PERFORMANCE OF A NOVEL ALGORITHM FOR CLUSTERING (HDBSCAN) APPLIED TO COVARIANCE MATRICES

By

Raúl Orlando Valerio Martínez

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

MATHEMATICS STATISTICS

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

2018

Approved by:

_____	_____
Pedro Torres Saavedra, Ph.D. President, Graduate Committee	Date
_____	_____
Dámaris Santana Morantn, Ph.D. Member, Graduate Committee	Date
_____	_____ Date
Wolfang Rolke, Ph.D. Member, Graduate Committee	
_____	_____
José Santivañez, Ph.D. Representative of Graduate Studies	Date
_____	_____
Victor Ocasio, Ph.D. Chairperson of the Department	Date

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

PERFORMANCE OF A NOVEL ALGORITHM FOR CLUSTERING (HDBSCAN) APPLIED TO COVARIANCE MATRICES

By

Raúl Orlando Valerio Martínez

July 2018

Chair: Pedro Torres Saavedra

Major Department: Mathematics

The use of grouping algorithms to divide and classify information in some phenomena is much easier in these modern times due to the advances in computing. However, measuring the efficiency and performance of these algorithms is necessary to identify the best methods to achieve the clustering. There exist several clustering or classification algorithms in literature such as KNN and k-means. More recently, the HDBSCAN, a density-based spatial hierarchical clustering algorithm, has been proposed. This algorithm has the ability to detect arbitrarily shaped clusters and it requires the specification of fewer parameters for achieving the best possible classification when compared to its competitors. Simulation studies have shown that this algorithm outperforms its competitors when clustering objects with several features. Nonetheless, the HDBSCAN algorithm has not been used for clustering of covariance matrices. Hence, this thesis proposes the use of the HDBSCAN

algorithm for clustering covariance matrices, a task that could have applications in different areas such as time series, image processing, among others.

A comparison of the performance of HDBSCAN with DBSCAN, K-NN and k-means is done using simulation studies. The scenarios of the simulation studies focus mainly on the sample size (number of matrices), number of clusters, size of the matrices, and distance metric. The relevance of this study is that, to our best knowledge, the HDBSCAN has not been implemented for clustering of covariance matrices. One of the factors having a large influence in the performance of a clustering algorithm is the distance metric. In this work, a revision of distance metrics between matrices is given. In particular, this thesis considers an affine invariant transformation (AIRM) to the calculate distance between symmetric positive definite matrices (SPD). This metric is compared with some popular distance metrics for matrices.

Simulation studies suggest that the combination of distance metric AIRM and HDBSCAN exhibit the higher computational cost for large arrays of matrices. Nonetheless, this combination is effective for clustering high-dimensional matrices. K-means and HDBSCAN have comparable results for small number of clusters and high-dimensional covariance matrices. However, when the input parameters of the algorithms change, purity values for HDBSCAN do not change considerably (i.e., HDBSCAN is more robust to changes of input parameters). K-means algorithm suffers when the input parameters are manipulated, a sensitive issue when dealing with real problems. These findings

demonstrate that HDBSCAN offers the highest robustness and performance for the four analyzed algorithms, a result that is consistent with previous finding for vectors.

Resumen de Disertación Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de maestría en Ciencias

DESEMPEÑO DE UN NUEVO ALGORITMO DE AGRUPAMIENTO (HDBSCAN) EN MATRICES DE COVARIANZA

Por

Raúl Orlando Valerio Martínez

Julio 2018

Consejero: Pedro Torres - Saavedra

Departamento: Matemática

El uso de algoritmos de agrupamiento para dividir y clasificar información en algunos fenómenos es mucho más fácil en estos tiempos modernos debido al avance en la computación. Sin embargo, medir la eficiencia y el desempeño de estos algoritmos es necesario para identificar los mejores métodos para lograr el agrupamiento. Existen muchos algoritmos de agrupamiento o de clasificación en la literatura tales como KNN y K-means. Mas recientemente, HDBSCAN, un algoritmo jerárquico espacial basado en densidad ha sido propuesto. Este algoritmo tiene la habilidad de detector clústeres de densidad arbitraria y requiere la especificación de menos parámetros para lograr la mejor clasificación posible comparado con sus competidores. Los Estudios de simulación han mostrado que este algoritmo supera a sus competidores cuando se agrupan objetos con muchas características. Sin embargo, el algoritmo HDBSCAN no ha sido usado para

agrupar matrices de covarianza. Por tanto, esta tesis propone el uso de HDBSCAN para agrupar matrices de covarianza, una tarea que podría tener aplicaciones en diferentes áreas como ser series de tiempo, procesamiento de imágenes, entre otros.

Una comparación de desempeño de HDBSCAN con DBSCAN, K-NN y K-means es hecha usando estudios de simulación. Los escenarios se enfocan principalmente en el tamaño de muestra (número de matrices), numero de clústeres, tamaño de matrices, y la métrica de distancia. La relevancia de este estudio es que, a nuestro saber, HDBSCAN no ha sido implementado para el agrupamiento de matrices de covarianza. Uno de los factores que tiene una gran influencia en el desempeño de los algoritmos de agrupamiento es la métrica de distancia. En este trabajo, una revisión de las métricas de distancia entre matrices es hecho. En particular, esta tesis considera la transformación invariante afín (AIRM) para calcular la distancia entre matrices definidas positivas simétricas (SPD). Esta métrica es comparada con algunas métricas de distancia para matrices muy populares.

Los estudios de simulación sugieren que la combinación de la métrica de distancia AIRM y HDBSCAN exhiben un mayor costo computacional para largos arreglos de matrices. En todo caso, esta combinación es efectiva para matrices de alta dimensión. K-means y HDBSCAN tienen resultados comparables para bajo número de clústeres y matrices de alta dimensión.

Sin embargo, cuando el parámetro de entrada de los algoritmos cambia, los valores de pureza para HDBSCAN no cambian considerablemente (i.e. HDBSCAN es más robusto

para cambios en los parámetros de entrada). K-means sufre cuando el parámetro de entrada es manipulado, un asunto delicado cuando se trata con problemas de la vida. Estos hallazgos demuestran que HDBSCAN ofrece la mayor robustez y desempeño para los cuatro algoritmos analizados, un resultado consistente con previos hallazgos para vectores.

Dedicated to:

To the Creator, my wife Sara and my kids Paolo and Alessandro with everlasting love.
Los amo.

ACKNOWLEDGEMENTS

- I wish to express my gratitude to my advisor Pedro A. Torres – Saavedra, Ph.D., for his exceptional guidance, constructive comments and patience during my studies.
- I am very grateful to Santiago Velasco – Forero, Ph.D., for his ideas and advice for the development of this work.
- My deepest gratitude to my wife Sara for being my support, for her patience and her unconditional love.
- My Kids, Paolo and Alessandro, the ones who I took time from and gave me their energy without hesitation.
- Thanks to my parents Raúl (R.I.P) and Mery Olinda for their care, endless love, support and tremendous sacrifices through all these years. And all my family who encourage me to take every step to achieve my goals.
- I am also grateful to all the staff of the Department of Mathematical Sciences for their collaboration in these years.
- To all my unforgettable friends I met during my studies in Puerto Rico,

Thank you !!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY OF TERMS

AIRM	Affine Invariant Riemannian Metric
Cpp	C++ programming language
DBSCAN	Density Based Spatial Clustering of Application with Noise
μ	Mean of a random variable (location parameter)
ϵ	Minimum distance between two points to become neighbors
HDBSCAN	Hierarchical Density Based Spatial of Clustering Application with Noise
KNN	k- Nearest Neighbor
NN	Nearest Neighbor
m_{pts}	Minimum points of neighbors
LERM	Logarithm Euclidean Riemannian Metric
R^2	Pearson correlation coefficient or coefficient of determination
R	R statistical programming language
p	Purity measure
σ	Standard deviation of a random variable (scale parameter)
Σ	Variance Matrix
σ^2	Variance of a random variable
σ_p	Standard error of proportion
w.r.t.	With respect to

1. INTRODUCTION

HDBSCAN (Hierarchical Density - Based Spatial Clustering of Applications with Noise), was developed by Campello, Moulavi, Zimek and Sander (2015), and it is a robust hierarchical version of DBSCAN (Density Based Spatial Clustering of Applications with Noise), a density-based clustering algorithm.

McInnes, L. Healy and Astels (2016) compared the performance between many clustering implementations (for example, K-means, Sklearn and FastCluster linkage) where the number of data points and the time taken to complete the clustering were evaluated. This comparison concluded that the HDBSCAN algorithm outperforms most of widely used clustering methods in terms of time. They have demonstrated that the HDBSCAN algorithm detects various density clusters, a task in which the DBSCAN algorithm falls short.

HDBSCAN was built for the real-world scenario of having data with varying density. It is relatively fast and allows to define which clusters are important for users based on size by manipulating the minimum number of neighbors. (m_{pts}). For some clustering algorithms, such as K-means, users must provide the number of clusters *k* to detect. Other density-based algorithms for clustering, such as DBSCAN, do not require the specification of this parameter, while hierarchical clustering avoids the problem completely.

This research assesses the performance and some well-known requirements of density–based algorithm, summarized by Matteucci (2014), such as:

- scalability,
- discovering clusters with arbitrary shape,
- minimal requirements for domain knowledge to determine input parameters,
- high dimensionality; dealing with considerable number of dimensions and substantial number of data items can be problematic because of time complexity,
- the effectiveness of the method depends on the definition of "distance".

This thesis examines the performance of HDBSCAN through simulation studies discussions and applications. The effectiveness of HDBSCAN is compared with other clustering algorithms, namely K-means, DBSCAN and the classifier KNN.

The goal of this work is to assess the effectiveness of HDBSCAN. However, instead of working with n-dimensional vectors, it considers the objects as n × n matrices, using from low-dimensional matrices (2×2) to high-dimensional matrices (200×200). Precisely, clustering algorithm are applied to positive-definite symmetric matrices (covariance matrices) by computing distances between these entities, the metrics on distances are analyzed and compared how they affect the performance of the algorithms.

The interest of working with this type of mathematical structure is to identify objects with the same or at least with the closest covariance matrix, as well as to test the dissimilarity of the covariance matrices of the distinct groups. This comparison is possible by estimating for each object its covariance matrix of size $n \times n$ in the raw data, where n means the number of variables or characteristics that are analyzed of the phenomenon under study, in addition to the object itself.

The importance of working with covariance matrices is discussed by Yger, Lotte and Sugiyama (2015) where they pointed out that covariance matrices are key in the brain-computer interface based on spatial pattern methods. In addition, they indicated that the non-Euclidean structure of the covariance matrices should be considered, taking for example, the Riemannian geometry that is effective handling such data. Moreover, the calculation of the distance between the covariance matrices before applying the clustering algorithm itself, avoids the use of Euclidean distance and its problems with the dimensionality of the data (Aggarwal et al, 2001).

Clustering matrices is used to analyze real world outcomes, such as:

- *Marketing*:

  Identify groups of stocks with similar Buy – Sell behavior in the Stock Market;

  Find groups of customers with similar behavior given a large database of customer data containing their properties and past buying records;

- *Biology*: classification of plants and animals given their features;
- *Crime Analysis*: Identifying areas with similar incidences of crime.
- *Image Detection*: Identify the similarity of objects; from the recognition of patterns in writing, behavior of traffic in a city to the patterns of anomalies in medical scanners.

The main objective on this thesis is utilize the new algorithm HDBSCAN for clustering covariance matrices in the programming language R.

Some secondary objectives of this work are:

1. Determine the most adequate metric used for calculating distances between covariance matrices.
2. Compare the performance of different supervised classifiers for clustering.
3. Compare the performance of different unsupervised classifiers for clustering
4. Evaluate the performance for the proposed algorithm HDBSCAN and compare it with its competitors.
5. Apply the HDBSCAN algorithm to describe and analyze a real data set using the covariance matrices approach.

This document is organized as follow: Chapter 2 features the four clustering algorithms employed in this work; Chapter 3 outlines the methodology applied for the simulation studies; Chapter 4 provides some experimental results. In Chapter 5, a pattern recognition problem is shown to prove the applicability of the HDBSCAN algorithm; finally, the main ideas and conclusions are discussed in Chapter 6.

2. LITERATURE REVIEW

The notion of *density* plays a significant role in statistics and in many data mining tasks. The fundamental idea behind density-based techniques for data analysis is that the dataset of interest represents a sample from an unknown probability density function (PDF), which describes the mechanism or mechanisms responsible for producing the observed data. The construction of an estimate of such a PDF from the observed data is a problem of relevance, for example, for analyzing and understanding the corresponding generating mechanism(s).

The density-based clustering approach is a methodology that is capable of finding arbitrarily shaped clusters, where clusters are defined as dense regions separated by low-density regions. Density-based algorithms need only one scan of the original data set and handle noise in the data. The number of clusters is not required in advanced since density-based clustering algorithms can automatically detect the clusters, along with the natural number of clusters (El-Sonbaty et al., 2004).

This thesis is based on the work of Campello et al (2015), which provides the definitions and structure of the HDBSCAN (hierarchical density-based spatial of clustering and applications with noise), as shown in Figure 1. The authors proposed a framework for hierarchical and nonhierarchical density-based clustering, global/local outlier detection and data visualization.

Besides, this work demonstrates the advancement that the framework represents in the areas of density-based clustering and unsupervised outlier detection through extensive experiments on a variety of synthetic and real-world datasets, where among some of its applications includes the grouping of objects such as covariance matrices by previously calculating their distances.

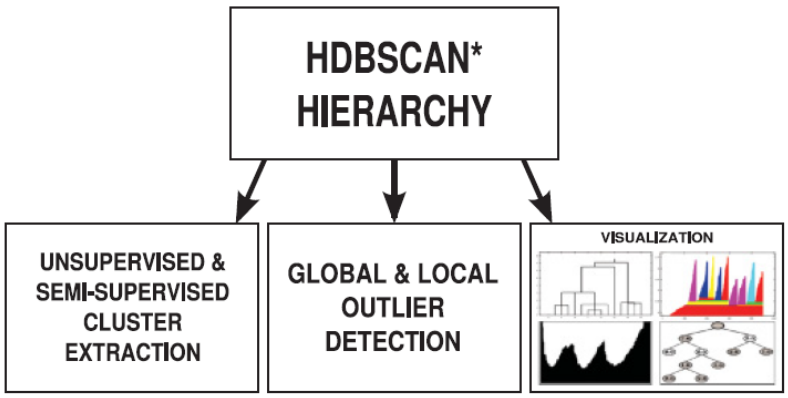


Figure 1. Proposed framework for hierarchical nonhierarchical density-based clustering, global/local outlier (Campello et al 2015)

In this section, a brief review of the literature related to each of the components of the proposed work is provided. First, the definitions of some important concepts related to density-based algorithms are given. Second, a discussion of the existing algorithms is provided to understand the differences with respect to HDBSCAN: these algorithms include: DBSCAN, K-means, HDBSCAN and KNN. Third, the theory of covariance matrices and the most used metrics to calculate distance between covariance matrices is presented. Finally, the definition of the purity measure and how to assess the

performance of clustering algorithms based in internal or external evaluations are discussed.

2.1 DBSCAN Algorithm

Ester et al. (1996) proposed a density-based clustering algorithm called DBSCAN (Density- Based Spatial Clustering of Applications with Noise) to discover arbitrarily shaped clusters. Only one input parameter is required, and the algorithm also supports the user in determining an appropriate value for this input parameter. They defined a cluster as a connected dense component which can grow in any direction that density leads.

The nonparametric method DBSCAN implements an algorithm to cluster based on density, and it can be used to identify groups with different shape in any dataset containing noise and outliers.

Some important characteristics of the DBSCAN algorithm are summarized as follow:

1. Clusters are defined as density – connected sets of points.
2. Density and connectivity are measured by local distribution of nearest neighbor
3. Target low dimensional spatial data

2.1.1 Definitions for DBSCAN

Figure 2 shows an example of two points p and q in a dataset. Definitions for DBSCAN are present below:

Definition 2.1.1. ε-neighborhood of a point p is defined as $N_\varepsilon(p) = \{q \in D \mid d(p,q)\} \leq \varepsilon\}$.

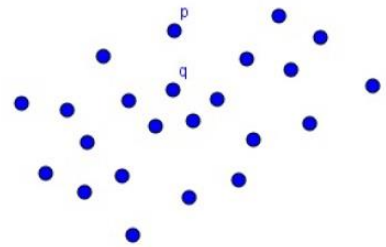
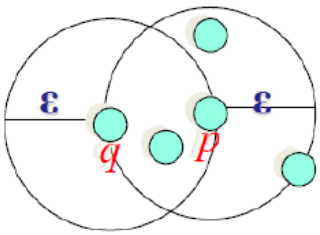


Figure 2. Two points in a dataset

Definition 2.1.2. A point q is called Core point if $|N_\varepsilon(q)| \geq m_{pts}$



MinPts = 4

Figure 3. p is a core point and q is border point for m_{pts} = 4.

Definition 2.1.3. Border point: A point q is called border point if $|N_\varepsilon(p)| < m_{pts}$ but it belongs to the ε-neighborhood for some core point p. (See Figure 3)

Definition 2.1.4. Directly density -reachable: A point p is directly – reachable from a point q if $p \in N_\varepsilon(q)$ and $|N_\varepsilon(q)| \geq m_{pts}$. (See Figure 4)

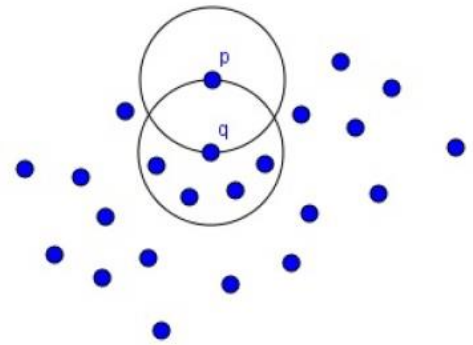


Figure 4. p is directly density reachable from q (with m_{pts}= 6) but q is not directly density – reachable from p.

Definition 2.1.5. Density reachable: A point q is density reachable from a point q if there is a chain of points $p_1, p_2, \ldots, p_n = q, p_n = p$ such that p_{i+1} is directly density – reachable from p_i. (See Figure 5)

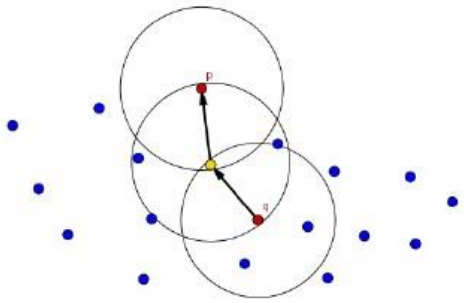


Figure 5. p is density reachable from q (with m_{pts}= 6) but q is not density reachable from p.

Definition 2.1.6. Density Connected: A point p is density – connected to a point q if there is a point o such that both, p and q are density – reachable from o. (See Figure 6)

Definition 2.1.7. Cluster: Let D be a database of points. A cluster C is a non-empty subset of D satisfying the following conditions:

a) $\forall\, p\,, q$: if p $\in C$ and q is density – reachable from q then $q \in C$. (Maximality)

b) $\forall\, p, q \in C$: p is density- connected to q. (Connectivity)

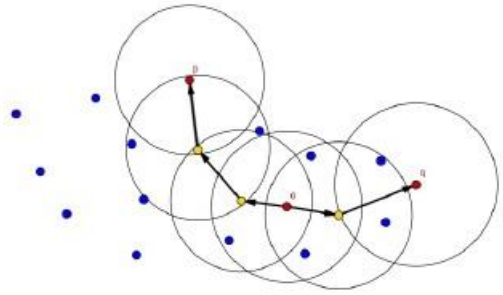


Figure 6. p is connected with q through o

Definition 2.1.8. Noise: Let $C_1, C_2, \ldots, C_k$ be clusters of the database D. Then we define the noise as the set of points in the database D not belonging to any cluster C_i , i.e. noise = $\{p \in D \mid \forall i : p \notin C_i\}$.

The main steps or pseudocode for DBSCAN are shown in the next Algorithm 1.

DBSCAN(X , $\mathcal{E}$, m_{pts})

1. Cluster = 0
2. **For** each unvisited point p in X
2.1 Mark p as visited
2.2 Obtain all neighbors for p
2.3 If size of neighbors $p < m_{pts}$
Mark p as noise
2.4 else
C= nextcluster
Expandcluster(p, neighbors, C, $\mathcal{E}$, m_{pts})

Expandcluster(p, neighbors of p, C, $\mathcal{E}$, m_{pts})
1. Add p to cluster C
2. **For** each q in neighbors
2.1 If q is not visited
Mark q as visited
Obtain neighbors for q
2.1.1 **If** size of neighbors of $q \geq m_{pts}$
Neighbors of p = neighbors of p joined with neighbors of q
2.2 If q is not member of any cluster
Add q to cluster c

Algorithm 1: DBSCAN main steps

2.1.2 Advantages and disadvantages of DBSCAN

Some advantages and disadvantages discussed by Dang (2015) for this algorithm are summarized as follow:

Advantages

1. It detects outliers or noise in the data.
2. It requires only two parameters and it is robust to the input order of the data in the database (ε and m_{pts}).
3. It does not require specifying the number of clusters in advance.
4. DBSCAN can find groups with arbitrary shapes.

Disadvantages

1. It depends on the function of the density measure used.
2. DBSCAN cannot cluster datasets very well with big differences in densities in the cloud of points, because the combinations of ε, m_{pts} cannot be chosen adequately for all groups (See Figure 12).
3. When the data dimension increases, it becomes more difficult to choose the optimal ε (curse of dimensionality).

2.2 HDBSCAN Algorithm

Hard clustering algorithms are subdivided into hierarchical and partitional algorithms. A partitional algorithm divides a data set into a single partition, whereas a hierarchical algorithm divides a data set into a sequence of nested partitions (Gan, Guojun. Ma,

Chaoqun. Wu, Jianhong. , 2007).Similarly, hierarchical algorithms are subdivided into agglomerative hierarchical and divisive hierarchical algorithms. Agglomerative hierarchical clustering starts with every single object in a single cluster. Then it continues merging the closest pair of clusters according to some similarity criteria until all the data are in a single cluster.

