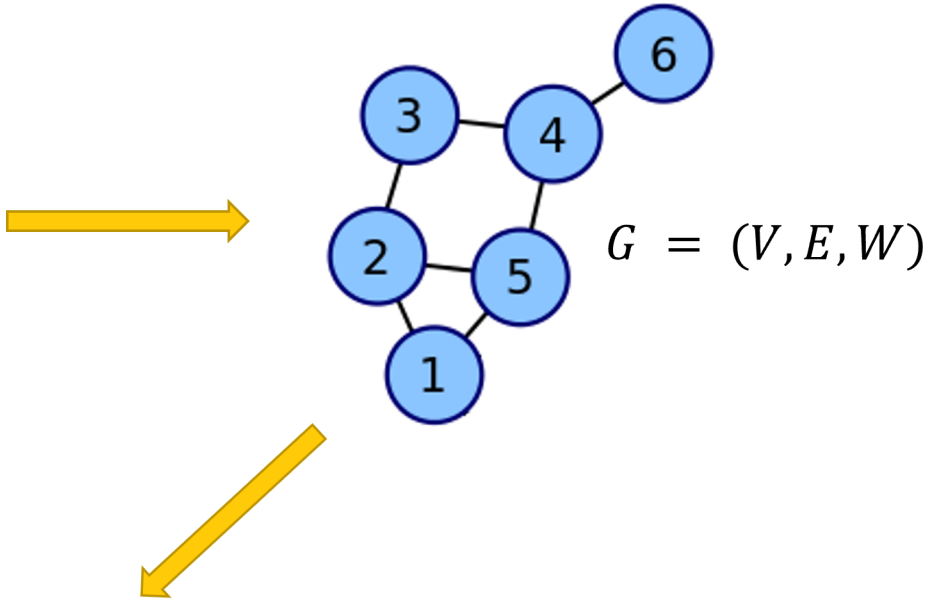
**Spectral** clustering is a **connectivity** method which transforms the dataset into its **similarity graph**, and converts that into an **eigenvalue problem**.

**Dataset**

| $i$ | $y_1$ | $y_2$ |
|---|---|---|
| $x_1$ | 0 | 1 |
| $x_2$ | 0 | 0 |
| $x_3$ | 2 | 0 |
| $x_4$ | 2 | 2 |
| $x_5$ | 0 | 2 |
| $x_6$ | 3 | 3 |




**Similarity Graph**



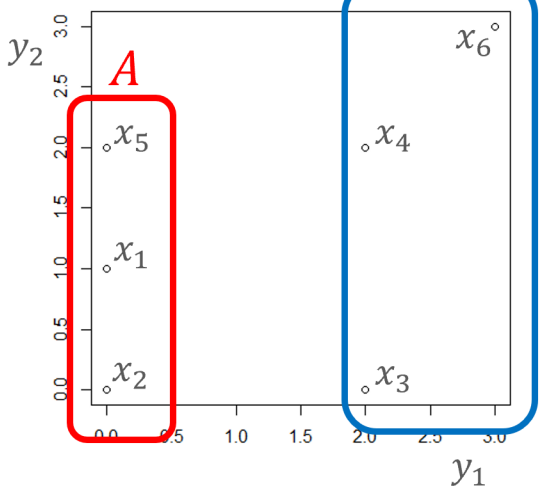

**Eigenvalue Problem**

$$L\vec{u} = \lambda\vec{u}$$

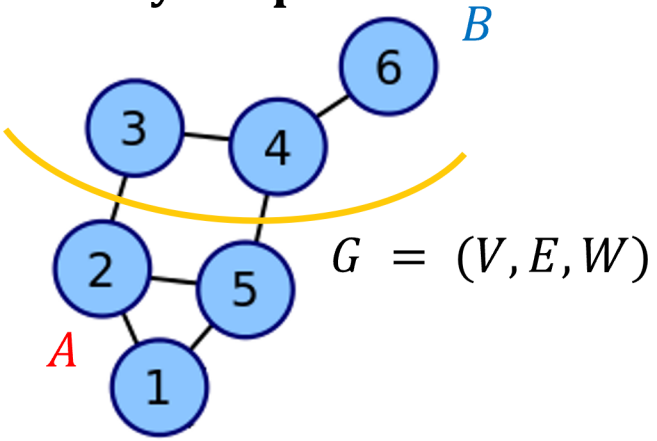
It then solve the **eigenvalue problem**, converts the solution into a **graph cut**, and translates the cut into dataset **clusters**.

**Clustered Dataset**

| $i$ | $y_1$ | $y_2$ |
|---|---|---|
| $x_1$ | 0 | 1 |
| $x_2$ | 0 | 0 |
| $x_3$ | 2 | 0 |
| $x_4$ | 2 | 2 |
| $x_5$ | 0 | 2 |
| $x_6$ | 3 | 3 |




**Similarity Graph with Cuts**




**Solved Eigenvalue Problem**

$$L\vec{u} = \lambda\vec{u}$$

# 4.6.1 – Graphs and Cuts

A **graph** is an object which connects **nodes** (**vertices**) together with **edges**.

The **edges** have **weights** and can also be **directed**.

Sometimes, the **weights** are identical and **bidirectional** $\Longrightarrow$ in that case, **edges** just represent that a **relationship** exists.

**Example:** airports (**vertices**) and flight paths (**edges**) form a graph in transportation networks; the **edges** can be **weighted** according to flight frequency and/or **directed** according to their origin and destination.

**Example:** people (**vertices**) and relationships (**edges**) form a graph in social networks; the **edges** can be **weighted** according to frequency of communication and/or **directed** according to who follows who on an app.

What is the link with clustering? Once a **similarity** measure $w$ is selected, a dataset can be represented by a **similarity graph** $G = (V, E, W)$:

1. observations $\mathbf{x}$ correspond to **vertices** $v \in V$;

2. if $i \neq j$, vertices $v_i, v_j \in V$ are connected by an **edge** $e_{i,j} = 1$ if the **similarity weight** $w_{i,j} = w(\mathbf{x}_i, \mathbf{x}_j) > \tau$ for a predetermined **threshold** $\tau \in [0, 1)$, and by **no** edge $(e_{i,j} = 0)$ otherwise;

3. the edges $(e_{i,j})$ form the **adjacency matrix** $E$;

4. the weights $(w_{i,j})$ form the **similarity matrix** $W$ (but $w_{i,i} = 0$, $\forall i$);

5. the **(diagonal) degree matrix** $D$ provides information about the number of edges attached to a vertex: $d_{i,i} = \sum_{j=1}^{n} e_{i,j}$.

**Example:** (data on p.125) we use the **Gower similarity measure**

$$w(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{1}{p}\sum_{k=1}^{p} \frac{|x_{i,k} - x_{j,k}|}{\text{range of } k\text{th feature in } \mathbf{X}} \implies p = 2,\ r_1 = r_2 = 3;$$

$$w_{3,4} = w_{4,3} = w(\mathbf{x}_3, \mathbf{x}_4) = 1 - \frac{1}{2}\left(\frac{|x_{3,1} - x_{4,1}|}{r_1} + \frac{|x_{3,2} - x_{4,2}|}{r_2}\right)$$

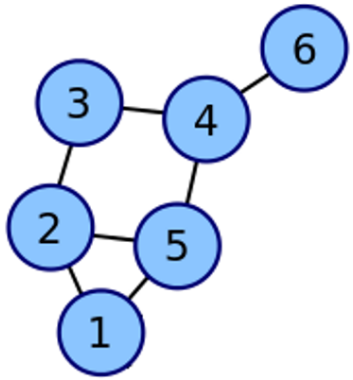
$$= 1 - \frac{1}{2}\left(\frac{|2-2|}{3} + \frac{|0-2|}{3}\right) = 1 - \frac{1}{2}\cdot\frac{2}{3} = \frac{2}{3}, \quad \text{for instance.}$$

With $\tau = 0.6$, the **similarity matrix** $W$ and the **adjacency matrix** $E$ are:

$$W = \begin{pmatrix} 0 & 5/6 & 1/2 & 1/2 & 5/6 & 1/6 \\ 5/6 & 0 & 2/3 & 1/3 & 2/3 & 0 \\ 1/2 & 2/3 & 0 & 2/3 & 1/3 & 1/3 \\ 1/2 & 1/3 & 2/3 & 0 & 2/3 & 2/3 \\ 5/6 & 2/3 & 1/3 & 2/3 & 0 & 1/3 \\ 1/6 & 0 & 1/3 & 2/3 & 1/3 & 0 \end{pmatrix} \quad E = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

The **degree matrix** $D$ are and the **similarity graph** $G$ can be read directly from $E$:

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$




The **degree matrix** $D$ can also be read **directly** from the **similarity graph** (depends on $\tau$), by counting the number of **edges** at each **node**.

A **graph cut** is the process by which we remove edges from the graph and separate the vertices into into groups (or **sub-graphs**).

**Clustering task:** split the **nodes** into distinct groups by **minimizing** the total **weight** of the **edges** broken in the process $\implies$ groups are as **dissimilar** as possible.

