A comparison between clustering algorithms

Andrea Barbieri (CLIQUE), Carlo De Nardin (GMM), Marta Tosolini (DBSCAN)

July 9, 2022

1 Introduction

Cluster analysis is one of the most common unsupervised-learning tasks. The main goal is to group a collection of objects into subsets called clusters such that objects belonging to the same group share similarities one another, and show significant differences with respect to objects belonging to different groups. The most famous clustering models fall under two distinct categories, namely hierarchical clustering, which aims to build a hierarchy of clusters, and partitional clustering, which simultaneously searches for a predefined number of clusters (such as the k-means algorithm). However, several other clustering methods have been proposed and with this project we are going to present some of these more refined an uncommon methods. In detail, we are going to examine three different approaches to cluster analysis, presenting one method for each approach, namely:

- Model-based clustering and Gaussian Mixture Models,
- Grid-based clustering and the CLIQUE algorithm,
- Density-based clustering and the DBSCAN algorithm.

We are also going to compare the three models presented using two scenarios: a synthetic scenario, where we use the models on a couple of artificial datasets in order to highlight pros and cons of each method, and a real scenario, using a real dataset and looking at the clusters found by the three methods.

2 Model-based Clustering: Gaussian Mixture Model

2.1 Introduction

A general idea to improve the disadvantage of K-means is to model each cluster as a probability distribution. In this article, a clustering approach based on **Gaussian Mixture Models** [1] will be analysed with a handy example on the **iris** dataset, taking into account the univariate case (Petal Width). The bivariate case is analogue.

2.2 Gaussian Mixture Models

A Gaussian Mixture is a function that involves multiple Gaussians each characterized by their own mean and variance. In addition, each distribution is multiplied by a weight of π to notice that there is not an equal number of samples for each category. In the *iris* dataset, data are generated from three Gaussian distributions.

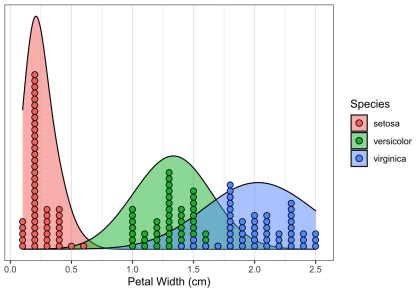


Figure 1: Iris dataset case study

GMM's purpose is to determine the probability of an observation coming from a Gaussian K; in mathematical terms: $P(x_i|\Theta)$ (Θ represents the components of each Gaussian). Optimal values for μ_k, $_k$ and π_k are values that maximize the likelihood of seeing an observation x_i.

$$P(x_i|\Theta) = \sum_{i=1}^{k} \pi_k \cdot N(x_i|\mu_k, \sigma_k)$$

To extend this to all observations, the probability of observing one sample is assumed to be independent from the others.

$$P(X|\Theta) = \prod_{n=1}^{N} \sum_{i=1}^{k} \pi_k \cdot N(x_n|\mu_k, \sigma_k)$$

Furthermore, log likelihood simplifies the expression.

$$log(P(X|\Theta)) = \sum_{n=1}^{N} log[\sum_{i=1}^{k} \pi_k \cdot N(x_n|\mu_k, \sigma_k)]$$

In the *iris* dataset, the belonging of each distribution is known since the response variable is given a priori. In this case, the parameters of each Gaussian are therefore estimated:

- π_k is the fraction of observations with label k: $\pi_k = \frac{1}{N} \sum_{n=1}^{N} [y_n = k]$
- μ_k is the mean of observations with label k: $\mu_k = \frac{\sum_{n=1}^{N} [y_n = k] x_n}{\sum_{n=1}^{N} [y_n = k]}$
- σ_k is the variance of observations with label k: $\sigma_k^2 = \frac{\sum_{n=1}^{N} [y_n = k] ||x_n - \mu_k||^2}{\sum_{n=1}^{N} [y_n = k]}$

In an unsupervised learning problem, the response variable is not given. The components of each Gaussian distribution are essential to cluster the data with GMM. The only trouble is that the belonging of each observation to a cluster remains unknown: we need an alternative solution to estimate these parameters.

2.3 Estimate Θ

In a single Gaussian distribution, the parameters can be estimated using the Maximum Probability Estimate (MLE). In this case, the weighted parameter is not included due to the fact that there is only one Gaussian.

$$P(X|\Theta) = \prod_{n=1}^{N} \pi_k \cdot N(x_n|\mu, \sigma)$$

Take the log likelihood.

$$log(P(X|\Theta)) = \sum_{n=1}^{N} log(\pi_k \cdot N(x_n|\mu, \sigma))$$

Set the partial derivate to 0 and obtain the MLE of the parameters.

$$\frac{d}{d\mu} log(P(X|\Theta)) \longrightarrow \mu_{MLE} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

$$\frac{d}{d\sigma} log(P(X|\Theta)) \longrightarrow \sigma^2_{MLE} = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)^2$$

On the other hand, it is impossible to estimate the parameters of a GMM for analytical reason. The **Expectation Maximization** algorithm can be used to solve this issue.

2.4 Expectation-Maximization (EM)

EM [2] is an iterative algorithm that uses **MLE** to estimate parameters in a statistical model with unobserved variables. It is divided into 2 steps:

- **expectation:** taking into account the parameters of each Gaussian (randomly initialized in the first step) compute the expectation of an observation to belong to each Gaussian.

$$\gamma_{ik} = \frac{P(k) * P(x_i|k)}{P(X)} = \frac{\pi_k \cdot N(x_i|\mu_k, \sigma_k)}{\sum_{k=1}^{K} \pi_k \cdot N(x_i|\mu_k, \sigma_k)}$$

  $\forall_{i,k}$, where γ_{ik} is the probability that x_i is generated by the distribution k.

- **maximization:** maximize the expectations computed in the previous step for the model parameters. This step involves updating the parameters for each Gaussian with the previous γ_{ik}.

$$\hat{\pi}_k = \frac{1}{N} \sum_{n=1}^{N} [y_n = k] \hat{\gamma}_{ik}$$

$$\hat{\mu}_k = \frac{\sum_{n=1}^{N} [y_n = k] \hat{\gamma}_{ik} x_i}{\sum_{n=1}^{N} [y_n = k] \hat{\gamma}_{ik}}$$

$$\hat{\sigma}^2_k = \frac{\sum_{n=1}^{N} [y_n = k] \hat{\gamma}_{ik} ||x_n - \mu_k||^2}{\sum_{n=1}^{N} [y_n = k] \hat{\gamma}_{ik}}$$

The following steps represent (approximately) the expectation maximization algorithm and its operation over on the iris dataset, considering the previous univariate case study.

1. Firstly, create a set of k Gaussian distributions which parameters are setted randomly. The number of k clusters is known a priori in this specific example and equal to 3.
2. Secondly, compute the likelihood that all K distributions will generate every observation x_i (E-step).

   Each observation colour in the *Fig. 4* represents the highest estimate of the probability explained above. Given a specific observation between the range [0,0.25] is more likely to be generated from the distribution 1 or 2 w.r.t. the blue one.
3. Thirdly, compute the estimates of every Gaussian distribution's component (M-step).

Actually, the last two steps are calculated in an iterative manner until a convergence is achieved.

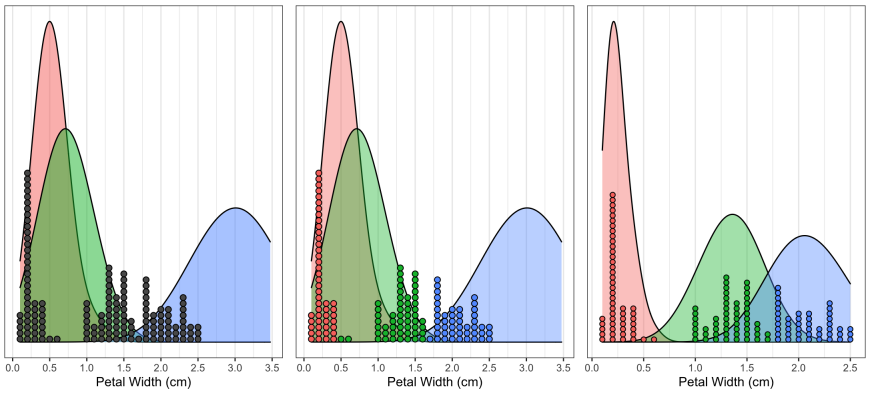


Figure 2: E-M algorithm steps

2.5 Model selection

In the example above, the number of clusters was known a priori so we landed in a global optimal solution. The problem is that if the algorithm begins from different initialization points it can land into different local optimal solutions.

One possible solution is to run the fitting procedure several times and compare each configuration with a criterion [3].

One of the most used criteria is the Bayesian Information Criterion (BIC). It provides an estimation on the accuracy of the GMM in terms of prediction of known data. In order to avoid overfitting, this criterion penalizes models with a large number of parameters. The lower is the BIC, the better is the model to predict known data.

2.6 R package: Mclust

Mclust [4] is a R package for model-based clustering, classification and density estimation based on **GMM**. It provides functionality for parameter estimation via the EM algorithm for GMM with various covariance structures.

In the bivariate case study the covariance matrix plays an important role as different types of covariance occur. It gives an idea of the "breadth" of the distribution as well as its orientation. According to the covariance matrix there are different situations in which the distributions may change significantly based on three attributes:

- **volume**: each cluster has its own volume, depending on the number of observations
- **shape**: each cluster can have the same variance (so that the distribution is spherical) or not. Otherwise is possible to find different situations like the oval one.
- **orientations**: each cluster has its own orientation, depending on the correlation of observations

In the **Mclust** package it is possible to perform all the cases for our cluster by setting an appropriate identifier.