In the same way, Gan et al (2007) summarizes some disadvantages for agglomerative hierarchical clustering, such as (a) data points that have been incorrectly grouped at an early stage cannot be reallocated and (b) different similarity measures for measuring the similarity between clusters may lead to different results. If we treat agglomerative hierarchical clustering as a bottom-up clustering method, then divisive hierarchical clustering can be viewed as a top-down clustering method. Divisive hierarchical clustering starts with all objects in one cluster and repeats splitting large clusters into smaller pieces.

HDBSCAN stands for Hierarchical Density-Based Spatial Clustering of Application with Noise. It generalizes and improves existing density-based clustering techniques with respect to several aspects (Campello et al., 2015):

- It provides a complete hierarchy that comprises all possible density-based clusters following the nonparametric model adopted. The resulting hierarchy can be easily processed.

- It post-processes data: a normalized score of outlierness can be assigned to each data object.
- A (nonhierarchical) clustering solution composed of clusters extracted from local cuts through the cluster tree can be obtained.
- HDBSCAN is a divisive hierarchical clustering algorithm.
- HDBSCAN is an unsupervised clustering, since it does not need training step.
- It does not label any point as border points as DBSCAN does.

2.2.1 Definitions from DBSCAN

In the following, some definitions required to define HDBSCAN are presented, the density-based clustering algorithm DBSCAN* is defined, which differs from DBSCAN (see definitions 2.1.1 to 2.1.8) where the clusters are defined based on the core objects alone.

Definition 2.4.1. (Core and Noise objects): An object x_p is called a core object with respect to ϵ and m_{pts} if its ϵ- neighborhood contains at least m_{pts} many objects, that is, if $\left| N_\epsilon(x_p) \right| \geq m_{pts}$, where $N_\epsilon(x_p) = \{x \in X | d(x, x_p) \leq \epsilon\}$ and $|.|$ denotes cardinality. An object is called noise if it is not a core object.

Definition 2.4.2. (ϵ - reachable): Two core objects x_p and x_q are ϵ- reachable with respect to m_{pts} if $x_p \in N_\epsilon(x_q)$ and $x_q \in N_\epsilon(x_p)$.

Definition 2.4.3. (Density Connected): Two core objects x_p and x_q are density – connected with respect to m_{pts} if they are directly or transitively ϵ-reachable.

Definition 2.4.4. (Cluster): A cluster w.r.t. ϵ and m_{pts} is a non-empty maximal subset such that every pair of objects in C is density-connected.

Based on these definitions, we can devise an algorithm DBSCAN* (similar to DBSCAN) that conceptually finds clusters as the connected components of a graph in which the objects of X are vertices and every pair of vertices is adjacent if and only if the corresponding objects are ϵ-reachable with respect to user defined parameters ϵ and m_{pts}. Noncore objects are labeled as noise. Border objects could also be included in a simple linear time postprocessing step (tracking and assigning each border object to, e.g., its closest core).

2.2.2 Conceptual HDBSCAN

This method requires a simple input parameter m_{pts}. Campello et al. (2015) provide the definitions for the hierarchical method, HDBSCAN in addition to definitions for DBSCAN*:

Definition 2.4.5 (core distance): The core distance of an object $x_p \in X$ with respect to m_{pts} , $d_{core}(x_p)$, is the distance from x_p to its m_{pts} – nearest neighbor (including x_p). (See Figure 8).

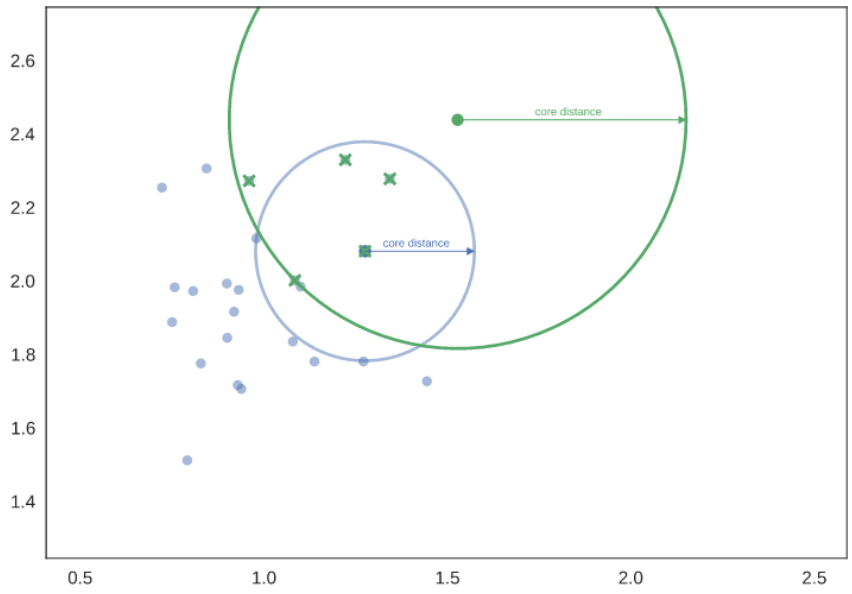


Figure 7. Core distances for two points (McInnes, L.Healy, J.Astels, S.,2016)

Definition 2.4.6 (Mutual reachability distance). The mutual reachability distance between two objects x_p and x_q in X with respect to m_{pts} is defined as $d_{mreach}(x_p, x_q) =$ max{ $d_{core}(x_p), d_{core}(x_q), d(x_p, x_q)$}.

Definition 2.4.7 (Mutual reachability graph). The mutual reachability graph is a complete graph, $G_{m_{pts}}$, in which the objects of X are vertices and the weight of each edge is the

mutual reachability distance (with respect to m_{pts}) between the respective pair of objects. (See Figure 9)

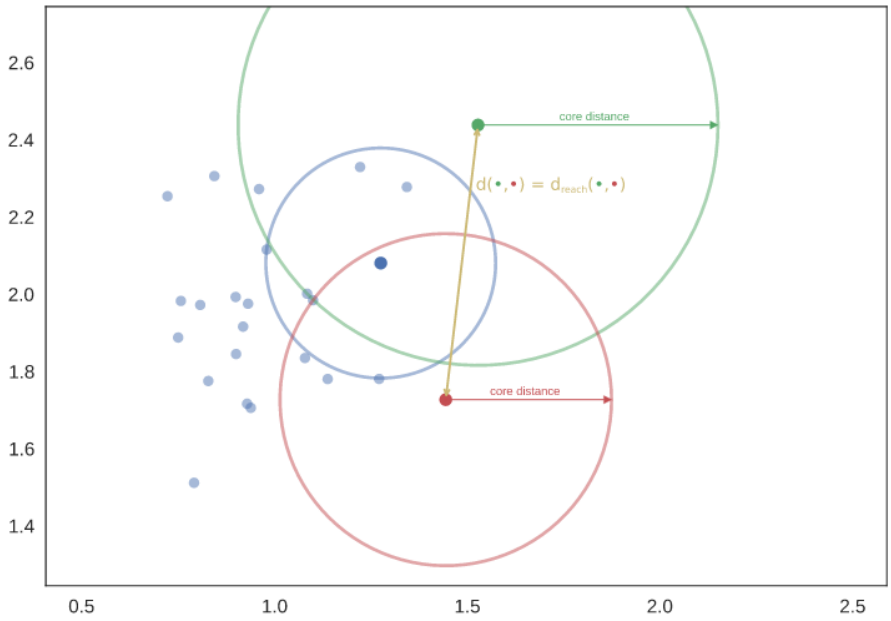


Figure 8. Example for mutual reachability distance between red and green points (McInnes et al, 2016)

Let $G_{m_{pts},\epsilon} \subseteq G_{m_{pts}}$ be the graph be the graph obtained by removing all edges from $G_{m_{pts}}$ having weights greater than some value of ϵ. From Definitions 2.4.4 and 2.4.7, it is straightforward to infer that clusters according to DBSCAN* with respect to m_{pts} and ϵ are then the connected components of core objects in $G_{m_{pts},\epsilon}$; and the remaining objects are noise. Consequently, all DBSCAN* clustering's for any $\epsilon \in [0, \infty]$ can be produced in a nested, hierarchical way by removing edges in decreasing order of weight from $G_{m_{pts}}$.

Proposition 2.4.1: Let X be a set of n objects described in a metric space by $n \times n$ pairwise distances. The clustering of this data obtained by DBSCAN* with respect to m_{pts} and some value ϵ is identical to the one obtained by first running Single - Linkage over the transformed space of mutual reachability distances (with respect to m_{pts}), then, cutting the resulting dendrogram at level ϵ of its scale, and treating all resulting singletons with $d_{core}(x_p) > \epsilon$ as a single class representing "Noise".

2.2.3 Algorithm of HDBSCAN

The fastest way to compute a Single-Linkage hierarchy is possibly by using a divise algorithm based on the Minimum Spanning Tree (MST), as shown in Figure 9. McInnes et al (2016) pointed out that it is necessary to consider the data as weighted graph with the data points as vertices and an edge between any points with weight equal to the mutual reachability distance of those points, then removing edges from a MST in decreasing order of weights starting from a threshold value and steadily lowering. Eventually, it will result a hierarchy of connected components at varying threshold levels, in practice, this is very expensive because there are n^2 edges.

The correct step to follow is to find a minimal set of edges such that dropping any edge from the set causes a disconnection of components. These disconnections form islands on the data and they may have different densities. To achieve this, the minimum spanning tree can be built very efficiently via Prim's algorithm, as McInnes et al (2016) argued.

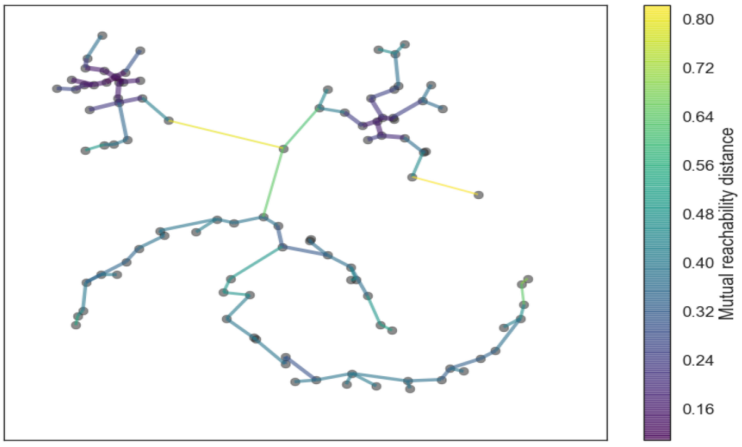


Figure 9. Example of Minimun Spanning Tree (MST) (McInnes et al, 2016).

HDBSCAN (m_{pts}; X)

1. Compute the core distance w.rt. m_{pts} for all data objects in X.

2. Compute an MST of $G_{m_{pts}}$, the Mutual Reachability Graph.

3. Extend the MST to obtain MS_{Text}, by adding for each vertex a "self edge" with the core distance of the corresponding object as weight.

4. Extract the HDBSCAN hierarchy as a dendrogram from MS_{Text}.

4.1 For the root of the tree assign all objects the same label (single "cluster").

4.2 Iteratively remove all edges from MS_{Text} in decreasing order of weights (in case of ties, edges must be removed simultaneously):

4.2.1 Before each removal, set the dendrogram scale value of the current hierarchical level as the weight of the edge(s) to be removed.

4.2.2 After each removal, assign labels to the connected component(s) that contain(s) the end vertex(-ices) of the removed edge(s), to obtain the next hierarchical level: assign a new cluster label to a component if it still has at least one edge, else assign it a null label ("noise").

Algorithm 2: HDBSCAN main steps (Campello et al. , 2015)

Given the minimal spanning tree, the next step is to convert this tree into the hierarchy of connected components. This is most easily done in the reverse order: sort the edges of the tree by distance (in increasing order) and then iterate through, creating a new merged cluster for each edge. The only difficult part here is to identify the two clusters each edge will join together, but this is easy enough via a union-find data structure. The result is a dendrogram as shown in Figure 10:

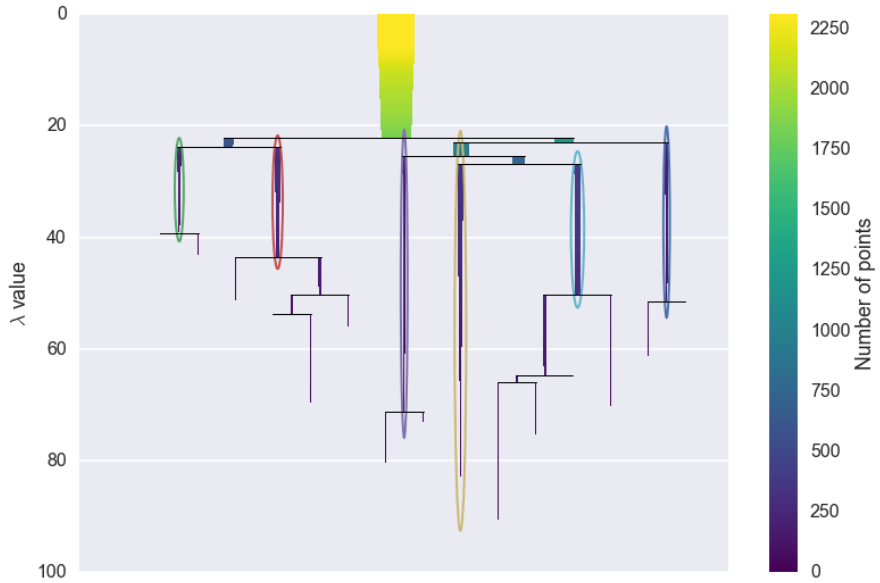


Figure 10 Example of dendrogram for a HDBSCAN application (McInnes, L.Healy, J.Astels, S., 2016)

2.3 K – nearest neighbor

K- nearest neighbors algorithm (KNN) is a non-parametric supervised algorithm method used for classification and regression. The input parameters consist of the k-closest elements in data training set (feature space). This algorithm is said to be a simple and lazy learning algorithm, because of its minimal or nonexistent training phase. It means, this technique does not use training points to perform any generalization. And it is non-parametric because there are no any data distribution assumptions.

Some k-NN characteristics are summarized below:

- The output is a class membership. The object element is classified by majority vote of its neighbors, with this element assigned to the most common class.
- It is a supervised classifier. The initial set or training set is a paired set consisting of input object (feature vector) and class labels values.
- It is affected by the curse of dimensionality, especially when it is used the Euclidean distance. It is high recommended to use dimension reduction techniques prior to applying k-NN algorithm.
- In k-NN regression, the output is the value for the element and it represents the average of the k-nearest neighbors values.
- The most common distances used are Euclidean's distance, Manhattan's distance and Minkowski's distance.

- It requires a lot of CPU and memory due to all data points are involved in the decision. It is the better choice for applications where predictions are not requested frequently but where accuracy is important (IBM 2018).

Some common applications for k-NN algorithm are:

- Identifying persons by comparing faces.
- Recommending related items to users in products consume.
- Handwritting recognition.
- Medical data mining. Find similarity between patients and reduce error of diagnosis. (Khamis, Kipruto. Kimani (2014))

2.4 K-means Algorithm

k - means is an unsupervised center-based clustering algorithm with a partitional approach where centroids become a component - wise means of data points in clusters. This algorithm aims to divide *n* observations into *k* clusters where each observation belongs to one cluster with the nearest mean (centroid). It starts randomly selecting the *k* centroids in data points and then refine the positions of centers until reaching a local optimum or satisfying initial conditions to stop.

2.4.1 Algorithm description

- k – means is a partitional clustering approach.
- The number of cluster *k* is an input parameter, it must be specified.
- A centroid (center point) defines a cluster, in other words, each cluster is associated with a centroid.
- Each point is in one exactly subset and is assigned to the cluster with the closest centroid.

The algorithm 3 presents the main steps for k-means,

K-means (number of clusters k)

1. Select K points as initial center points.
2. **Repeat**
3. Assign each observation to the clusters with the closest centroid.
4. Recompute the new centroids to be the intrinsic means of the observations in the new clusters.
5. **Until** Assignments cease to change.

Algorithm 3. K – means

2.4.2 Limitations of k-means

Some limitations of k – means are summarized by Zhang (2012) as follows:

- k – means algorithm exhibits problems when clusters are of differing on sizes, densities or non-globular shapes.

- k – means algorithm has problems when data contains outliers
- The algorithm must be run several times with different values of *k*.

Table 1 presents a summary of the most important characteristics for all clustering algorithm considered in this thesis:

Algorithm	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	Number of clusters	Very large number of samples, medium number of clusters	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
DBSCAN	Neighborhood size and eps	Very large number of samples, medium number of clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
KNN	Neighborhood size	Small number of training data	Supervised scenarios	Distances between points
HDBSCAN	Neighborhood size	Very large number of samples, medium number of clusters	Non-flat geometry, uneven cluster sizes	Mutual reachability distance

Table 1 Summary of Clustering Algorithms (Pedregosa et al, 2016)

Figure 11 shows the clustering differences for some algorithms using synthetic data with 6 groups. It can be seen that even providing k = 6 (see 6 different colors in the image), k-means cannot detect the clusters appropriately and does not discover the noise or outliers present in the data. DBSCAN fails to detect the right number of clusters and the varying density shapes. HDBSCAN outperforms these algorithms, because it is able to detect the right number of varying density shapes as well as the noise existing in the data.

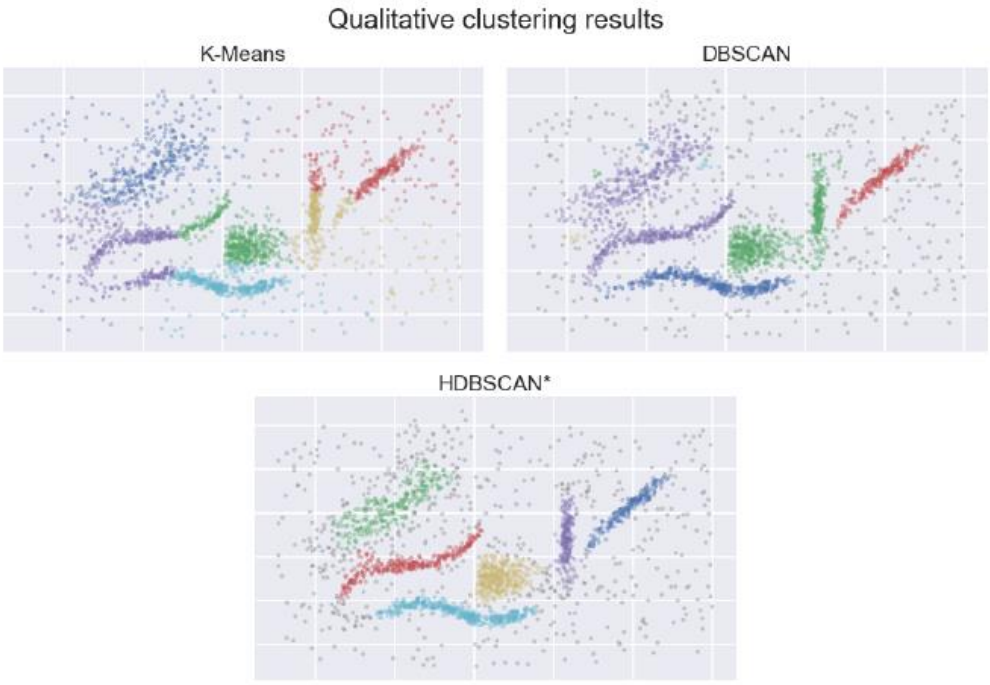


Figure 11 A qualitative comparison of clustering algorithm (Mc Innes et al. 2017).

2.5 Covariance Matrices

A covariance matrix (also called dispersion matrix or variance-covariance matrix) refers to the symmetric square matrix that contains the variances and covariances associated with several variables. Many statistical applications calculate the variance-covariance matrix for the estimators of parameters in a statistical model. Covariance is a measure of how much two random variables move together in the same direction; hence the covariance describe the unidirectional and bidirectional influences of several variables on each other.

A variance-covariance matrix is a square matrix that contains the variances and covariances associated with several variables, where the element in the *i, j* position is the covariance

between the i^{th} and j^{th} elements of a random vector contain the covariances between, having this for all possible pairs of variables.

2.5.1 Definitions

X and **Y** denote to random vectors, and X_i and Y_i denote to random variables. If the entries in the column vector

$$\boldsymbol{X} = \begin{bmatrix} X_1 \\ . \\ . \\ . \\ X_n \end{bmatrix}$$

are random variables, each with finite variance, then the covariance matrix Σ is the matrix whose *(i, j)* entry is the covariance

$$\Sigma_{i,j} = cov(X_i, X_j) = E\left[(X_i - \mu_i)(X_j - \mu_j)\right] = E[X_i X_j] - \mu_i \mu_j$$

where $\mu_i = E(X_i) < \infty$ is the expected value of the i^{th} entry in the vector **X**. In other words,

$$\Sigma = \begin{bmatrix} E[(X_1 - \mu_1)(X_1 - \mu_1)] & \cdots & E[(X_1 - \mu_1)(X_n - \mu_n)] \\ \vdots & \ddots & \vdots \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & \cdots & E[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}$$

2.5.2 Properties of covariance matrices

Following Taboga's (2010) definitions, for $\Sigma = E\ [\ (\ \boldsymbol{X} - E[\boldsymbol{X}])(\boldsymbol{X}\ -\ \ E[\boldsymbol{X}])^T\ ]$ and $\mu = E(\boldsymbol{X})$, the following basic properties apply to a covariance matrix Σ:

1. $\Sigma = E(\boldsymbol{X}\boldsymbol{X}^T) - \boldsymbol{\mu}\boldsymbol{\mu}^T$
2. Σ is a positive - semidefinite and symmetric. This is, $X^*\Sigma\, X > 0$, where $X \in \boldsymbol{C}^n$ and X^* is the conjugate transpose of X.
3. If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent, then $cov(\boldsymbol{X}, \boldsymbol{Y}) = \mathbf{0}$.

2.6 Distance between matrices

Dryden, Koloydenko and Zhou (2009) summarize the most commonly matrix distances used as follow:

"If a sample of covariance matrices is available, we wish to estimate an average covariance matrix, or we may wish to interpolate in space between two or more estimated covariance matrices, or we may wish to carry out tests for equality of mean covariance matrices in diverse groups."

The usual approach to estimating mean covariance matrices in statistics is to assume a scaled Wishart distribution for the data, and then the maximum likelihood estimator (MLE) of the population covariance matrix is the arithmetic mean of the sample covariance matrices. The estimator can be formulated as a least squares estimator using Euclidean distance. However, since the space of positive semi-definite symmetric matrices is a non-Euclidean space, it is more natural to use alternative distances.