This is known as the **minimum cut problem** (**MinCut**). There are variants (**Normalized Cut**, **Ratio Cut**, **Min-Max Cut**, etc.).

The **MinCut** problem is **NP-hard** $\implies \nexists$ **theoretically guaranteed** efficient way to solve it.

With $n$ observations, the number of possible **cuts** is **bounded below** by $2^n$. For any reasonable dataset size $n$, the **brute force approach** is an exercise in **futility**.

The **clustering** approach generalizes the **MinCut** problem by imposing properties on the **similarity graph** in order to **approximate** the true **MinCut** solution in a **computationally efficient** manner.

**Important:** the approximate solution has to be near the right one.

Formally, **MinCut** involves finding a **partition** $\{A_1, ..., A_k\}$ of $G$ which **minimizes** the objective function

$$\text{Cut}(A_1, ..., A_k) = \frac{1}{2}\sum_{i=1}^{k} \mathcal{W}(A_i, \overline{A_i}), \quad \text{where} \quad \mathcal{W}(A, B) = \sum_{i \in A, j \in B} w_{i,j}$$

and $\overline{A}$ is the **(set-theoretic) complement** of $A$.

The factor $\frac{1}{2}$ is there to remove **double-counted** edges.

The **spectral** clustering approach instead solves the **Normalized Cut** (**NCut**) problem, where we are minimizing the weight of edges **escaping** a cluster relative to the **total weights** in the cluster: the objective function is

$$J_{\text{NCut}}(A, B) = \text{Cut}(A, B)\left(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)}\right), \text{ where } \text{Vol}(C) = \sum_{i \in C} w_{i,*}.$$

We **minimize** $J_{\text{NCut}}$ against the set of **all possible partitions** $(A, B)$ of $G$ and proceed **recursively** on the cluster **sub-graphs**, as often as necessary.

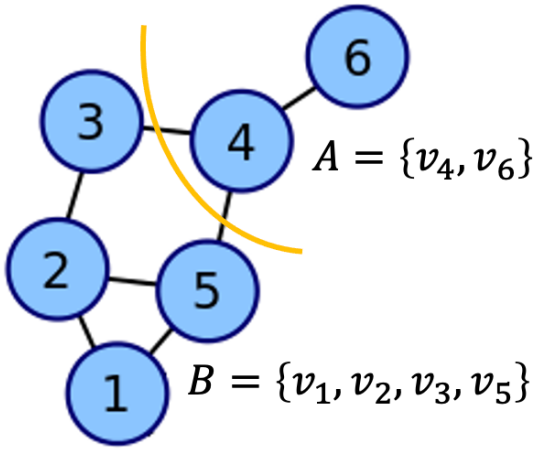
Intuitively, $J_{\text{NCut}}$ is **small** when the observations **within** each sub-graph are **similar** to one another ($\text{Vol}(A), \text{Vol}(B)$ are **large**) and the observations **across** are dissimilar to one another ($\text{Cut}(A, B)$ is **small**).

**Advantages:** considers **size of clusters** and **intra-group variance**; avoids **isolating vertices**; can be "solved" using **algebraic means**.

Well... that's not quite true. Solving the **NCut** problem is also **NP-hard**.

There is a simpler problem that **can** be solved with **purely algebraic means** (an **eigenvalue problem**) which yields solutions that are usually very similar to the **NCut** solutions.

THAT's **spectral clustering**.




$$\text{Cut}(A,B) = \sum_{i\in A, j\in B} w_{ij} = 3$$
$$\text{Vol}(A) = \sum_{i\in A, j\in V} w_{ij} = 13/3$$
$$\text{Vol}(B) = \sum_{i\in V, j\in B} w_{ij} = 32/3$$

$$J_{\text{NCut}}(A,B) = 0.97$$

# 4.6.2 – Similarity, Degree, and Laplacian Matrices

For a distance $d$, there are different ways to construct a **similarity graph**:

- **fully connected** graphs (all pairs $(v_i, v_j)$ with $w_{i,j} \neq 0$ have $e_{i,j} = 1$);
- $r-$**neighbourhood** graphs, where $e_{i,j} = 1$ iff $v_j \in B_d(v_i, r)$; $r$ is tuned to capture the **local structure** of data;
- $k$**NN** graphs (and variants), where each vertex is connected to its $k$ nearest neighbours (according to $d$), with $k$ **pre-selected**, and
- **mixtures** of the last two, to better capture **sparsity** in the data, etc.

The **similarity measure** $w$ is usually picked from a list that includes: **Gaussian** (most common), **cosine**, **fuzzy**, **Euclidean**, **Gower**, etc.

The **similarity matrix** $W$ is symmetric and has **zeros** along the **diagonal**; the **non-diagonal entries** represent the **similarity strength** between the corresponding graph **vertices**.

We build the **adjacency matrix** $E$ from $W$ and a **threshold** $\tau \in [0, 1)$.

The only component **not directly** captured by $W$ is the **degree matrix** $D$ (the number of edges at each vertex, assuming an **undirected** graph).

The **diagonal** of $D$ holds the degrees for all vertices. We can combine $W$ or $E$ with $D$ to create the **Laplacian** $L$, which has properties linked to the **topology** of the **similarity graph** $G$.

The **Laplacian** of a graph is defined by

$$L_0 = D - \Theta, \quad \Theta \in \{E, W\};$$

the **symmetric Laplacian** by

$$L_S = D^{-1/2} L D^{-1/2} = \mathbf{I}_n - D^{-1/2} \Theta D^{-1/2},$$

and the **asymmetric Laplacian** by

$$L_A = D^{-1} L = \mathbf{I}_n - D^{-1} \Theta.$$

In all cases, the **off-diagonal** entries are **non-positive**, and the **diagonal entries** contain the **degree** of the corresponding **node**.

The Laplacians have the following useful properties:

- $L_0$, $L_S$ are **symmetric**; $L_A$ not necessarily so;
- all their eigenvalues $\lambda$ are **real** and **non-negative**;
- every row and column adds up to $0 \implies \lambda_0 = 0$ is $\min\{\lambda\}$;
- the # of **connected components** in $G$ is $\dim$ of the nullspace associated to $\lambda_0 = 0 \implies \approx$ **# of clusters** in data (?), and
- the second smallest eigenvalue gives the graph's **sparsest cut** (**NCut**).

Let's see how this all comes together.

# 4.6.3 – SC Algorithm

In the case of **two** clusters, the objective function $J_{\text{NCut}}$ is **minimized** when finding the **eigenvector** $\mathbf{f}$ corresponding to the **smallest positive** eigenvalue of $L_S$, also known as the **spectral gap**.

The clustering in the **original data** is obtained by assigning $\mathbf{x}_i$ to $A$ when $f_i > 0$ and $\mathbf{x}_j$ to $B$ **otherwise**. This **deterministic** algorithm is a special case of the **spectral clustering algorithm**.

To split $\mathbf{X}$ into $k$ (**stochastic?**) clusters, we use the following steps:

1. form a **similarity matrix** $W$ and a **degree matrix** $D$, using a **threshold** $\tau \in [0, 1)$;

2. construct a **Laplacian** matrix $L_\xi$, $\xi \in \{0, S, A\}$, using $\Theta = W$;

3. compute the first $k$ eigenvectors $\{\mathbf{u}_1, ..., \mathbf{u}_k\}$ of $L_\xi$ corresponding to the $k$ **smallest positive** eigenvalues of $L_\xi$;

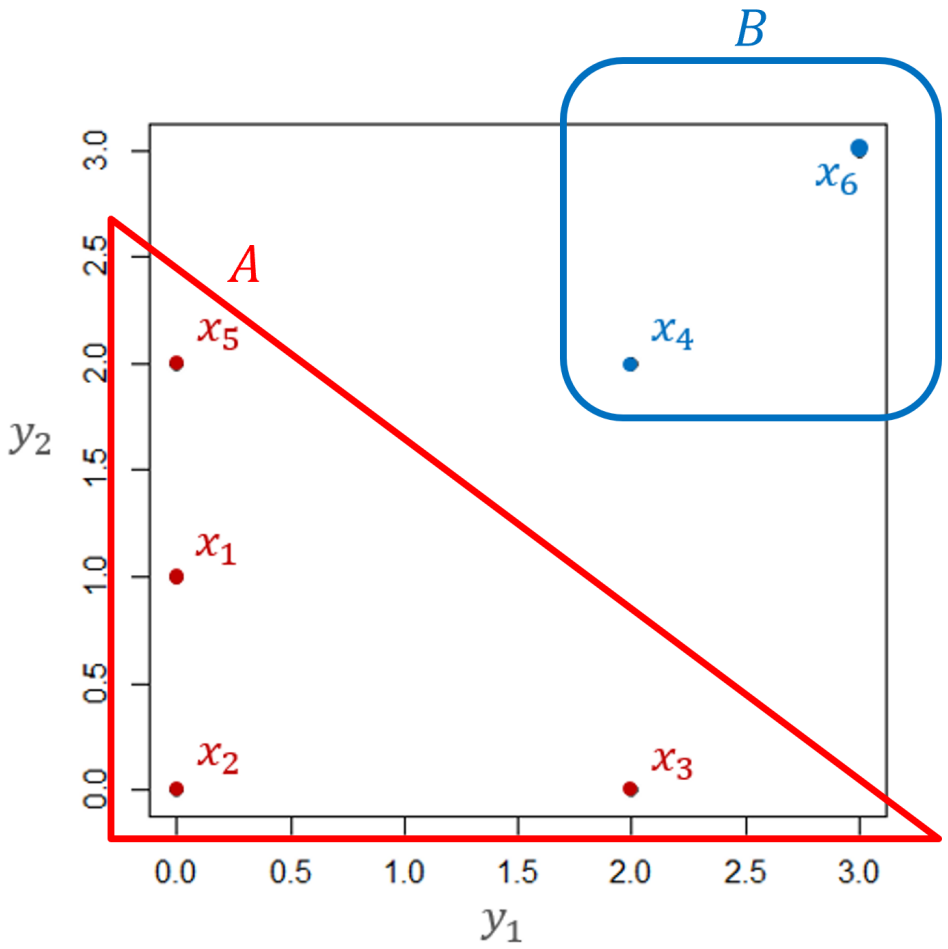
4. construct the $n \times k$ matrix $\mathbf{U}$ containing the vectors $\{\mathbf{u}_1, ..., \mathbf{u}_k\}$ as **columns**;

