# Uma avaliação empírica de transformações aleatórias aplicadas a *ensembles* de agrupamento de dados

**Gabriel Damasceno Rodrigues**


Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2018

**Gabriel Damasceno Rodrigues**

# Uma avaliação empírica de transformações aleatórias aplicadas a *ensembles* de agrupamento de dados

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Marcelo Keese Albertini

Uberlândia
2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

R696a
2018

Rodrigues, Gabriel Damasceno, 1993-
Uma avaliação empírica de transformações aleatórias aplicadas a ensembles de agrupamento de dados [recurso eletrônico] / Gabriel Damasceno Rodrigues. - 2018.

Orientador: Marcelo Keese Albertini.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2019.311
Inclui bibliografia.
Inclui ilustrações.

1. Computação. 2. Aprendizado do computador. I. Albertini, Marcelo Keese, 1984- (Orient.) II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Maria Salete de Freitas Pinheiro - CRB6/1262

# Agradecimentos

Agradeço a Deus e minha família por me proporcionar essa oportunidade e me apoiar com carinho, especialmente minha mãe e esposa. Aos professores Márcia, Gina, Elaine e Humberto, meu sincero agradecimento por todo o conhecimento passado durante as disciplinas e fora delas. Também agradeço ao apoio prestado por meus colegas de trabalho no IFTM. Em destaque, um agradecimento ao meu orientador, por me guiar e ensinar imensuráveis coisas ao longo de todos esses anos.

*"Não devemos nos questionar porque algumas coisas nos acontecem,*
*e sim o que podemos fazer com o tempo que nos é dado.*
*(J. R. R. Tolkien)*

# Resumo

O número de técnicas de *ensemble* de agrupamento de dados cresceu nos últimos anos, oferecendo um melhor desempenho médio entre diversos domínios e conjuntos de dados. Benefícios colaterais são encontrar novos agrupamentos inatingíveis por um único algoritmo de agrupamento e também fornecer estabilidade de agrupamento. As principais estratégias de *ensemble* de agrupamento de dados são: combinar resultados de diferentes algoritmos de agrupamento; produzir resultados diferentes por meio de reamostragem dos dados, como nas técnicas de *bagging* e *boosting*; e executar um determinado algoritmo várias vezes com diferentes parâmetros ou inicialização. Muitas vezes, as técnicas de *ensemble* são desenvolvidas para ambientes supervisionados e, posteriormente, adaptadas para ambientes não supervisionados. Recentemente, Blaser e Fryzlewicz propuseram uma técnica de agrupamento para classificação baseada em reamostragem e transformação dos dados de entrada. Especificamente, eles empregaram rotações aleatórias para melhorar significativamente o desempenho de *Random Forests*. Neste trabalho, estudamos empiricamente os efeitos de transformações aleatórias baseadas em matrizes de rotação, distância de Mahalanobis e proximidade usando densidade para melhorar o *ensemble* de agrupamento de dados. Nossos experimentos consideraram 12 conjuntos de dados e 25 variações de transformações aleatórias, considerando então um total de 5100 conjuntos de dados aplicados a 8 algoritmos de agrupamento que foram avaliados por 4 medidas de avaliação. Testes estatísticos identificaram 17 transformações viáveis para serem aplicadas previamente em *ensembles* e em agrupamento de dados comum produzindo consistentemente efeitos positivos na qualidade do agrupamento. Em nossos experimentos, as melhores transformações foram as baseadas em Mahalanobis. Os algoritmos de agrupamento em *ensemble* que mais se beneficiaram com as transformações foram o CBA e o bClust.

**Palavras-chave:** Agrupamento de dados; *Ensembles* de agrupamento de dados; Transformações Aleatórias.

# An empirical evaluation of random transformations applied to ensemble clustering

**Gabriel Damasceno Rodrigues**


Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU
FACULDADE DE COMPUTAÇÃO – FACOM
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO – PPGCO

The undersigned hereby certify they have read and recommend to the PPGCO for acceptance the dissertation entitled **"An empirical evaluation of random transformations applied to ensemble clustering"** submitted by **"Gabriel Damasceno Rodrigues"** as part of the requirements for obtaining the **Master's degree in Computer Science**.

Uberlândia, ___ de _____ de _____

Supervisor: _____
Prof. Dr. Marcelo Keese Albertini
Universidade Federal de Uberlândia

Examining Committee Members:

_____
Prof. Dr. Murillo Guimarães Carneiro
Universidade Federal de Uberlândia

_____
Prof. Dr. Murilo Coelho Naldi
Universidade Federal de São Carlos

# Abstract

The number of ensemble clustering techniques have grown in recent years offering better average performance among domains and datasets. Other expected benefits are to find novelty clustering which are unattainable by any single clustering algorithm and to provide clustering stability, such that the quality is little affected by noise, outliers or sampling variations. The main clustering ensemble strategies are: to combine results of different clustering algorithms; to produce different results by resampling the data, such as in bagging and boosting techniques; and to execute a given algorithm multiple times with different parameters or initialization. Often ensemble techniques are developed for supervised settings and later adapted to the unsupervised setting. Recently, Blaser and Fryzlewicz proposed an ensemble technique to classification based on resampling and transforming input data. Specifically, they employed random rotations to improve significantly Random Forests performance. In this work, we have empirically studied the effects of random transformations based in rotation matrices, Mahalanobis distance and density proximity to improve ensemble clustering. Our experiments considered 12 data sets and 25 variations of random transformations, given a total of 5100 data sets applied to 8 algorithms and evaluated by 4 clustering measures. Statistical tests identified 17 random transformations that are viable to be applied to ensembles and standard clustering algorithms, which had positive effects on cluster quality. In our results, the best performing transforms were Mahalanobis-based transformations. The ensemble algorithms which best profited from these were CBA and bClust.