Family	**Volume**	**Shape**	**Orientation**	**Identifier**
Spherical	Equal	Equal	NA	EII
Spherical	Variable	Equal	NA	VII
Diagonal	Equal	Equal	Axes	EEI
Diagonal	Variable	Equal	Axes	VEI
Diagonal	Equal	Variable	Axes	EVI
Diagonal	Variable	Variable	Axes	VVI
General	Equal	Equal	Equal	EEE
General	Equal	Variable	Equal	EVE
General	Variable	Equal	Equal	VEE
General	Variable	Variable	Equal	VVE
General	Equal	Equal	Variable	EEV
General	Variable	Equal	Variable	VEV
General	Equal	Variable	Variable	EVV
General	Variable	Variable	Variable	VVV

Table 1: Parameterisations of the within-group covariance matrix Σ_k for multidimensional data [4]

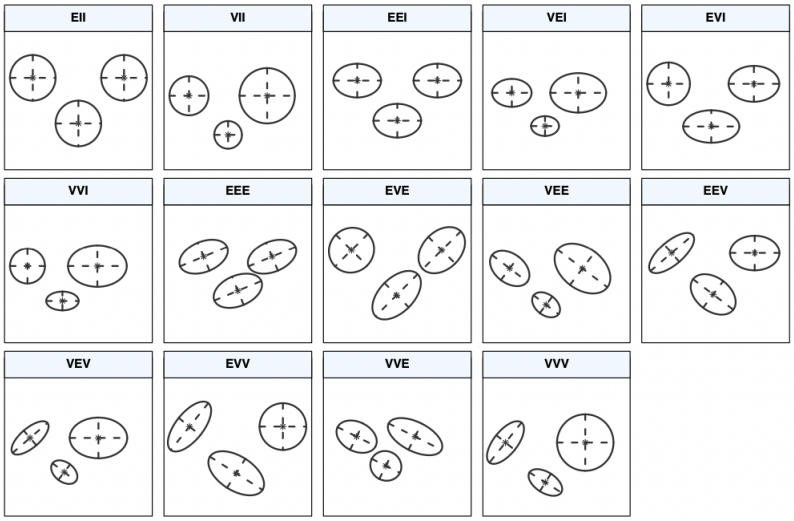


Figure 3: Plot of the 14 different Gaussian models in case of three groups in two dimensions [4]

The following example demonstrates the use of the **Mclust** package on the *iris* dataset in a bivariate case study (Petal Width and Petal Length).

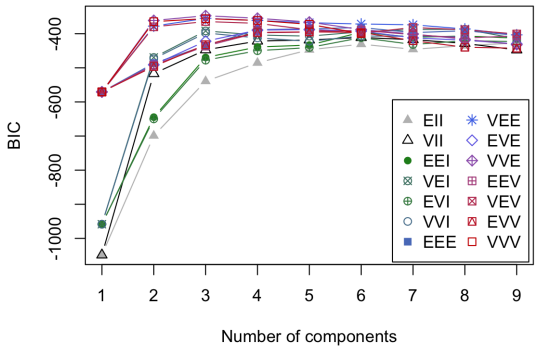


Figure 4: BIC summarize on the iris dataset considering 2 dependent variables

Based on the Bayesian Information Criterion, in this specific case, the best model is fitted with a number of components (cluster) equal to **3** and an identifier equal to **VVE**. Refering to *Tab. 1* **VVE** means: variable volume, variable shape and equal orientation.

The classification and uncertainty plots are reported below. In the uncertainty plot, the larger the observations, the greater the uncertainty.

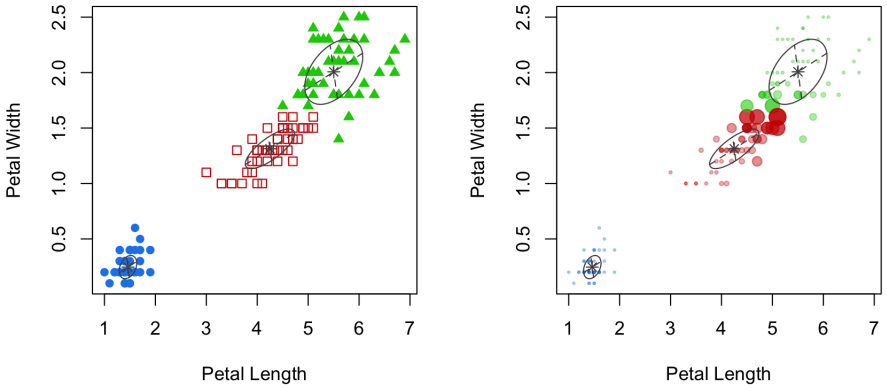


Figure 5: Classification (left) and Uncertainty (right) on the iris dataset

2.7 Conclusion

Gaussian Mixture Models are a very powerful tool used in a variety of tasks that involve data clustering.

The Expectation-Maximization algorithm suffers from the issue of convergence towards local maxima as well as the less obvious problem of overflow. In order to take these issues into account, the parameters could be selected by a heuristic method:

- Use k-means algorithm to group data first, then set weights based on k-means memberships;
- Use hierarchical clusters to group data (**mclust** default approach).

3 Grid-based Clustering: CLIQUE

3.1 Introduction

The next subgroup of clustering algorithms we're going to discuss are grid-based ones. The main idea of such algorithms is to partition the data space in a finite number of cells (commonly following hyper-rectangle shapes), in order to get a grid-version of the space, and then search for clusters inside this structure. Speaking of grid-based clustering algorithms, CLIQUE certainly stands out as the most famous one. As presented above, CLIQUE creates a grid version of the original data space and searches for clusters using a density-based approach. It is a quite fast algorithm and works nicely even in high-dimensional settings, landing also easily interpretable results.

3.2 High-level idea and parameters

CLIQUE was first presented in 1998 by Agrawal et al. [5] with the main goal of automatically find dense clusters inside subspaces of the maximal dimensionality. This method takes as input a d-dimensional set of data points $V = \{v_1, v_2, \ldots, v_m\}$, where $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$. The data space $\mathcal{S} = A_1 \times A_2 \times \cdots \times A_d$ (where A_j is the domain of the j-th component) is divided into non-overlapping hyper-rectangles called *units*, by dividing each dimension in a fixed number of intervals. We then check the number of points contained in each unit and compute, for each unit u, the *selectivity*, defined as

$$selectivity(u) = \frac{\#\text{points} \in u}{m},$$

where m is the total number of data points. If a unit has a selectivity value above a fixed threshold is called *dense*. Two k-dimensional units are called *connected* if they have a common face or if there exists a third unit which is connected to both. Note that two units can be connected even if they do not share a face in all d dimensions. A *cluster* is defined as a maximal set of connected dense units, hence the algorithm, following the above defined rules, will search for clusters in the generated grid, searching for connected dense units.

The algorithm has two tunable parameters:

- The number ξ of intervals in which every dimension is partitioned,
- The density threshold τ to classify a unit as dense.

Note that the algorithm can be generalized to allow different ξ values for different dimensions, however τ must be scaled accordingly in order to account for the difference in relative volumes when computing the density of a unit. The model can also be adapted to handle categorical variables by introducing an arbitrary order in the categorical variable domain. To partition such dimension, each interval will contain one categorical value and another empty interval will be inserted between two values.

3.3 Algorithm

CLIQUE algorithm is divided in three steps:

1. identification of subspaces containing clusters,
2. identification of clusters,
3. generation of minimal description for the clusters.

We now briefly discuss each one of these steps.

3.3.1 Subspace identification

The first step, given ξ units in d dimensions, is to identify those units above the τ threshold and to check whether clusters are present. This is done using a bottom-up algorithm, based on the following lemma:

Lemma (Monotonicity): *If a collection of points S is a cluster in a k-dimensional space, then S is also part of a cluster in any (k-1)-dimensional projections of this space.*

The algorithm is iterative and stops when no more candidates are generated. For a generic step k, the algorithm starts by passing over the data and creating the set D_{k-1}, containing all $k-1$ dimensional dense units. Then, through a procedure called *candidate generation*, a set C_k containing all the k-dimensional dense units is generated, eventually the algorithm discards from that set the dense units which have a projection in $k-1$ dimensions not included in C_{k-1}. The last step follows from the cited Lemma 1, which basically states that any $k-1$ dimensional projection of k-dimensional dense unit must also be dense.

Some modifications of this step have been proposed, we cite the MDL-based pruning, which prunes subspaces with low *coverage* (low number of units inside the cluster) making the overall algorithm faster at the cost of deleting potentially interesting subspaces.

3.3.2 Finding clusters

The second step is cluster search inside a set of dense units. Let D be a set of dense units in the same k-dimensional space S. To find clusters a depth-first algorithm is used: starting from a random unit $u \in D$ the algorithm searches for all the units it is connected to, ending assigning the first cluster number. The same procedure is applied to all the dense units which were not "visited" in the previous steps. The output of the procedure is a partition of D into $D_1, \ldots, D_q$, where q is the total number of clusters found. All units inside D_i will be connected and no two units $u_i \in D_i, u_j \in D_j$ with $i \neq j$ are connected.

3.3.3 Generating minimal cluster descriptions

The third and last step of the CLIQUE algorithm is devoted to generate concise descriptions for clusters. Taking as input disjoint sets of connected k-dimensional units in the same subspace, this step includes two further sub-steps, involving one cluster at a time:

- *Greedy growth*: in this step the algorithm greedily grows a number of rectangles (*regions*) in order to cover the cluster (i.e. to get a set $\mathcal{R}$ of rectangles such that each unit $u \in C$ is also in a rectangle $R \in \mathcal{R}$). The algorithm grows hyper-rectangular regions following this simple scheme: start from a unit $u_1 \in C$, grow it as much as possible along dimension a_1 both to the left and to the right using connected dense units and grow in the same way along dimension $a_2, a_3, \ldots, a_d$ using connected dense units, obtaining a maximal region covering u. The algorithm reiterates starting from a unit $u_2 \in C$ not yet included in a region $R \in \mathcal{R}$, repeating the same procedure until all the units inside C are covered by some region $R \in \mathcal{R}$.
- *Minimal cover*: this step aim to minimize the number of regions used to cover a cluster, taking as input the output of the previous step. This is done removing the smallest (in terms of number of units) redundant maximal regions, i.e. regions in which every unit is included in another maximal regions. The removal is repeated until no other maximal regions can be removed.

3.4 Variants

Over the years, a couple of variants of the presented algorithm were introduced. A first slight modification of is the algorithm ENCLUS (ENtropy-based CLUStering) [6], which differs for the criterion used for subspace selection: to determine whether a subspace is interesting or not, this algorithm uses an entropy-based criterion, defining as "good" subspaces the ones with entropy below an input threshold ω.

Another interesting variant is MAFIA (Merging of Adaptive Finite IntervAls) [7], which includes a major modification of the CLIQUE algorithm: this algorithm uses adaptive grids in each dimension, identifying interesting clusters using a dedicated technique based on histograms. In contrast to CLIQUE, MAFIA does not have constraints in the dimension used for clustering (CLIQUE creates high-dimensional clusters only if they share the first dimensions), resulting in a higher cluster number as output.

3.5 Pros and Cons

One of the major pros of CLIQUE is the ability to find clusters in a high dimensional space leaving untouched the features of the dataset: given the particular way the algorithm works, there's no need to reduce the dimensionality of a dataset, leading to more interpretable clusters based on the original features. Furthermore, the procedure identifies automatically the highest-dimensional clusters. Being a grid-size method, it is also fast to compute and easily scalable. Lastly, it is also a flexible method since no canonical distribution of the data is assumed.