5. **normalize** the rows of $\mathbf{U}$ into a matrix $\mathbf{Y}$ with rows $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ having **unit length**;

6. **cluster the rows** of $\mathbf{Y}$ into $k$ clusters (**stochastic**?);

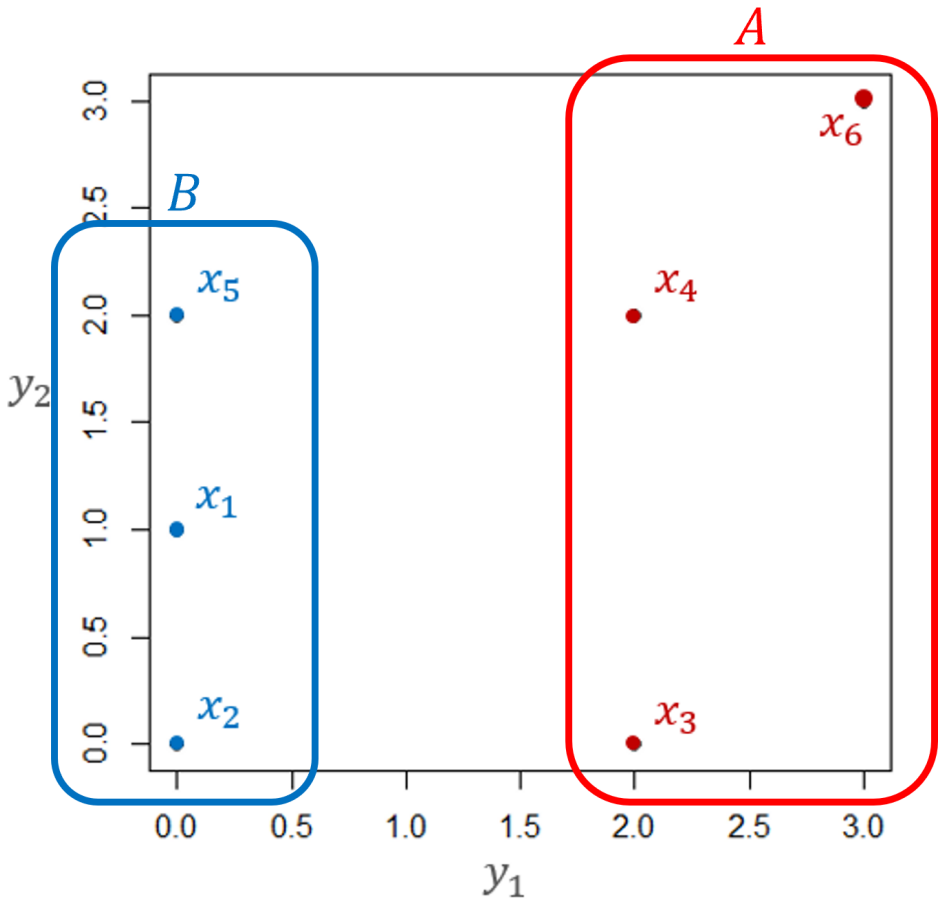
7. assign $\mathbf{x}_i$ to cluster $j$ of $\mathbf{X}$ if $\mathbf{y}_i$ was assigned to cluster $j$ step 6.

| $i$ | $f_i$ | $A/B$ |
|---|---|---|
| $x_1$ | -0.39 | $A$ |
| $x_2$ | -0.30 | $A$ |
| $x_3$ | -0.18 | $A$ |
| $x_4$ | 0.03 | $B$ |
| $x_5$ | -0.16 | $A$ |
| $x_6$ | 0.84 | $B$ |




Special spectral $2-$clustering case, with simple Laplacian $L_0$

| $i$ | $f_i$ | $A/B$ |
|---|---|---|
| $x_1$ | 0.36 | $B$ |
| $x_2$ | 0.54 | $B$ |
| $x_3$ | -0.03 | $A$ |
| $x_4$ | -0.43 | $A$ |
| $x_5$ | 0.14 | $B$ |
| $x_6$ | -0.61 | $A$ |




Special spectral $2-$clustering case, with symmetric Laplacian $L_S$

Spectral clustering is an attractive approach because it is **easy to implement** and reasonably **fast**, especially for **sparse** data: it is a **graph partitioning problem** with no assumption on the **shape** of the clusters.

Spectral clustering has **variants**, which depend on the **choices** that can be made at various points in the process:

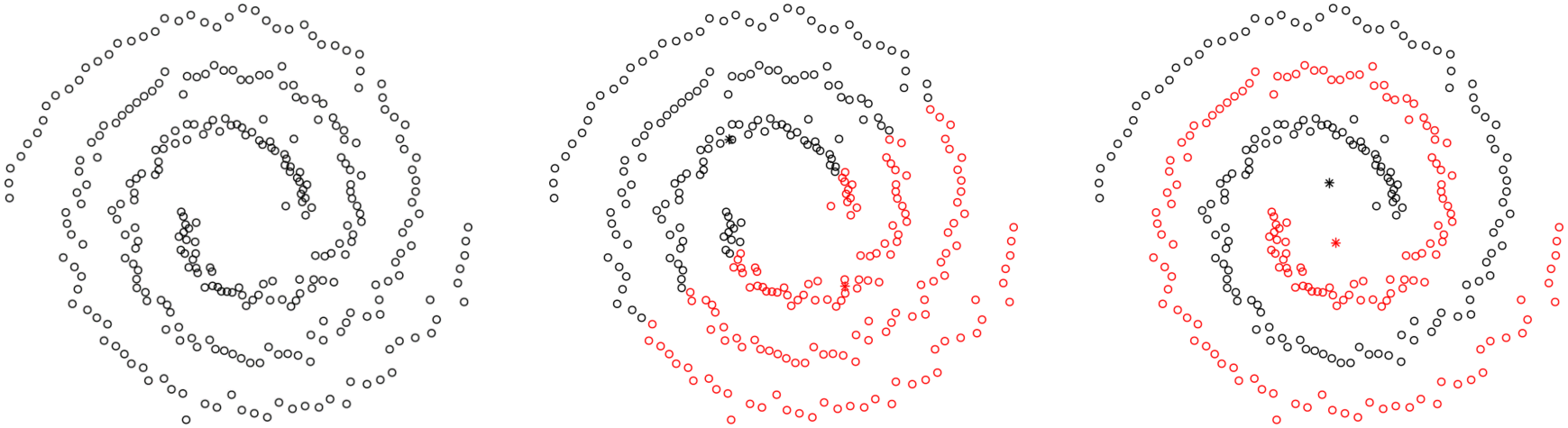
1. **pre-processing** (choices: **number of cluster** $k$, **similarity measure** $w$, **threshold** $\tau$);

2. **spectral representation** (choice of **Laplacian**);

3. **clustering algorithm** (choice of **compact**-based, potentially **non-deterministic** algorithm used on the **rows** of $\mathbf{Y}$).

The **NJW algorithm** uses $L_S$ for the spectral representation and **$k-$means** in the clustering step.

It can be interpreted as **kernalized $k-$means**: if we select a **kernel** (see Chapter 5) which transforms the points to their **mapped value** in the Laplacian of the similarity graph, we (almost directly) recover **NJW**.



Original data (left); $2-$means (middle); $2-$NJW (right)

# 4.6.4 – Practical Details and Limitations

In general, there is virtually **no theoretical justification** for determining what clustering approach to use; even after an approach has been selected, it can be quite difficult to pick appropriate **parameter values**.

SC suggest using **sparse similarity/adjacency matrix**: we must strike a balance between a Laplacian which is **too densely connected**, and one in which all the observations are **dissimilar** to one another.

Also: **computational** challenge of **finding the eigenvalues** of $L_\xi$. Fairly easy to do if the matrix is **sparse enough** $\Longrightarrow$ use a **relatively-high** threshold $\tau$.