2.6.1 Euclidean distance

Consider n sample covariance matrices (symmetric and positive semidefinite $k \times k$ matrices) $S_1, \ldots, S_n$. We assume that the S_i are independent and identically distributed (i.i.d.) from a distribution with mean covariance matrix Σ.

The Euclidean distance between two matrices is given by:

$d_E(S_1, S_2) = \|S_1 - S_2\| = \sqrt{trace\{(S_1 - S_2)^T(S_1 - S_2)\}}$, where $\|\boldsymbol{X}\| = \sqrt{trace(\boldsymbol{X}_T\boldsymbol{X})}$ is the Euclidean norm (also known as the Frobenius norm).

A drawback of the Euclidean distance is when extrapolating beyond the data, nonpositive semi-definite estimates can be obtained.

2.6.2 Log-Euclidean distance

Let $S = U\,\Lambda \mathrm{U}^{\mathrm{T}}$ be the usual spectral decomposition, with $U \in O(k)$ and $\log\,\Lambda$ be a diagonal matrix with logarithm of the diagonal elements of Λ on the diagonal. The logarithm of S is given by $\log S = U\,(\log \Lambda)\,\mathrm{U}^{\mathrm{T}}$ and likewise the exponential of the matrix S is $\exp S = U(\exp \Lambda)\,U^T$.

$$d_E(S_1, S_2) = \|\log(S_1) - \log(S_2)\|$$

Using this metric avoids extrapolation problems into matrices with negative eigenvalues, but it cannot deal with positive semi-definite matrices of deficient rank, this is, when one column is a linear combination of others.

2.6.3 Non- Euclidean size-and-shape metric

The non-Euclidean size-and-shape metric between two $k \times k$ covariance matrices S_1 and S_2 is defined as $d_S(S_1, S_2) = \inf_{R \in O(k)} \|L_1 - L_2 R\|$, where L_i is the decomposition of S_i such that $S_i = L_i L_i^T$ and $L_i = chol(S_i)$ is lower triangular matrix in the Cholesky's decomposition.

For example, the Cholesky decomposition could be used, since$S = LL^T$, then the decomposition is represented by an equivalence class$\{LR = R \in O(k)\}$. For practical computation it is often needed to choose representative from this class, called an icon, and in our computations, we shall choose the Cholesky decomposition. This metric has been used previously in the analysis of point set configurations where invariance under translation, rotation and reflection is required.

2.6.4 Affine - Invariant Riemannian distance

This metric was written by Forstner and Moonen (2003). They presented a metric for positive definite covariance matrices. It is a natural expression involving traces and joint eigenvalues of the matrices. It is shown to be the distance coming from a canonical invariant Riemannian metric on the space $Sym^+(n, \mathbb{R})$ of real symmetric positive definite matrices. In contrast to known measures (Grafarend, 1972), the metric is invariant under affine transformations and inversion. It can be used for evaluating covariance matrices or for optimization of measurement designs.

Forstner and Moonen (2003) proposed the next definition for the new metric:

$$d(\boldsymbol{A},\boldsymbol{B}) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(\boldsymbol{A},\boldsymbol{B})}$$

between symmetric positive definite matrices, A and B, with the eigenvalues $\lambda_i(\boldsymbol{A},\boldsymbol{B})$ from $|\lambda\boldsymbol{A} - \boldsymbol{B}|$.

Some properties for this metric are summarized next:

Let for all $M(n,\boldsymbol{R}) \coloneqq \{\boldsymbol{A} = (a_{ij}) | 1 \leq i, 1 \leq n, a_{ij} \in \boldsymbol{R}\}$ be the space of real $n \times n$ matrices, and let $S^+ \coloneqq Sym^+(n,R) \coloneqq \{\boldsymbol{A} \in M(n,R) | \boldsymbol{A} = \boldsymbol{A}^T, \boldsymbol{A} > 0\}$. For all $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C} \in Sym^+(n,\boldsymbol{R})$.

1. d is invariant with respect to affine transformations of the coordinate system.

$$d(\boldsymbol{A},\boldsymbol{B}) = d(\boldsymbol{XAX^T},\boldsymbol{XBX^T})$$

   $\boldsymbol{X} \in GL(n,\boldsymbol{R})$, where GL means General Linear Group.

2. d is invariant with respect to an inversion of the matrices.

$$d(\boldsymbol{A},\boldsymbol{B}) = d(\boldsymbol{A^{-1}},\boldsymbol{B^{-1}})$$

3. It is claimed that d is a metric.
   a) Positivity: $d(A,\boldsymbol{B}) \geq 0$, and $d(\boldsymbol{A},\boldsymbol{B}) = 0$ only if $\boldsymbol{A} = \boldsymbol{B}$.
   b) Symmetry: $d(\boldsymbol{A},\boldsymbol{B}) = d(\boldsymbol{B},\boldsymbol{A})$
   c) Triangle inequality: $d(\boldsymbol{A},\boldsymbol{B}) + d(\boldsymbol{A},\boldsymbol{C}) \geq d(\boldsymbol{B},\boldsymbol{C})$

A summary of the most used metrics to calculate covariance matrices distances is given in Table 2:

Name	Notation	Form
Euclidean	$d_E(S_1, S_2)$	$\|S_1 - S_2\|$
Log – Euclidean (LERM)	$d_L(S_1, S_2)$	$\|\log(S_1) - \log(S_2)\|$
Cholesky	$d_C(S_1, S_2)$	$\|chol(S_1) - chol(S_2)\|$
Riemannian (AIRM)	$d_R(S_1, S_2)$	$\left\| \log(S_1^{-\frac{1}{2}} S_2 S_1^{-\frac{1}{2}}) \right\|$

Table 2 Form of Metrics on Covariance matrices

Table 3 provides the common properties discussed for SPD. AIRM is the metric with more invariance properties, while Cholesky's metric does not meet any of these properties.

Distance	Affine invariance	Scale invariance	Rotation invariance	Inversion invariance
Euclidean	No	No	Yes	No
Log – Euclidean (LERM)	No	Yes	Yes	Yes
Cholesky	No	No	No	No
Riemannian (AIRM)	Yes	Yes	Yes	Yes

Table 3 Properties of Metrics on covariance

2.7 Purity Measure

There exist different methods to measure the effectiveness of a clustering algorithm. However, before applying some of these performance measures, it is necessary to indicate what criteria evaluation will be used. There are two kinds of clustering evaluation:

- Internal evaluation: based in data that was clustered itself, this is, considering the information intrinsic to the data alone.
- External evaluation: consisting in using data not used for clustering (e.g. class labels), this is, based on previous knowledge about the data.

When simulated data is used knowing a priori the class labels for each dataset then the use of an external criteria index to evaluate performance or effectiveness is the appropriate option.

Purity is a simple and transparent external evaluation where each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N (Manning C., 2009). In addition, Purity is a measure used for many authors to assess and validate the performance of clustering algorithms based in existing known of set of ground truth or class labels (Satya et al, 2011).

The formula to purity is defined as

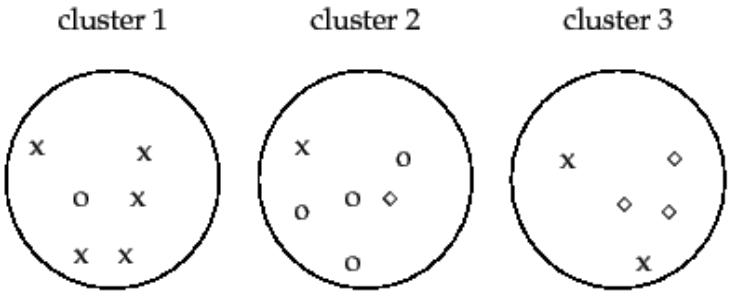
$$Purity(\Omega, C) = \frac{1}{N} \sum_k \max_j \left|\omega_k \cap c_j\right|,$$

where $\Omega = \{\omega_1, \omega_2, \dots, \omega_k\}$ is the set of clusters and $C = \{c_1, c_2, \dots, c_j\}$ is the set of classes.

Some characteristics of the purity measure are:

- When the clustering is perfect (perfect matching between comparing pairs) then purity is equal to 1.
- When no matching between compared pairs is obtained (bad clustering) then purity is equal to 0.
- It is not recommended for unbalanced (nonequal number of elements per group) data.

Figure 12 provides an example of the calculation of Purity.



Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ◇, 3 (cluster 3). Purity is $(1/17) \times (5 + 4 + 3) \approx 0.71$.

Figure 12 Example of purity measure. (Manning, 2009)

3 METHODOLOGY

This chapter discusses with the methodology used to measure the performance of the proposed algorithm HDBSCAN in different situations and to compare it with other known clustering methods like K-means and DBSCAN and the supervised classifier KNN. In addition, the input parameters for each algorithm are varied to verify their consistency.

To assess the high performance of HDBSCAN, different sets of covariance matrices are used. The size of the matrices varies from low dimensional to high dimensional covariance matrices. After generating the set of matrices, the distance between each pair of matrices is calculated choosing the desired metric (AIRM, LERM, Cholesky or Euclidean), as presented in section 2.6. Synthetic data is generated to obtain diverse situations varying from 2, 4 and 8 groups. The different datasets were simulated to mimic different scenarios based on the difficulty to cluster the matrices. The characteristics of every dataset is described in Tables 3,4 and 5.

3.1 Data Generation

The statistical programming language R is used to generate artificial data to assess and implement the clustering algorithms. Using package ***CovTools*** (Lee et al, 2018) to generate covariance matrices where ***Samplecovs*** function generates a 3d array of stacked sample covariances where in 3rd dimension, the first half are sample covariances of samples generated independently from normal distribution with identity covariance, where the latter

half consists of samples covariances from random population covariance. The same package is used to call the function **covDist**, which is used to calculate the distance between covariance metrics in an array by selecting the metric AIRM, Euclidean, LERM or Cholesky.

The Wishart distribution $W(df, \Sigma)$ and the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ are used to generate $n \times n$ dimensional covariance matrices, where their parameters are modified or altered to generate the datasets of covariance matrices to implement in the simulation studies. For the Wishart distribution the covariance matrix Σ is called the Scale matrix where the degree of freedom (df) is set as $df = n + 1$, where n is the matrix size; $n * \Sigma$ stands for the expected mean value in this probability distribution.

The ***rWishart*** package (Barnard, 2017) is used to generate a 3d array of covariance matrices from ***Wishart*** distributions. Here, it is necessary to provide the degrees of freedom, number of covariance matrices and the scale covariance matrix.

Some covariance structures used in the simulation studies are summarized next:

- First order autoregressive covariance matrix:

$$\sigma^2 * \begin{pmatrix} 1 & \rho & \rho^2 & \dots & \rho^n \\ \rho & 1 & \rho & \dots & \rho^{n-1} \\ \rho^2 & \rho & 1 & \dots & \rho^{n-2} \\ \dots & \dots & \dots & \dots & \dots \\ \rho^n & \rho^{n-1} & \rho^{n-2} & \dots & 1 \end{pmatrix}$$

- Banded main diagonal covariance matrix:

$$\begin{pmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \sigma_n^2 \end{pmatrix}$$

- The method, denoted by "eigen", first randomly generates eigenvalues $(\lambda_1, \dots, \lambda_p)$ for the covariance matrix (Σ), then uses columns of a randomly generated orthogonal matrix $(Q = (\alpha_1, \dots, \alpha_p))$ as eigenvectors. The covariance matrix (Σ) is then contructed as $Q * diag(\lambda_1, \dots, \lambda_2) * Q^T$ (Qiu, 2015).

- Randomly generated covariance matrices from normal population are obtained through generating random arrays of n-dimensional vectors with $\boldsymbol{\mu} = \mathbf{0_n}$ and $\sigma^2 = 1$, binding them with other random n-dimensional vectors with $\boldsymbol{\mu} = \mathbf{5_n}$ and $\sigma^2 = 1$. Finally, the covariance matrix of the array is obtained.

The ***ClusterGeneration*** package (Qiu et al, 2015) is used to generate a covariance matrix by using ***genPositiveDefMat*** with the option "eigen". The package ***mvtnorm*** (Genz et al, 2018) is also used to generate synthetic covariance matrices coming from multivariate normal distribution data, where using the function ***rmvnorm*** it is possible to modify the mean and standard deviation for the n-dimensional vectors employed to generate the covariance matrices.

3.2 Clustering Algorithms

HDBSCAN and DBSCAN algorithm are called using the package ***dbscan*** (Hahsler, Piekenbrock, Arya and Mount, 2017), whereas k-means algorithm is called with the package ***stats***. The package ***class*** (Ripley et al, 2015) is used for calling KNN algorithm.

3.3 Simulation scenarios

For the first scenario, three different datasets are generated each one varying its parameters and consisting of two separated groups as shown in table 4.

Set Number	Group	Mean $\boldsymbol{\mu}$	Covariance Σ	Probability Distribution
1	1	$\mathbf{0}_n$	Identity	Multivariate Normal
	2	$\mathbf{0}_n$	Randomly generated from normal population	Multivariate Normal
2	1	$\mathbf{1}_n$	Identity	Multivariate Normal
	2	$\mathbf{1}_n$	First order Autoregressive ($\rho = 0.2, \sigma^2 = 2$)	Multivariate Normal
3	1	$n * \Sigma$	Identity	Wishart
	2	$n * \Sigma$	First order Autoregressive ($\rho = 0.6, \sigma^2 = 2$)	Wishart

Table 4 Characteristics of generated data with two groups

Identically, the parameter for scenario four is shown in table 5.

Set Number	Group	Mean μ	Covariance Σ	Probability Distribution
1	1	$n * \Sigma$	Generated using eigenvalues ($\mu = \mathbf{1_n}$)	Singular Wishart
	2	$n * \Sigma$	Generated from normal population ($\mu = \mathbf{0_n}$)	Wishart
	3	$n * \Sigma$	Randomly generated from normal population	Wishart
	4	$n * \Sigma$	Randomly generated from normal population	Singular Wishart
2	1	$\mathbf{1_n}$	First order Autoregressive ($\sigma^2 = 2, \rho = 0.6$)	Normal
	2	$\mathbf{4_n}$	First order Autoregressive ($\sigma^2 = 2, \rho = 0.2$)	Normal
	3	$\mathbf{1_n}$	Identity	Normal
	4	$\mathbf{1_n}$	Banded Main Diagonal (Generated from random normal population with $\mu = 3$ and $\sigma^2 = 0$)	Normal

Table 5 Characteristics for generated data with 4 groups

Table 6 shows the parameters for scenario 3.

Set Number	Group	Mean μ	Covariance Σ	Probability Distribution
1	1	$n * \Sigma$	Generated using eigenvalues ($\mu = \mathbf{1_n}$)	Singular Wishart
	2	$n * \Sigma$	Banded Main Diagonal (Generated from random normal population with $\mu = 4$ and $\sigma^2 = 0$)	Wishart
	3	$n * \Sigma$	Identity ($\boldsymbol{I_n}$)	Wishart
	4	$n * \Sigma$	Randomly generated from normal population	Singular Wishart
	5	$\mathbf{1_n}$	First order Autoregressive ($\sigma^2 = 2, \rho = 0.6$)	Normal
	6	$\mathbf{8_n}$	First order Autoregressive ($\sigma^2 = 2, \rho = 0.2$)	Normal
	7	$\mathbf{1_n}$	Identity ($\boldsymbol{I_n}$)	Normal
	8	$\mathbf{1_n}$	Banded Main Diagonal (Generated from random normal population with $\mu = 4$ and $\sigma^2 = 0$)	Normal

Table 6 Characteristics for generated data with eight groups

To avoid drawbacks of the purity measure, this study will consist of generated balanced synthetic data to evaluate the performance of the clustering algorithms.

A summarized version of the procedure used to simulate data, implement the clustering algorithms and measure their performance is given next.

Steps for generating synthetic data

1. Set the size of the covariance matrix to generate ($2 \times 2, 5 \times 5, 10 \times 10, 50 \times 50, 200 \times 200$)
2. Select the true number of cluster k in the set of covariance matrices $(2, 4, 8)$
3. Choose the dataset with N-covariance matrices samples (N/k per group) by the difficulty of the separation between each group in the set (1, 2, and 3)

   3.1 Select the metric on distance to use (AIRM, LERM, Cholesky, Euclidean)

   3.2 Pick the clustering algorithm to apply (k-means, HDBSCAN, DBSCAN, KNN)

   3.3 Calculate the purity measure to the resulting clustering comparing with the true grouping of the data.
4. Repeat this simulation m times and calculate the average $\bar{p}$ of the purity measures and the sampling error $\sigma_p = \sqrt{\bar{p} * \frac{1-\bar{p}}{N}}$ of the m purity measures.
5. Go back to step 1 and repeat the steps 3-4 modifying some characteristics of the covariance matrices to generate.

4 SIMULATION STUDIES

This simulation studies are limited by the computational resources. Therefore, it is necessary to evaluate the execution time for calculating the distance between matrices, and the clustering procedure. Because of this preliminary analysis, a relatively small number of matrices per group and number of repetitions is considered in each scenario of the simulation studies.

An array of N covariance matrices (20 matrices by group in every set) is generated to calculate the purity measure for each algorithm. Then using a sample size of 50 repetitions per set, the mean and standard error of proportion $\sigma_{\hat{p}}$ for all purity results are calculated and summarized in Tables 13-15, 17-19. It is worth mentioning that KNN was trained before to be applied and then the result for testing are shown in the comparative tables for the purity measure. DBSCAN was chosen with the same number of neighbors (m_{pts}) than HDBSCAN but a $\epsilon -$ minimum neighbors distance value only was chosen to show how selecting this parameter affects the performance of the algorithm.

Sections 4.1 to 4.5 provide the results for execution time and accuracy with purity measure for four different clustering algorithms varying the size of the covariance matrices as well as the predefined number of clusters.

4.1 Comparison of outputs for metrics

Consider an example of an array of 20 covariance matrices with size 2×2 displayed in Table 7. The first element belongs to Group 1 in set 1 with two clusters.

No.	Group	AIRM	Euclidean	LERM	Cholesky
2	1	1.53	0.51	1.24	0.51
3	1	3.32	0.98	3.16	0.90
4	1	3.01	1.37	2.97	1.19
5	1	2.68	1.55	2.64	1.41
6	1	0.62	0.14	0.59	0.20
7	1	3.14	2.10	3.12	1.89
8	1	2.63	1.15	2.54	0.99
9	1	2.06	1.15	1.72	1.10
10	1	2.47	1.42	2.45	1.31
11	2	5.39	32.32	5.24	32.21
12	2	5.15	30.34	4.86	30.24
13	2	5.49	34.88	5.31	34.75
14	2	4.97	35.06	4.70	35.01
15	2	3.69	7.56	3.50	7.47
16	2	2.30	0.86	2.29	0.66
17	2	4.61	29.24	4.31	29.20
18	2	3.99	10.58	3.85	10.51
19	2	5.63	17.63	4.92	17.57
20	2	4.38	15.75	4.24	15.69

Table 7 Comparative of distances by metric for first element in 2×2 matrices

If the distances between the first element and the other elements are ordered, it is possible to find some errors in all metrics, that is, some distances between element 1 from group 1 to other element in group 2 could be less than the other element in the same group 1. For example, consider the distance between element 1 with the element 7 (3.14 with AIRM) and with the element 16 (2.30 with AIRM) respectively, as shown in Table 7. However, if

the size of the matrices is increased, the errors are reduced for all metrics, as shown in Table 8.

Now, if an array of covariance matrices with size 50×50 from set 1 with 2 clusters is used then the accuracy to calculate distance for elements in the same group is increased in all metrics:

No.	Group	AIRM	Euclidean	LERM	Cholesky
2	1	8.79	7.16	8.54	6.21
3	1	8.80	7.20	8.57	6.23
4	1	9.07	7.24	8.83	6.32
5	1	8.72	7.21	8.47	6.41
6	1	8.82	7.17	8.59	6.34
7	1	8.91	7.01	8.67	6.16
8	1	9.11	7.42	8.87	6.52
9	1	9.16	7.55	8.94	6.61
10	1	9.00	7.40	8.77	6.40
11	2	12.00	333.91	11.56	334.02
12	2	11.71	238.26	11.29	238.37
13	2	11.65	237.85	11.23	237.94
14	2	11.74	271.26	11.30	271.36
15	2	11.63	382.90	11.20	383.02
16	2	11.90	278.93	11.49	279.03
17	2	11.93	413.54	11.51	413.65
18	2	11.50	251.24	11.10	251.34
19	2	11.78	241.93	11.35	242.03
20	2	11.48	374.59	11.47	374.70

Table 8 Distance comparative by metric for first element in 50×50 matrices

Table 9 shows 100 repetitions to calculate a true percent to order an array with 20 covariance matrices based in the distance offered for each metric, taking in consideration not only the first element but all the elements in the array.

Metric	Ordering based in distance		
	2×2	3×3	50×50
AIRM	0.772	0.857	1.0
Euclidean	0.904	0.979	1.0
LERM	0.764	0.849	1.0
Cholesky	0.922	0.985	1.0

Table 9 Comparison of ordering distance for each metric

Table 9 shows that, when the size of the matrices is increased all the distance in each metrics tend to be more precise as it was shown in table 7 and table 8 when the first element is used as reference. Now, consider increasing the length of the array of covariance matrices. Group 1 with 500 covariance matrices and 500 covariance matrices for group 2 in set 1 with two clusters. Simulation of 100 arrays with 1,000 covariance matrices varying their size were repeated.

Method	Order based in distance	
	2×2	50×50
AIRM	0.760	1.0
Euclidean	0.892	1.0
LERM	0.750	1.0
Cholesky	0.904	1.0

Table 10 Comparison of ordering based in distance by metrics for 1000 covariance matrices.

Tables 9 and 10 show equivalent results: for low dimensional covariance matrices the accuracy to divide the array in two differentiate sets is less precise than for high dimensional covariance matrices sets. This result will be very important when the algorithm is taken into consideration because the less precise the metric is, the less precise will the algorithm be cluster correctly.

4.2 Comparison of execution time in distance metrics and algorithms

In this section, an analysis of the execution times of distance metrics and the clustering algorithms is presented. This analysis considers the implications in the computational cost of the number of simulations, the size of the matrix and the length of the set of covariance matrices to execute the simulation studies.

Consider varying the size of the array to examine the execution time for each distance metric, as shown in Figure 13.