The most evident drawback of CLIQUE resides in the grid-based approach to clustering, because such methods heavily depend on the positioning of the grids. Hence, a poorly chosen ξ parameter may lead to low clustering accuracy. The same applies to τ threshold, so the two parameters must be carefully tuned to get the best results. Another drawback is the fixed τ threshold, which remains the same in classifying both low and high dimensional subspaces as dense, with the possibility to overlook some interesting subspaces due to the curse of dimensionality.

4 Density-based clustering: the DBSCAN algorithm

4.1 Introduction

Density-based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering non-parametric algorithm proposed in 1996.

The algorithm groups point that are close to each other. The closeness is computed with two parameters, that have to be defined at the beginning:

- Epsilon (*eps*), a distance parameter that specifies how close points should be to each other.
- Minpts, the minimum number of points to form a dense region

Other than grouping, it also marks points that are outside of the regions as outliers.

The distances between points that are computed are usually the Euclidean distances, but it is possible to compute them in different ways, we won't discuss other measures in this project.

This is considered as one of the most popular clustering algorithm because it can be efficient in clustering both easy and complex region shapes (example: non-convex shapes). The hardest part is the estimation of *epsilon* and *minpts* parameters, we will see that if we don't choose the correct values the algorithm won't work efficiently [8].

4.2 Steps to follow

- We choose a random data point to begin with, let's call it P
- We draw the radius of P with epsilon as a parameter
- We check how many k points are below the radius.

  The algorithm will classify the point P into one of these three categories:

  - Core point

    In this case k is equal or greater than *minpts*, this means that the points can be aggregated in a cluster. The point P is labeled as *core* point.

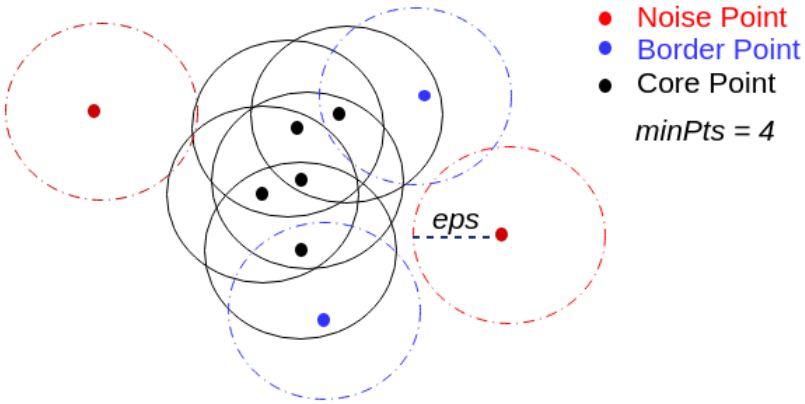
  - Border point

    In this case k is lower than *minpts*, but the point P is near to one or more *core* points, so it is labeled as *border* point. *Border* points are included in the cluster of the closest *core* point.

  - Noise point

    The last case in which k is lower than *minpts* and the point P isn't close enough to a *core* point. It is labeled as *noise* point.

    In the next picture we can observe and example of this classification with $minpts = 4$:



- We repeat the process with the other remaining points, until the density-connected cluster is completely found.

4.3 Advantages

- DBSCAN does not require a number of clusters to be chosen at the beginning. It retrieves it on his own.
- It handles outliers (noise points) well.
- Thanks to the density approach, it can find particular-shaped clusters
- The semplicty of the algorithm can be considered as an advantage. It requires the setting of just two parameters, and the steps that are followed are easily understandable.

4.4 Disadvantages

- It is difficult to cluster data with different densities.
- The estimation of epsilon becomes more difficult with high dimensional data, infact, it is suggested to apply the algorithm on a maximum of 3-dimensional data.
- The estimation of the two parameters affects the algorithm a lot. It is important to choose them correctly.[9]

4.5 Estimation of the parameters

Many estimations of the two parameters have been discussed, I will describe the most common ones:

- Minpts

  The best number for the minimum number of points can be computed from the number of dimensions (D) in the dataset. So, *minpts* should be equal or greater than D + 1. The minimum value must be 3, the larger the dataset, the larger the parameter should be.

- Epsilon (eps)

  The most used method to try to estimate the parameter is by using the k-distance graph, plotting the minpts-NN (nearest neighbour) distance ordered from the largest to the smallest value. Good values of epsilon are where this plot shows an "elbow". If the *eps* is too small, a large part of the data will not be clustered, and there will be a lot of noise points. If the parameter is too high, the majority of points will be merged in the same cluster.

4.6 DBSCAN on R Studio

The library "dbscan" is used to apply the algorithm.

These are the exact steps to follow [10]:

- We choose a parameter for minpts, that can be based on the previous explain estimations.
- Then we estimate the epsilon showing the kNNdistplot, looking for the elbow to choose and appropriate number.

  We will use kNNdistplot(data, k) function on R to display the plot.
- We apply the algorithm with the dbscan(data, eps, minPts) function.
- If we are not satisfied with the result, we can try to change the parameters to optimize the clustering.

4.7 Conclusions

The fact that this algorithm is one of the most used ones is not a surprise.

The density approach and the simplicity of it are both good qualities, and it becomes very interesting dealing with particular shaped clusters, unlike other algorithms (such as k-means) that may show great difficulties in these cases.

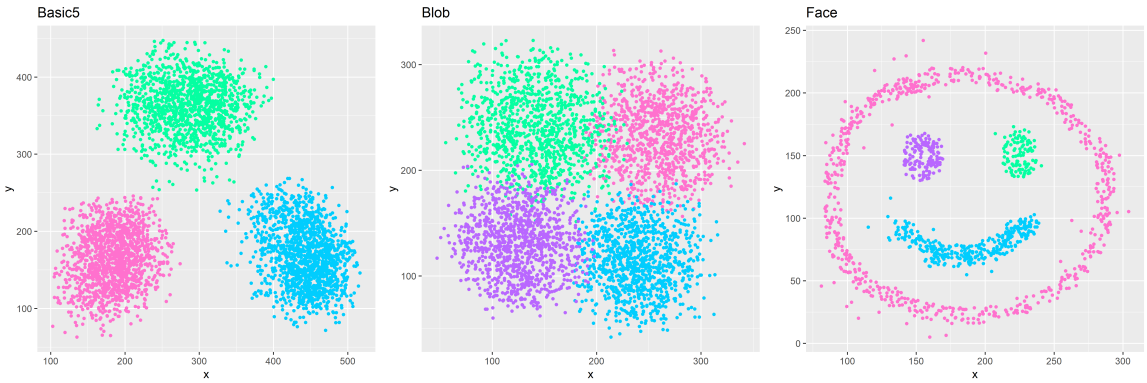
Anyway, the estimation of the two parameters, especially epsilon, will be difficult and very important to obtain a good result, and we will pay great attention on it.

The ability of finding noise points is another interesting quality, we will see the results in the next practical applications.

Different implementations of this algorithm were found, we will try just the R Studio one.

5 Artificial Dataset Applications

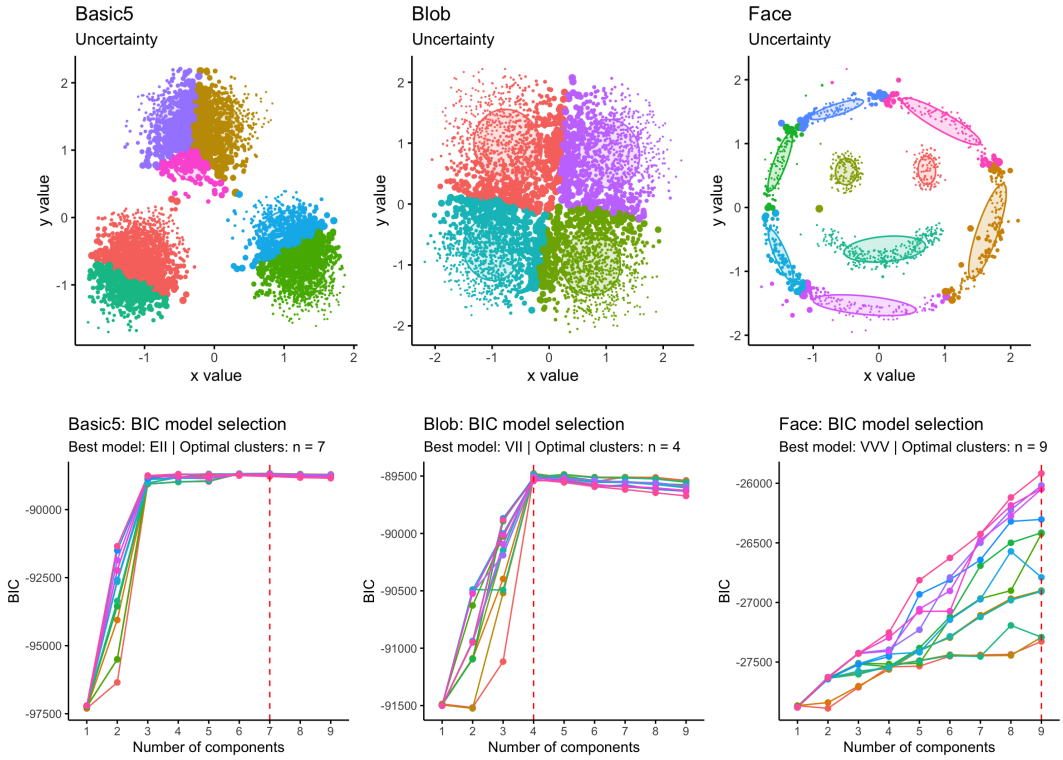
In this section we are going to use the presented model on three different artificial datasets, chosen from a collection of datasets for clustering available on the Kaggle website [11]. We decided to focus on 2-dimensional clusters, in order to easily plot them and check the models' performance just by looking at the scatterplots. In detail, the dataset chosen are the following: `Basic5` (4000 observations), showing three high-density and well separated clusters, `Blob` (4086 observations), showing four high-density clusters very close one another, `Face` (1273 observations), including four clusters placed in order to form a smiley face on the plane. A plot of the three dataset can be seen below.