There are ways to help SC **tune** for the best parameter values (including $\tau$) **automatically**, but they are **resource-intensive**.

SC methods are effective because they do not require assumptions about **distributions** and **centers**, are fairly **easy to implement**, and are **transparent** and **interpretable**.

However, they suffer from some of the same drawbacks as other methods:

- how to select the **initial parameter values**;
- run-times **do not scale** with larger datasets, and
- determining optimal ways to **visualize** the results.

As in all clustering scenarios, analysts are faced with decisions at various levels of the process; they must be prepared to run multiple algorithms, in multiple configurations, in order to get a sense for the data structure. Some strategies specific to spectral clustering are presented in.

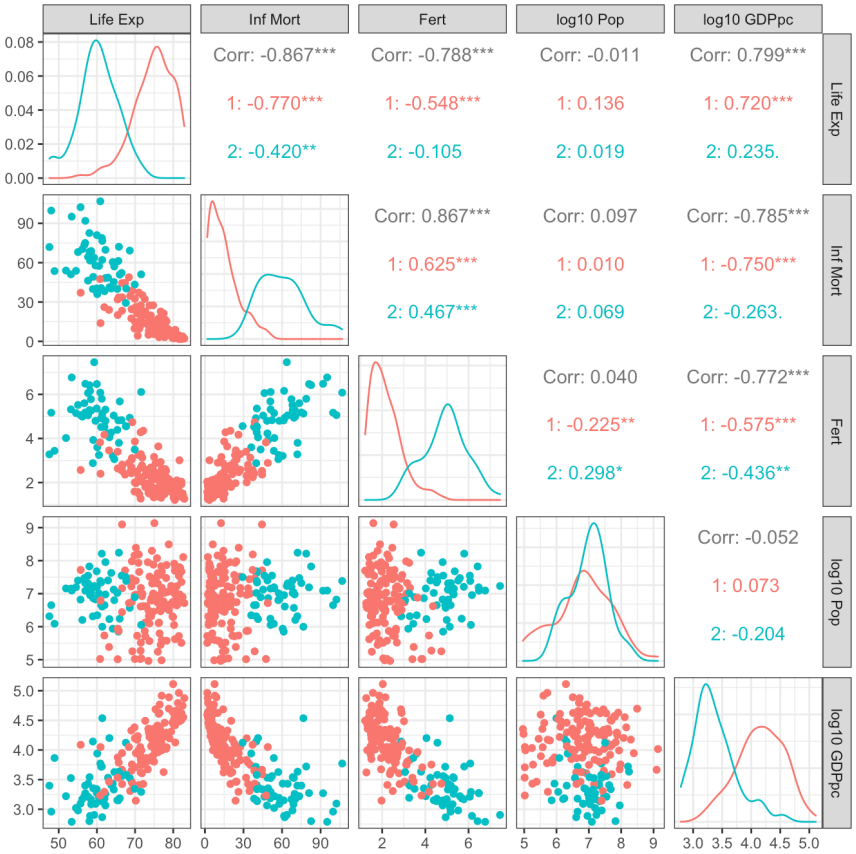
# 4.6.5 – Example: Gapminder Dataset

We re-visit the (scaled) 2011 Gapminder dataset using Euclidean dissimilarity. We use the `kernlab` implementation of the NJW algorithm found in `R`'s `specc()`, with the default settings.

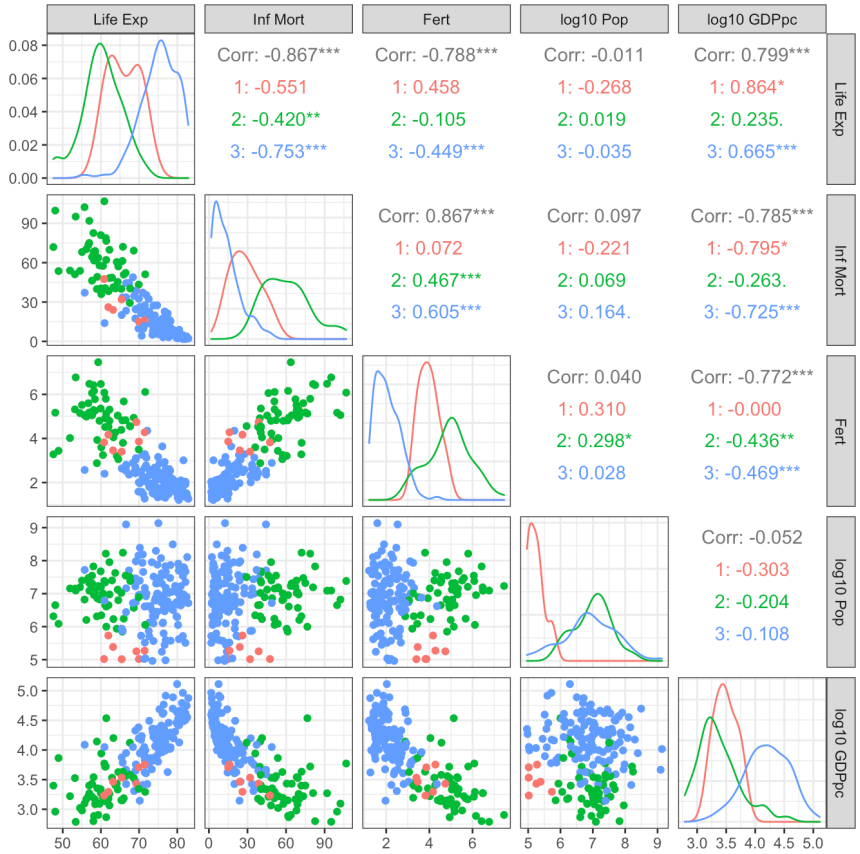
We run one instance of the algorithm for $k = 2$ to $k = 7$ clusters (see DUDADS, 22.4.2, *Spectral Clustering* for code).

None of our clustering attempts (not just SC) have found what one might call **natural groups** in the 2011 Gapminder data.

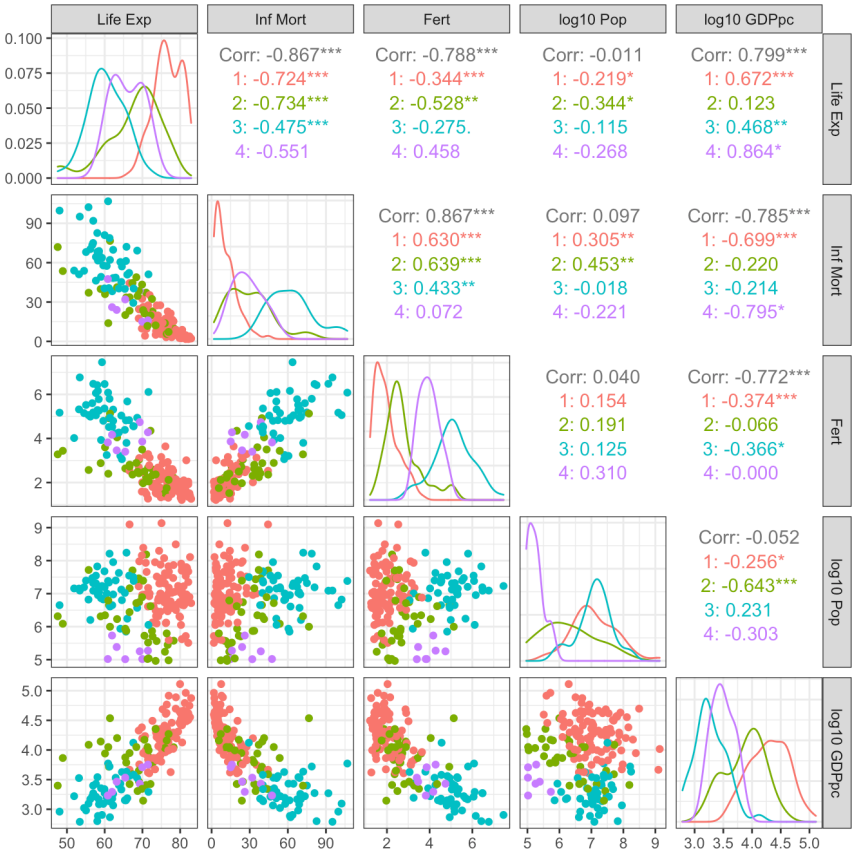
We might not have hit on the right method yet... but at what point do we decide that the task is futile and no such groups exist in the first place?