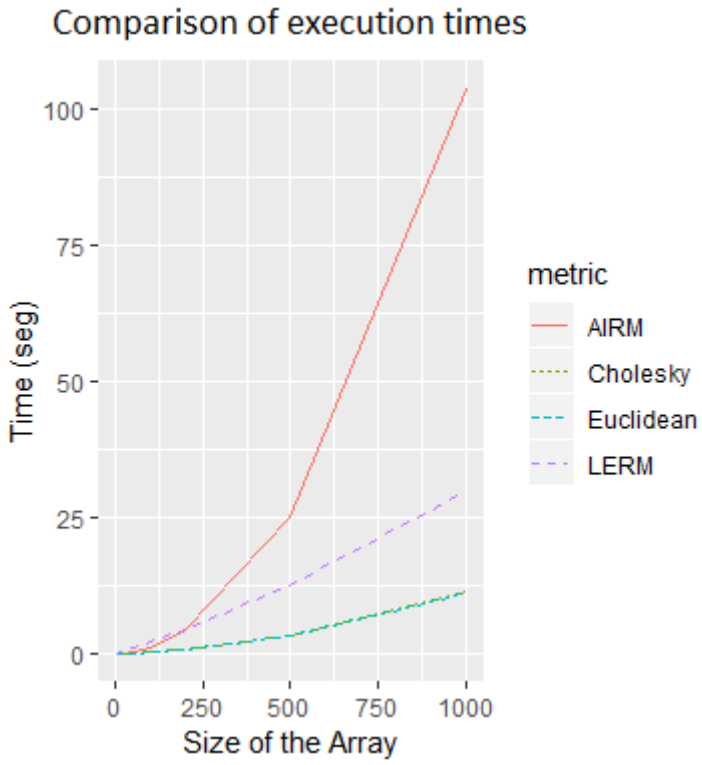


Figure 13 Comparison of execution times for the distance metrics varying the size of the array

The data evidences that AIRM takes considerably more time to calculate the distance between the covariance metrics when the size of the array is significantly large. However, if the size of the matrix is small, the LERM metric takes more time to calculate the distances than the other metrics.

Now, if the size of the covariance matrix is increased, the LERM metric takes more time to calculate the distances between the covariance matrices as shown in Figure 14.

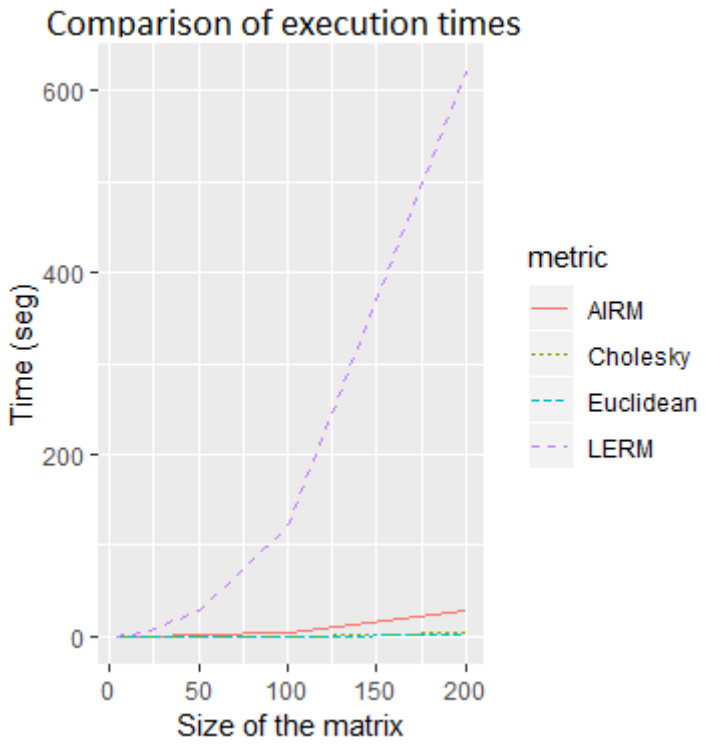


Figure 14 Comparative of execution times for the distance metrics varying the size of the matrix

Consider calculating the time to cluster the array of covariance matrices, first applying the distance metric and then the clustering algorithm. Tables 11 and 12 summarize the worst scenarios in both cases, varying the size of the matrix and the size of the array of covariance matrices, where the time in seconds spent on calculations seen by the user is labeled as user.self.

Metric	user.self (seg)	Algorithm	user.self(seg)	Total time (seg)
AIRM	103.67	HDBSCAN	9.98	113.65
Euclidean	11.14	K-means	0.145	11.285

Table 11 Execution times for an array of 1000 matrices with size 5×5

The AIRM - HDBSCAN pair is the worst combination in terms of runtime when using large array of covariance matrices, as shown in Table 11. However, if a small array with high-dimensional matrices is used for the grouping, then the LERM-HDBSCAN pair results in the worst combination, as shown in Table 12.

Metric	user.self (seg)	Algorithm	user.self(seg)	Total time (seg)
LERM	620.09	HDBSCAN	0.009	620.099
Euclidean	2.27	K-means	0.001	2.271

Table 12 Execution times for an array of 40 matrices with size 200×200.

It is evident that Euclidean and Cholesky's methods exhibit faster execution times in all scenarios. This is very simple to understand due to the definitions of each metric, as it was discussed it in Section 2.4. Similarly, the HDBSCAN clustering algorithm implies a higher computational cost for its execution regardless the number of matrices. Note,

however, that KNN is not considered in the calculation in the computational cost because of it requires additional time for training.

4.3 Comparing clustering algorithm

Sections 4.3.1 – 4.3.3 are organized according to the predefined number of clusters. A comparison of the resulting clustering using the purity measure calculated for each algorithm is presented.

4.3.1 Scenario 1: Two groups

The covariance matrices sets used in the simulation studies for this scenario can be appreciated in Figures 15, 16 and 17, where a detailed description of the different sets is given in Table 4. To obtain a graph for a covariance matrix A, it is necessary to use Cholesky Decomposition for the matrix ($chol(A)$) and then scaling it by factor 1 to convert it into an ellipse centered at the data centroid ellCtr in $(0,0)$. These plots of ellipses are representation of the eigenvalues for each 2×2 covariance matrices in every group.

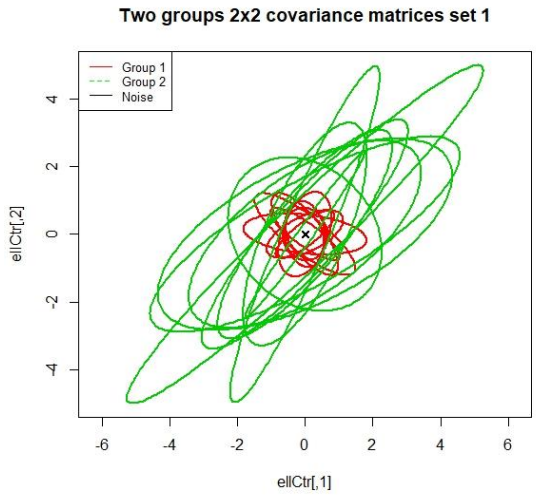


Figure 15 Set 1 for two groups comparison

Dataset 1 (Set 1) was generated to have more separated groups and being easier to cluster than datasets 2 and 3, as shown in Figure 15. The difficulty for clustering is increasing as it is observed in Tables 13-15 for all purity means of clustering algorithms based on 50 simulations, where the values in parenthesis represents standard error of the mean purity σ_p .

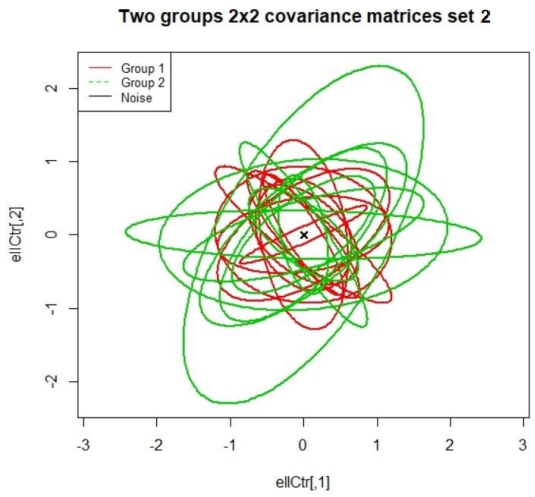


Figure 16 Set 2 for two groups comparison

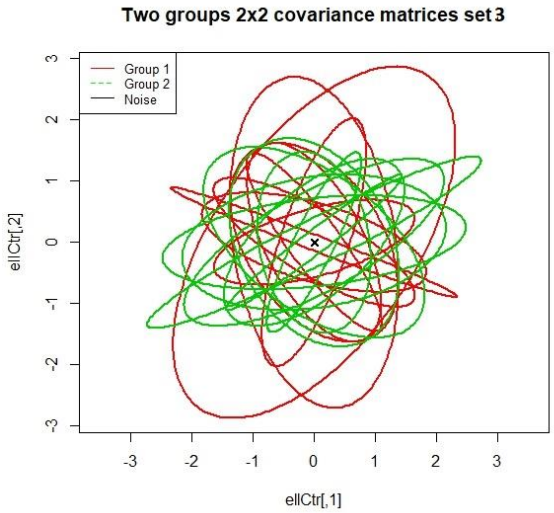


Figure 17 Set 3 for two groups comparison

Matrix Size	Metric	SET	HDBSCAN		DBSCAN	KNN		Kmeans	
			m_{pts} = 4	m_{pts} = 10	m_{pts} = 4 Eps=45	K=3	K=5	K=2	K=3
2 × 2	AIRM	Set1	0.557(0.078)	0.179(0.061)	0.500(0.079)	0.908(0.046)	0.912(0.045)	0.842(0.058)	0.733(0.070)
		Set2	0.254(0.069)	0.0(0.0)	0.500(0.079)	0.585(0.078)	0.596(0.078)	0.542(0.077)	0.516(0.079)
		Set3	0.265(0.069)	0.0(0.0)	0.500(0.079)	0.619(0.077)	0.622(0.077)	0.545(0.079)	0.539(0.079)
	Euclidean	Set1	0.585(0.078)	0.0(0.0)	0.526(0.079)	0.748(0.069)	0.761(0.067)	0.700(0.072)	0.726(0.070)
		Set2	0.249(0.068)	0.0(0.0)	0.500(0.079)	0.647(0.076)	0.638(0.076)	0.633(0.076)	0.591(0.078)
		Set3	0.394(0.077)	0.0(0.0)	0.503(0.079)	0.649(0.075)	0.637(0.076)	0.639(0.076)	0.610(0.077)
	LERM	Set1	0.507(0.079)	0.546(0.079)	0.500(0.079)	0.915(0.044)	0.920(0.043)	0.872(0.053)	0.765(0.067)
		Set2	0.330(0.074)	0.242(0.068)	0.500(0.079)	0.592(0.078)	0.605(0.077)	0.537(0.079)	0.511(0.079)
		Set3	0.336(0.075)	0.233(0.067)	0.500(0.079)	0.621(0.077)	0.619(0.077)	0.555(0.079)	0.455(0.079)
	Cholesky	Set1	0.607(0.077)	0.079(0.043)	0.572(0.078)	0.700(0.072)	0.721(0.071)	0.692(0.073)	0.744(0.069)
		Set2	0.522(0.079)	0.030(0.027)	0.500(0.079)	0.619(0.077)	0.634(0.076)	0.659(0.075)	0.735(0.070)
		Set3	0.651(0.075)	0.031(0.027)	0.500(0.079)	0.596(0.078)	0.616(0.077)	0.657(0.075)	0.777(0.066)
5 × 5	AIRM	Set1	0.963(0.030)	0.941(0.037)	0.500(0.079)	0.996(0.010)	0.996(0.010)	0.999(0004)	0.824(0.060)
		Set2	0.205(0.064)	0.058(0.037)	0.500(0.079)	0.910(0.045)	0.900(0.047)	0.653(0.075)	0.658(0.075)
		Set3	0.260(0.069)	0.098(0.047)	0.500(0.079)	0.900(0.047)	0.885(0.050)	0.523(0.077)	0.605(0.077)
	Euclidean	Set1	0.765(0.067)	0.842(0.058)	0.788(0.064)	0.945(0.036)	0.943(0.037)	0.861(0.055)	0.797(0.063)
		Set2	0.450(0.079)	0.062(0.038)	0.500(0.079)	0.933(0.040)	0.925(0.042)	0.807(0.062)	0.790(0.064)
		Set3	0.619(0.077)	0.259(0.069)	0.506(0.079)	0.878(0.052)	0.902(0.047)	0.763(0.067)	0.756(0.068)
	LERM	Set1	0.955(0.033)	0.962(0.030)	0.500(0.079)	1.0(0.0)	1.0(0.0)	0.998(0.007)	0.875(0.052)
		Set2	0.313(0.073)	0.037(0.030)	0.500(0.079)	0.917(0.044)	0.905(0.046)	0.711(0.072)	0.655(0.075)
		Set3	0.292(0.072)	0.016(0.020)	0.500(0.079)	0.925(0.042)	0.917(0.043)	0.620(0.077)	0.642(0.076)
	Cholesky	Set1	0.828(0.060)	0.826(0.060)	0.575(0.078)	0.975(0.025)	0.975(0.025)	0.863(0.054)	0.820(0.061)
		Set2	0.716(0.071)	0.421(0.078)	0.500(0.079)	0.988(0.017)	0.985(0.019)	0.880(0.051)	0.815(0.061)
		Set3	0.701(0.072)	0.416(0.078)	0.498(0.079)	0.826(0.060)	0.867(0.053)	0.810(0.062)	0.796(0.064)

Table 13 Mean purity results based on 50 simulations. The number on parenthesis are the Monte Carlo standard error of the mean purity. Two groups. Part I

Matrix Size	Metric	SET	HDBSCAN		DBSCAN	KNN		Kmeans	
			m_{pts} = 4	m_{pts} = 10	m_{pts} = 4 Eps=45	K=3	K=5	K=2	K=3
10 × 10	AIRM	Set1	1.0(0.0)	1.0(0.0)	0.500(0.079)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.837(0.058)
		Set2	0.662(0.075)	0.398(0.077)	0.500(0.079)	0.998(0.007)	0.998(0.007)	0.998(0.007)	0.848(0.057)
		Set3	0.317(0.074)	0.160(0.058)	0.500(0.079)	0.976(0.024)	0.978(0.023)	0.575(0.078)	0.747(0.069)
	Euclidean	Set1	0.852(0.056)	0.937(0.038)	0.752(0.068)	0.998(0.007)	0.998(0.007)	0.979(0.022)	0.846(0.057)
		Set2	0.579(0.078)	0.426(0.078)	0.500(0.079)	0.996(0.001)	0.993(0.013)	0.988(0.017)	0.775(0.066)
		Set3	0.739(0.069)	0.591(0.078)	0.000(0.079)	0.993(0.013)	0.995(0.011)	0.923(0.042)	0.838(0.058)
	LERM	Set1	1.0(0.0)	1.0(0.0)	0.500(0.079)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.845(0.057)
		Set2	0.642(0.076)	0.503(0.079)	0.500(0.079)	0.998(0.007)	0.998(0.007)	0.992(0.014)	0.835(0.059)
		Set3	0.475(0.079)	0.226(0.066)	0.500(0.079)	0.983(0.020)	0.985(0.019)	0.979(0.023)	0.798(0.063)
	Cholesky	Set1	0.953(0.033)	0.946(0.036)	1.0(0.0)	0.998(0.007)	0.998(0.007)	0.982(0.021)	0.857(0.055)
		Set2	0.941(0.037)	0.916(0.044)	0.500(0.079)	1.0(0.0)	1.0(0.0)	0.991(0.015)	0.826(0.060)
		Set3	0.771(0.066)	0.730(0.070)	0.0(0.0)	0.998(0.007)	0.998(0.007)	0.916(0.044)	0.850(0.056)
50 × 50	AIRM	Set1	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.825(0.060)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.850(0.056)
		Set3	0.845(0.057)	0.835(0.059)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.911(0.045)
	Euclidean	Set1	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.815(0.061)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.782(0.065)
		Set3	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.852(0.056)
	LERM	Set1	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.800(0.063)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.842(0.058)
		Set3	0.917(0.043)	0.910	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.907(0.046)
	Cholesky	Set1	0.955(0.033)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.785(0.065)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.835(0.059)
		Set3	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.842(0.058)

Table 14 Mean purity results based on 50 simulations. The number on parenthesis are the Monte Carlo standard error of the mean purity. Two groups. Part I. Two groups. Part II

Matrix size	Metric	SET	HDBSCAN		DBSCAN	KNN		Kmeans	
			m_{pts} = 4	m_{pts} = 10	m_{pts} = 4 Eps=45	K=3	K=5	K=2	K=3
200 × 200	AIRM	Set1	1.0(0.0)	1.0(0.0)	0.500(0.079)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.816(0.061)
		Set2	1.0(0.0)	1.0(0.0)	0.500(0.079)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.815(0.061)
		Set3	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.935(0.039)
	Euclidean	Set1	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.818(0.061)
		Set2	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.837(0.058)
		Set3	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.822(0.060)
	LERM	Set1	1.0(0.0)	1.0(0.0)	0.500(0.079)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.805(0.063)
		Set2	1.0(0.0)	1.0(0.0)	0.500(0.079)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.787(0.065)
		Set3	1.0(0.0	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.936(0.037)
	Cholesky	Set1	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.826(0.060)
		Set2	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.832(0.059)
		Set3	1.0(0.0)	1.0(0.0)	0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	0.825(0.060)

Table 15 Mean purity results based on 50 simulations. The number on parenthesis are the Monte Carlo standard error of the mean purity. Two groups. Part I. Two groups. Part III

Notice that all the algorithms are able to detect the groups in dataset 1 better than for other datasets. In addition, the Cholesky's metric performs much better than AIRM for low dimensions covariance matrices (less or equal than 10×10), with a higher mean purity (0.565) when HDBSCAN is used for clustering. However, the purity means for the LERM metric and the AIRM metric are higher when k-means is used to cluster, 0.736 and 0.714, respectively, compared to 0.699 for the Cholesky's metric.

k-means and KNN work much better than HDBSCAN for two groups in low dimensions regardless the distance metric used, however, when the size of the matrices increases the purity measure is almost the same: when varying the input parameters HDBSCAN keeps the purity value, but k-means fails when the true number of clusters is not given. KNN has a very high accuracy value for testing (it was trained before as mentioned in the beginning of Section 4) with a mean purity of 0.809 in low dimensional matrices and 0.982 for high dimensional covariance matrices. Meanwhile, HDBSCAN has a purity mean of 0.975 for high dimension matrices, the purity mean of k-means is 0.86 in the same case.

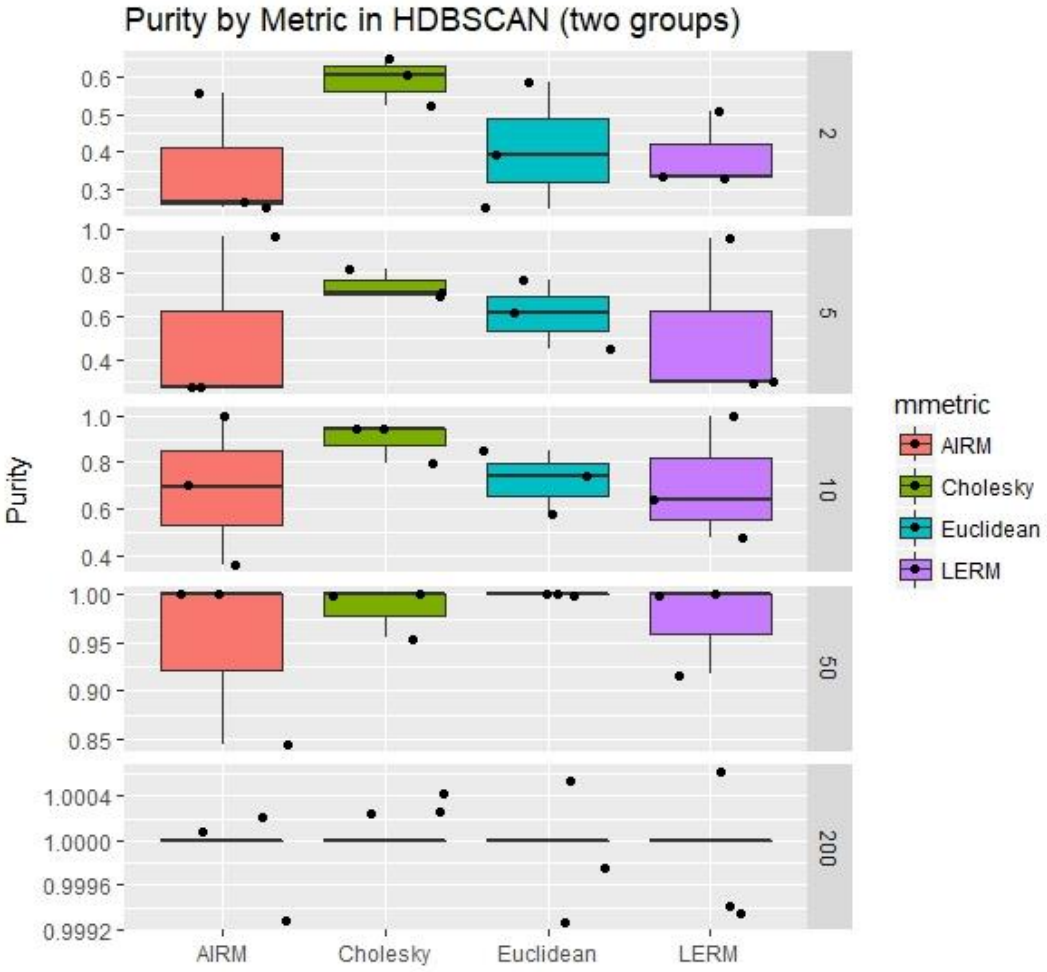


Figure 18 Comparison of Purity by distance metrics and Matrix size for two clusters.

Some results are presented in Figure 18 for $m_{pts} = 4$, k = 5 for KNN and k=2 for k-means. Cholesky's metric performs much better than other metrics under low dimensions, but once the size of the covariance matrices is increased this difference disappears.