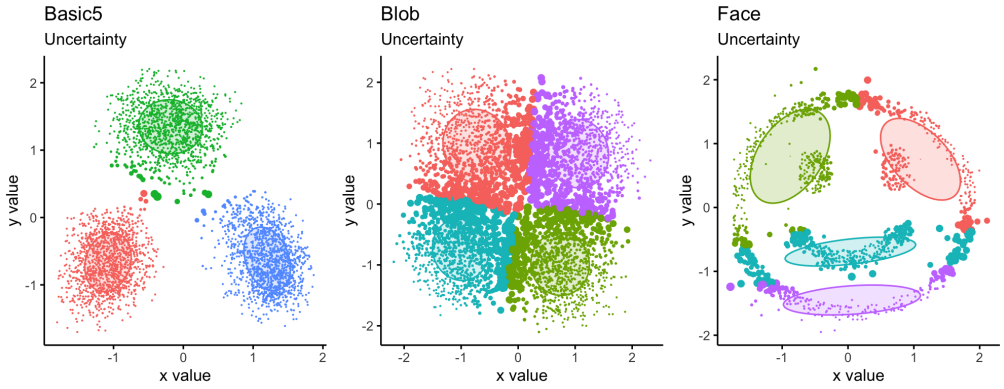
We expect that every model can easily find the three clusters on `Basic5`, however DBSCAN and CLIQUE should have some problems isolating the four clusters on `Blob`, since the latter are very close one another, creating a unique dense region that can be easily interpreted as a single cluster. On the other hand, we expect GMM to fail identifying the four clusters on `Face`, given the irregular-shaped clusters present that can hardly be modeled with Gaussian distributions.

5.1 Model-based Clustering: Gaussian Mixture Model

Initially, the different datasets were tested without an assumed number of clusters and the BIC criteria were used to analyze the different models.

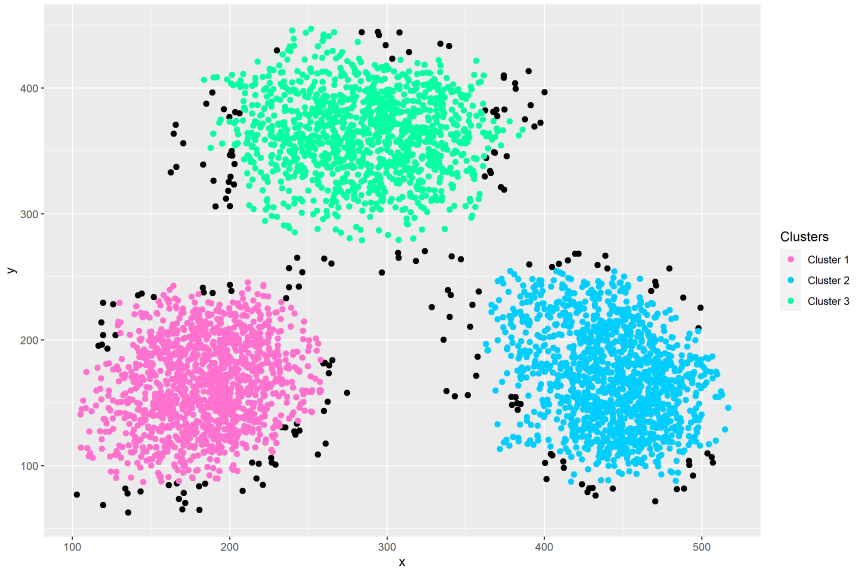


- **basic5:** Examination of the BIC curve for various models showed that a number of components of 3 was optimal. The following uncertainty plot shows a good result.
- **blob:** Looking at the BIC curve for various models, it was found that a cluster selection of 4 was optimal, so that the previous result without assumptions was optimal.
- **face:** When examining the BIC curve for various models, it was found that a cluster selection of 9 was optimal.Here it is possible to observe that the number of clusters equal to 9 is very high. Knowing a priori the cluster number equal to 4 is possible to force the clustering process but the result was not optimal.

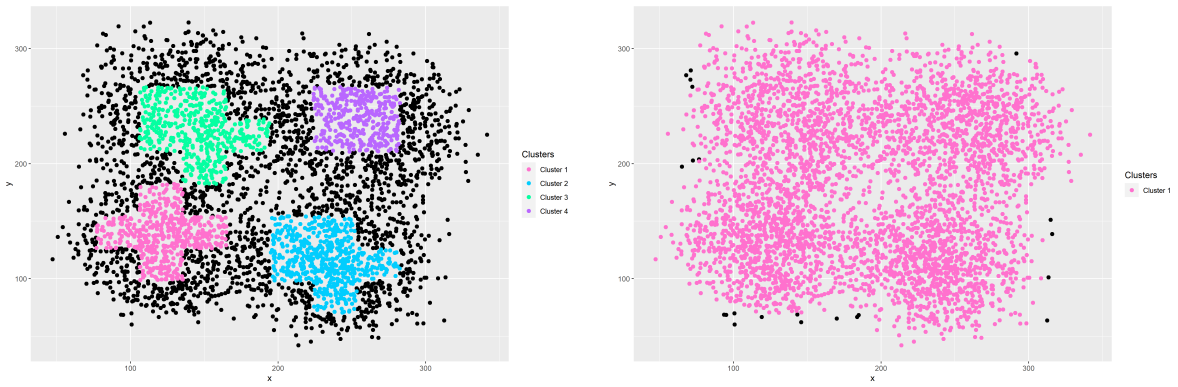


5.2 Grid-based Clustering: CLIQUE

We now take a look at the results obtained with the CLIQUE algorithm. To fit the model the `subspace` R library was used, together with `ggplot2` to visualize the results. Since no literature about parameters' tuning was found, for each dataset we manually tune them in order to get a number of bidimensional clusters equal to the number of clusters actually present in it.

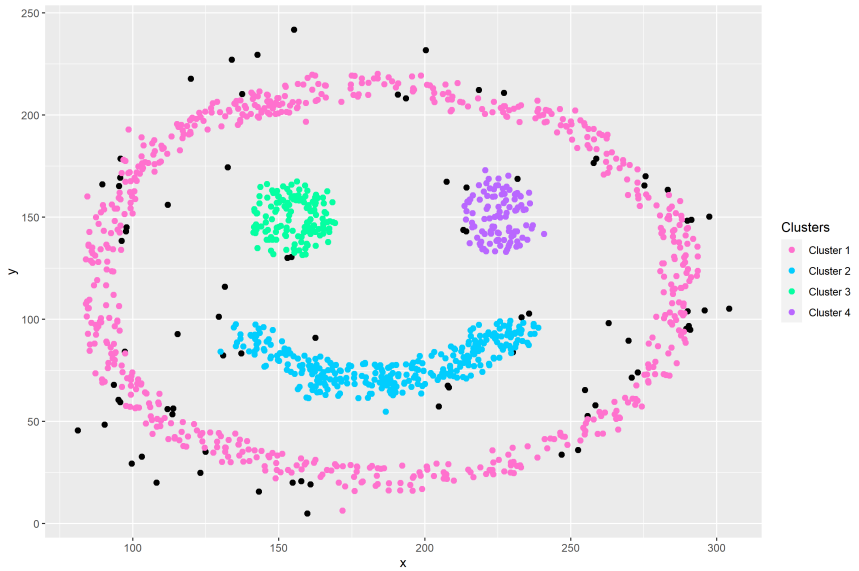


We first examine the results on the `basic5` dataset: using the parameters $\xi = 16$ and $\tau = 0.002$ the algorithm can easily isolate the three clusters, including almost all correct observations for each one of them.



Moving on to the `blob` dataset, we can clearly see that the algorithm fails to identify the four clusters present: on the left side ($\xi = 10, \tau = 0.017$) CLIQUE can find four high-density areas, corresponding to the four clusters actually present, however most of the observations do not get included in any of the four clusters.

We should also note that we have tuned the parameters in order to find exactly four bidimensional clusters: if we totally ignore this information (taking a more naïve route) the image on the right will be the most likely output of the algorithm. Given the way the points are distributed, creating four high-density regions very close one another, CLIQUE will find many connected dense units, leading to a single big cluster. In detail, the parameters used were $\xi = 10$ and $\tau = 0.001$, however similar results can be obtained with different values. As side note, one can clearly see on the left image how the algorithm works, creating rectangular-shaped units and generating clusters by looking for connected units.

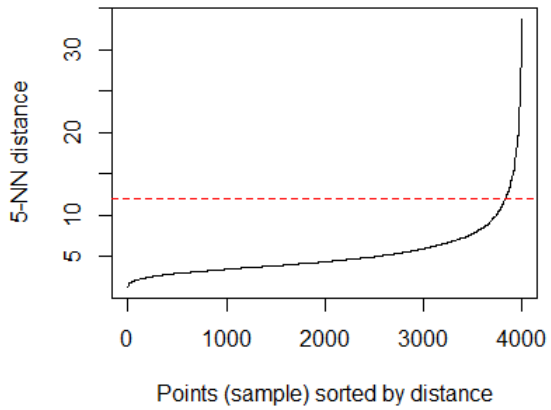


We eventually look at the `face` dataset, where the CLIQUE algorithm as expected easily finds the clusters present in the data. Using $\xi = 15$ and $\tau = 0.003$ the four clusters present are isolated with a low number of black points (not belonging to any cluster) that can be interpreted as "noise".

5.3 Density-based clustering: DBSCAN

- Example1: basic5 (3 colors) [4,000 rows]

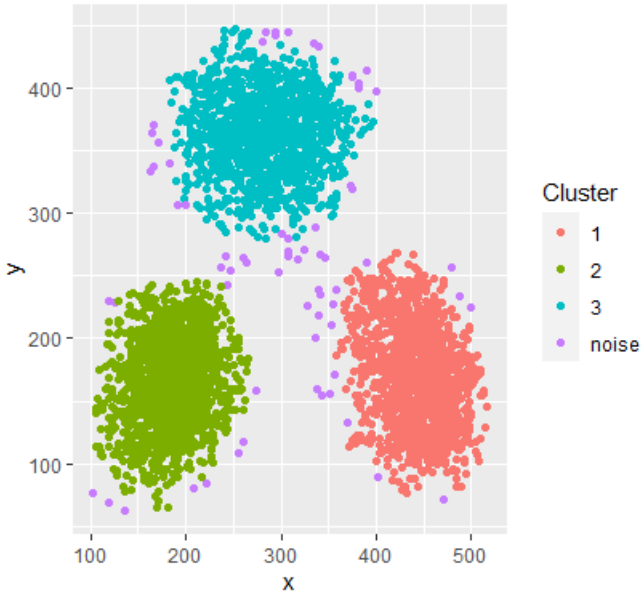
  The fist example is a basic one. We will start choosing minpts = 5 (for a large dataset). We draw the red line to try an optimal value of eps in the 5-nn distance plot. In this case epsilon = 12:



DBSCAN clustering for 4000 objects (eps = 12, minPts = 5)

The clustering contains 3 cluster(s) and 68 noise points:

Noise (68) - 1 (1369) - 2 (1311) - 3 (1252)



The result is not bad, the algorithm found the 3 clusters, but with some noise points.