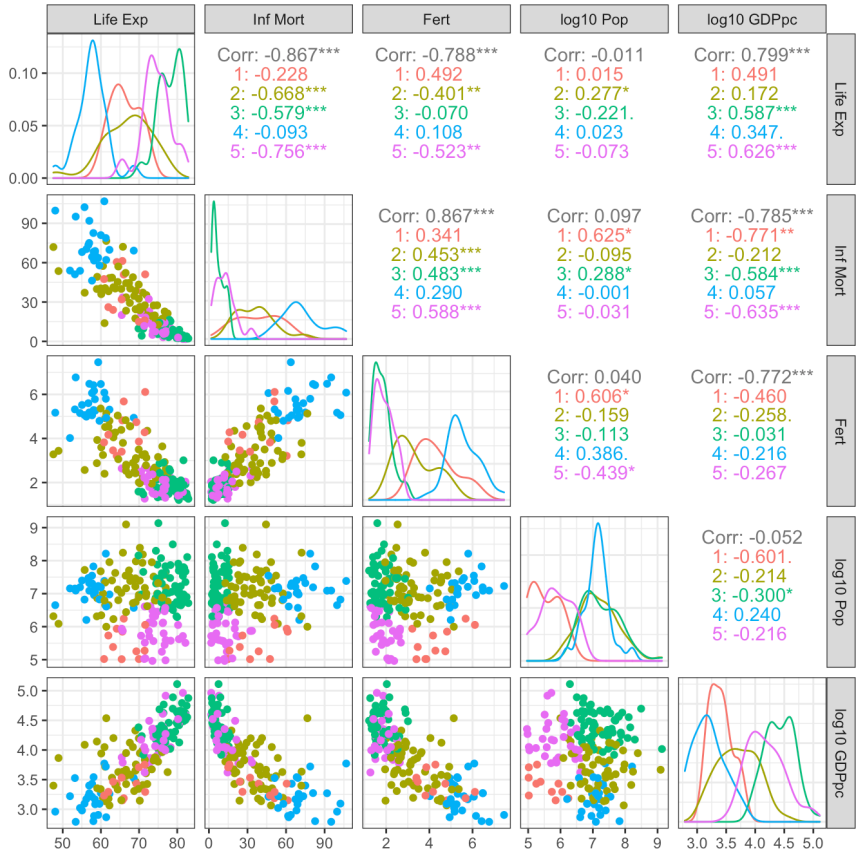

2−SC results



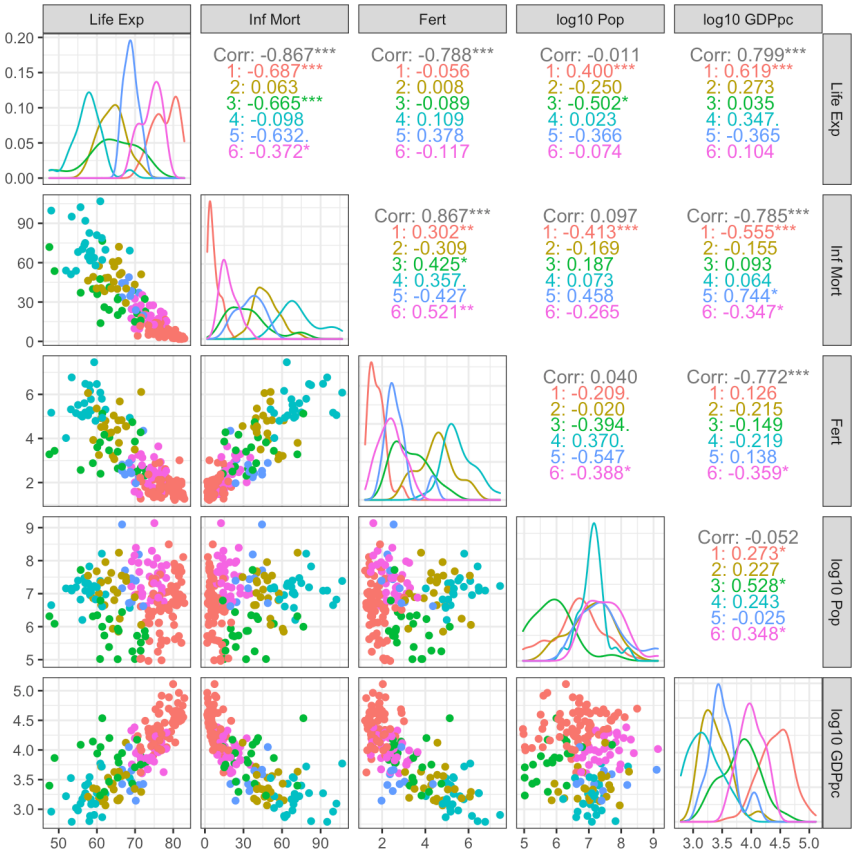

3−SC results



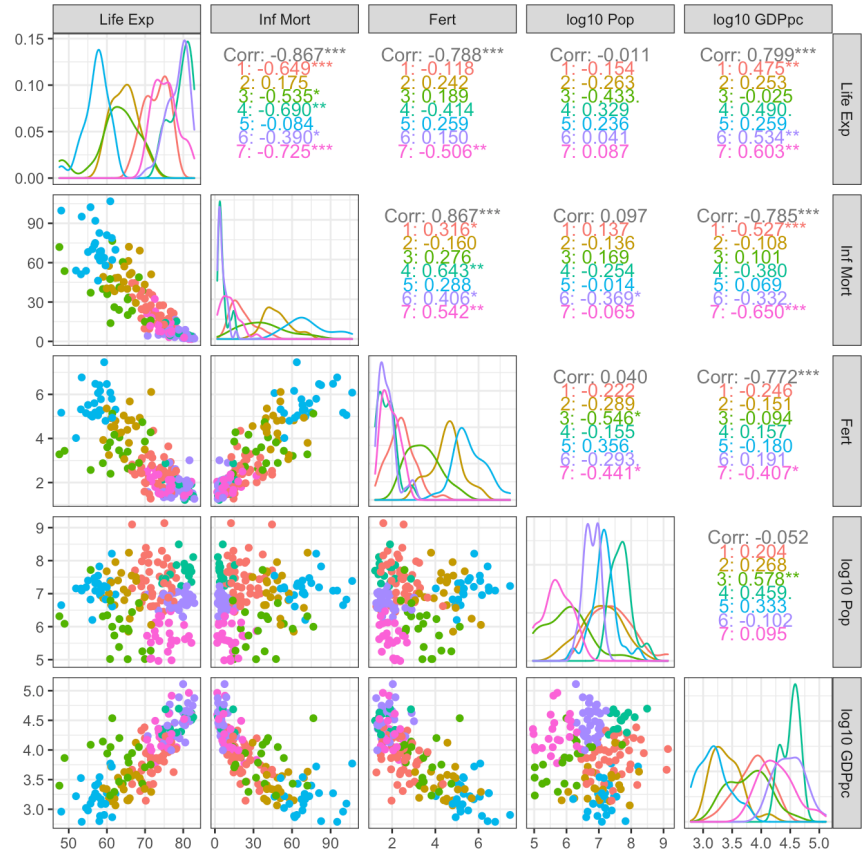

4−SC results




5−SC results




6−SC results




7−SC results

# 4.7 – Probability-Based Clustering

**Density-based** clustering and **spectral** clustering are **model-free**.

In contrast, **probabilistic-based clustering** optimizes the fit between the **observed data** and a **mathematical model** of clustering, assuming that the data is generated *via* a number of **underlying probability distributions**.

In practice, we assume that clusters are represented by **parametric probability distributions**, allowing analysts to use **probability theory** (joy!).

**Objective:** **learn the parameters** of each of these distributions.

# 4.7.1 – Mixture Models

**Main assumption**: in a **mixture model** (MM), each observation is **drawn** (or **generated**) from one of several **mechanisms** (or **components**).

In **model-based clustering**, we learn the parameters that provide the **optimal fit** to the data; i.e., we make predictions about which **component(s)** generated **each** of the observations.

Observations generated by a given component belong to the **same cluster**.

Formally, we let

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \in M_{n,p}(\mathbb{R}).$$

We assume that $k$ mechanisms generate $\mathbf{X}$, and that each of is determined by a **parameter vector** $\boldsymbol{\theta}_\ell$, $1 \leq \ell \leq k$.

For $1 \leq j \leq n$, let $P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell)$ be the probability that $\mathbf{x}_j$ is **generated by component** $\ell$, $1 \leq \ell \leq k$; the **cluster label** of $\mathbf{x}_j$ is $z_j \in \{1, \ldots, k\}$.

The **mixture vector** $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$ is such that $\pi_\ell \in [0, 1]$ for all $1 \leq \ell \leq k$ and $\pi_1 + \cdots + \pi_k = 1$.

If $P(z_j = \ell) = \pi_\ell$, for $1 \leq \ell \leq k$, and if $P(\mathbf{x}_j \mid z_j = \ell) = P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell) \quad \forall j, \ell$, then the probability of **observing** $\mathbf{x}_j$ is

$$P(\mathbf{x}_j) = \sum_{\ell=1}^{k} \pi_\ell P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell) = \sum_{\ell=1}^{k} P(z_j = \ell) P(\mathbf{x}_j \mid z_j = \ell),$$

according to the **Law of Total Probability**.

Alternatively, we could use

$$\mathbf{z}_j \in \{0,1\}^k, \quad \|\mathbf{z}_j\|_2 = 1$$

to denote the **cluster signature** of $\mathbf{x}_j$. The **norm condition** implies that exactly **one** of the components of $\mathbf{z}_j$ is $1$; all others are $0$.