**Keywords:** Data clustering; Clustering Ensembles; Random transformations.

# List of Figures

# List of Tables

# Acronyms list

**ARI** Adjusted Rand Index

**COP** Constraint-Partitioning

**CBA** Constraint Based Algorithm

**CSPA** Cluster-based Similarity Partitioning Algorithm

**CQ** Cluster Quality measure

**DBSCAN** Density-Based Spatial Clustering of Applications with Noise

**DA** Density Attraction

**DR** Density Repulsion

**DN** Dunn Index

**EM** Expectation–Maximization

**FCM** Fuzzy $c$-means

**HGPA** Hypergraph Partitioning Algorithm

**HCL** Hierarchical Complete Link

**HSL** Hierarchical Single Link

**KCC** $k$-means Consensus Clustering

**MCLA** Meta-Clustering Algorithm

**MST** Minimum Spanning Tree

**M100** Mahalanobis transformation on 100% of features.

**M20** Random selection of 20% of features and transform using Mahalanobis transformation

**M50** Random selection of 50% of features and transform using Mahalanobis transformation

**M80** Random selection of 80% of features and transform using Mahalanobis transformation

**RM100** Mahalanobis transformation and Random Rotations

**RM20** Random selection of 20% of features and transform using Mahalanobis transformation and Random Rotations

**RM50** Random selection of 50% of features and transform using Mahalanobis transformation and Random Rotations

**RM80** Random selection of 80% of features and transform using Mahalanobis transformation and Random Rotations

**N100** Original dataset

**N20** Random selection of 20% of features of original dataset

**N50** Random selection of 50% of features of original dataset

**N80** Random selection of 100% of features of original dataset

**R100** Random Rotations

**R20** Random selection of 20% of features and transform using Random Rotations

**R50** Random selection of 50% of features and transform using Random Rotations

**R80** Random selection of 100% of features and transform using Random Rotations

**SWC** Silhouette Width Criterion

**UM100** Uniform Mahalanobis transformation

**UM20** Random selection of 20% of features and transform using Uniform Mahalanobis transformation

**UM50** Random selection of 50% of features and transform using Uniform Mahalanobis transformation

**UM80** Random selection of 80% of features and transform using Uniform Mahalanobis transformation

**RUM100** Mahalanobis transformation and Random Rotations

**RUM20** Random selection of 20% of features and transform using Uniform Mahalanobis transformation and Random Rotations

**RUM50** Random selection of 50% of features and transform using Uniform Mahalanobis transformation and Random Rotations

**RUM80** Random selection of 80% of features and transform using Uniform Mahalanobis transformation and Random Rotations

**UCI** University of California Irvine

# List of notations

| Notation | Description |
|---|---|
| $n$ | Number of elements |
| $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ | A dataset with $n$ elements |
| $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ | A vector of weights |
| $k$ | Number of clusters |
| $\Pi = \{\pi_1, \ldots, \pi_k\}$ | A set of $k$ partitions |
| $b$ | Number of bootstrap samples |
| $u$ | Number of random initializations |
| $T$ | Maximum number of iterations |
| $m$ | Number of base algorithms |
| $o$ | Number of partitions in the ensemble |
| $d$ | Number of data dimensions |
| $c$ | A cluster label |
| $\mathcal{CQ}$ | A cluster quality criteria |
| $\mathcal{A}$ | A base algorithm |
| $\mathcal{M}$ | An association map between two partitions |
| $\mathcal{R}$ | A constraint set |
| $\mathbf{S}$ | Covariance matrix |

# Contents

I hereby certify that I have obtained all legal permissions from the owner(s) of each third-party copyrighted matter included in my thesis, and that their permissions allow availability such as being deposited in public digital libraries.

Gabriel Damasceno Rodrigues

# Chapter 1

# Introduction

Machine Learning studies the ability of a computer to learn from and make predictions on data. The literature mostly organizes the algorithms of Machine Learning in two settings, supervised and unsupervised. In the supervised setting, each training example has a class label typically provided by a specialist in the dataset area and the supervised techniques learn how to classify a set of examples. The supervised learning is oriented to reduce the error rate by working with its bias and variance. Bias is the effect of assumptions in machine learning algorithms which can coincide with the data patterns and well modeling them or differ from them, and therefore incurring in error. Variance is the amount by which the learning function changes while it is being trained on data. Supervised algorithms are known to produce errors according to their combination of bias and variance (2). Thus, aiming to avoid scenarios of high bias or high variance, studies have proposed to combine several algorithms with distinct capabilities into an ensemble.

There are many successful ensemble techniques for supervised setting. The most well-known techniques of this type are: boosting (3), stacking (4) and bagging with bootstrapping (5). These techniques can generally be described as ways to profit from bias and reduce variance considering different learning algorithms.

Recently, Blaser and Fryzlewicz (6) introduced an ensemble bagging method for classification which improves results obtained from Random Forests by transforming data randomly before the use of base learning techniques. It improves models diversity in the ensemble with minimum impact in computational time. This method is interesting due to these rotations are easy to incorporate on existing algorithms or standard techniques.

For unsupervised environment the idea of profit considering different algorithms is also valid. However, usually, it is not possible to mechanically apply ensemble algorithms from supervised to the clustering domain. Thus, researchers have studied how to convert techniques originally developed for supervised setting to work in an unsupervised setting (7).

The main difference between the supervised learning and unsupervised algorithms is the presence of external information about the classes. In supervised learning, there

is a fixed set of class labels which algorithms use as reference to the learning process. In unsupervised algorithms, clustering labels produced by different algorithms are not directly related and examples do not include class information. Thus, the learning process is typically based on proximity, similarity or density (8).

There are different ways of identifying correspondence of clusters among two or more clustering. This abstract problem is often known as the meta-mapping problem (7). In a simplified scenario, let $F$ and $G$ be two clustering containing $F_k$ and $G_k$ clusters. Meta-mapping functions are 1-to-1 mappings which translate each cluster identifier $i$ in $F$ to an identifier $j$ in $G$:

$$map(i) : \{1, \ldots, F_k\} \rightarrow \{1, \ldots, G_k\}.$$

Implicitly, this problem corresponds to the ensemble learning for clustering, and there are studies to solve the meta-mapping problem: using graph-based, using statistical or information-theoretic methods, and an approach which does not explicitly solve the label correspondence problem (9).