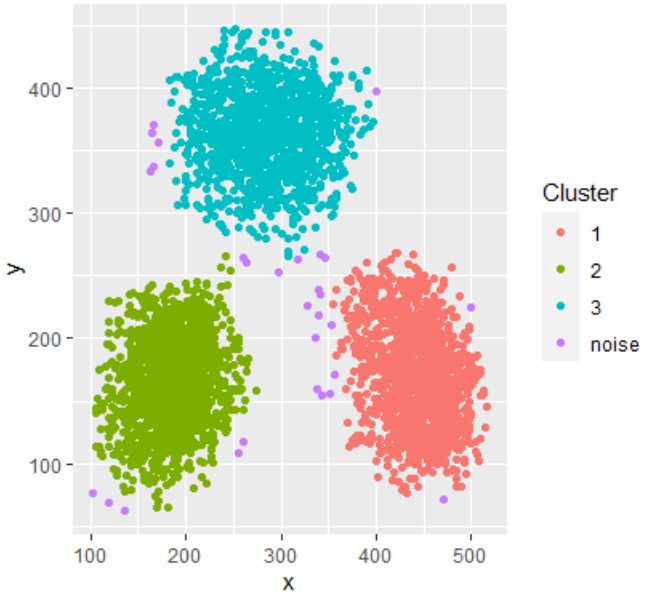
We can see that it had some difficulties for the external points of the clusters, that are more distant to the dense areas.

In this case, we can simply try to increase the epsilon for a better result. Let's try epsilon = 15:

DBSCAN clustering for 4000 objects (eps = 15, minPts = 5)

The clustering contains 3 cluster(s) and 29 noise points.

Noise (29) - 1 (1376) - 2 (1320) - 3 (1275)

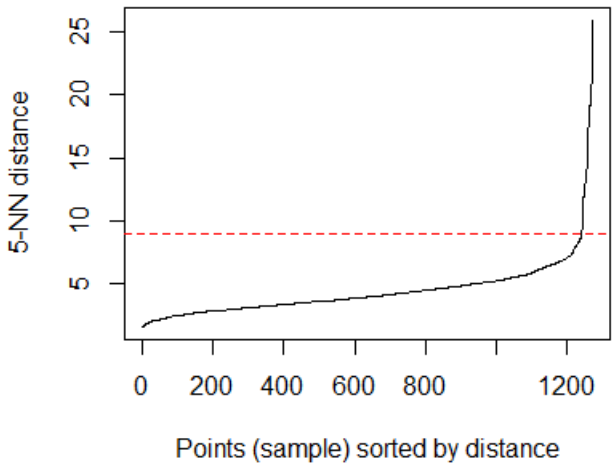


The algorithm worked better and found less noise points.

- Example 2: face (4 colors) [1273 rows]

  The second example is a complex shape. DBSCAN should not have problems finding the clusters.

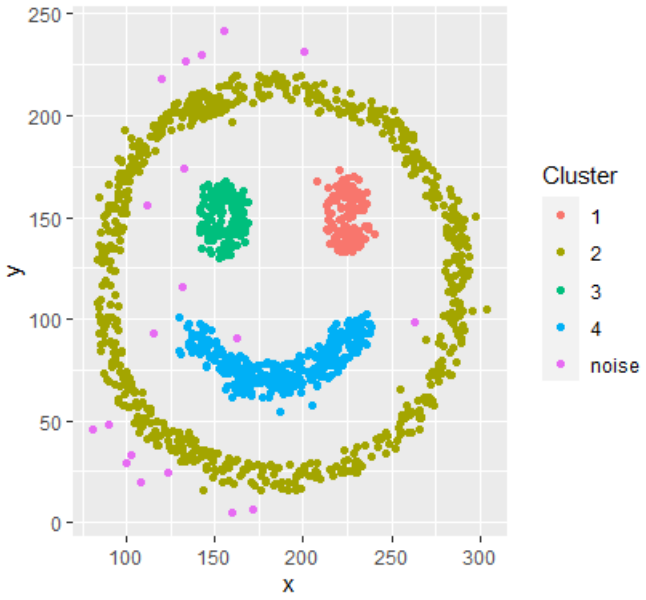
  We draw the red line to try an optimal value of eps in the 5-nn distance plot. In this case epsilon = 9:



  DBSCAN clustering for 1273 objects (eps = 9, minPts = 5)

  The clustering contains 4 cluster(s) and 19 noise points.

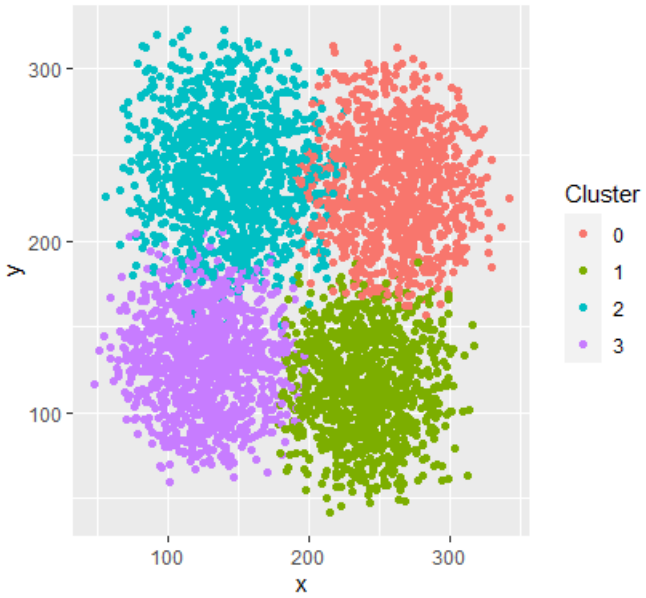
  Noise (19) - 1 (110) - 2 (692) - 3 (138) - 4(314)



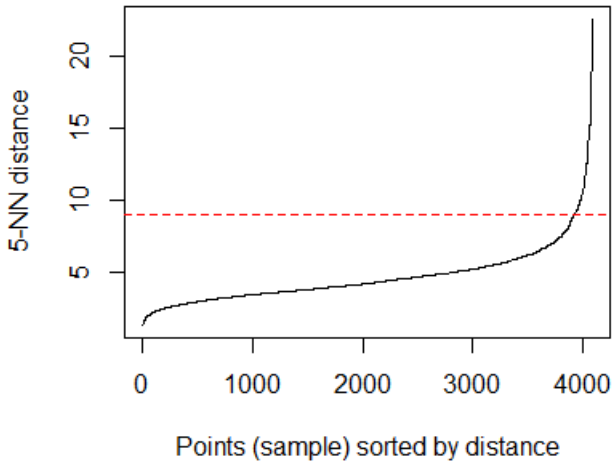
As we expected, the algorithm worked fine even with a complex shape (face).

- Example 3: blob (4 colors) [4,086 rows]

  The third example could be very difficult for DBSCAN. Let's see the 4 clusters we expect:



We can see that the points are very close to each other. The problem here is that, with a density approach, the algorithm will aggregate points in an unique cluster. Let's see the estimation for epsilon in the 5-nn plot:

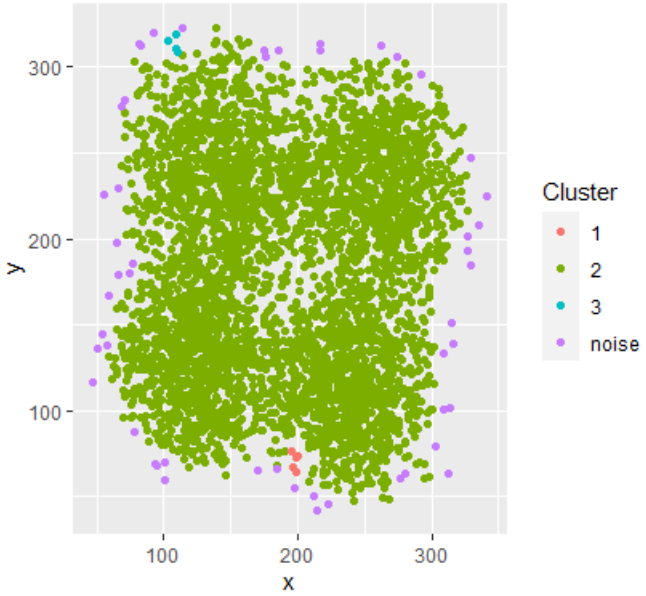


We can choose epsilon = 9 (red line).

DBSCAN clustering for 4086 objects (eps = 9, minPts = 5)

The clustering contains 3 cluster(s) and 51 noise points.

Noise (51) - 1 (5) - 2 (4026) - 3 (4)



As we expected, the algorithm found just one big cluster and some noise points. It aggregated few points in other clusters.

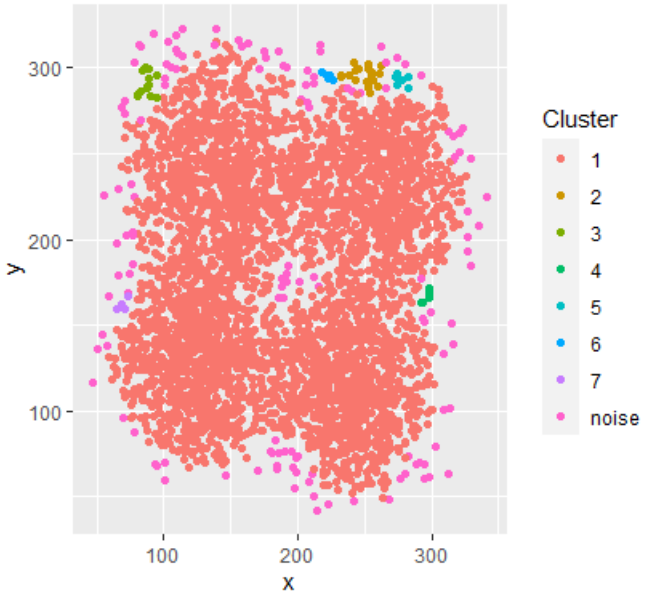
Usually, if the algorithm aggregates all the points in a big cluster, it is a correct approach to decrease the epsilon parameter.

Let's try epsilon = 7:

DBSCAN clustering for 4086 objects (eps = 7, minPts = 5)

The clustering contains 7 cluster(s) and 136 noise points

Noise (136) - 1 (3894) - 2 (20) - 3 (14) - 4 (5) - 5 (6) - 6 (6) - 7 (5)



Like the previous case, DBSCAN still could't find the 4 clusters.

Increasing or decreasing the epsilon even more won't help us. Same for minpts.

This is an example of DBSCAN not working because the 4 clusters are not well separated from each other.