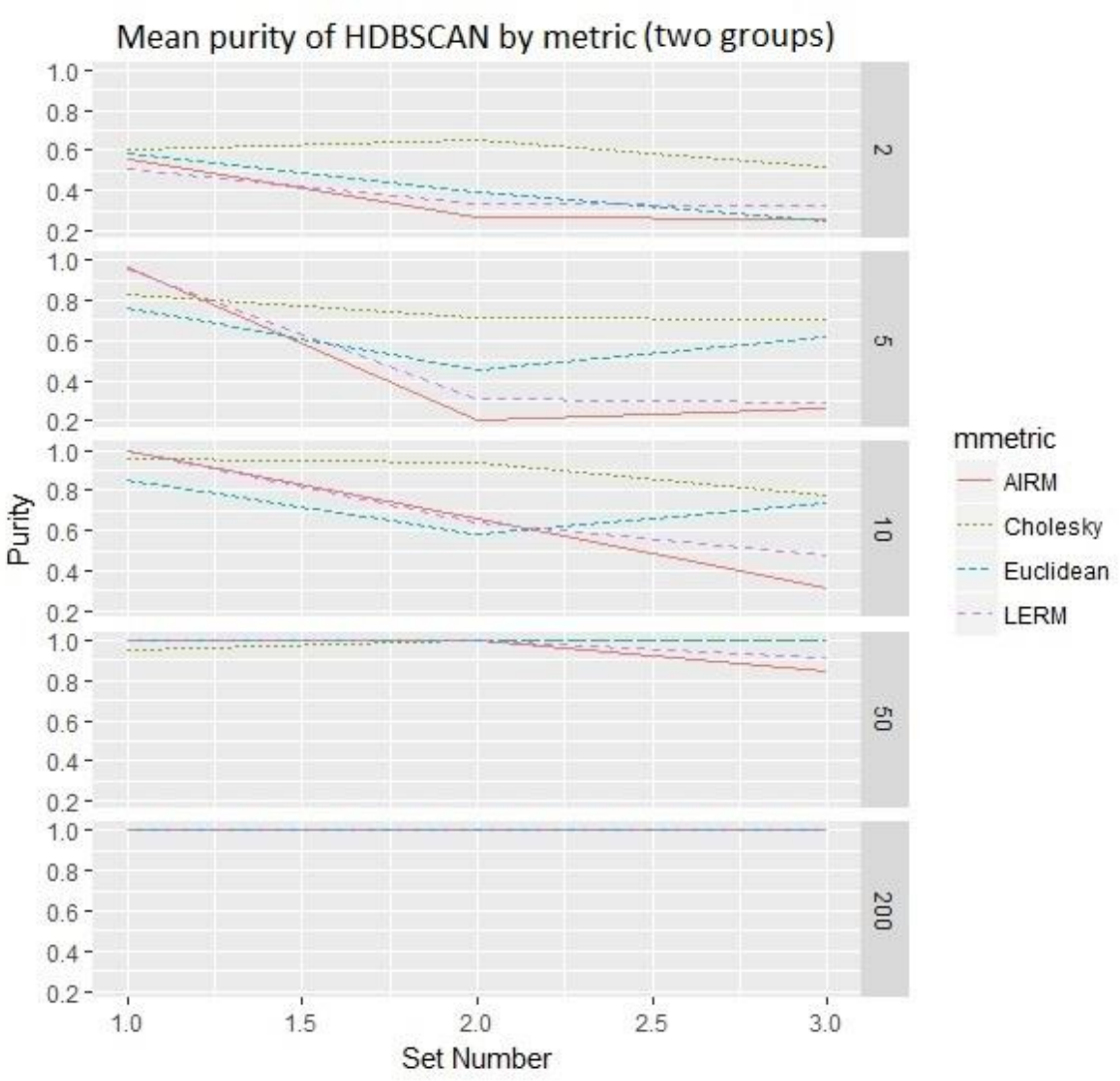


Figure 19 Mean purity of HDBSCAN by metric and matrix (two groups)

Mean purities by set for the HDBSCAN algorithm are shown in Figure 19. Cholesky's metric seems the best option for low dimensions as it has the same purity's average for all datasets. On the other hand, Figure 20 shows the same comparison for k-means, where mean purity for AIRM metric decays for low dimensional matrices.

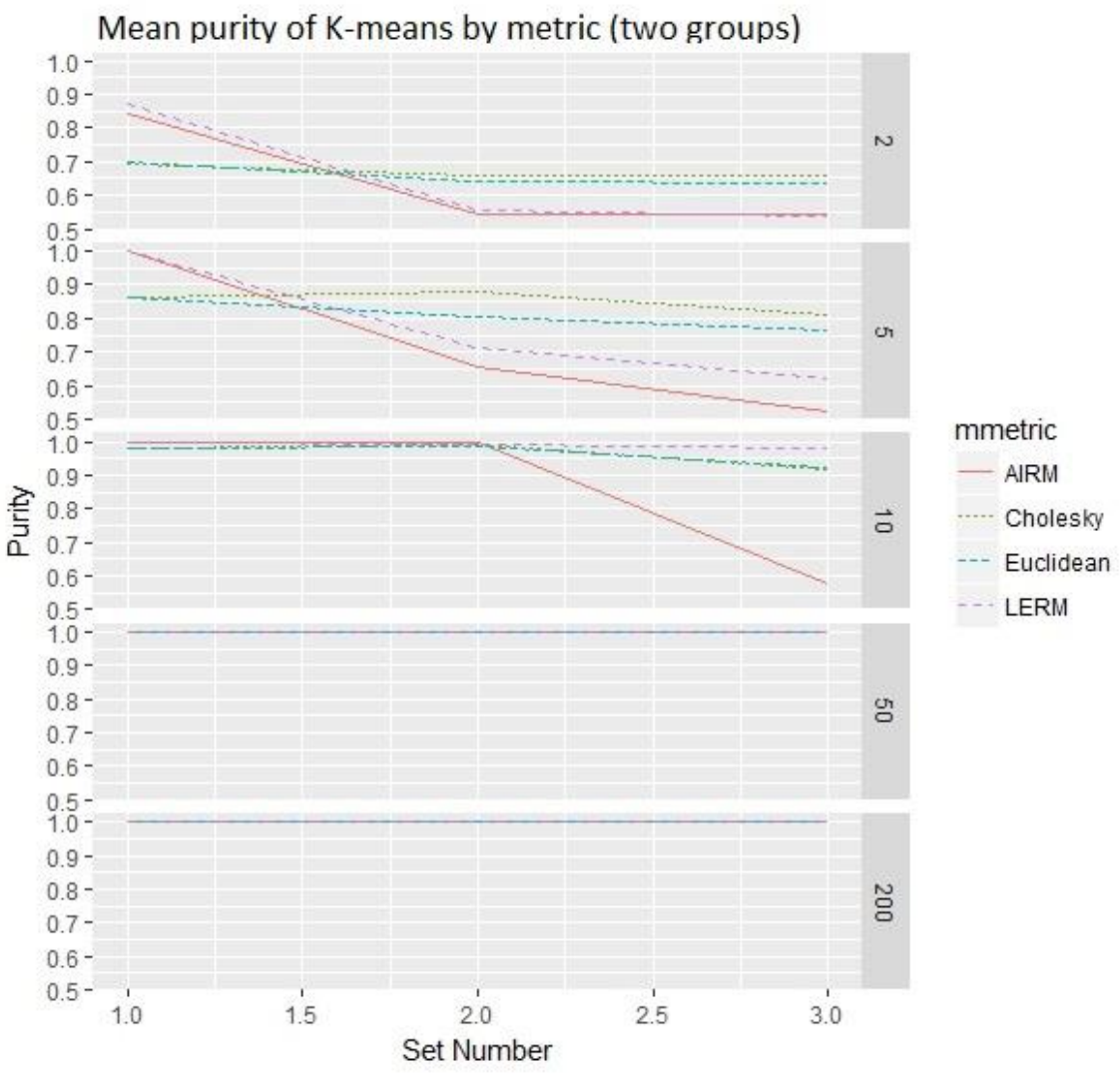


Figure 20 Mean purity of K-means by metric and matrix size (two groups and k=2).

Figures 21 and 22 show the mean purity for all clustering algorithms. An algorithmic tendency line is imposed when the covariance matrices size changes. Comparison of these algorithms by set is shown in Figure 23.

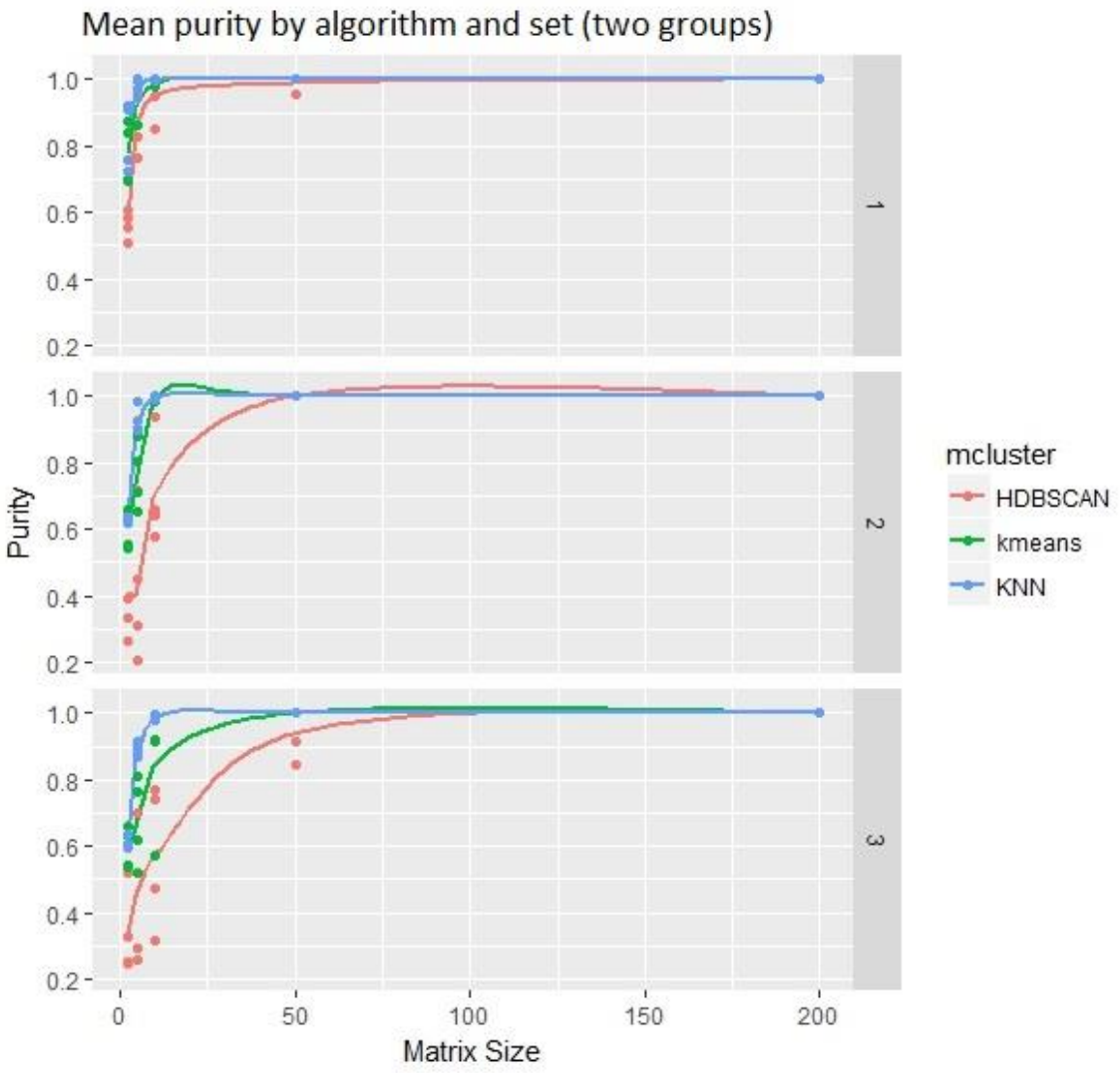


Figure 21 Comparison of purity by Matrix size and Sets for Clustering Algorithms (two groups).

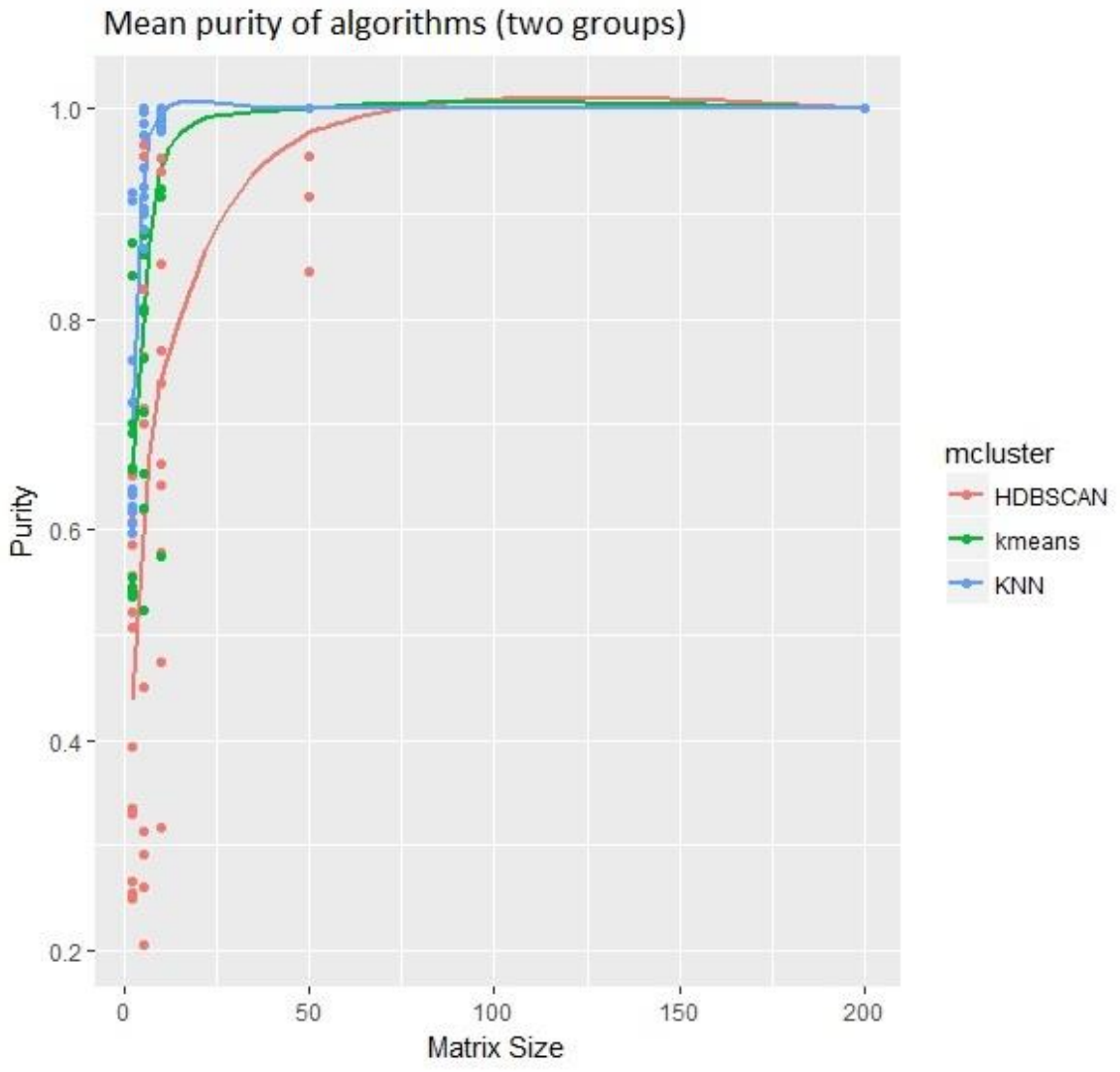


Figure 22 Comparison of Purity by matrix size for clustering Algorithms in data with two clusters, k=2 and m_{pts}= 4.

It is evident that HDBSCAN works better when for high-dimensional covariance matrices, although scenarios with two clusters vary, as shown in Figure 21. In general , no difference between the algorithms can be appreciated when the data consists of two clusters, as shown in Figure 22.

4.3.2 Scenario 2: Four groups

Tables 17 and 18 show the means of purity measure, where following the characteristics of the sets summarized in Table 5, the input parameters vary and arrays of 20 covariance matrices per group were used. An example of using HDBSCAN with the AIRM metric to obtain clusters for set 1 consisting of 4 distinct groups is shown in Figure 23.

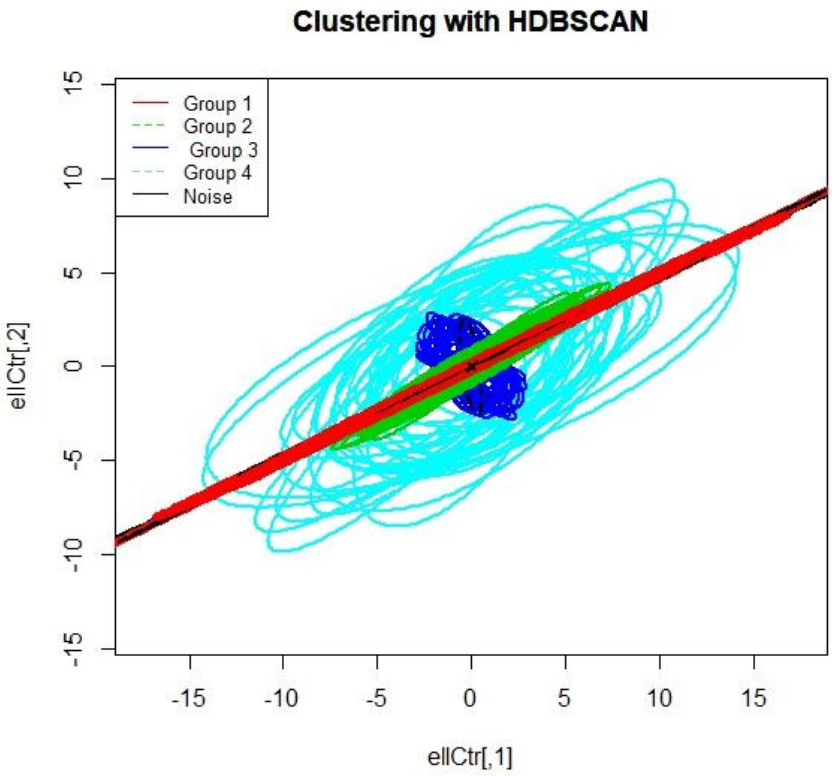


Figure 23 Four groups with 2x2 covariance matrices, set 1

The elements of set 1 were easy to identify by representing ellipses, however, there is no clear identification of the groups when set 2 is used, as shown in Figure 24.

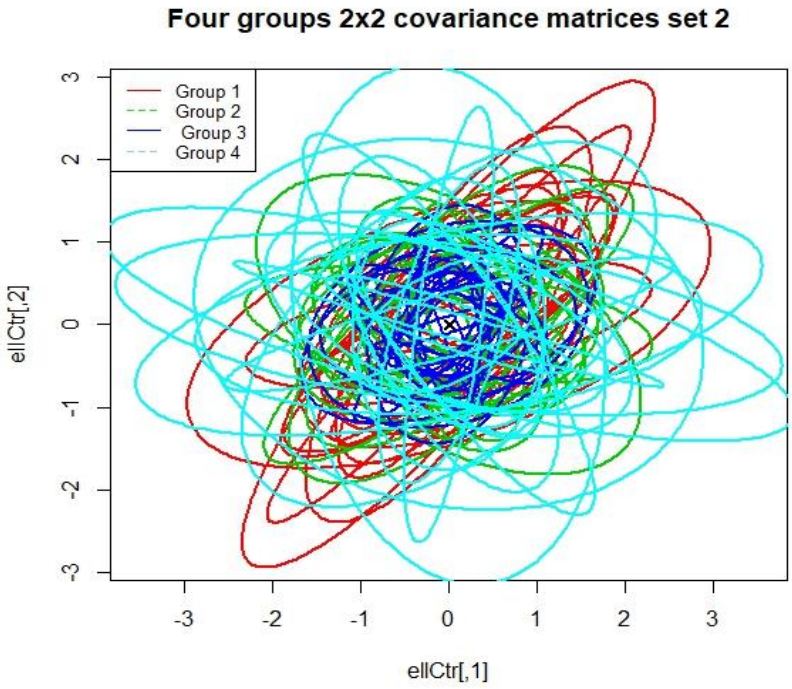


Figure 24 Four groups 2x2 matrices, set 2

Although KNN was already trained, it still has problems identifying groups in dataset 2 for each distance metric when using low dimensions covariance matrices (10 or less attributes), as shown in Table 17. The purity means for KNN, K-means and HDBSCAN are 0.692, 0.613 and 0.281, respectively, when the set 2 is used. However, the HDBSCAN's mean purity is 0.956 when the size of the covariance matrices in the set 2 is increased to high dimensions, while the mean purity of k-means is 0.81.

In addition, AIRM and LERM metrics have higher mean purity when distances between high-dimensional covariance matrices are used and the data set consists of four clusters, as shown in Figures 25 and 26.