Minaei-Bidgoli et al. (1) presented a taxonomy that shows different ways to develop ensemble clustering algorithms. Figure 1 shows a simplified version of Minaei-Bidgoli's taxonomy.

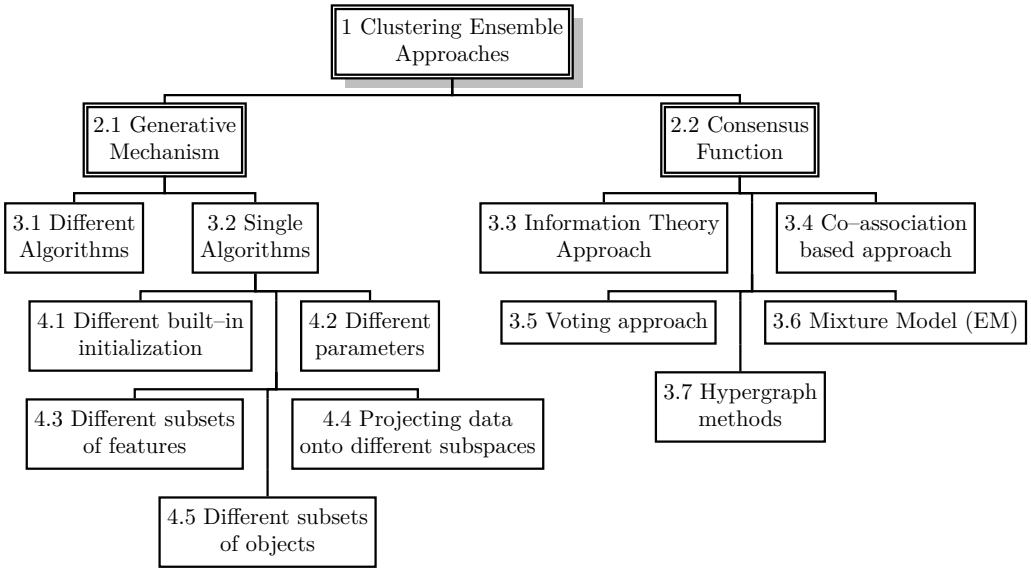


Figure 1 – A simplified taxonomy of ensemble clustering based on work by Minaei-Bidgoli (1)

Examples of ensemble clustering algorithms are bagging of clustering bclust (10), a constraint-rule based algorithm referenced in this dissertation as CBA (11), $k$-means Consensus Clustering (KCC) (12) and hkclustering (13).

Studying the related work about clustering ensembles, we did not find a technique which produces random rotations on input data, similar to random rotations for ran-

dom forest introduced by Blaser and Fryzlewicz (6). Also, we did not find a technique which produces random transformation on input data in general.[1] Thus, we observed an interesting gap of study, since Blaser e Fryzlewicz obtained good results in their work.

Therefore, following the idea of converting a supervised technique into an unsupervised scenario, we adapted the idea of transform input data randomly to current clustering algorithms and investigate the application viability of random data transformations. For this purpose, we studied the effects of using random data transformations applied to data sets on ensemble algorithms and evaluated the results using 4 measures. Our motivation is to slightly transform the data so that the original groups do not get lost.

We used three types of random transformations: rotation-based (6), generalized Mahalanobis-based and density-based. We applied the transformations on 12 different data sets and used as input for 8 clustering algorithms whose source code is freely available.

The algorithms selected were: bclust (10), CBA (11), KCC (12), hkclustering (13), Hierarchical Complete and Single Link (15), $k$-means (16), and DBSCAN (17). The non-ensemble algorithms were included for comparison purposes.

Four evaluating measures were picked: Accuracy, Adjusted Rand Index (18), Dunn Index (19), and Silhouette Width Criterion (SWC) (20). Also, we analyzed our results with Friedman test (21) and Nemenyi's procedure (22) to evaluate their significance.

The experimental results showed that applying 17 of 25 random transformations studied in this work improves clustering results. The best random transformations are the Uniform Mahalanobis-based. The worst random transformations are the density-based. These resulting experiments are reproducible with the randomClustering R package, available in www.github.com and more detailed in Appendix A.

This dissertation chapters are organized as follows. First, we introduce the main ensemble techniques which serve a background to our work in Chapter 2. Then, we discuss about the selected ensemble clustering algorithms used in the experiments in Chapter 3. In Chapter 4, we present our transformations. Chapter 5 presents the experiments details and analysis, followed by our conclusions in Chapter 6.

[1] During the final development of this dissertation, a work about an ensemble clustering framework which produces data transformations, were published by Ronan et al (14).

# Chapter 2

# Background

The number of ensemble clustering techniques have grown significantly in recent years, where various authors proposed ensemble clustering techniques.

Fred and Jain introduced a co-association consensus function to find a combined partition (23, 24). Further, they also proposed how to combine different partitions generated by a $k$-means algorithm using different parameters of initialization.

Topchy et al. propose to use weak clustering components and a new consensus function based in intra-class variance criteria (25, 26).

Strehl and Ghosh (27) made many important contributions regarding clustering ensemble. Three examples are: a study of feature-distributed and object-distributed formulations of clustering ensembles, a study of finding consensus partitions using hypergraph-based algorithms, and a study about combining partitions with a deterministic overlap of examples between data subsets.

The bootstrap method or bagging, initially proposed by Efron (28) as a method of resampling with replacement, has been successfully applied to supervised learning approaches by Breiman (5).

Also, Jain and Moreau have analyzed how to estimate the number of clusters and how to evaluate cluster validity using bootstrap (29). Dudoit and Fridlyand (30), and Ben-Hur et al. (31) also studied cluster validity using resampling methods.

Figure 1 shows a simplified taxonomy of clustering ensemble approaches based in the taxonomy introduced by Minaei-Bidgoli (1). We introduce each approach in the next paragraphs.

Clustering ensemble approaches have two main processes: the generative mechanism (item 2.1), which generates the partitions used in ensemble, and the consensus function (item 2.2), used to combine the multiple partitions received as input resulting in a final partition.

The generative mechanism can use different algorithms like instances of hierarchical and $k$-means or single algorithms, just only k-means instances.