6 Real Dataset Application

6.1 Overview

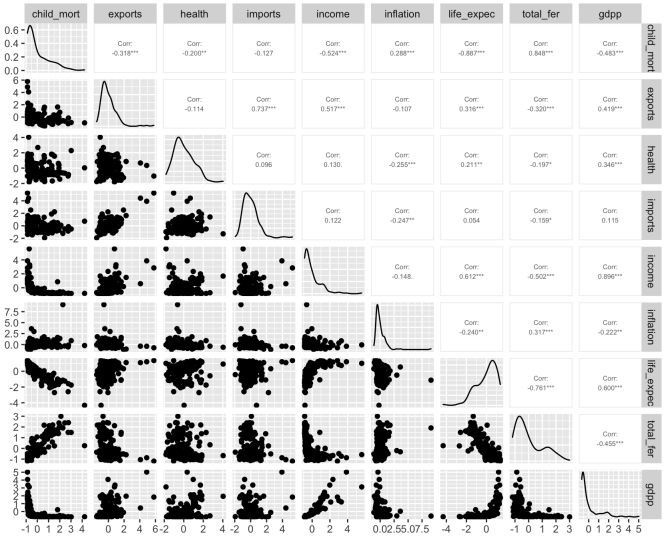
As a final achievement, we are going to use the models presented on a real dataset [12] that contains variables on health and socio-economic factors from 167 countries. The purpose of this analysis is to understand and categorize countries that need support to increase development in terms of health and economy. Variables in this dataset are:

- `country`: Name of the Country
- `child mort`: Death of children under 5 years of age per 1000 live births,
- `exports`: Exports of goods and services per capita,
- `health`: Total health spending per capita,
- `imports`: Imports of goods and services per capita
- `income`: Net income per person,
- `inflation`: The measurement of the annual growth rate of the Total GDP,
- `life expec`: The average number of years a new born child would live if the current mortality patterns are to remain the same,
- `total fer`: The number of children that would be born to each woman if the current age-fertility rates remain the same,
- `gdpp`: The GDP per capita. Calculated as the Total GDP divided by the total population.

All the variables take real values (except `country`), however the how and what they measure is very different: `exports`, `health` and `imports` are expressed as percentage of the GDPP, and together with this one they measure economic phenomena. Some variables express demographic phenomena instead, such as `child mort`, `life expec` or `total fer`, manipulating originally integer values. As a consequence of that, both means and variances of the variables will be very different from each other, leading us to the decision to standardize the dataset, centering and scaling each feature to get null mean and variance equal to 1.

6.2 Data exploration

We now observe possible relationships between variables and check the distribution of the variables.



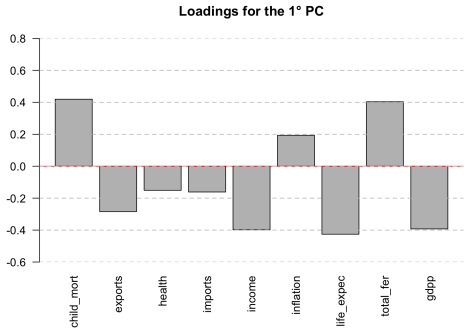
From this pairwise plot is possible to grasp some information. Child mortality rate, life expectancy and total fertility are highly correlated. gdpp is highly correlated with exports, health, income and imports. Looking at each variable at a time we can generally see that the distribution resembles the Gaussian bell-shaped curve, with some notable exceptions such as `total fer`, which has a clearly bimodal distribution.

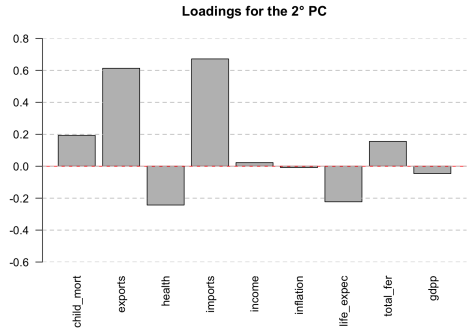
6.3 Assessment of Clustering Tendency: Hopkins test

We can conduct the Hopkins Statistic test to assess the presence of an uniform distribution across all the dimension. Fixing the α threshold to 0.01, the Hopkins Statistic is almost equal to 1 (0.99999), with a p-value equal to 0, far lower than the fixed threshold α. Given the strong evidence, we decide to reject the null hypothesis and go on with the clustering phase.

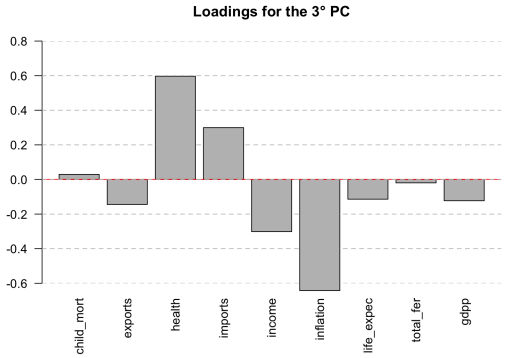
6.4 Principal Component Analysis

The different methods will be applied on the whole standardized dataset, without performing dimensionality reduction. However, a principal component analysis is carried out on the standardized dataset, to make a possible a single graphic representation containing as much information as possible regarding all the dimensions. To achieve that, we have created a scatterplot based on the first two principal components, which together explain more than the 60% of the total variance, highlighting each time the obtained clusters. We now take a look at the resulting components, trying to interpret them.





From the first plot we can conclude that child mortality, inflation and total fertility have positive weights, while life expectancy, income, gdpp have negative weights in the first PC. Summarizing, the first PC can be interpreted as "negative socio-economic dimension". The second component shows positive weights on export and import, and mixed weights (both positive and negative) in other demographic variables (child mort, life expec, total fer) even though with lower coefficients with respect to the first couple of variables. We can conclude that this component represents "international trade" of a country.



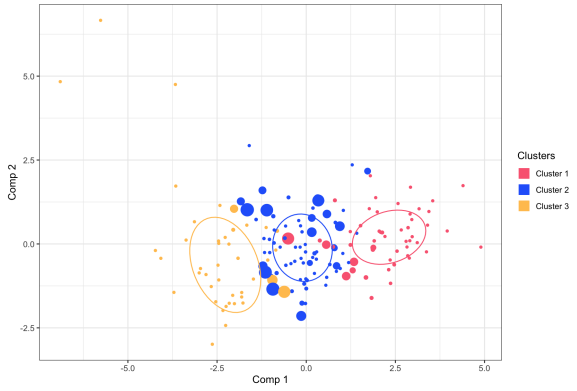
The plot with the loadings of the third PC is necessary for the case of DBSCAN algorithm, that will present a 3d plot with the result. In this case health is positively correlated and inflation is negatively correlated to the third PC.

These plots will help us understanding better the role of the Principal Components.

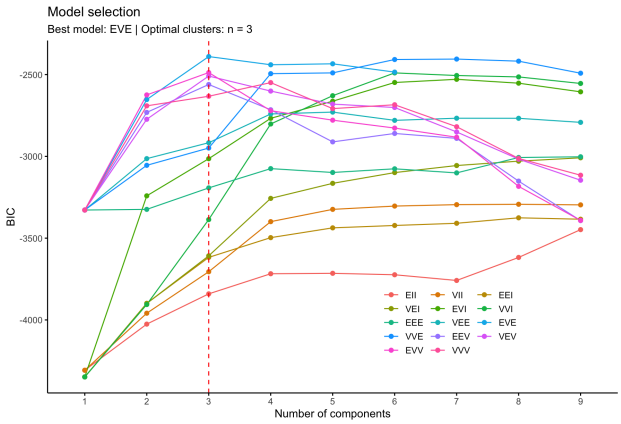
6.5 Model-based Clustering: Gaussian Mixture Model

We now examine the results achieved with the GMM clustering method on the dataset.

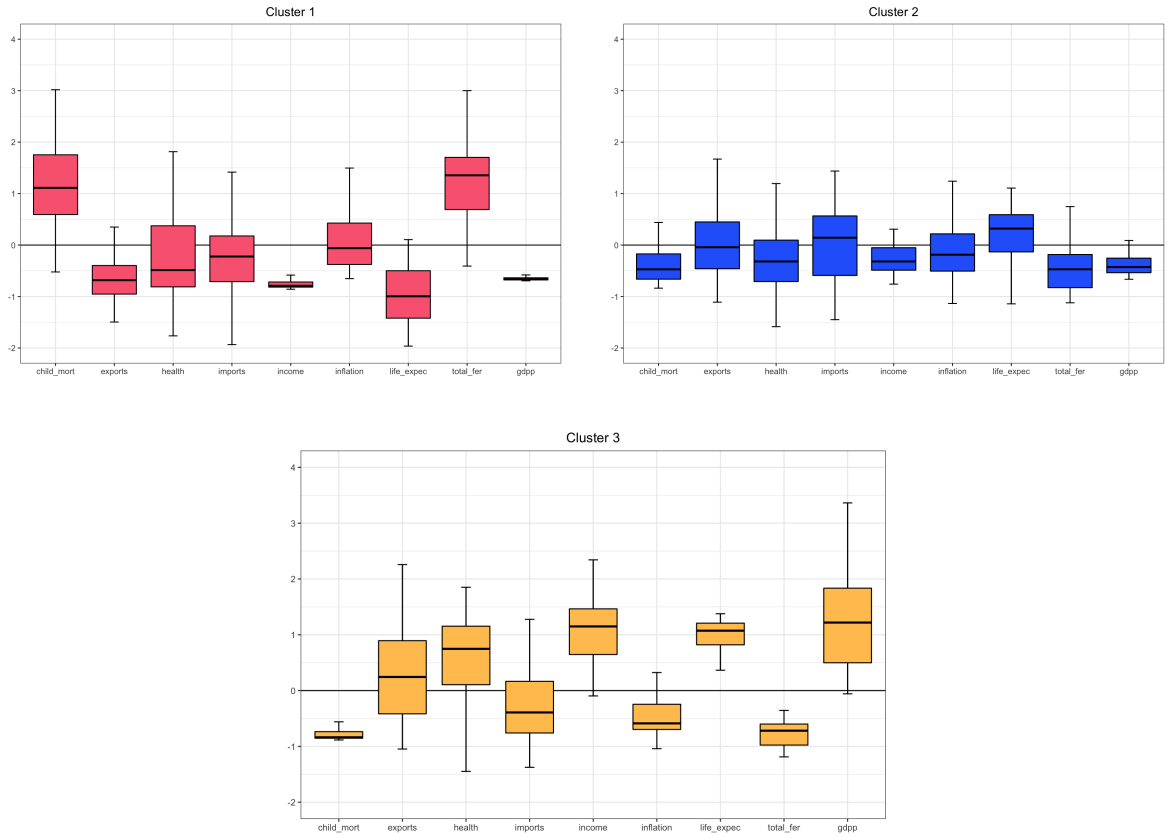
Initially, the dataset was tested without making assumption on the number of clusters. Different models were trained with different cluster shapes and number of clusters. Then, using the BIC criteria an optimal number of cluster was chosen.



Using the BIC criteria, it is possible to note that the number of components equal to 3 is optimum.