For instance, if there are $k = 5$ mechanisms (clusters) in the data and $\mathbf{x}_j \in C_4$, then $\mathbf{z}_j = (0,0,0,1,0)$.

This can be generalized to **fuzzy clusters**: the cluster signature of $\mathbf{x}_j$ is

$$\mathbf{z}_j \in [0,1]^k, \quad \|\mathbf{z}_j\|_2 = 1.$$

If $\mathbf{z}_j = (0,0,\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0)$, say, then interpret $\mathbf{x}_j$ as belonging **equally** to clusters $C_3$ **and** $C_4$ or as having **probability** $1/2$ of belonging to **either.**

If we write

$$P(\mathbf{z}_j) = \pi_1^{z_{j,1}} \times \cdots \times \pi_k^{z_{j,k}} = \prod_{\ell=1}^{k} \pi_\ell^{z_{j,\ell}}$$

and

$$P(\mathbf{x}_j \mid \mathbf{z}_j) = P(\mathbf{x}_j \mid \boldsymbol{\theta}_1)^{z_{j,1}} \times \cdots \times P(\mathbf{x}_j \mid \boldsymbol{\theta}_k)^{z_{j,k}} = \prod_{\ell=1}^{k} P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell)^{\ z_{j,\ell}},$$

we recover the **mixture model**:

$$P(\mathbf{x}_j) = \sum_{\ell=1}^{k} \pi_\ell P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell) = \sum_{\ell=1}^{k} P(\mathbf{z}_j \in C_\ell) P(\mathbf{x}_j \mid \mathbf{z}_j \in C_\ell).$$

# 4.7.2 – Generative Process

We can imagine that $\mathbf{X}$ was **generated** as follows. For $1 \leq j \leq n$:

1. draw a **cluster signature** $\mathbf{z}_j \sim \mathcal{G}_k(\boldsymbol{\pi}) = \text{Mult}_k(\boldsymbol{\pi})$, and

2. draw an observation $\mathbf{x}_j$ from the **corresponding mechanism** according to $P(\mathbf{x}_j \mid \mathbf{z}_j)$.

We do not usually have access to this **generative process**; instead, we are **given** $\mathbf{X}$ and the clustering task is to **determine how likely it is** that component $C_\ell$, $1 \leq \ell \leq k$, is **responsible** for observation $\mathbf{x}_j$, $1 \leq j \leq n$.

To do so, we need to compute the probabilities

$$\gamma(z_{j,\ell}) = P(\mathbf{z}_j \in C_\ell \mid \mathbf{x}_j), \quad \forall j, \ell.$$

This is hard to do directly, so we use **Bayes' Theorem**:

$$\gamma(z_{j,\ell}) = P(\mathbf{z}_j \in C_\ell \mid \mathbf{x}_j) = \frac{P(\mathbf{z}_j \in C_\ell)P(\mathbf{x}_j \mid \mathbf{z}_j \in C_\ell)}{P(\mathbf{x}_j)}$$
$$= \frac{P(\mathbf{z}_j \in C_\ell)P(\mathbf{x}_j \mid \mathbf{z}_j \in C_\ell)}{\sum_{\nu=1}^{k} P(\mathbf{z}_j \in C_\nu)P(\mathbf{x}_j \mid \mathbf{z}_j \in C_\nu)} = \frac{\pi_\ell P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell)}{\sum_{\nu=1}^{k} \pi_\nu P(\mathbf{x}_j \mid \boldsymbol{\theta}_\nu)}.$$

The **clustering objective** is to infer $\boldsymbol{\Theta} = \{\pi_1, \ldots, \pi_k, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_\ell\}$ from $\mathbf{X}$ for a fixed $k$, so as to compute the desired probabilities $\gamma(z_{j,\ell})$.

If the observations $\mathbf{x}_j$ are **independently** drawn in the process, then:

$$P(\mathbf{X} \mid \boldsymbol{\Theta}) = \prod_{j=1}^{n} \sum_{\ell=1}^{k} \pi_k P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell).$$

This is often re-written as:

$$\mathsf{LL}(\boldsymbol{\Theta}) = \ln P(\mathbf{X} \mid \boldsymbol{\Theta}) = \sum_{j=1}^{n} \ln \left( \sum_{\ell=1}^{k} \pi_k P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell) \right).$$

The **maximum likelihood estimator** (MLE) of $\boldsymbol{\Theta}$ is

$$\boldsymbol{\Theta}_{\mathsf{MLE}} = \arg\max_{\boldsymbol{\Theta}} \big\{ \ln P(\mathbf{X} \mid \boldsymbol{\Theta}) \big\};$$

if we have information about the **prior** $P(\boldsymbol{\Theta})$, then we may use the **maximum a posteriori estimator** (MAP) instead:

$$\boldsymbol{\Theta}_{\mathsf{MAP}} = \arg\max_{\boldsymbol{\Theta}} \big\{ \ln P(\mathbf{X} \mid \boldsymbol{\Theta}) + \ln P(\boldsymbol{\Theta}) \big\}.$$

# 4.7.3 – Gaussian Mixture Models

**Standard assumption:** all clusters are generated by **Gaussian mechanisms** $\Longrightarrow$ $P(\mathbf{x}_j \mid \boldsymbol{\theta}_\ell)$ arises from a **multivariate Gaussian** distribution:

$$\mathcal{N}(\mathbf{x}_j \mid \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell) = \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_\ell|}} \exp\left(-\tfrac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_\ell)\right),$$

where $\boldsymbol{\mu}_\ell \in \mathbb{R}^p$ and $\boldsymbol{\Sigma}_\ell$ is a **covariance matrix**. With $k$ mixture components:

$$\mathsf{LL}(\boldsymbol{\Theta}) = \ln P(\mathbf{X} \mid \boldsymbol{\Theta}) = \sum_{j=1}^{n} \ln\left(\sum_{\ell=1}^{k} \pi_\ell \mathcal{N}(\mathbf{x}_j \mid \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)\right).$$

It is straightforward to show that

$$\nabla \text{LL}_{\boldsymbol{\mu}_\ell}(\boldsymbol{\Theta}) = \boldsymbol{\Sigma}_\ell^{-1} \sum_{j=1}^{n} \gamma(z_{j,\ell})(\mathbf{x}_j - \boldsymbol{\mu}_\ell),$$

so that the **MLE estimators** or the **mean vectors** are **weighted** averages:

$$\hat{\boldsymbol{\mu}}_\ell = \sum_{j=1}^{n} \gamma(z_{j,\ell})\ \mathbf{x}_j \Bigg/ \sum_{j=1}^{n} \gamma(z_{j,\ell}).$$

Simultaneously, we can show that

$$\nabla \text{LL}_{\boldsymbol{\Sigma}_\ell}(\boldsymbol{\Theta}) = \sum_{j=1}^{n} \frac{\pi_\ell}{P(\mathbf{x}_j \mid \boldsymbol{\Theta})} \cdot \frac{\partial \mathcal{N}(\mathbf{x}_j \mid \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)}{\partial \boldsymbol{\Sigma}_\ell}.$$

Slightly more complicated manipulations show that the **MLE estimators** of the **covariance matrices** are also **weighted averages**:

$$\hat{\boldsymbol{\Sigma}}_\ell = \sum_{j=1}^{n} \gamma(z_{j,\ell})(\mathbf{x}_j - \hat{\boldsymbol{\mu}}_\ell)(\mathbf{x}_j - \hat{\boldsymbol{\mu}}_\ell)^\top \Big/ \sum_{j=1}^{n} \gamma(z_{j,\ell}).$$

Finally, we obtain the **mixture probabilities** $\pi_\ell$ by maximizing $\mathrm{LL}(\boldsymbol{\Theta})$ with respect to $\boldsymbol{\pi}$, subject to $\pi_\ell \in [0,1]$ and $\pi_1 + \cdots + \pi_k = 1$. **Lagrange multipliers** $\Longrightarrow$ **MLE estimators** of **mixture probabilities** are **averages**:

$$\hat{\pi}_\ell = \frac{1}{n} \sum_{j=1}^{n} \gamma(z_{j,\ell}).$$

**Problem:** **clustering probabilities** $\gamma(z_{j,\ell})$ $\Longrightarrow$ **MLE estimators** $\hat{\boldsymbol{\Theta}}$ $\Longrightarrow$ **clustering probabilities** $\gamma(z_{j,\ell})$ $\Longrightarrow$ ... which came first?!?

# 4.7.4 – EM Algorithm

There is no **closed form** way to express $\mathbf{z}_j$ directly in terms of $\mathbf{X}$, but the **Expectation-Maximization** (EM) algorithm may converge to the solution.