Using different algorithms (item 3.1) (27), intuitively, generates distinct partitions.

Otherwise, using a single algorithm (item 3.2) to get truly distinct partitions only by running the algorithm multiple times is not enough, thus, it uses some approaches like: different built-in initializations (item 4.1) (26), different parameters (item 4.2) (32), different sub-set of features (item 4.3) (27), research planning data onto different subspace (item 4.4) (26, 33) and different subset of objects (item 4.5) (27, 1, 9).

We present five approaches to implement consensus functions: information theory approach (item 3.3), co-association based approach (item 3.4), voting approach (item 3.5), mixture models using Expectation-Maximization (EM)(item 3.6) and Hypergraphs approaches (item 3.7).

The information theory approach (item 3.3) uses the mutual information between the partitions, like Quadratic Mutual Information, to resolve the meta-mapping problems (26). The co-association based approach (item 3.4) provides a co-association matrix between the partitions to a similarity based algorithm, like the hierarchical algorithms (Single, Complete, Average Link) (1, 32).

Consensus function by voting approach (item 3.5) collects votes of which cluster each partition assign an element. After computing all votes, it generates the final partition by assigning each element to the most voted cluster for the element (25).

The mixture model (EM)(item 3.6) creates the consensus partition based on a probability model in the space of contributing clusters and by maximizing a log-likelihood objective (34).

Hypergraph approaches (item 3.7) represent clusters in an ensemble as hyperedges on a graph, where each hyperedge describes a set of elements that belong to the same cluster. The clusters of consensus partition are the connected components after cutting the graph (35).

Some works have applied the theoretical framework of supervised ensembles to unsupervised ensembles, as examples are Boost-clustering algorithm (36) and Ensemble semi-supervised (37). They show the viability and the improvements of ensemble clusterings using approaches initially designed to supervised learning.

Boost-clustering method proposed by Frossyniotis et al.(36) is an iterative multiple clustering approach which uses simple clustering algorithms as weak learners, then it aggregates the resulting partitions using weighted voting to produce final clustering. According to the authors, the method is promising and leads to better clustering results when compared to solutions obtained from the base algorithms. However, its source code is not available.

Another example is the evidence accumulation method proposed by Fred et al. (23), which uses evidence accumulation of multiple running of a $k$-means algorithm.

In the following sections, we detail the main ensemble techniques used in supervised and unsupervised setting.

## 2.1 Ensemble strategies for supervised learning

This section details the following techniques: Boosting, Bagging with bootstrap, Stacking, and Random rotations. These are the main ensemble techniques used in supervised learning, except Random rotations, which are included because of its relevance on this dissertation's objectives.

### 2.1.1 Boosting

Boosting is a method used in machine learning ensemble algorithms (3) which aims to convert a weak learner to a strong one. A weak learner is a classifier which slightly improves the correlation with the true classification when compared to a random guessing algorithm. In other hand, a strong learner classifies arbitrarily well when compared to the true classification.

Typically, the technique trains a set of learners sequentially and combines them into a strong ensemble classifier, in which the later learners focus on the mistakes of the earlier learners. It weights the dataset examples according to accuracy of the weak learner recently added on each iteration. These weights work to inform which examples have to receive more focus in the training process. If a weak learner misclassifies an example, the learner increases this element weight to sample it with more probability in the next iteration and also increasing the chance of being classified correctly.

The boosting techniques has been well succeeded to take full advantage of the weak learners. Schapire and Freund introduced AdaBoost (38), one of the first boosting algorithms. AdaBoost is an adaptive boosting algorithm that won the prestigious Gödel Prize. The algorithm has two phases: training phase, described in Algorithm 1, and classification phase.

The training phase initializes the weights of each element of data set with a probability $1/n$, where $n$ is the number of elements. Then, the algorithm adds a base learning algorithm $\mathcal{A}$ applied on $\mathbf{X}$ and the hypothesis learned $h_t$ to the ensemble set $\mathcal{H}$. Also it calculates the error $\varepsilon_t$ which is the total sum weight of the incorrectly classified examples. Adaboost stops if $\varepsilon_t$ is higher than 0.5, otherwise, it updates the weights using $\beta_t$ for next iteration $t + 1$. The classification phase defines the class of an example $d$ by computing the vote of each hypothesis in $\mathcal{H}$ and then assigns the most voted class to $d$.

### 2.1.2 Bagging with bootstrap method

In machine learning context, bootstrapping (5), also called bagging, aims to improve stability and accuracy of machine learning algorithms. Given a training set $\mathbf{X}$ of size $n$, we generate $q$ new training sets $\mathbf{X}_i$, sampled with replacement and with size $n$. Then,

**Algorithm 1** AdaBoost Training

**Require:** Dataset $\mathbf{X}$ with $n$ examples
A base learning algorithm $\mathcal{A}$
The maximum number of iterations $T$
**Ensure:** Hypothesis set $\mathcal{H}$
Initialize weights $\mathbf{w} = \{1/n, \ldots, 1/n\}$ of each element in $\mathbf{X}$
Let $\mathcal{H}$ the empty set of classification
**for** $t$ from 1 to $T$ **do**
  Learn a hypothesis $h_t$ from the weighted examples: $h_t = \mathcal{A}(\mathbf{X})$
  Add $h_t$ to $\mathcal{H}$
  Calculate the error $\varepsilon_t$ of the hypothesis $h_t$
  **if** $\varepsilon_t > 0.5$ **then**
    **return** $\mathcal{H}$
  **end if**
  Let $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
  Multiply the weights of the examples that $h_t$ classifies correctly by $\beta_t$
  Rescale the weights of all examples
**end for**
**return** $\mathcal{H}$

the algorithm creates $b$ models for each resample $\mathbf{X}_i$ and the final model is these models combination, by averaging the output for regression or voting for classification.

The main advantage of using bagging is the prediction accuracy improvement. However bagging bad classifiers can degrade their accuracy even further and produce loss of interpretability. In the case of loss of interpretability in a classification tree, the final bagged classifier is not a tree, thus the model is not a white box anymore.