From the first plot of the first two principal components we can observe that the cluster 3 involves lower values of the first principal component with respect to the others two clusters. Cluster 2 involves values of the first principal component between the values of the first and third clusters and also presents many uncertain observations. This uncertainty on many observations may infer that the cluster number is likely to be 2. With respect to the second principal component there is not a clear pattern like the first principal component. We now look at the distribution of the features inside the three clusters.



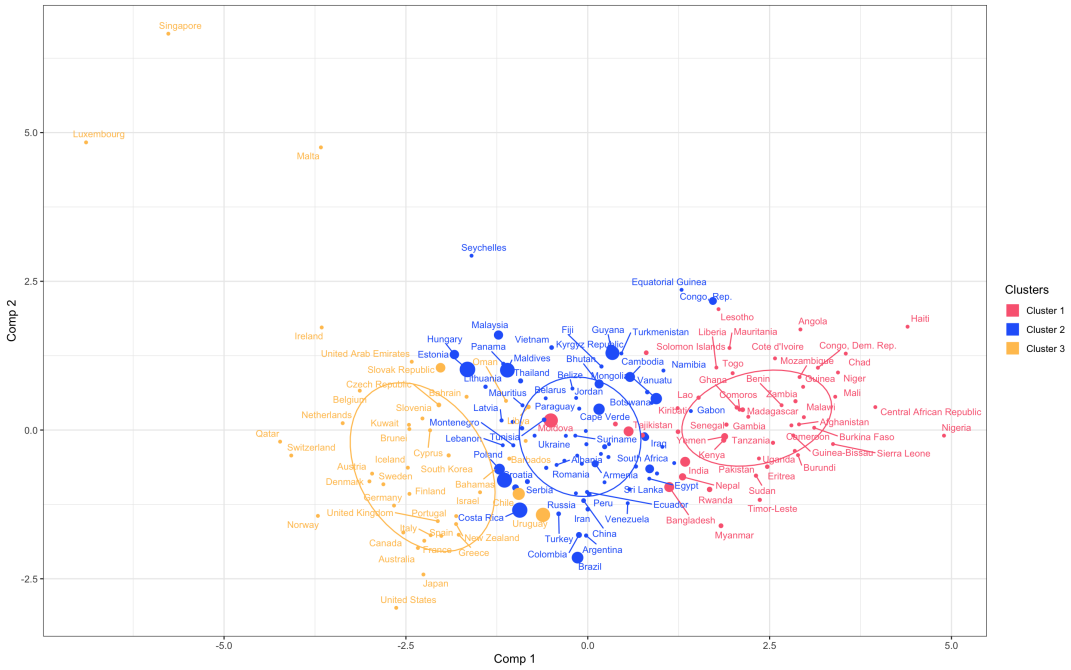
Given that the values have been standardized, the average for all countries is 0.

The first cluster shows that child mortality and total fertility are above the global average of the countries. Inflation and imports show no significant value. Instead, features like exports, health income, life expectancy and gdpp are lower than the global average of countries. From this consideration is possible to conclude that this cluster contains third-world countries (Ghana, Nigeria, Togo, Anola, etc).

The second cluster generally shows little significance with respect to the global average. Some features are below the world average of countries such as child mortality, healt, income and total fertility. This cluster is made up of the "first" third-world countries (Albania, Morocco, Algeria, etc).

The third cluster shows different features lower than the global average of the countries: child mortality, imports, inflation and total fertility. We can also see that features like exports, health, income, life expectancy and gdpp are above the global average. In general, these values correspond to rich countries (Italy, Austria, Germany, France, etc).

These consideration can be validated by plotting the first and the second principal component with the name of the countries.

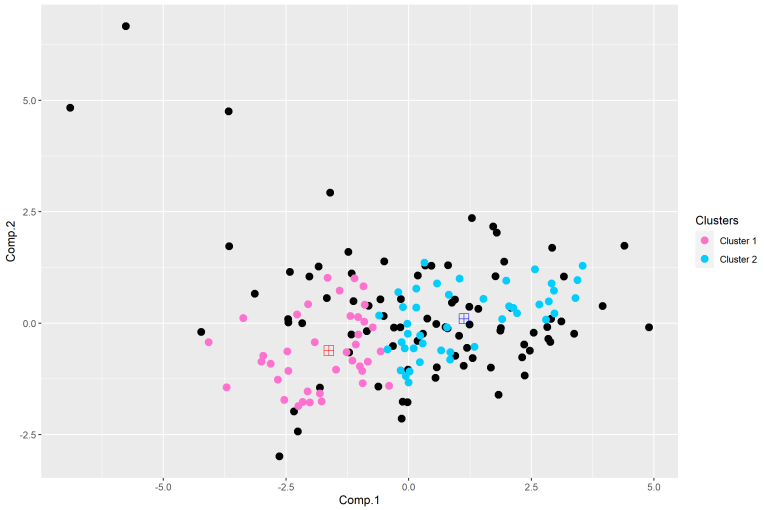


In conclusion, the GMM approach for clustering has found good clusters with respect to our idea of countries partitions for these features. However from the BIC curve it is possible to point out that we could choose a number of cluster equal to two. The problem with three cluster is that we have many uncertainty observations in the second cluster.

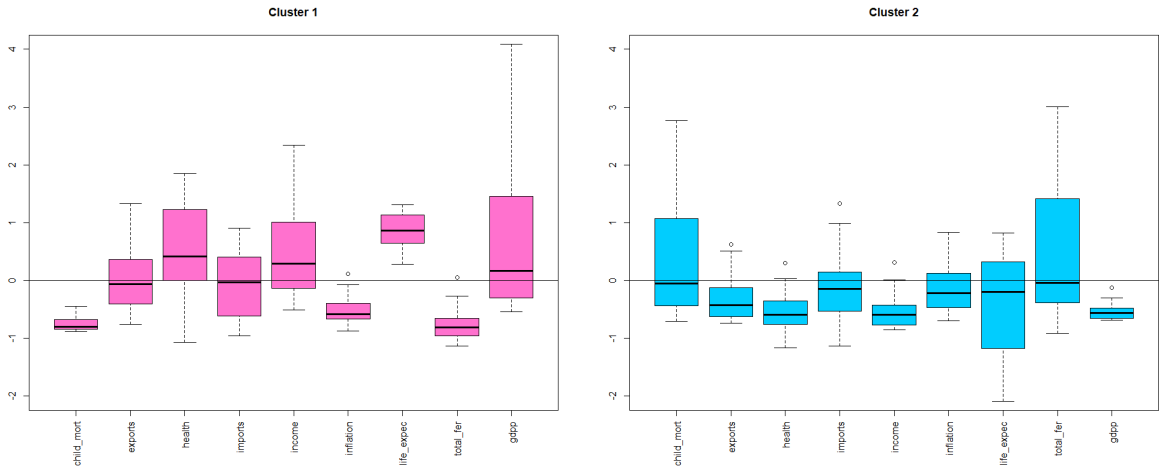
6.6 Grid-based Clustering: CLIQUE

We now take a look at the results obtained with CLIQUE on the dataset: using the normalized feature space, we ran the algorithm choosing different values, aiming to find the largest number of clusters in the highest dimensional subspace. Our final choice was $\xi = 10, \tau = 0.01$: lower ξ values and higher τ ones led us to small and insignificant clusters, while using high ξ and low τ made the algorithm take a very long time to fit, forcing us to stop the entire process before its completion.

Using the parameters fixed above CLIQUE has found a total of 3827 clusters across all the dimensions, ranging from one dimension to all of them simultaneously. Looking a the results we decided to focus on two 6-dimensional clusters, since they showed some significant differences between each other, involving a sufficiently large number of observations (they both contain 43 observations). We first plot the first two PCA components, highlighting the two obtained clusters.



Note that the red and blue squares with the plus sign inside represents the respective centroids of the clusters in the space of the principal components. We can clearly see that the first cluster involves lower values of the first principal component while the second cluster involves larger values. As for the second component, we can see that the first cluster includes observations with lower values with respect to the second cluster, however it is a more subtle difference compared to the one seen in the first component. We now look at the distribution of the standardized features inside the two clusters.



The first cluster shows a child mortality, inflation and total fertility median values below the average of all the countries (which is 0 since we have standardized the data), as well as health, income and life expectancy above the average. One can also see that median GDPP is slightly above the average. On the other hand, imports and exports are centered in zero, not showing any significant values with respect to the global average. We can conclude that this cluster depicts first-world countries, and this is confirmed by looking at the countries actually present in the cluster, such as most of the European countries (Finland, Norway, UK, Italy, Germany, Spain and others), Canada and South Korea.

Looking now at the second cluster, we can first see that it shows GDPP, health and income median values below the average. We can also see that child mortality and total fertility have a median value close to zero, however a long tail towards higher values is visible. also export and import median values show values below the average, especially the first one. Summarizing, this cluster probably depicts third-world countries, an idea confirmed looking at the countries present, comprehending countries such as China, India and many African countries (Ghana, Togo, Mali, Morocco and others).

Interestingly enough, four countries are present simultaneously in both clusters, namely Albania, El Salvador, Peru and Tunisia. This can be due to the fact that all these countries can be classified as "rich" third-world countries (showing lower values of the first principal component with respect to the centroid) or "poor" first world-countries (showing higher values of the first principal component with respect to the centroid) simultaneously.

In conclusion, we can say that CLIQUE has found some good clusters however it is not the most straightforward model for clustering: the algorithm automatically searches for clusters, however we do not know a-priori if those clusters can be actually interesting and as the total number of clusters found increases it becomes more difficult to search for useful clusters, since we have to manually look at a higher number of groups. Another drawback is the parameters' tuning, since a poor choice of the initial parameters leads to poor clusters as output. This can be easily seen for instance in the model used for the world data example: with the given parameters we have found some sensitive clusters, however also some uninformative 8-dimensional clusters were found, including only two observations each. On the other hand, the same parameters led to a large number of 2-dimensional clusters including almost all the observations, resulting ineffective in isolating interesting groups in the data. A good rule of thumb could be to select the τ threshold value based on how many dimensions we want for each cluster, setting a large threshold for low-dimensional clusters and smaller one for high-dimensional clusters (a direct consequence of the curse of dimensionality issue).

6.7 Density Based Clustering: DBSCAN Algorithm

Finally, we applied the DBSCAN algorithm to find some possible clusters on the real dataset.

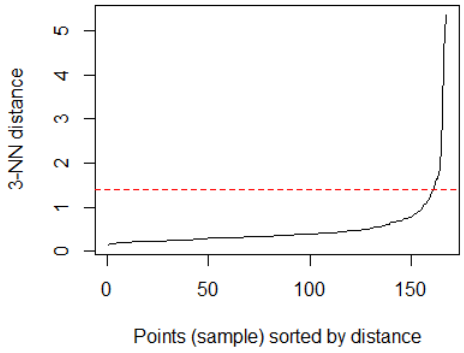
Note that DBSCAN doesn't work well with large dimensions (more than 3). In this case we have 9 dimensions as an input, so the algorithm will be applied after the reduction of dimensions.