Matrix size	Metric	SET	HDBSCAN		DBSCAN	KNN		K-means	
			m_{pts}=4	m_{pts}=10	$m_{pts} = 4$ eps= 8.8	K=3	K=5	K=4	K=5
2×2	AIRM	Set1	0.836(0.041)	0.761(0.048)	0.808(0.044)	0.908(0.032)	0.916(0.031)	0.888(0.035)	0.869(0.038)
		Set2	0.178(0.043)	0.0 (0.0)	0.249(0.048)	0.319(0.052)	0.322(0.052)	0.361(0.054)	0.338(0.053)
	Euclidean	Set1	0.430(0.055)	0.052(0.025)	0.047(0.024)	0.477(0.056)	0.489(0.056)	0.362(0.054)	0.391(0.055)
		Set2	0.251(0.048)	0.00 (0.00)	0.261(0.059)	0.299(0.051)	0.309(0.052)	0.370(0.054)	0.397(0.055)
	LERM	Set1	0.644(0.053)	0.755(0.048)	0.677(0.058)	0.886(0.035)	0.891(0.035)	0.887(0.035)	0.853(0.040)
		Set2	0.201(0.045)	0.181(0.043)	0.245(0.048)	0.324(0.052)	0.334(0.053)	0.392(0.055)	0.378(0.054)
	Cholesky	Set1	0.576(0.055)	0.220(0.032)	0.186(0.043)	0.564(0.055)	0.591(0.055)	0.364(0.054)	0.380(0.054)
		Set2	0.441(0.055)	0.196(0.044)	0.304(0.051)	0.355(0.053)	0.368(0.054)	0.501(0.056)	0.597(0.055)
5×5	AIRM	Set1	0.599(0.055)	0.507(0.056)	0.187(0.044)	0.999(0.004)	0.999(0.004)	0.836(0.041)	0.836(0.041)
		Set2	0.171(0.042)	0.022(0.016)	0.241(0.048)	0.849(0.04)	0.846(0.040)	0.607(0.055)	0.575(0.055)
	Euclidean	Set1	0.627(0.054)	0.415(0.055)	0.0(0.0)	0.590(0.055)	0.578(0.055)	0.586(0.055)	0.541(0.056)
		Set2	0.243(0.048)	0.0(0.0)	0.248(0.048)	0.752(0.048)	0.772(0.047)	0.515(0.056)	0.563(0.055)
	LERM	Set1	0.643(0.053)	0.605(0.055)	0.426(0.055)	0.995(0.008)	0.995(0.008)	0.822(0.043)	0.834(0.042)
		Set2	0.202(0.045)	0.007(0.009)	0.245(0.048)	0.862(0.039)	0.852(0.040)	0.594(0.055)	0.595(0.055)
	Cholesky	Set1	0.641(0.054)	0.512(0.056)	0.0(0.0)	0.932(0.028)	0.950(0.024)	0.611(0.054)	0.654(0.053)
		Set2	0.493(0.056)	0.398(0.055)	0.329(0.052)	0.725(0.050)	0.712(0.051)	0.629(0.054)	0.671(0.052)

Table 16 Mean purity results based on 50 simulations. The number on parenthesis are the Monte Carlo standard error of the mean purity. Four clusters. Part I

Matrix size	Metric	SET	HDBSCAN		DBSCAN	KNN		K-means	
			m_{pts}=4	m_{pts}=10	m_{pts}=4 eps= 8.8	K=3	K=5	K=4	K=5
10 × 10	AIRM	Set1	0.993(0.009)	0.976(0.017)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.896(0.034)	0.887(0.035)
		Set2	0.496(0.056)	0.391(0.055)	0.248(0.048)	0.994(0.008)	0.992(0.010)	0.914(0.031)	0.844(0.041)
	Euclidean	Set1	0.735(0.049)	0.826(0.042)	0.0(0.0)	0.875(0.037)	0.886(0.035)	0.665(0.053)	0.771(0.047)
		Set2	0.458(0.056)	0.177(0.043)	0.250(0.048)	0.968(0.027)	0.966(0.020)	0.749(0.048)	0.737(0.049)
	LERM	Set1	0.967(0.020)	0.936(0.027)	0.0(0.0)	0.996(0.007)	0.995(0.008)	0.816(0.043)	0.875(0.037)
		Set2	0.562(0.055)	0.434(0.055)	0.249(0.048)	0.995(0.008)	0.993(0.009)	0.932(0.028)	0.867(0.038)
	Cholesky	Set1	0.876(0.037)	0.787(0.046)	0.0(0.0)	0.994(0.008)	0.991(0.010)	0.600(0.055)	0.715(0.050)
		Set2	0.696(0.051)	0.551(0.055)	0.290(0.051)	0.877(0.035)	0.853(0.040)	0.815(0.043)	0.782(0.046)
50 × 50	AIRM	Set1	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.767(0.047)	0.834(0.042)
		Set2	0.998(0.005)	0.998(0.005)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.828(0.042)	0.870(0.038)
	Euclidean	Set1	0.875(0.037)	0.998(0.005)	0.0(0.0)	0.994(0.008)	0.992(0.010)	0.841(0.041)	0.775(0.047)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.785(0.046)	0.795(0.045)
	LERM	Set1	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.833(0.042)	0.849(0.04)
		Set2	0.999(0.003)	0.999(0.003)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.795(0.045)	0.798(0.045)
	Cholesky	Set1	0.999(0.003)	0.996(0.007)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.817(0.043)	0.787(0.046)
		Set2	0.776(0.046)	0.768(0.047)	0.160(0.041)	0.938(0.027)	0.925(0.029)	0.690(0.051)	0.845(0.04)
200 × 200	AIRM	Set1	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.856(0.039)	0.809(0.044)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.811(0.044)	0.855(0.039)
	Euclidean	Set1	0.890(0.035)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.818(0.043)	0.801(0.044)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.792(0.045)	0.841(0.041)
	LERM	Set1	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.785(0.046)	0.895(0.034)
		Set2	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.793(0.045)	0.875(0.037)
	Cholesky	Set1	1.0(0.0)	1.0(0.0)	0.0(0.0)	1.0(0.0)	1.0(0.0)	0.860(0.039)	0.827(0.042)
		Set2	1.0(0.0)	0.987(0.012)	0.0(0.0)	0.996(0.007)	0.996(0.007)	0.810(0.044)	0.803(0.044)

Table 17 Mean purity results based on 50 simulations. The number on parenthesis are the Monte Carlo standard error of the mean purity. Four clusters. Part II

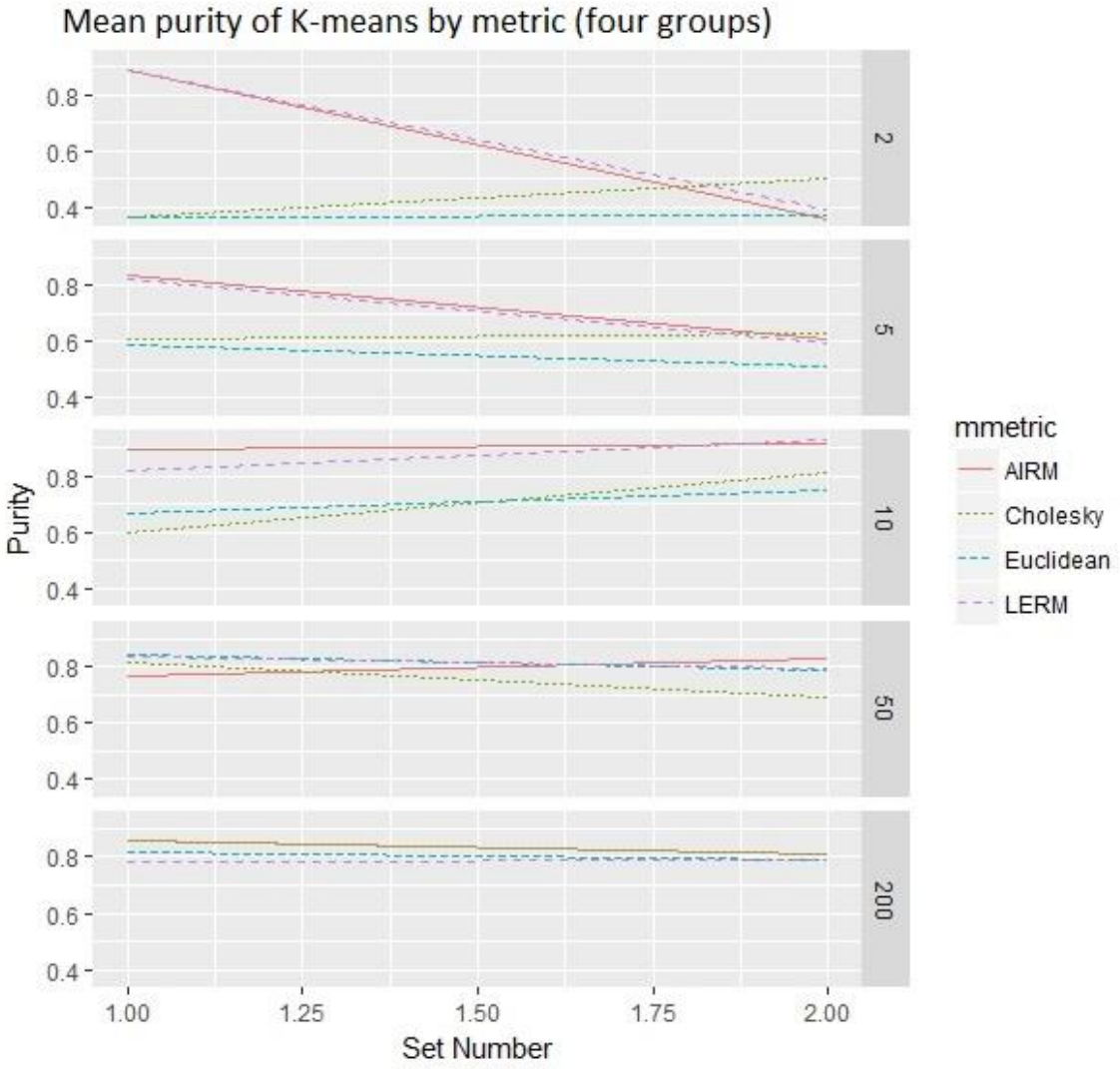


Figure 25 Comparison of Purity by Matrix size for K-means (k=4)

Using $m_{pts} = 4$, k = 5 for KNN and k=4 for k-means, the Cholesky's metric appears to keep the same mean Purity for each matrix size, as shown in Figure 25 for K-means and Figure 26 for HDBSCAN. Moreover, it is notable that AIRM and LERM have better performance than Cholesky's metric in high dimensional matrices using K-means or HDBSCAN.

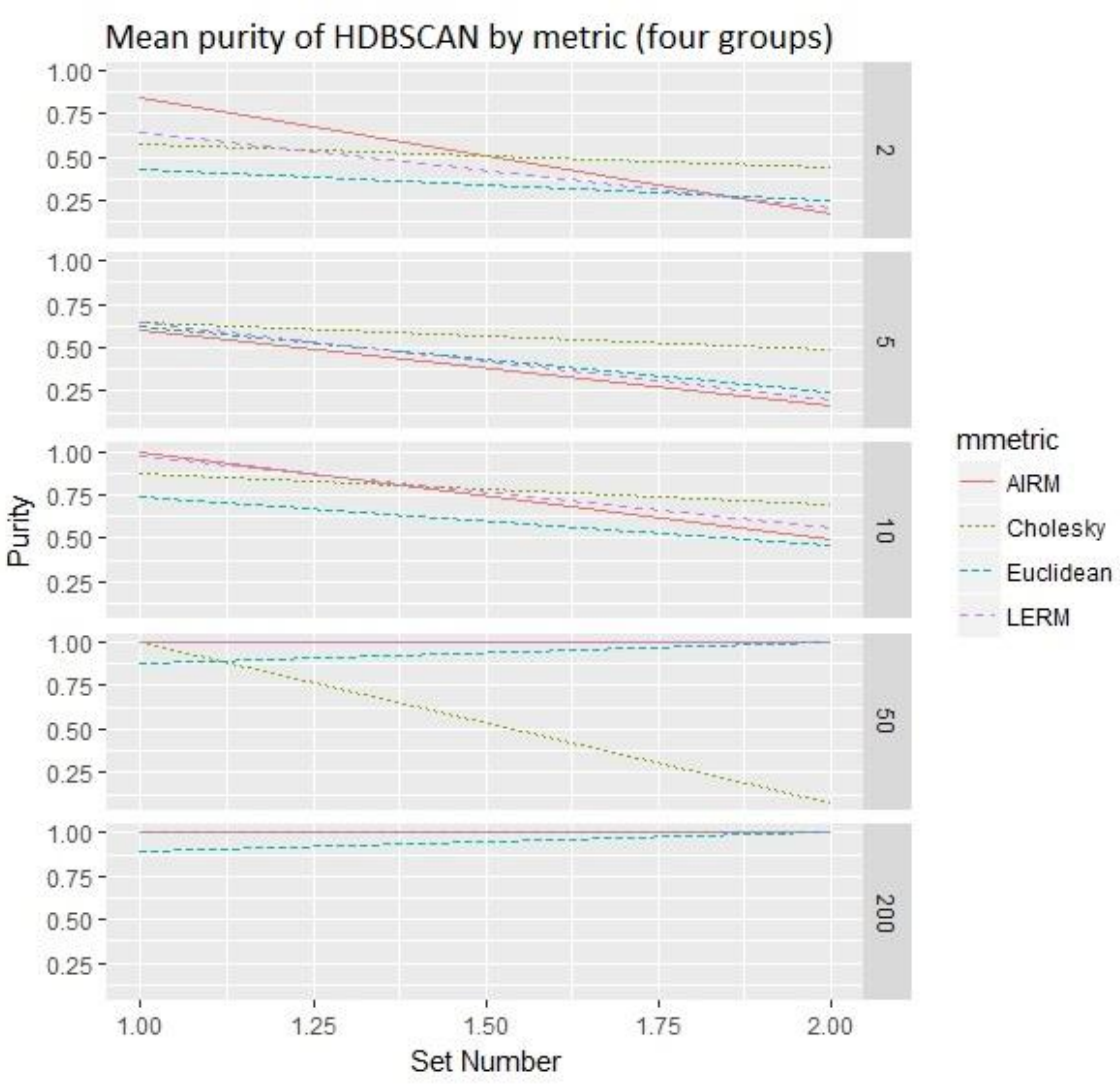


Figure 26 Comparison of Purity by Matrix size for HDBSCAN ($m_{pts} = 4$)

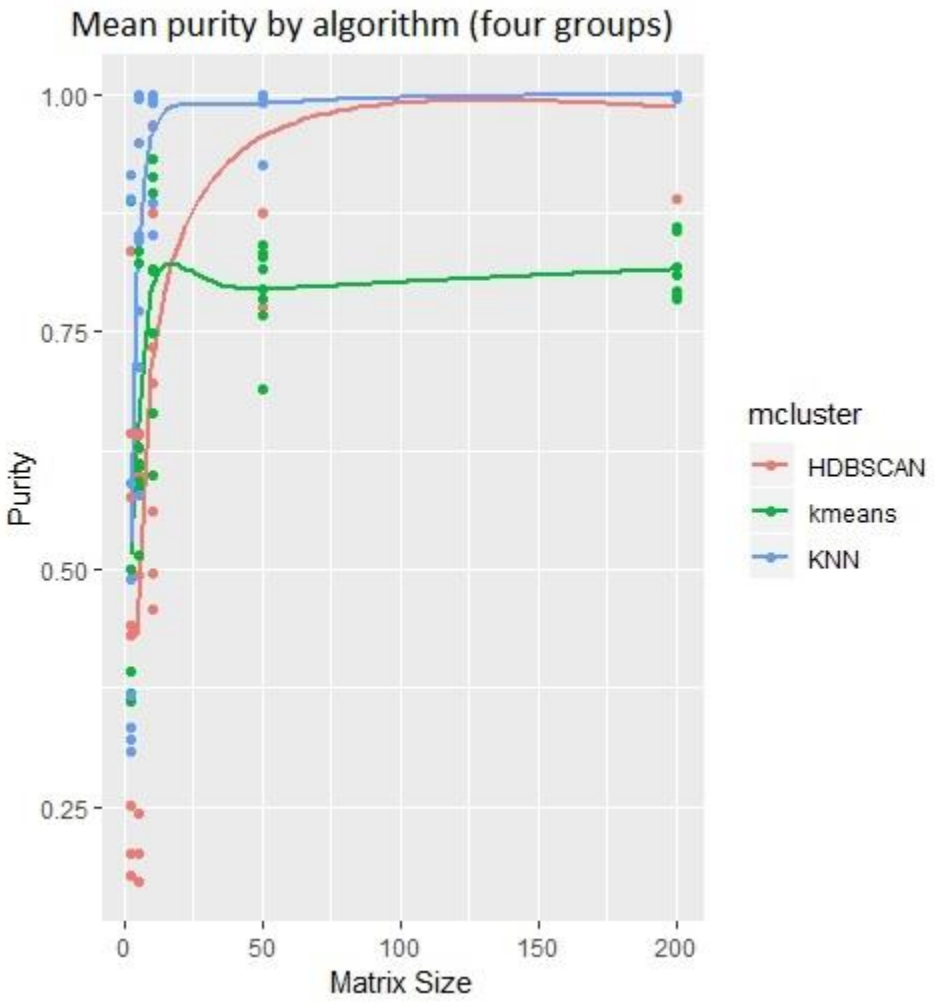


Figure 27 Comparison of Purity by Matrix size and clustering Algorithms (Four clusters)

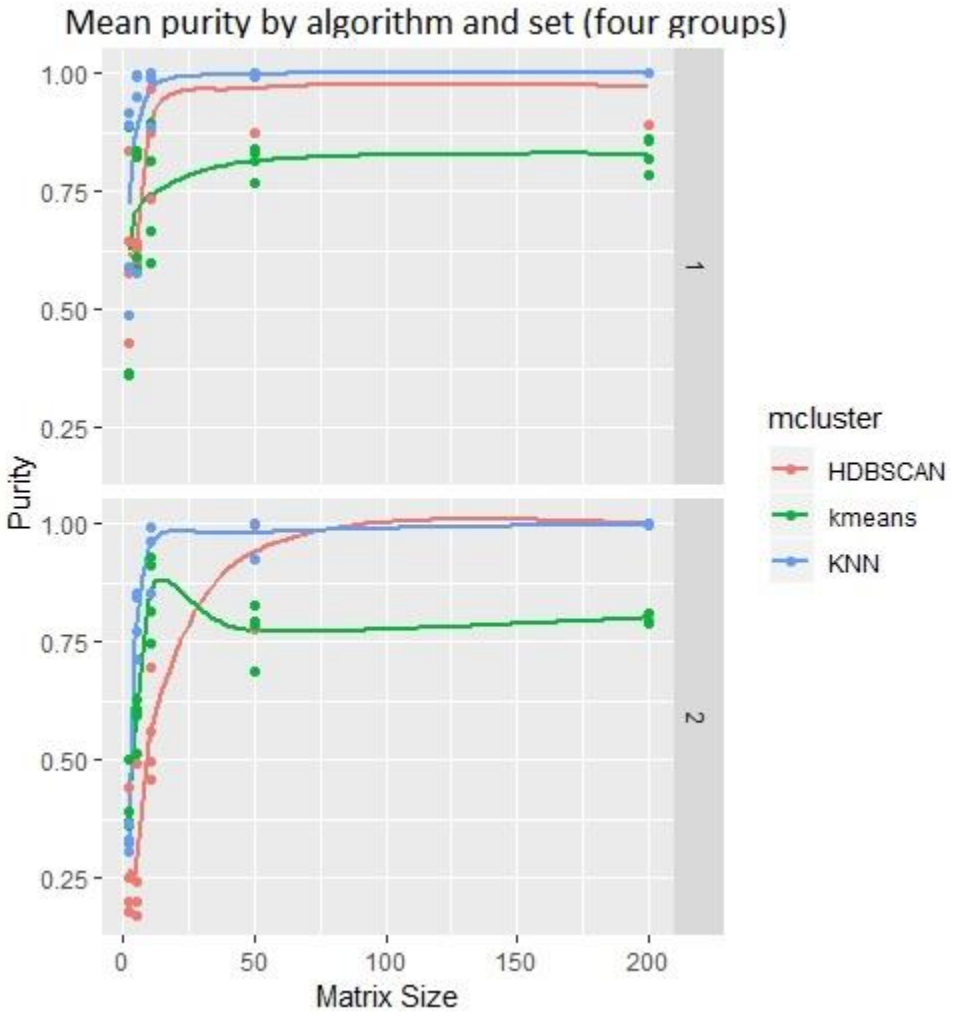


Figure 28 Comparison of Purity by Matrix size and Set for clustering Algorithms (four clusters).

The K-means algorithm manifests problems to detect the 4 clusters in the array for each dataset, as shown in Figure 27 and 28. In addition, HDBSCAN and KNN exhibit higher mean purity when using distances from high dimensional matrices. Although, KNN has excellent results, it was necessary a training step before, while HDBSCAN did not.

4.3.3 Scenario 3: Eight groups

Table 19 presents the purity results when arrays with 20 covariance matrices per group were used to simulate the data, as detailed in table 6.

Matrix size	Metric	Dataset	HDBSCAN		DBSCAN	KNN		K-means	
			m_{pts}=4	m_{pts}=10	m_{pts}=4 eps=45	K=3	K=5	K=8	K=9
2 × 2	AIRM	Set1	0.289(0.036)	0.206(0.032)	0.125(0.026)	0.557(0.039)	0.581(0.039)	0.483(0.039)	0.474(0.039)
	Euclidean	Set1	0.204(0.032)	0.006(0.006)	0.143(0.027)	0.322(0.037)	0.315(0.037)	0.293(0.036)	0.282(0.035)
	LERM	Set1	0.326(0.037)	0.195(0.031)	0.125(0.027)	0.558(0.039)	0.574(0.039)	0.480(0.039)	0.474(0.039)
	Cholesky	Set1	0.411(0.038)	0.279(0.035)	0.256(0.034)	0.569(0.039)	0.567(0.039)	0.434(0.039)	0.463(0.039)
5 × 5	AIRM	Set1	0.252(0.034)	0.260(0.035)	0.147(0.028)	0.838(0.029)	0.830(0.029)	0.628(0.038)	0.654(0.037)
	Euclidean	Set1	0.316(0.036)	0.073(0.021)	0.124(0.026)	0.394(0.038)	0.387(0.038)	0.415(0.038)	0.416(0.039)
	LERM	Set1	0.263(0.035)	0.261(0.035)	0.135(0.027)	0.843(0.028)	0.837(0.029)	0.676(0.037)	0.672(0.037)
	Cholesky	Set1	0.510(0.039)	0.351(0.037)	0.249(0.034)	0.727(0.035)	0.718(0.035)	0.522(0.039)	0.502(0.039)
10 × 10	AIRM	Set1	0.554(0.039)	0.550(0.039)	0.125(0.027)	0.995(0.005)	0.992(0.007)	0.787(0.032)	0.863(0.027)
	Euclidean	Set1	0.660(0.037)	0.499(0.039)	0.125(0.027)	0.598(0.038)	0.603(0.038)	0.733(0.034)	0.806(0.031)
	LERM	Set1	0.563(0.039)	0.526(0.039)	0.125(0.027)	0.994(0.006)	0.993(0.006)	0.851(0.028)	0.842(0.028)
	Cholesky	Set1	0.724(0.035)	0.614(0.038)	0.128(0.026)	0.936(0.019)	0.935(0.019)	0.670(0.037)	0.696(0.036)
50 × 50	AIRM	Set1	1.0(0.0)	1.0(0.0)	0.625(0.038)	1.0(0.0)	1.0(0.0)	0.781(0.032)	0.752(0.034)
	Euclidean	Set1	0.993(0.006)	0.996(0.005)	0.237(0.033)	0.613(0.038)	0.615(0.038)	0.802(0.031)	0.797(0.032)
	LERM	Set1	1.0(0.0)	1.0(0.0)	0.625(0.038)	1.0(0.0)	1.0(0.0)	0.693(0.036)	0.795(0.032)
	Cholesky	Set1	0.879(0.025)	0.870(0.026)	0.248(0.034)	0.987(0.008)	0.983(0.010)	0.736(0.034)	0.768(0.033)
200 × 200	AIRM	Set1	1.0(0.0)	1.0(0.0)	0.500(0.039)	1.0(0.0)	1.0(0.0)	0.744(0.034)	0.795(0.032)
	Euclidean	Set1	0.990(0.007)	1.0(0.0)	0.375(0.038)	0.609(0.038)	0.617(0.038)	0.830(0.029)	0.833(0.029)
	LERM	Set1	1.0(0.0)	1.0(0.0)	0.500(0.039)	1.0(0.0)	1.0(0.0)	0.741(0.034)	0.738(0.034)
	Cholesky	Set1	0.875(0.026)	0.875	0.250(0.034)	0.998(0.003)	0.999(0.002)	0.726(0.035)	0.817(0.030)

Table 18 Mean purity results based on 50 simulations. The number on parenthesis are the Monte Carlo standard error of the mean purity. Eight groups

The comparison of purities in Figure 29 shows that, Cholesky's metric still has higher mean purity at low dimensional covariance matrices, however, LERM and AIRM metrics have better evaluations when high-dimensional covariance matrices are used. It is important to note that in this scenario with high dimensional matrices, Cholesky's metric did not achieve a perfect purity value, Euclidean distance shows higher purity mean value when compared to Cholesky's metric, as shown in Table 19.