A bagging algorithm have a training phase and a classification phase. The training phase, described in Algorithm 2, generates $b$ samples from the original data set $\mathbf{X}$ drawn with replacement and builds $b$ classifiers for each resample of the training set. The algorithm adds the classifiers to the final ensemble $\mathcal{E}$, and in classification phase, it chooses the class with the greatest number of votes from each classifier in $\mathcal{E}$ for example $d$.

**Algorithm 2** Bagging algorithm for training

**Require:** Data set $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$
$b$ classifiers to train
**Ensure:** The ensemble of classifiers $\mathcal{E}$
**for** $i = 1, \ldots, b$ **do**
  Generate a bootstrap sample $\mathbf{X}_i$ of size $n$ from $\mathbf{X}$ by sampling with replacement
  Build a classifier $\mathcal{E}_i$ using $\mathbf{X}_i$ as a training set
  Add the classifier $\mathcal{E}_i$ to the current ensemble, $\mathcal{E} = \mathcal{E} \cup \mathcal{E}_i$
**end for**
**return** $\mathcal{E}$

### 2.1.3 Stacking

Stacking is a method to combine learning models which introduces the meta-learning approach (4). Differently from bagging and boosting, stacking is mostly used to combine models of different types generated by different algorithms like Naive Bayes learner or decisions trees.

A meta-learning approach proposed by Witten et. al (4) learns which classifiers in an ensemble are reliable and discovers the best way to combine the base learners models using a different algorithm from the base ones. A meta-model, denoted by *level-1 model* receives as input the base learner model decisions, denoted by *level-0 models*, and combine them to produce the final prediction. They defined a *level-1 model* instance as $\mathbf{x_i} = \{x_{i,1}, \ldots, x_{i,m}\}$ for $m$ base learners, where attribute value $x_{i,m}$ is the class designed to instance $i$ by base learner $m$.

The method transforms *level-0* to training data into *level-1* training data using holdout or cross-validation methods. The use of same instances for training in both levels produces a preference to overfitting classifiers for the meta-learner (4).

Holdout method reserve some instances to compose the training set of *level-1* learner and the compose the training set of *level-0* with remaining instances. The method generates the classifiers of *level-0* and uses them to classify the hold outed instances, thus, the predictions of *level-0* are unbiased. The procedure transforms the *level-0* training data into *level-1* training data. Then, it reapplies the *level-0* learners on the full training set to provide data *level-1* combination.

The other approach uses cross-validation method in every *level-0* learner, making each instance in the training set occur in exactly one over the cross-validation folds. Thus, cross-validation generates a training instance of *level-1* for each *level-0* instance. Therefore, cross-validation uses all instances as training set, differently from the holdout procedure, which uses some instances to produce training data for *level-1*.

### 2.1.4 Random rotations

Transformed data is the key to diversity of an ensemble (6). Blaser and Fryzlewicz propose to use random rotations of feature space before applying base learners in the supervised context. They proposed a method to apply random rotations on data by multiplying a random matrix to data matrix. A rotation matrix $R$ is an orthogonal square matrix, $d \times d$, with a unit determinant as defined in Equation 1 and $n$ representing the feature space number of elements.

$$R^T = R^{-1} \text{ and } |R| = 1 \tag{1}$$

The sets of these matrices form two groups, a special orthogonal group $SO(n)$ with determinant $|R| = 1$ and an orthogonal subgroup $O_{sub}(d)$ with determinant $|R| = d, d \in$

$\{-1, 1\}$.

The approach performs the random rotations by sampling over all possible rotations using the Haar measure (39), which is a uniform distribution of the $O_{sub}(d)$ group. The random matrix $R$ will be uniformly distributed if $P(R \in U) = P(R \in \Gamma U)$, $\forall\, U \subset O_{sub}(d)$ and $\Gamma \subset O_{sub}(d)$. Using Haar measure over $O_{sub}(d)$, there are many algorithms to generate these random orthogonal matrices (6).

Their method applies a Householder $QR$ decomposition (40) to get a factorization $A = QR$, with square matrix $A_{d\times d}$, orthogonal matrix $Q$ and upper triangular matrix $R$ with positive diagonal elements. It composes the matrix $A$ of independent univariate standard normal random variates. The resulting matrix $Q$ can have a positive or negative determinant. If $Q$ determinant is negative, it is not a valid rotation matrix as described in Equation 1. In this case, it flips the sign of a random column vectors of $A$ to get $A^+$, and consequently obtains a valid rotation matrix $Q^+$ by repeating the decomposition.

Incorporating this approach on existing algorithms or standard techniques provides a diversity improvement with minimum computational impact on the base learners classification performances. It has shown effectiveness for base learners which have axis parallel decision boundaries, like most of tree based algorithms. Also, in axis-aligned methods for image processing, it can use this method to ensure axis-independent approach.

## 2.2 Ensemble clustering

This section details the following techniques: Boosting-based clustering, Data Clustering Using Evidence Accumulation, Mixture Model for Clustering Ensembles, a framework for combining multiple partitions, Adaptive Clustering Ensembles, and Combining Multiple Weak Clusterings. These are the main ensemble techniques used in unsupervised setting, and some of them applies techniques originally developed for supervised learning.

### 2.2.1 A clustering method based on boosting

Frossyniotis et al. (36) had proposed an iterative multiple clustering approach called boost-clustering. This approach, described in Algorithm 3, interactively computes a new distribution over the training points and creates a new training set by sampling data from the original dataset.

Then, Algorithm 3 applies a base clustering algorithm on this new training set, producing a partition to the ensemble. It uses these partitions to produce the final clustering by aggregating using weighted voting.

Similar to AdaBoost, it samples each element with same probability, $\frac{1}{n}$. For each iteration $t$ the algorithm generates a bootstrap sample $\mathbf{X}^t$ of original dataset $\mathbf{X}$ according of each element probability in iteration $t$. The algorithm runs a base clustering algorithm like $k$-means on $\mathbf{X}^t$ to get the partition $H^t$.