We will try the algorithm on both 2 and 3 dimensions.

- Two Principal Components

  We will start by applying the algorithm on the first two PCs.

  In this case dimension D = 2, so we will choose minpts = D + 1 = 3. Let's see the plot of 3-nn distance looking for the "elbow".



  A good value for epsilon is 1.4 (red line).

  DBSCAN clustering for 167 objects (eps = 1.4, minPts = 3)

  The clustering contains 1 cluster(s) and 3 noise points.

  noise (3) - 1 (164)

  The algorithm clustered most of the points in 1 single cluster, and identified 3 noise points. We expected these result, remember that DBSCAN is based on a density approach, plotting the first two PCs we can observe one dense area, and the algorithm worked fine clustering the points in a unique cluster.

  The noise points represent the states: Malta, Luxembourg and Singapore.

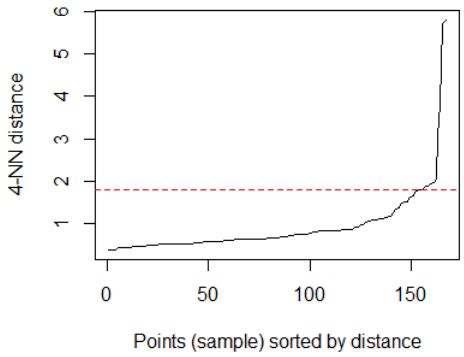
  But we have to consider the fact that the first two PCs covered about 63 percentage of the total variance.

So, it is a useful approach adding another Principal Component.

- Three Principal Components

  Let's try to apply the algorithm on the first three PCs, that cover about more than 70 percentage of the total variance. Remember that DBSCAN works well even with 3 dimensions.

  In this case dimension D = 3, so we will choose minpts = D + 1 = 4. Let's see the plot of 4-nn distance looking for the "elbow".

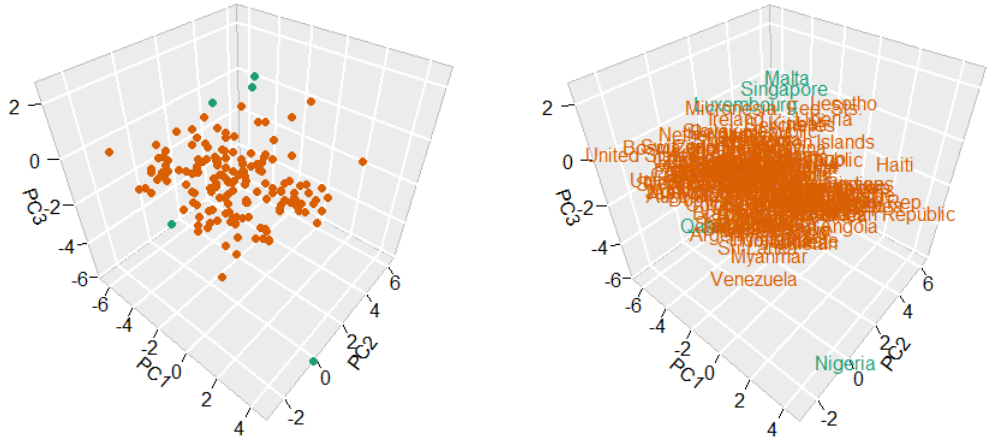


  In this case we choose epsilon = 1.8.

  DBSCAN clustering for 167 objects (eps = 1.8, minPts = 4)

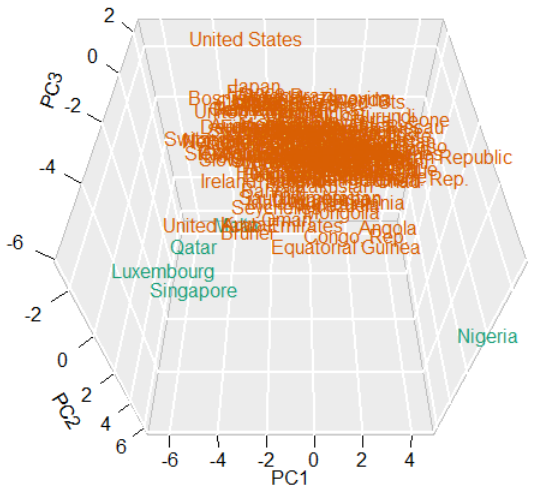
  The clustering contains 1 cluster(s) and 5 noise points. noise (5) - 1 (162)

  We will use the library plot3D to plot 3 dimensions.

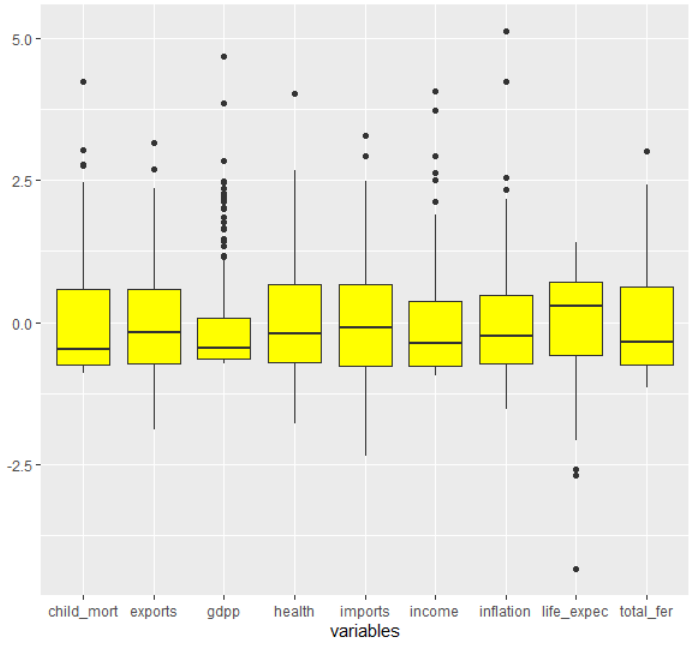


  As we can observe in the result, plotting the first three principal components, there's only one big cluster. DBSCAN identified 5 noise points. We can see that Malta, Luxembourg, Singapore, Nigeria, Qatar are states considered as noise points, and they don't take part into the single big cluster.

  From the first 3d plot, it is not very clear that Qatar is a noise point, so, a rotation of the plot shows better that this is actually true:



We can say that DBSCAN applied to the first two and three PCs didn't help us too much finding more than 1 cluster, but it identified some noise points.



Even if most of the states are grouped in one cluster, we did the boxplot to understand how the values are distributed for each variable. In this case we observe that the standardized median is near zero for each feature, but we observe some outliers, expecially for gdpp feature.

Let's try to compare the values of features in the noise points (Malta, Luxembourg, Singapore, Nigeria, Qatar) from the boxplot.

country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
Luxembourg	-0.4923947	0.6877300	0.8294923	0.5270953	0.5654215	-0.4459947	0.4813931	-0.4004083	1.38204233
Malta	-0.4207215	0.3960943	1.1443988	0.6861784	-0.7500912	-0.4413250	0.3729712	-0.5390111	-0.69099646
Nigeria	1.7868122	-1.2967185	-0.1366980	-1.1247175	-1.2304401	1.7861135	-1.7737818	1.7607696	-1.15477402
Qatar	-0.3813013	-0.8062402	-1.3032835	-1.0398732	1.2563768	-0.3712798	0.2862337	-0.1745369	0.52465918
Singapore	-0.4923947	1.0191343	-0.5339097	0.9513169	0.1587330	-0.5275140	0.6331837	-0.6468133	-0.06093103

Starting from Luxembourg, just health and gdpp are a little bit higher from the standardized median, for the case of Malta the health is higher.

Nigeria's values are all far from the mean (except from health), in particular for gdpp feature,

the value is the lowest, this suggest that it is correct consider it as a noise point.

In the case of Qatar health and imports are lower than usual, income is high, again it was correct to classify it as a noise point.

Finally, Singapore's export value is higher than usual, the other features are not very far from the mean of features in the main cluster.

As a final consideration, Luxembourg and Malta are both very small states (low population), Nigeria is one of the poorest countries, Singapore is famous for high export, and finally Qatar is one of the richest nations, famous for oil production.

So, these 5 countries are actually considered different from all the usual countries, so in general we may be satisfied from the fact that the algorithm, at least, classified some particular countries as noise points.

On the other hand, this algorithm suggested us that all the other countries take part on a single cluster, which is not useful for the task of classifying countries in different clusters.

To conclude we could say that, for this task, Grid-based and Model-based algorithms were useful to find multiple clusters of countries while Density-based one was more useful to find some "noise" countries (considered very different from the others).

References

[1] Melnykov, V., Maitra R.: *Finite mixture models and model-based clustering.* Statistics Surveys, 2010.

[2] Biarnes, A.: *Gaussian mixture models and expectation-maximization (full explanation)*, 2021. https://towardsdatascience.com/gaussian-mixture-models-and-expectation-maximization-a-full-explanation-50fa94111ddd.

[3] Huang, T., Peng H. Zhang K.: *Model selection for gaussian mixture models.* Statistica Sinica, 2017.

[4] Scrucca, L., Fop M. Murphy T. Raftery A.: *mclust 5: Clustering, classification and density estimation using gaussian finite mixture models.* The R Journal, 2016.

[5] Agrawal, Rakesh, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan: *Automatic subspace clustering of high dimensional data for data mining applications.* SIGMOD Rec., 27(2):94–105, jun 1998, ISSN 0163-5808. https://doi.org/10.1145/276305.276314.

[6] Cheng, C. H.: *Entropy-based subspace clustering for mining numerical data.* 1999.

[7] S. Goil, H. Nagesh and A. Choudhary: *Mafia: Efficient and scalable subspace clustering for very large data sets.* 1999.

[8] Ester, Martin; Kriegel, Hans Peter; Sander J org; Xu Xiaowei (1996). Simoudis Evangelos;Han Jiawei; Fayyad Usama M. (eds.): *A density-based algorithm for discovering clusters in large spatial databases with noise.* 1996.

[9] Schubert, Erich; Sander, Jörg; Ester Martin; Kriegel Hans Peter; Xu Xiaowei (July 2017): *Dbscan revisited, revisited: Why and how you should (still) use dbscan.*

[10] Pegoraro, Enrico: *Statistica per data science con r - v. 03.*

[11] Joonas: *Clustering exercises*, 2021. https://www.kaggle.com/datasets/joonasyoon/clustering-exercises.

[12] Kokkula, Rohan: *Unsupervised learning on country data*, 2020. https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data.