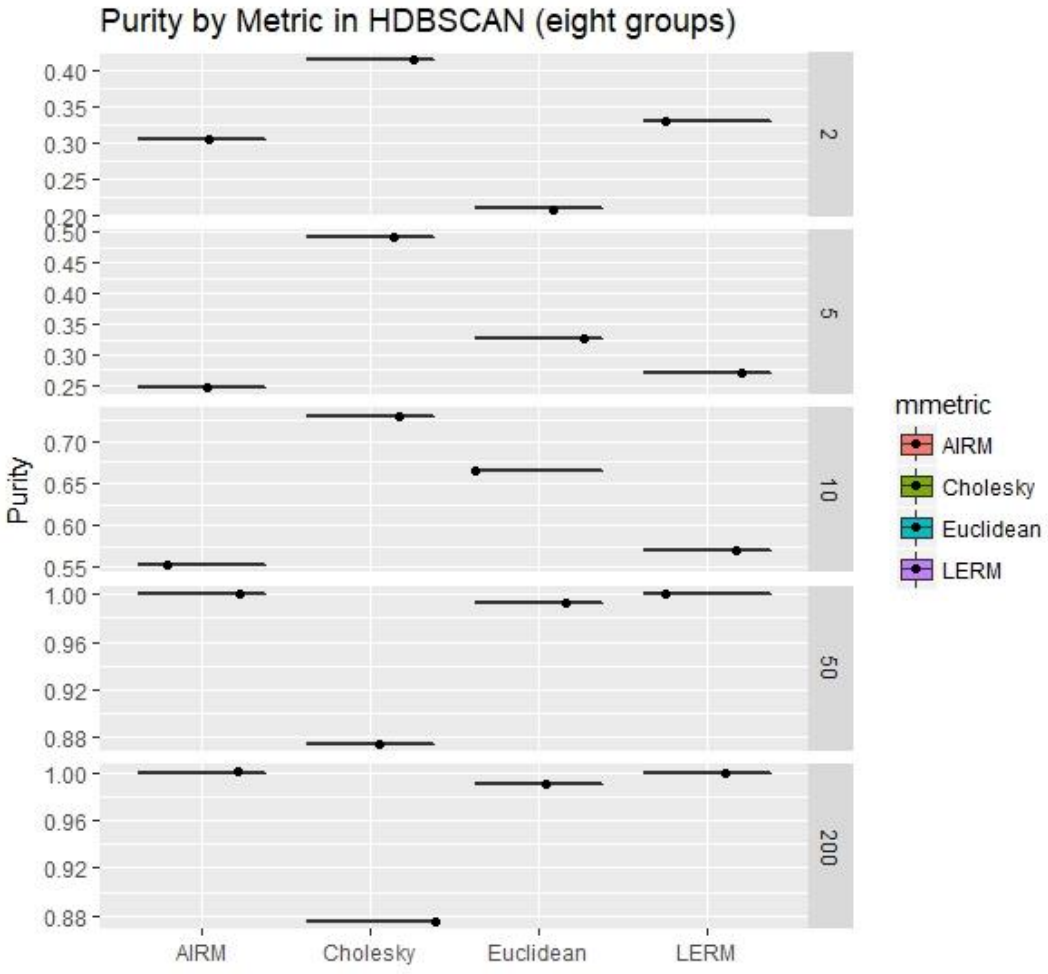


Figure 29 Comparison of Purity by Metric and Matrix size for HDBSCAN ($m_{pts} = 4$), eight groups.

HDBSCAN overperforms k-means when covariance matrices sizes are increased, as shown in Figure 30. Notice, even with a training step, KNN has a lower mean purity than HDBSCAN. In this scenario, k-means has the lowest values.

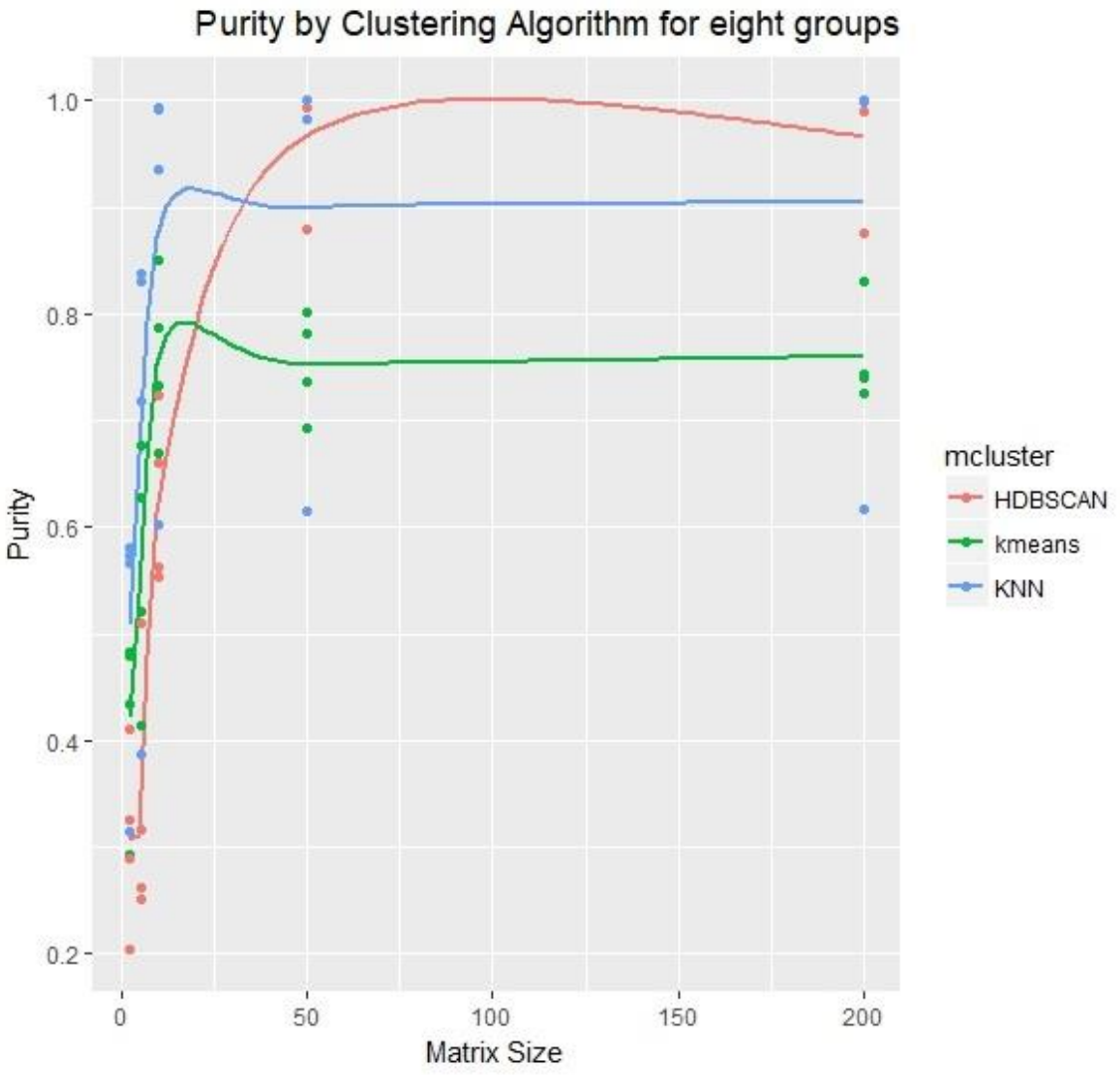


Figure 30 Mean Purity by Clustering Algorithms (m_{pts} = 4, k = 5 for KNN and k=8 for k-means)

4.3.4 Summary

The comparison between distance metrics is shown in Figures 31 and 32. Cholesky's metric evidences better performance in low dimensions than the other metrics.

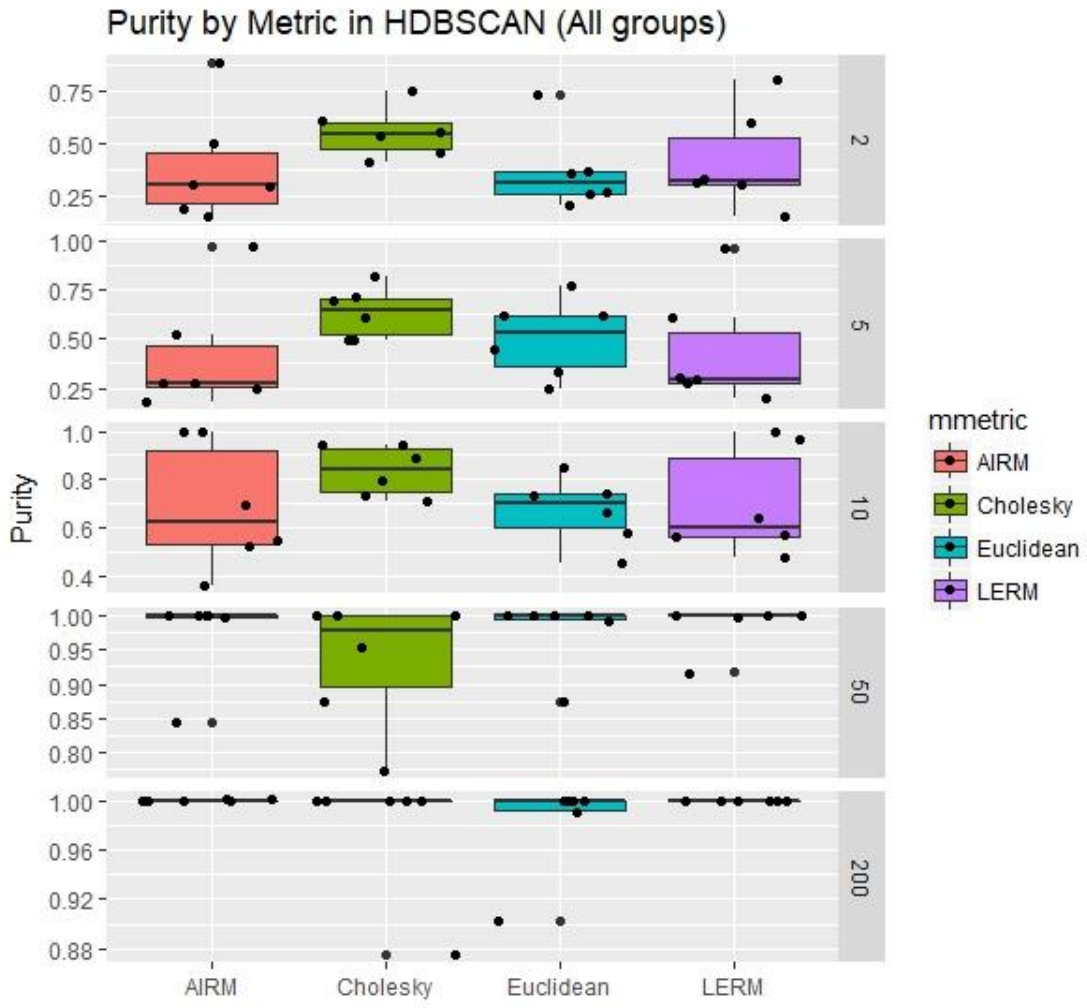


Figure 31 Purity by metric for HDBSCAN (All groups)

Even though AIRM and LERM metrics manifest to have higher mean purity for high-dimensional covariance matrices when HDBSCAN is used, as shown in Figure 32, an

ANOVA test is necessary to confirm and to generalize this proposition for other algorithms.

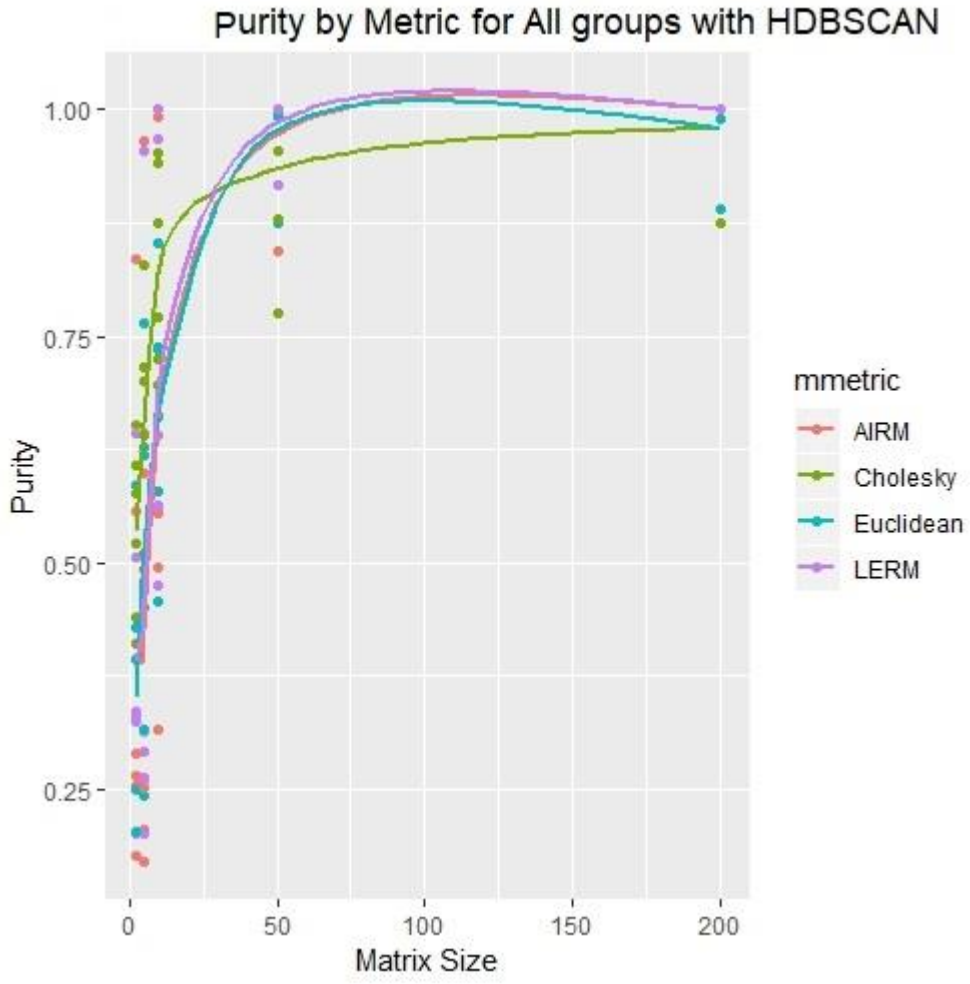


Figure 32 Comparison of Purity by metric using HDBSCAN

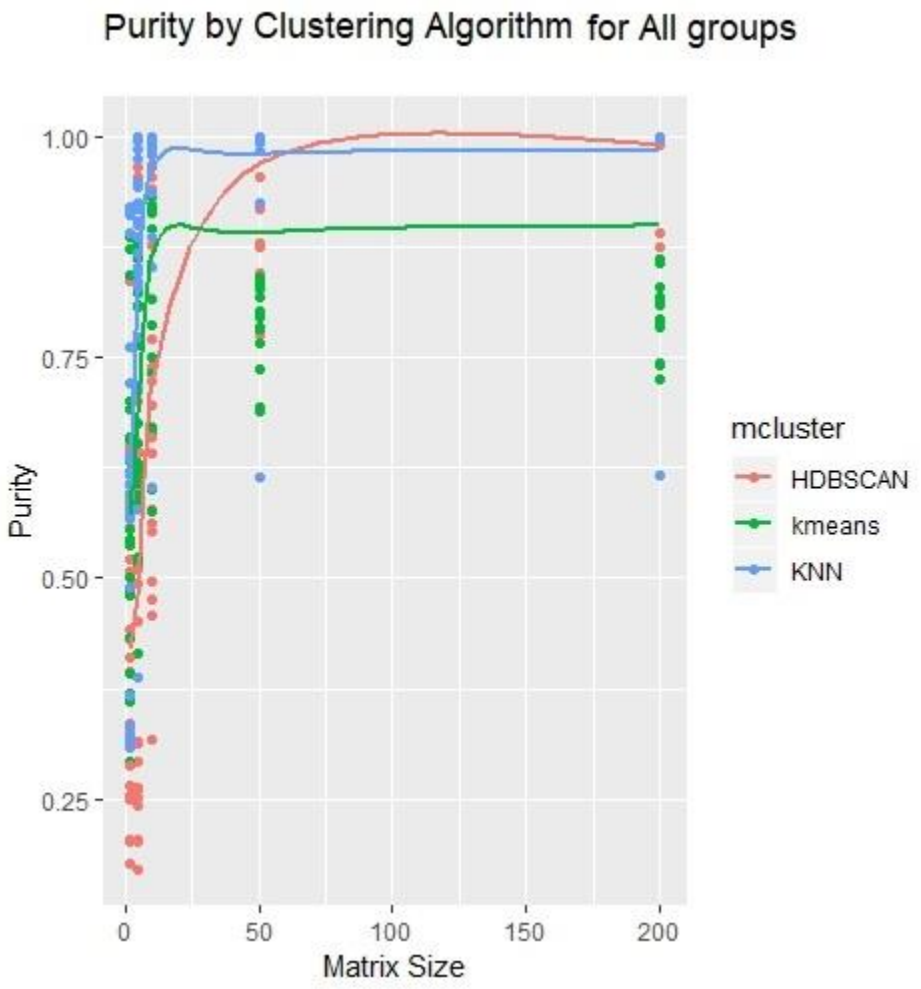


Figure 33 Purity for clustering algorithms (All groups)

If the number of clusters present in the data is increased, the purity for the k-means algorithm decreases, it is more difficult to detect appropriately to what group each covariance matrix belongs to, as shown Figure 33. Moreover, Figure 33 shows that HDBSCAN can achieve the performance of KNN for high dimensions, without any training steps.

In order to determine the significant factors associated to the performance of the clustering algorithms according to the simulation studies, a linear model involving the main factors in the simulation studies as well as some interaction terms are presented next.

Consider the following linear model to explain the mean purity in terms of the simulation factors:

$$Purity_{i\,j\,k\,l} \sim Metric_i + Algorithm_j * Matrix\ Size_k + Number\ of\ groups_l + \epsilon_{ijlk} \ ,$$

where $var(\epsilon_{ijkl}) = \sigma^2(|v_k|^{\delta_j})$.

$i = 1.AIRM, 2.Cholesky, 3.Euclidean, 4.LERM.$

$j = 1.HDBSCAN, 2.K-means, 3.KNN$

$k = 2, 5, 10, 50, 200$

$l = 2, 4, 8$

The heterogeneous variance for the error was tested and Table 19 presents the sum of squares Type III for the model described previously, when K-means takes k = true number of clusters (2,4 and8), k = 5 for KNN and m_{pts}=4 for HDBSCAN.

Response: Purity	Df	Chisq	Pr(>Chisq)
(Intercept)	1	1187.888	<2.20E-16***
Algorithm	2	31.010	1.84E-13***
Matrix Size	1	84.880	<2.20E-16***
Metric	3	3.805	0.1468
Number of groups	1	63.652	1.483E-15***
Algorithm*Matrix size	2	25.673	1.03E-05***

Table 19 ANOVA type III for purity when the true number of clusters is provided to k-means.

The distance metric is not significant in the performance of clustering, however, the interaction between matrix size and clustering algorithm is significant as shown in Table 19. This is, the interaction between these variables is significative and their choice affects the purity measure as shown in Figure 34.

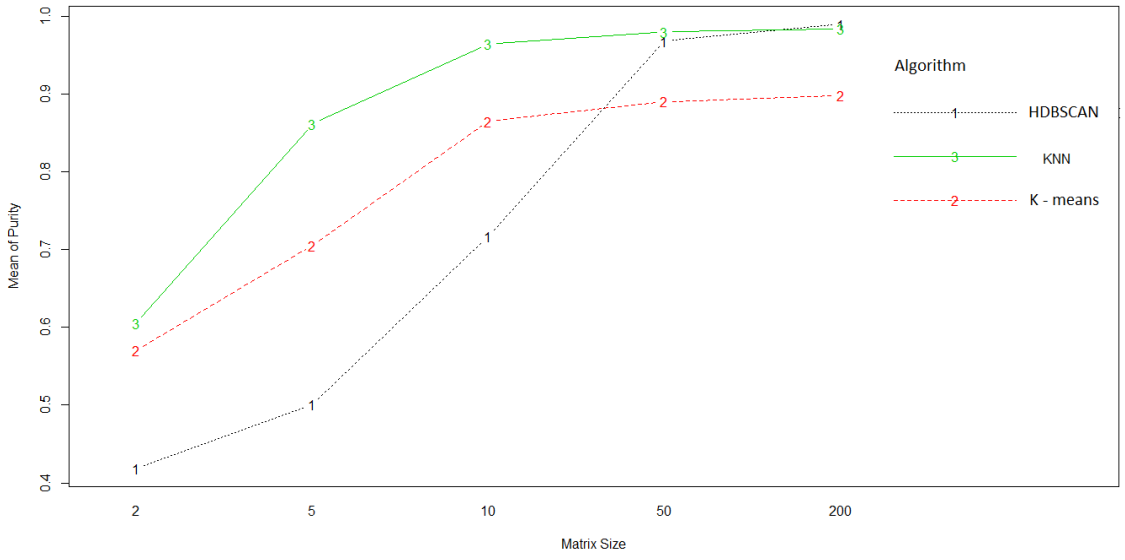


Figure 34 Interaction plot: Matrix Size – Algorithm with true number of cluster

Even though, K-means is provided with the true number of clusters, HDBSCAN has better results when the data is consisting of high-dimensional matrices.