**Algorithm 3** Boost-clustering algorithm

**Require:** Dataset $\mathbf{X} = \{x_1, \ldots, x_n\}$;
A base clustering algorithm $\mathcal{A}$;
A number $k$ of clusters;
The maximum number of iterations $T$.
**Ensure:** The number of iterations $T_f$;
The final cluster hypothesis $H_{ag}^t$.

Initialize $p_i^1 = \frac{1}{n}$ for $i = 1, \ldots, n$.
Set $t = 1$, $\varepsilon_{max} = 0$ and $ic = 0$.
**while** $t \leq T$ **do**
  $\mathbf{X}^t \leftarrow$ A bootstrap sample of $\mathbf{X}$ according with weight probabilities $w$
  $H^t \leftarrow \mathcal{A}(\mathbf{X}^t)$
  Get the cluster hypothesis $H_i^t = (h_{i,1}^t, h_{i,2}^t, \ldots, h_{i,k}^t)$ for all $i = 1, \ldots, n$, where $h_{i,j}$ is the membership degree of instance $i$ to cluster $j$.
  If $t > 1$, renumber the cluster indexes of $H^t$ according to $H_{ag}^{t-1}$ .
  Calculate the pseudoloss $\varepsilon_t$
  $\beta_t \leftarrow \frac{1-\varepsilon_t}{\varepsilon_t}$.
  # Stopping criteria
  **if** $\varepsilon_t > 0.5$ **then**
    $T' = t - 1$
    break
  **end if**
  **if** $\varepsilon_t < \varepsilon_{max}$ **then**
    $ic = ic + 1$
    **if** $ic = 3$ **then**
      $T_f = t$
      break
    **else**
      $ic = 0$
      $\varepsilon_{max} = \varepsilon_t$
    **end if**
  **end if**
  Update weight distribution $\mathbf{w}$
  Compute the aggregate cluster hypothesis $H_{ag}^T$ using Equation 6
  $t \leftarrow t + 1$
**end while**
**return** $T_f$ and $H_{ag}^T$

With the partition $H^t$ is possible to get the cluster hypothesis $H_i^t$, which is a set of membership degree of each element regarding the $k$ clusters. Thus, the boost-clustering method is non-parametric, representing the final partition in terms of membership degree instead of cluster centers, which gives the flexibility to define arbitrarily shaped data partitions.

After $t > 1$, the algorithm renumbers the cluster index due to meta-mapping problem. It renumbers the clusters in $H^t$ following the highest matching score of a correspondence function, which identifies the common patterns between $H_{ag}^{t-1}$ and $H^t$, where $H_{ag}^{t-1}$ is the consensus hypothesis until iteration $t-1$.

The next step in Algorithm 3 calculates the pseudoloss $\varepsilon_t$ using Equation 2, verifying if the loss increased with the last added sample to ensemble, where $w_i^t$ is the weight of element $i$ in iteration $t$ and $n$ is the number of examples. AdaBoost uses the cluster quality criteria $\mathcal{CQ}_i^t$ to calculate pseudoloss $e_t$ for each instance by two different ways.

$$\varepsilon_t = \frac{1}{2}\sum_{i=1}^{n} w_i^t \mathcal{CQ}_i^t \tag{2}$$

The first one, minmax-$\mathcal{CQ}$ defined in Equation 3, computes a relation between the maximum membership degree of $d_i$ ($h_{i,good}^t$) and the minimum membership degree of $d_i$ ($h_{i,bad}^t$). Where $h_{i,j}^t$ the membership degree of element $i$ to cluster $j$, for visualization purposes we represented the element $d_i$ just by the index $i$.

$$\text{minmax-}\mathcal{CQ}_i^t = 1 - h_{i,good}^t + h_{i,bad}^t \tag{3}$$

In second one, entropy-$\mathcal{CQ}$ is based on entropy and defined in Equation 4. entropy-$\mathcal{CQ}_i^t$ has high values when the membership degree $h_{i,j}^t$ of an example $i$ has not been well-clustered.

$$\text{entropy-}\mathcal{CQ}_i^t = -\sum_{j=1}^{C} h_{i,j}^t \log(h_{i,j}^t) \tag{4}$$

If the algorithm does not reach stopping criteria, update the probabilities for distribution $W^{t+1}$ regarding the pseudoloss and a normalized factor $Z_t$, as defined in Equation 5. $W^{t+1}$ is a distribution due to $\sum_{i=1}^{n} w_i^{t+1} = 1$.

$$w_i^{t+1} = \frac{w_i^t \beta^{\mathcal{CQ}_i^t}}{Z_t} \tag{5}$$

Equation 6 calculates the final aggregate cluster hypothesis $H_{ag}^T$, where $\beta = \{\beta_1, \ldots, \beta_t\}$ is a weight update and $k$ is the number of clusters.

$$H_{ag}^T = \arg\max_{1,\ldots,k} \sum_{\tau=1}^{T} \left[ \frac{\log(\beta_\tau)}{\sum_{j=1}^{t} \log(\beta_j)} h_{i,k}^\tau \right] \tag{6}$$

The boost-clustering with minmax-$\mathcal{CQ}$ and with entropy-$\mathcal{CQ}$ have been evaluated using $Z$-score and compared over four datasets with the simple $k$-means (16) and Fuzzy $c$-means

(FCM) (41). Overall, the boost-clustering outperformed Lloyd's $k$-means algorithm in 10 out of 16 experiments, using either of cluster quality criteria. When compared with FCM algorithm, boost-clustering with minmax-$\mathcal{CQ}$ outperforms in 13 out of 16 experiments, and entropy-$\mathcal{CQ}$ in 11 out of 16.

### 2.2.2 Data Clustering Using Evidence Accumulation

Fred et al. propose to use multiple runnings of well-known $k$-means algorithm with evidence accumulation (23). First, they split data into a large number of compact and small clusters and then map multiple clusterings into a co-association matrix which provides a measure of similarity between patterns. The algorithm obtains the final data partition by clustering this new similarity matrix, corresponding to the merging of clusters.