**EM Input:** $\mathbf{X}$; **EM Output:** $\boldsymbol{\Theta}^*$ which maximizes $\text{LL}(\boldsymbol{\Theta})$

0. Initialize $\boldsymbol{\Theta}^{[0]} = \left\{\boldsymbol{\mu}_\ell^{[0]}, \boldsymbol{\Sigma}_\ell^{[0]}, \pi_\ell^{[0]}\right\}_{\ell=1}^{k}$ and set $\text{LL}^{[0]} = \textbf{LL}(\boldsymbol{\Theta}^{[0]})$, $i := 0$;

1. **E**(xpectation)-**step**: compute the **responsibilities**

$$\gamma(z_{j,\ell}^{[i]}) = \pi_\ell^{[i]} \mathcal{N}(\mathbf{x}_j \mid \boldsymbol{\mu}_\ell^{[i]}, \boldsymbol{\Sigma}_\ell^{[i]}) \Big/ \sum_{\nu=1}^{k} \pi_\nu^{[i]} \mathcal{N}(\mathbf{x}_j \mid \boldsymbol{\mu}_\nu^{[i]}, \boldsymbol{\Sigma}_\nu^{[i]}), \quad \forall j, \ell;$$

2. **M**(aximization)-**step**: update the **parameters**

$$\boldsymbol{\mu}_\ell^{[i+1]} = \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]})\ \mathbf{x}_j \Big/ \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]}), \quad \forall \ell;$$

$$\boldsymbol{\Sigma}_\ell^{[i+1]} = \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]})(\mathbf{x}_j - \boldsymbol{\mu}_\ell^{[i]})(\mathbf{x}_j - \boldsymbol{\mu}_\ell^{[i]})^\top \Big/ \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]}), \quad \forall \ell,$$

$$\pi_\ell^{[i+1]} = \frac{1}{n} \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]}), \quad \forall \ell;$$

3. Set $\text{LL}^{[i+1]} = \text{LL}(\boldsymbol{\Theta}^{[i]})$ and check for convergence: if satisfied, set $\boldsymbol{\Theta}^* = \boldsymbol{\Theta}^{[i+1]}$; otherwise, set $i := i + 1$ and repeat steps 1 to 3.

There are two main **limitations** to using EM for GMM:

- EM is **costlier** (longer run-time) than $k-$**means**; also depending on the **initialization**, the algorithm may converge to a **local critical point** which is not necessarily the **global** maximizer;
- as the algorithm iterates, **two** (or more) GMM clusters can **collapse** into a **single** GMM cluster.

The EM algorithm can be sped-up by **first running** $k-$**means** and using the **mean vector**, **covariance matrix**, and **prop of observations** in the cluster $C_\ell$ to initialize $\boldsymbol{\mu}_\ell^{[0]}$, $\boldsymbol{\Sigma}_\ell^{[0]}$, and $\pi_\ell$ for $1 \leq \ell \leq k$.

The **collapsing** of clusters can be mitigated by **monitoring** $\|\boldsymbol{\Sigma}_\ell^i\|_2$ and **randomly resetting** $\boldsymbol{\mu}_\ell^{[i]}$, $\boldsymbol{\Sigma}_\ell^{[i]}$ when some **threshold** is reached.

# Special Cases and Variants

In a GMM with $k$ components, if $\boldsymbol{\Sigma}_\ell = \boldsymbol{\Sigma} = \sigma^2 \mathbf{I}_n$ for all $\ell$, then

$$P(\mathbf{x}_j \mid \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^p}\sigma} \cdot \exp\left(-\frac{1}{2\sigma^2}\|(\mathbf{x} - \boldsymbol{\mu}_\ell)\|_2^2\right);$$

the EM algorithm applied to this special case leads to

**E-step:** $$\gamma(z_{j,\ell}^{[i]}) = \pi_\ell^{[i]} e^{\left(-\|\mathbf{x}_j - \boldsymbol{\mu}_\ell^{[i]}\|_2^2/2\sigma^2\right)} \Big/ \sum_{\nu=1}^{k} \pi_\nu^{[i]} e^{\left(-\|\mathbf{x}_j - \boldsymbol{\mu}_\nu^{[i]}\|_2^2/2\sigma^2\right)}$$

**M-step:** $$\boldsymbol{\mu}_\ell^{[i+1]} = \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]})\ \mathbf{x}_j \Big/ \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]}); \quad \pi_\ell^{[i+1]} = \frac{1}{n}\sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]}).$$

When $\sigma \to 0$, we can show that

$$\gamma(z_{j,\ell}) \to \begin{cases} 1 & \text{if } \ell = \arg\min_\nu \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_\nu\|_2^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

which is simply the formulation for $k-$**means**.

If the dataset of observations is **binary** (image data, say), we can modify GMM so $P(\mathbf{x}_j \mid \boldsymbol{\mu}_\ell)$ arises from a **multivariate Bernoulli** distribution:

$$\mathcal{B}(\mathbf{x}_j \mid \boldsymbol{\mu}_\ell) = \prod_{\nu=1}^{p} \mu_{\ell,\nu}^{\ x_{j,\nu}} (1 - \mu_{\ell,\nu})^{\ 1-x_{j,i}},$$

where $\boldsymbol{\mu}_\ell \in [0,1]^p$.

Thus, if there are $k$ components, then

$$\mathrm{LL}(\boldsymbol{\Theta}) = \ln P(\mathbf{X} \mid \boldsymbol{\Theta}) = \sum_{j=1}^{n} \ln \left( \sum_{\ell=1}^{k} \pi_\ell \prod_{\nu=1}^{p} \mu_{\ell,\nu}^{\ x_{j,\nu}} (1 - \mu_{\ell,\nu})^{\ 1-x_{j,i}} \right).$$

We can find $\boldsymbol{\Theta}^*$ that **maximizes** $\mathrm{LL}(\boldsymbol{\Theta})$ by using the EM algorithm for the **Bernoulli Mixture Model** (BMM):

**Initialization:** $\pi_\ell^{[0]} = \dfrac{1}{k}; \quad \boldsymbol{\mu}_\ell^{[0]} \sim \prod_{\nu=1}^{p} \mathcal{U}(0.25, 0.75)$

**E-step:** $\gamma(z_{j,\ell}^{[i]}) = \pi_\ell^{[i]} \prod_{\nu=1}^{p} \left(\mu_{\ell,\nu}^{[i]}\right)^{\ x_{j,\nu}} (1 - \mu_{\ell,\nu}^{[i]})^{\ 1-x_{j,i}}$

**M-step:** $\boldsymbol{\mu}_\ell^{[i+1]} = \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]})\ \mathbf{x}_j \Big/ \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]}); \quad \pi_\ell^{[i+1]} = \dfrac{1}{n} \sum_{j=1}^{n} \gamma(z_{j,\ell}^{[i]}).$

**Other Variants:** **generalized EM**, **variational EM**, **stochastic EM**.

The **essence** of EM methods is the same for all algorithms:

- **E-step** – we attempt to "**guess**" the **responsibilities** for each cluster;
- **M-step** – we **update** the model parameters based on the approximated **responsibilities** found.

**Big Strength:** EM can detect **overlapping** clusters (unlike $k-$**means**).

**Limitations:** we've already mentionned two (no guarantee that convergence will be to a global maximizer; convergence may be quite slow), but we also need to know the **number of clusters** prior to analysis.

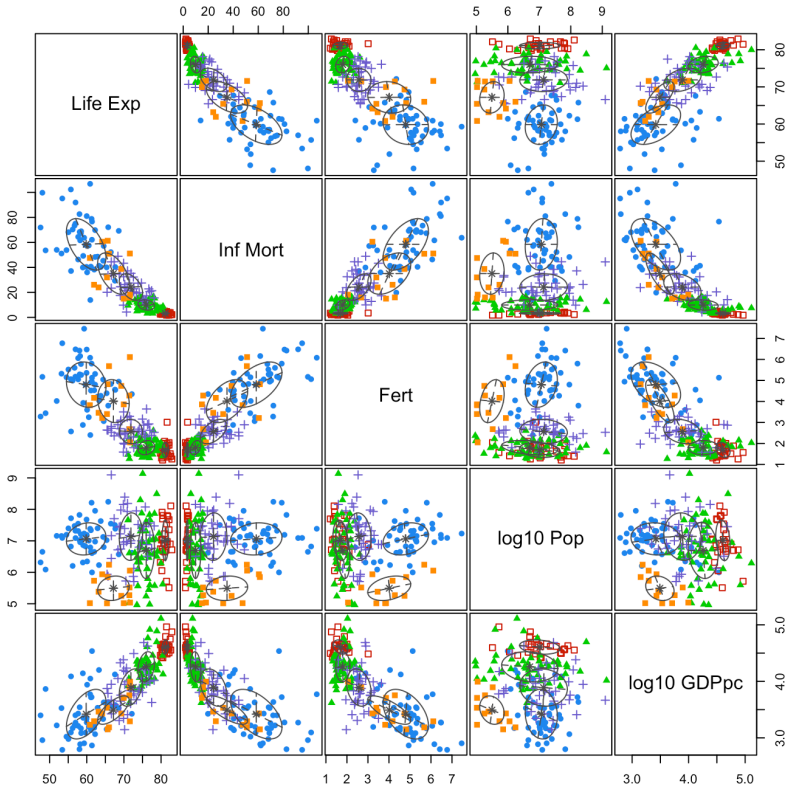
# 4.7.5 – Example: Gapminder Dataset

We attempt to cluster the 2011 Gapminder dataset for the final time using EM in `R`; no parameters need to be specified (unless we want to use a different dissimilarity measure).