Consider now varying the parameters for the clustering algorithms. Table 20 presents the results of ANOVA type III for the all data from Tables 13 -18, except for DBSCAN. The analysis evidences that all variables and interaction are significant for the model.

Response: Purity	Df	Chisq	Pr(>Chisq)
(Intercept)	1	2245.5688	< 2.2E-16***
Algorithm	2	69.7561	7.123E-16***
Matrix Size	1	166.3418	< 2.2E-16***
Metric	3	8.8679	0.0311*
Number of groups	1	66.2438	3.98E-16***
Algorithm*Matrix size	2	62.3771	2.85E-14***

Table 20 ANOVA type III for all the data

The results now provide evidence for the fact that the choice of a distance metric is significant for the quality of the clustering, as shown in table 20. In addition, the interaction term between the clustering algorithm and the size of the covariance matrices is significant in the results of the purity measure. Hence, no algorithm evidences better purity results if the size of the covariance matrices is not take in account.

Identically, the number of groups or clusters present in the synthetic data used in the simulation studies are significant of the performance for clustering. These findings are directly in line with previous studies discussed in the Section 2.4. The interaction between the size of the covariance matrices and the distance metric is shown in Figure 35. Notice, that the choice of the Euclidean metric derives in lower mean purity.

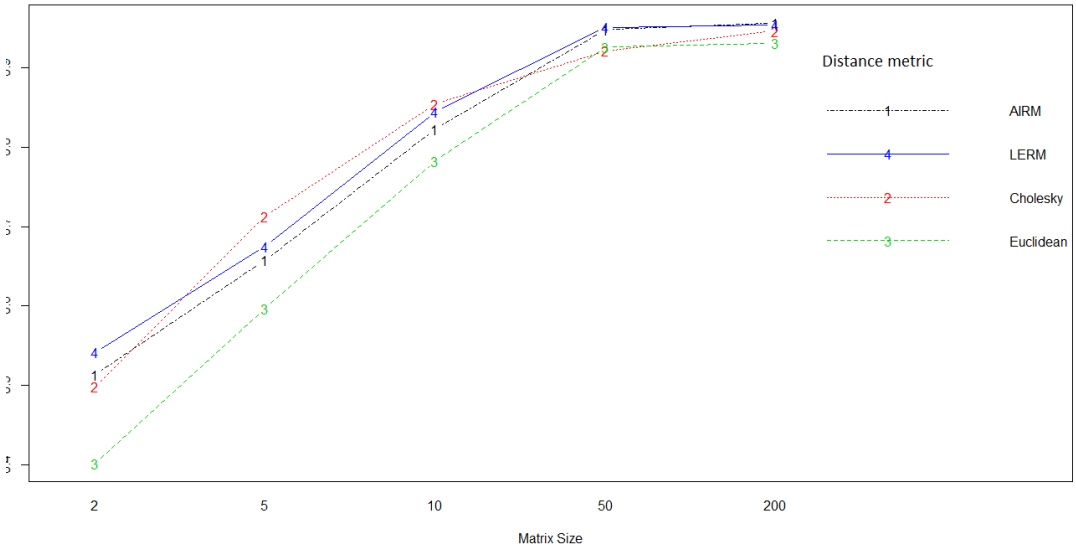


Figure 35 Interaction plot: Matrix size - distance metric for all input parameters

The implications of high covariance matrices are shown in Figure 36, where the K-means algorithm manifests the lowest mean purity for all distance metrics used to calculate the distance between the covariance metrics.

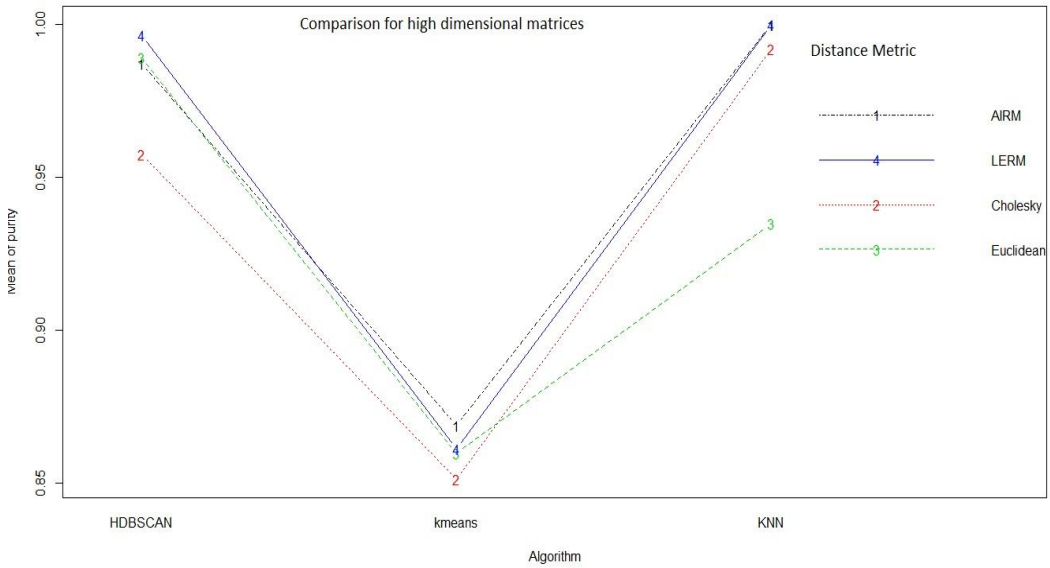


Figure 36 Interaction plot for distance metric and algorithm with high dimensional covariance matrices.

The result of the ANOVA test confirms the importance of the choice of the clustering algorithm for arrays consisting of high dimensional matrices, as shown in table 21. Furthermore, the selection of the distance metric is significant in the performance of the clustering, where the mean purity is lowest when the Cholesky's metric is used for HDBSCAN and for K-means, as shown in Figure 36 and Table 21.

Response: Purity	Df	Chisq	Pr(>Chisq)
(Intercept)	1	4367.8582	< 2.2e-16***
Algorithm	2	57.8413	2.53E-13***
Matrix Size	1	2.7282	0.16714
Metric	3	7.0265	0.07106.
Number of groups	1	59.85	1.02E-14***
Algorithm*Matrix size	2	0.911	0.6339

Table 21 ANOVA test type III for high dimensional covariance matrices

5 REAL DATA PROBLEMS

A real data problem is presented to show the applicability of the algorithms in daily life phenomena.

5.1 Letter Recognition Problem

For this example, provided by (Frey and Slate, 1991), the objective was to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. Table 19 shows the variables used in the study and the first 10 rows of data.

No.	lettr	x.box	y.box	Width	high	onpix	x.bar	y.bar	x2bar	y2bar
1	T	2	8	3	5	1	8	13	0	6
2	I	5	12	3	7	2	10	5	5	4
3	D	4	11	6	8	6	10	6	2	6
4	N	7	11	6	6	3	5	9	4	6
5	G	2	1	3	1	1	8	6	6	6
6	S	4	11	5	8	3	8	8	6	9
7	B	4	2	5	4	4	8	7	6	6
8	A	1	1	3	2	1	8	2	2	2

Table 22 Letter recognition problem

That is, this problem involved the clustering of matrices to identify the groups of letters by their form of writing in the English alphabet. First, the covariance matrices of each letter must me calculated using the 16 variables or attributes.

Using three different algorithms with the AIRM distance, the letters were clustered and the obtained results are shown in Table 23.

No.	**Letter**	**HDBSCAN**		**DBSCAN**		**K - means**	
		m_{pts}=2	m_{pts}=3	eps=8, m_{pts}=2	eps=10, m_{pts}=3	k=7	k=10
1	B	7	2	1	1	7	4
2	R	7	2	1	1	7	4
3	P	6	2	2	1	7	1
4	F	6	2	2	1	7	1
5	C	5	2	3	1	2	3
6	G	5	2	3	1	2	3
7	I	2	2	0	0	3	10
8	J	2	0	0	0	3	10
9	U	1	1	0	2	4	6
10	V	1	1	0	2	4	6
11	M	3	2	4	1	5	2
12	N	3	2	4	1	5	2

13	O	5	2	3	1	2	3
14	E	5	2	0	1	6	8
15	Q	5	2	0	1	2	9
16	T	1	1	0	2	1	7
17	K	4	2	5	1	6	5
18	X	4	2	5	1	6	5
19	Y	1	1	0	2	4	6
20	S	0	2	0	1	6	5
21	H	4	2	5	1	5	2

Table 23 Classification comparison for each clustering algorithm with AIRM

Table 23 shows how the choice of the input parameter significantly modifies the number of clusters and the clustering of the letters. DBSCAN has several problems to cluster I and J, and it labeled them as noise, but it clustered the letters *K*, *X* and *H* in the same group. In addition, K-means presents problems to detect noise and sensitivity to changes of the input parameter k.

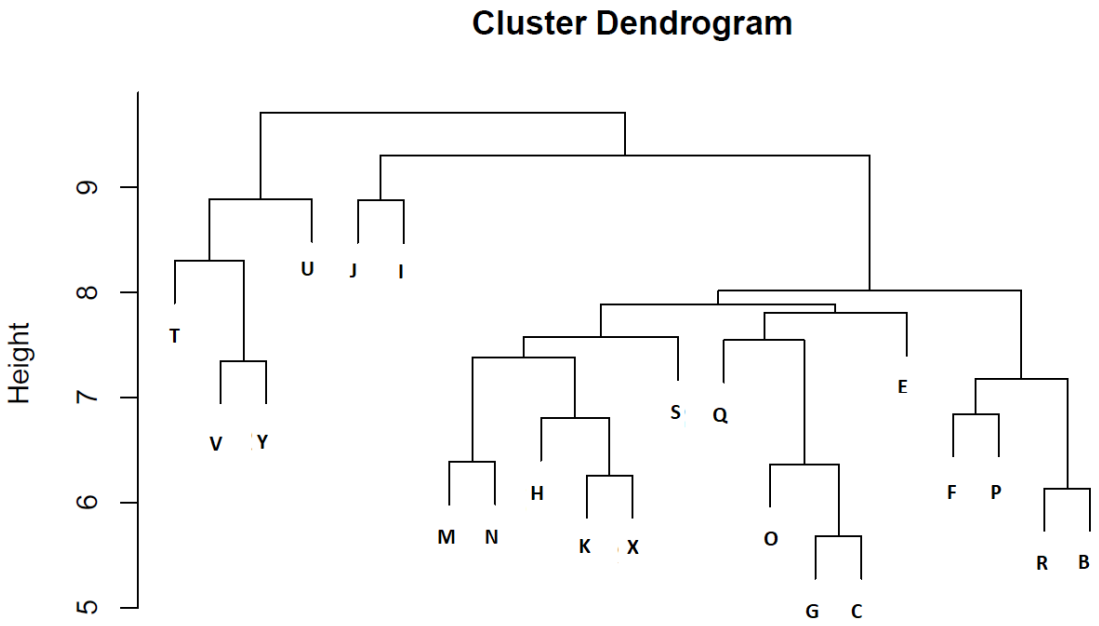
HDBSCAN offers a better performance when grouping the letters by their shape. This is confirmed by observing how the letter S is classified, for example. It was much more consistent to clustering than K-means, given that the true number of clusters is not known. Now, consider if the distance metric is varied and using HDBSCAN algorithm with $m_{pts} = 2$, the results are shown in Table 24.

No.	Letter	AIRM	LERM	Euclidean	Cholesky
1	B	7	7	3	3
2	R	7	7	3	3
3	P	6	6	3	3
4	F	6	6	3	3
5	C	5	5	3	3
6	G	5	5	3	3
7	I	2	2	3	2
8	J	2	2	0	0
9	U	1	1	1	0
10	V	1	1	3	2
11	M	3	3	1	0
12	N	3	3	1	0
13	O	5	5	3	1
14	E	5	5	3	1
15	Q	5	5	3	1
16	T	1	1	2	0
17	K	4	4	3	0
18	X	4	4	3	1
19	Y	1	1	2	0
20	S	0	0	3	1
21	H	4	4	1	0

Table 24 Classification comparison for each distance metric with HDBSCAN

From Table 24, AIRM and LERM metrics offer better clustering than other metrics based in number of clusters and the letters classified to each group. For example, letter C cannot be in the same group than R, as Euclidean and Cholesky metrics predicted.

Figure 37 offers a dendrogram for the recognition problem using HDBSCAN and AIRM metric. Figure 38 shows a cluster tree for this problem, where y axis indicates the ϵ - value chosen to have each form group.



Cluster dendrogram with mpts= 2
hdbscan (*, "robust single")

Figure 37 Dendrogram for HDBSCAN and m_{pts}=2

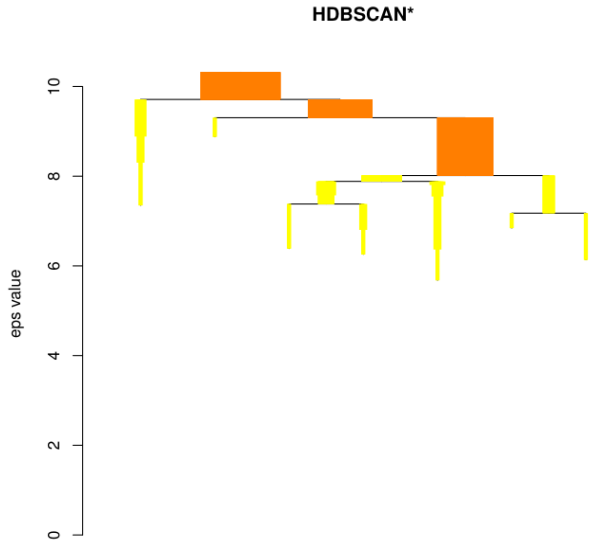


Figure 38 Cluster tree for letter recognition problem

6 CONCLUSIONS

- AIRM and HDBSCAN is the most expensive running time combination for long arrays, but it is worth due to its effectiveness for high dimensional matrices.

- LERM and AIRM metrics have similar purity evaluations in all scenarios, but AIRM exhibits more computational costs for larger arrays, however the LERM metric needs more execution time when the size of the matrices increases. The choice of a distance metric depends of the application and resources available to the analyst.

- K-means and HDBSCAN have comparable results for small number of clusters and high dimension covariance matrices, however when the input parameters vary, HDBSCAN's purity values do not change considerable; k-means suffers when the input parameters are manipulated, this is a very important issue when dealing with real problems. These findings demonstrate that HDBSCAN offers the highest robustness for the four analyzed algorithms, and it is consistent with previous finding for n-dimensional vectors (McInnes et al, 2016).

- Cholesky and AIRM distance metrics present diverse results: Cholesky's distance metric is more effective at low dimensions but AIRM distance is recommended for high dimensional matrices (more than 10×10).

- HDBSCAN is the most effective unsupervised algorithm; K-means is also effective, but the number of clusters must be known in advance, besides K-means should not be used for considerable number of clusters. This is a problem when we are dealing with real problems application as it was illustrated in the letter recognition example.

- When using high dimensional matrices, HDBSCAN is slightly affected by the metric chosen, however DBSCAN yields different results by choosing ϵ with the same m_{pts}, where it is necessary to scan different distance results before selecting the right ϵ value. For example, with DBSCAN when changing from set 1 to set 2 in any number of groups, it was necessary to change the value of the parameter ϵ but it was not necessary with HDBSCAN.

- K-means presents difficulties to cluster multi group datasets, HDBSCAN outperformed it in purity evaluations for 3 different datasets. For matrices with low dimensions there are no noticeable differences between these algorithms. When the dimensions of the covariance matrices dimensions were increased, these differences were remarkable as shown in Figures 34 and 36.

- The crucial aspect in clustering problems is that there is no optimal solution, the analyst should decide or not if the solution is appropriated for the problem.

RECOMMENDATIONS

- Given the considerable execution time the HDBSCAN takes to clustering, the implementing of the new Accelerated HDBSCAN (McInnes et al 2017) in R is recommended. It provides comparable performance to DBSCAN in time but superior results in qualitative clustering.

- Use high performance cluster computers to increase the number of simulations and increase the precision when comparing the performance of the algorithm.

- Future implementations should consider the second-order approximation to AIRM (SOA-AIRM) to reduce execution time while keeping its performance.

- Propose different scenarios to generate synthetic data with multiple groups, for instance, comparing Auto Regressive SPD varying the ρ value only.

REFERENCES

Aggarwal C.C., Hinneburg A., Keim D.A. (2001) On the Surprising Behavior of Distance Metrics in High Dimensional Space. In: Van den Bussche J., Vianu V. (eds) Database Theory — ICDT 2001. ICDT 2001. Lecture Notes in Computer Science, vol 1973. Springer, Berlin, Heidelberg

Barnard, Ben (2017). rWishart: random wishart matrix generation. https://cran.r-project.org/web/packages/rWishart/index.html Last visit: 12/29/2017

Campello, R. Moulavi, D. Zimek, A. Sander, J. (2015) Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. ACM Transactions on Knowledge Discovery from Data,New York : ACM,v. 10, n. 1, p. 5:1-5:51, Jul.2015.

Dang, Shilpa. (2015). Performance Evaluation of Clustering Algorithm Using Different Datasets. IJARCSMS. 3. 167-173.

Dryden, Ian. Koloydenko, Alexey. Zhou, Diwei (2009) Non-euclidean statistics for covariance matrices with application to diffusion tensor imaging. The Annals of statistics Vol. 3 No. 3. 1102-1123.Institute of Mathematicals statisticas. 2009

El-Sonbaty, Y., Ismail, M., and Farouk, M. (2004). An efficient density-based clustering algorithm for large databases. *In 16th IEEE international conference on tools with artificial intelligence*, pages 673 - 677. Boca Raton FL: IEEE Computer Society.

Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *In Second international*

conference on knowledge discovery and data mining, pages 226 - 231. Portland, OR: AAAI Press.

Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *In Second international conference on knowledge discovery and data mining,* pages 226 - 231. Portland, OR: AAAI Press.

Förstner, Wolfgang. Moonen,Boudewijn (2003). A metric on covariance matrices. In Geodesy-The Challenge of the 3rd Millennium, pages 299 $-$ 309. Publisher: Springer Berlin Heidelberg. Online ISBN 978-3-662-05296-9.

Gan, Guojun. Ma, Chaoqun. Wu, Jianhong. (2007) Data clustering: theory, algorithms, and applications. *ASA-SIAM series on statistics and applied probability*. Pages 219 - 220. ISBN: 978-0-898716-23-8.

Grafarend, E. W. (1972): Genauigkeitsmasse geodätischer Netze. DGK A 73, Bayerische Akademie der Wissenschaften, München, 1972.

Genz, Alan. Bretz, Frank. Miwa, Tetsuhisa. Mi, Xuefei (2015). Mvtnorm: multivariate normal and t distributions. https://cran.r-project.org/web/packages/mvtnorm/index.html Last visit 5/30/2018

Hahsler, Michael. Piekenbrock, Matthew. Arya, Sunil. Mount, David (2017). Package 'dbscan'. https://cran.r-project.org/web/packages/dbscan/dbscan.pdf
Last visit: 3/29/2017

Lee, Kyoungjae. Lin, Lizhen. You, Kisung (2018). Statistical tools for covariance analysis. https://cran.r-project.org/web/packages/CovTools/index.html Last visit: 5/30/2018.

Matteucci, Matteo (2014) A tutorial on clustering algorithm. https://home.deib.polimi.it/matteucc/clustering/tutorial_html/ Last visit: 1/24/2018.

McInnes, L.Healy, J.Astels, S.(2016). The HDBSCAN clustering library. http://hdbscan.readthedocs.io/en/latest/index.html Last visit: 3/28/2017.

McInnes, Leland. Healy, John (2017). Accelerated Hierarchical Density Clustering. 2017 IEEE International Conference on Data Mining Workshops (ICDMW). 18-21 Nov. 2017. Page(s):33 – 42. Electronic ISSN: 2375-9259

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12 (2011), 2825-2830.

P. W. Frey and D. J. Slate (1991). "Letter Recognition Using Holland-style Adaptive Classifiers".Machine Learning Vol 6 #2 March 91.

Piekenbrock, Matt. Hahsler, Michael (2016). HDBSCAN with the dbscan package. https://cran.r-project.org/web/packages/dbscan/vignettes/hdbscan.html. Last visit: 9/30/2017.

Ripley, Brian (2015). Functions and classification. https://cran.r-project.org/web/packages/class/index.html Last visit 5/30/2018

Qiu, Weiliang. Joe, Harry (2015). Random cluster generation (with specified degree of separation). https://cran.r-project.org/web/packages/clusterGeneration/index.html. Last visit: 12/30/2017.

Satya, Chaitanya & , Sripada & M Sreenivasa Rao, Dr. (2011). Comparison of purity and entropy of k-means clustering and fuzzy c means clustering. Indian Journal of Computer Science and Engineering. 2.

Taboga, M. (2010) "Lectures on probability and statistics", https://www.statlect.com.

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, Journal of machine learning research. 12, pp. 2825-2830, 2011.

Thanh N. Tran, Klaudia Drab, Michal Daszykowski (2015), "Revised DBSCAN algorithm to cluster data with dense adjacent clusters", Chemometrics and Intelligent Laboratory Systems, 120:9296. DOI: 10.1016/j.chemolab.2012.11.006

Yger, Florian. Lotte, Fabian. Sugiyama, Masashi (2015). Averaging covariance matrices for EGG signal classification based on the CSP*: An empirical study. Signal Processing Conference (EUSIPCO), 2015 23rd European*. Electronic ISSN: 2076-1465

Zhang, Zijun (2012). K- means algorithm (Online). Available: http://user.engineering.uiowa.edu/~ie_155/lecture/K-means.pdf