The proposed strategy follows a split-and-merge approach. The split phase uses $u$ random initialization of $k$-means to produce various partitions. In the combine phase they proposed a voting mechanism, which composes a new similarity measure between the patterns. This measure, $co\_assoc(i, j)$, is a co-association matrix $n \times n$, defined by the number of pairs of objects $(i, j)$ placed in the same cluster in different clusterings, divided by the number $n$ of clusterings. In the merge phase, a minimum spanning tree (MST) algorithm cuts weak links at a threshold $t$ and produces the final clustering aiming to recover natural clusters. The number of $k$ small clusters must be bigger than the expected number of clusters, typically, $k$=$\sqrt{n}$ and threshold $t = 0.5$.

They execute the experiments using the following datasets: spiral, half-rings and iris (42), random data and Gaussian data with varying cluster separability and based on 200 $k$-means clusterings combination to present the results.

The main advantage of this strategy is to identify well separated, arbitrarily shaped clusters. Nevertheless, when clusters overlapping occurs, the strategy performs poorly.

### 2.2.3 A Mixture Model for Clustering Ensembles

According to the taxonomy in Figure 1, Topchy et al. (9) work is a prototypical example of item 3.6 which focuses on the primary problem of clustering ensembles, namely the consensus function (meta-mapping problem), to create the combined clustering. They proposed a new fusion method for these kinds of unsupervised decisions based on a probability model of the consensus partition in the space of contributing clusters.

The approach finds the consensus partition as a solution to the maximum likelihood problem for a given clustering ensemble. Also, it optimizes the likelihood function of an ensemble with respect to the parameters of a finite mixture distribution. Each component in this distribution corresponds to a cluster in the target consensus partition, and the approach assumes that is a multivariate, multinomial distribution.

Given a set $\mathbf{X} = \{x_1, \ldots, x_n\}$ with $n$ data points and a set of partitions $\Pi = \{\pi_1, \ldots, \pi_k\}$ with $k$ partitions of $\mathbf{X}$. Each partition returns a different set of labels for each point. Thus, the consensus function uses the notation $y_{ij} = \pi_j(x_i)$, or for simplicity, $y_i = \pi(x_i)$ denoting the label assigned to point $x_i$ by the $j$-th algorithm. The consensus function aims to find a partition $\pi_C$ which summarizes the partitions in $\Pi$ information. The consensus function assumes that labels $y_i$ are modeled as random variables drawn from a probability distribution, which is a multivariate component of densities mixture as described in Equation 7, where $\alpha_m$ is a mixing coefficients which corresponds to the prior probabilities of the clusters, $\theta_m$ is a parameter for each component and $\Theta$ is a set of $\theta_m$ parameters. The approach generates each data points $y_i$ by drawing a component using the mass function probability $\alpha_m$ and sampling a point from the distribution $P_m(y_i|\theta_m)$, assuming that the data $Y = \{y_i\}, i \in \{1, \ldots, n\}$ is independent and identically distributed.

The consensus function objective is to maximize the log-likelihood of distribution parametrized by $\Theta$ according to data $Y$ defined in Equation 8.

$$P(y_i|\Theta) = \sum_{m=1}^{M} \alpha_m P_m(y_i|\theta_m) \tag{7}$$

$$\log L(\Theta|Y) = \log \prod_{m=1}^{N} P(y_i|\Theta) \tag{8}$$

The Expectation-Maximization (EM) algorithm (34) solves this maximum likelihood problem.

The main advantage of this approach is that it avoids solving the label correspondence problem. Another advantage is the relatively low computational complexity of the EM consensus function $O(kno)$ for $k$ clusters in the target partition, $n$ patterns, and $o$ clusterings in the ensemble. Topchy et al. (9) states this results in fast convergence that is comparable to the $k$-means algorithm.

### 2.2.4 Cluster ensembles, a knowledge reuse framework for combining multiple partitions

Meta-mapping problem is central to clustering ensembles. In order to attack this problem, Strehl and Ghosh introduced three techniques to obtain consensus functions to combine partitions (35) by transforming partitions into hypergraph representations.

They formulated the hypergraph-based method as a solution to the $k$-way min-cut hypergraph partitioning problem. This method represents all the clusters in the ensemble as hyperedges on a graph with $n$ vertices, in which, each hyperedge describes a set of elements that belong to the same cluster. The clusters of consensus partition are the

connected components after the cut. The main algorithms which use this method are CSPA, HGPA and MCLA (35).

The first technique is the Cluster-based Similarity Partitioning Algorithm (CSPA), which establishes a pair-wise similarity measure based in the relationship between elements in the same cluster. CSPA creates a $n \times n$ binary similarity matrix to each partition, where $n$ is the number of elements. The matrix $S$ represents the fraction of partitions which two elements belongs to same cluster. The algorithm computes $S$ in one sparse matrix multiplication. Then, it uses a similarity-based clustering algorithm which uses the matrix $S$ to recluster the elements. CSPA is a simple algorithm and has high computational and storage costs.

HyperGraph Partitioning Algorithm (HGPA) represents clusters as hyperedges of a hypergraph and approximates the maximum mutual information objective with a constrained minimum cut objective. HGPA partitions the hypergraph by cutting a minimal number of hyperedges and obtaining a set of unconnected components with approximately same size.

Similarly to HGPA, Meta-CLustering Algorithm (MCLA) also represents clusters by hyperedges and the main idea is to group and collapse these hyperedges. It assigns each element to the collapsed hyperedge which it highly participates.

### 2.2.5 Combining Multiple Weak Clusterings

Topchy et al. defined a weak clustering algorithm as a clustering algorithm which produces slightly better partitions than a random algorithm (26). They proposed two algorithms to generate weak partitions to clustering combination. The main motivation is that the synergy of multiple weak partitions combined possibly finds better partitions than a single algorithm partition (26). This motivation is akin to supervised boosting.