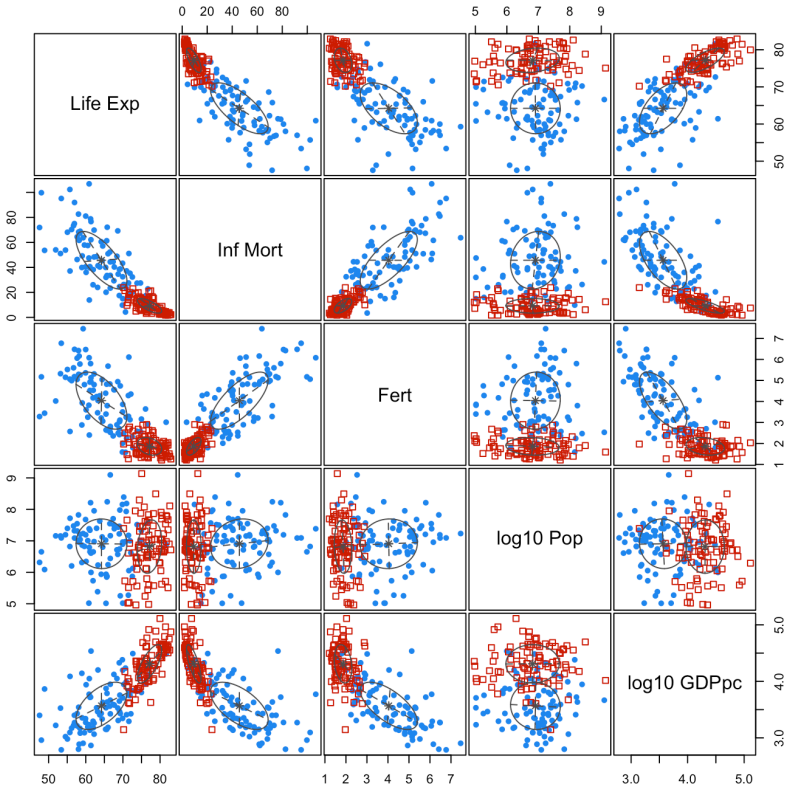
We cluster both the **raw** data and the **scaled** data, to showcase the impact scaling can have.

This implementation determines the optimal number of clusters using BIC (5 for the raw data; 2 for the scaled data).

(see DUDADS, 22.4.3, *Probability-Based Clustering* for code)




EM with 5 clusters, raw data




EM with 2 clusters, scaled data

# The Rest of the Clustering Picture

**Affinity propagation clustering:** clustering by passing information

**Fuzzy clustering:** observations can belong to more than one cluster

**Cluster ensembles:** UL's version of ensemble learning

**Time-series clustering:** what to do with time-series data

**Network clustering:** what to do with network data

**Semisupervised clustering:** leveraging known/partial information for clustering

**Latent Dirichlet allocation:** a form of probability-based clustering

etc.

# References

G.James, D.Witten, T.Hastie, and R.Tibshirani, An Introduction to Statistical Learning: With Applications in R. Springer, 2014.

C.C.Aggarwal and C.K.Reddy, Eds., Data Clustering: Algorithms and Applications. CRC Press, 2014.

C.C.Aggarwal, Data Mining: The Textbook. Cham: Springer, 2015.

J.Cranshaw, R.Schwartz, J.I.Hong, and N.M.Sadeh, "The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City," in ICWSM.

F.R.Bach and M.I.Jordan, "Learning spectral clustering, with application to speech separation," J. Mach. Learn. Res., vol. 7, pp. 1963–2001, Dec. 2006.

H.T.Kung and D.Vlah, "A spectral clustering approach to validating sensors via their peers in distributed sensor networks," Int. J. Sen. Netw., vol. 8, no. 3/4, pp. 202–208, Oct. 2010, doi: 10.1504/IJSNET.2010.036195.

C.Plant et al., "Automated detection of brain atrophy patterns based on MRI for the prediction of Alzheimer's disease," NeuroImage, vol. 50, no. 1, pp. 162–174, 2010.

Wikipedia, "Cluster analysis algorithms."

E.Schubert, J.Sander, M.Ester, H.P.Kriegel, and X.Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," ACM Trans. Database Syst., vol. 42, no. 3, Jul. 2017, doi: 10.1145/3068335.

N.Harris, "Visualizing DBSCAN clustering."

M.Grootendorst, "9 distance measures in data science," Towards Data Science, Feb. 2021.

M.Harmouch, "17 types of similarity and dissimilarity measures used in data science," Towards Data Science, Mar. 2021.

Y.Murat and Z.Cakici, "An integration of different computing approaches in traffic safety analysis," Transportation Research Procedia, vol. 22, pp. 265–274, Dec. 2017, doi: 10.1016/j.trpro.2017.03.033.

R.C.Amorim, "Feature relevance in ward's hierarchical clustering using the lp norm," J. Classif., vol. 32, no. 1, pp. 46–62, Apr. 2015, doi: 10.1007/s00357-015-9167-1.

"Ward's method." Wikipedia.

C.C.Aggarwal and C.K.Reddy, Eds., Data Clustering: Algorithms and Applications. CRC Press, 2014.

Wikipedia, "Cluster analysis algorithms."

B.Desgraupes, ClusterCrit: Clustering Indices. 2018.

D.H.Wolpert and W.G.Macready, "No free lunch theorems for optimization," IEEE Transactions on Evolutionary Computation, 1997.

L.Vendramin, R.J.G.B.Campello, and E.R.Hruschka, "Relative clustering validity criteria: A comparative overview," Stat. Anal. Data Min., vol. 3, pp. 209–235, 2010.

J.M.Lewis, M.Ackerman, and V.R.de Sa, "Human cluster evaluation and formal quality measures: A comparative study," Cognitive Science, vol. 34, 2012.

R.Yedida, "Evaluating clusters." Beginning with ML, 2019.

U.von Luxburg, "Clustering stability: An overview," Foundations and Trends in Machine Learning, vol. 2, no. 3, pp. 235–274, 2010, doi: 10.1561/2200000008.

Wikipedia, "Cluster analysis algorithms."

E.Amigó, J.Gonzalo, J.Artiles, and F.Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," Inf. Retr., vol. 12, no. 5, p. 613, 2009.

T.Lange, M.Braun, V.Roth, and J.Buhmann, "Stability-based model selection," Advances in Neural Information Processing Systems (NIPS 2002): 2002, Jun. 2003.

T.Hastie, R.Tibshirani, and J.Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. Springer, 2008.

V.U.Panchami and N.Radhika, "A novel approach for predicting the length of hospital stay with DBSCAN and supervised classification algorithms," in ICADIWT, 2014, pp. 207–212.

X.L.Xie and G.Beni, "A validity measure for fuzzy clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no. 8, pp. 841–847, 1991.

G.Schoier and G.Borruso, "Individual movements and geographical data mining. Clustering algorithms for highlighting hotspots in personal navigation routes," in Computational science and its applications - ICCSA 2011, 2011, pp. 454–465.

N.Harris, "Visualizing DBSCAN clustering."

N.X.Vinh, J.Epps, and J.Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," J. Mach. Learn. Res., vol. 11, pp. 2837–2854, Dec. 2010.

A.Y.Ng, M.I.Jordan, and Y.Weiss, "On spectral clustering: Analysis and an algorithm," in Proceedings of the 14th international conference on neural information processing systems: Natural and synthetic, 2001, pp. 849–856.

Y.Bengio, P.Vincent, and J.-F.Paiement, "Learning Eigenfunctions of Similarity: Linking Spectral Clustering and Kernel PCA," Département d'informatique et recherche opérationnelle, Université de Montréal, 1232, 2003.

U.von Luxburg, "A Tutorial on Spectral Clustering," Stat. Comput., vol. 17, no. 4, pp. 395–416, 2007.

F.Tung, A.Wong, and D.A.Clausi, "Enabling scalable spectral clustering for image segmentation," Pattern Recognition, vol. 43, no. 12, pp. 4069–4076, 2010.

L.Zelnik-Manor and P.Perona, "Self-tuning spectral clustering," in Advances in Neural Information Processing Systems, 2005, vol. 17.

D.Dueck, "Affinity propagation: Clustering data by passing messages," Ph.D. Thesis, Jan. 2009.

B.Frey and D.Dueck, "Clustering by passing messages between data points," Science (New York, N.Y.), vol. 315, pp. 972–6, Mar. 2007, doi: 10.1126/science.1136800.

L.Kaufman and P.J.Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley, 1990.

A.Bagnaro, F.Baltar, and G.Brownstein, "Reducing the arbitrary: Fuzzy detection of microbial ecotones and ecosystems – focus on the pelagic environment," Environmental Microbiome, vol. 15, no. 16, 2020.

D.Gustafson and W.C.Kessel, "Fuzzy clustering with a fuzzy covariance matrix," 1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes, pp. 761–766, 1978.

S.H.Kwon, J.Kim, and S.H.Son, "Improved cluster validity index for fuzzy clustering," Electronics Letters, vol. 57, no. 21, pp. 792–794, 2021.