The first algorithm splits data points using random hyperplanes which cut data into $r+1$ clusters, where $r$ is the number of desired hyperplanes. It generates each hyperplane by sampling an origin random point in multidimensional region and randomly choosing a normal vector $\mathbf{r}$ to define the hyperplane. Two objects, represented by vectors $\mathbf{p}$ and $\mathbf{q}$, will be in the same cluster if the scalar product $(\mathbf{rp})(\mathbf{rq}) > 0$ and in different cluster if $(\mathbf{rp})(\mathbf{rq}) < 0$. Considering multiples hyperplanes, the probability of two objects belong to the same cluster is the product of probabilities of two objects be split by each hyperplane. Thus, the algorithm generates multiple partitions by clustering data using random hyperplanes, and obtains the final clustering using a consensus function.

The second algorithm generates random subspaces by projecting sample data in a 1-dimensional space, a random line. The projection cost is $O(nd)$, where $n$ is the number of objects and $d$ the number of dimensions, which is very cheap. Then, to clustering data, the algorithm uses a fast and simple clustering algorithm like $k$-means, and combines the generated partitions into a final partition by also using a consensus function. Note that

in this case, the algorithm uses predefined number $k$ of clusters related to the $k$-means parameter. If chosen $k$ is too high or too low, the capture of true structure of data is difficulty.

### 2.2.6 Adaptive Clustering Ensembles

Topchy et al. (43) proposed an adaptive approach that uses an algorithm multiple times. The algorithm computes the clustering consistency (or points) in each run. The history of clustering used to evaluate the data points and then build the ensemble clustering. This method is adaptive due to its use of consistency to weigh the clustering history.

The goal of adaptation is to improve confidence in cluster assignments by concentrating sampling distribution on problematic regions of the feature space.

Their approach extends the studies of ensembles which generate partitions via data resampling, where small subsamples can represent entire data set structure and reduce computational complexity (43).

The adaptive partition generation mechanism aims to reduce inter-class decision boundaries variance. Unlike the regular bootstrap method that draws subsamples uniformly from a given data set, adaptive sampling favors points from regions close to the decision boundaries.

At the same time, it samples the points located far from the boundary regions less frequently. Therefore, the technique updates the probability to sample a data point when adds a clustering to ensemble using a consistency index. It also updates the consistency index when adds a new partition to ensemble.

The experiments used $k$-means and four different consensus functions (9): mutual information and the hypergraph-based methods HGPA, MCLA and CSPA (35), briefly described in Section 2.2.4.

The mutual information method uses the shared information between the empirical target and given partitions, that is, consensus partitions and ensemble partitions. Assuming these partitions are independent, the algorithm calculates the sum of pair-wise of the shared information between the partitions.

They generated partitions with $n$ size in their experiments, drawn with replacement, and repeated each experiment 20 times showing average values of errors. Results show the clustering error decreased by $1 - 5\%$ using the proposed sampling adaptation. Accuracy improvement depends on the number of clusters in the ensemble partitions. They observed the most significant progress when the algorithm combines from 25 to 75 partitions.

## 2.3 Concluding remarks

In this chapter we summarized the main works for ensemble learning and ensemble clustering. For ensemble clustering we also introduced the meta-mapping problem and the main approaches and techniques to produce an ensemble, solving the meta-mapping problem or avoiding it.

We observed which ensemble methods, boosting, bagging, and stacking, were successfully applied to unsupervised setting. This achievement leads us to study random transformations, originally supervised techniques, to apply in ensemble clustering algorithms (6).

Chapter 3

# Selected clustering ensemble algorithms

In the literature we found ensemble clustering algorithms which use different approaches of ensemble clustering taxonomy in Figure 1. We selected algorithms from different items of taxonomy whose source code were available or whose implementation is straightforward. In this chapter we detail these algorithms used in our experiments.

We selected the following ensemble algorithms for our experiments: Constraint Based Algorithm (CBA) (11), bClust (10), KCC (12) and hkclustering (13). We chose these algorithms because they provide different ensemble approaches: CBA is rule-based; bClust is bagging-based; KCC is $k$-means stacking-based; and hkclustering is based in hierarchical aggregation by stacking.

We also included non-ensemble clustering algorithms, described below, to serve as baseline to ensemble algorithms. The non-ensemble algorithms are: Hierarchical Complete Link (15), Hierarchical Single Link (15), $k$-means (16) and DBSCAN (17).

## 3.1 Constraint-Based Algorithm

CBA algorithm is a constraint-based algorithm which uses a modified version of COP-$k$-means to produce the final clustering (11). The main idea of CBA is to produce a set of must-belonging constraints to restrict elements to movement among the clusters in reassignment phase of modified COP-$k$-means. CBA represents the items 3.1 of taxonomy in Figure 1.

The algorithm has three main phases. Algorithm 4 describes the first phase, which resolves meta-mapping problem by creating an association map. It computes an association map $\mathcal{M}$ between clusters from clustering $\pi_1$ and $\pi_2$.

The second phase defines a must-belonging constraint set using Algorithm 5. Observing the agreements between algorithms, this step assembles a constraint map $\mathcal{R}[x] = c_i$ as follows: marks an instance $x$ with a constraint to *must-belong* to cluster $c_i$. Only if clusterings $\pi_1$ and $\pi_2$ put $d$ into cluster $c_i$ the algorithm defines the constraint rule. Subsequently, the algorithm sends each instance not yet marked, denoted by $y$, to a reallocation process.

The reallocation process observes the constraints of the nearest instances regarding each instance not marked, defined on $\mathcal{RM}$ map. This process resembles the kNN algorithm. It verifies if there is a consensus on these constraints by calculating the normalized entropy of the nearest constraints to separation threshold $h$, which is a parameter of the proposed algorithm[1]. Equation 9 defines a cluster constraint probability $p_i$ in $\mathcal{RM}$, where $\mathcal{RM}$ is a set of nearest elements constraints, $nc$ is the quantity of constraints of cluster id $i$. Also, Equation 10 defines the normalized entropy, where $dc$ is the number of distinct constraints of $\mathcal{RM}$. The algorithm obtains the final constraint set $\mathcal{R}$ after the reallocation process.

$$p_i = \frac{t}{\|\mathcal{RM}\|} \tag{9}$$

$$entropy(\mathcal{RM}) = \frac{-\sum_{i \in s} p_i \log p_i}{dc} \tag{10}$$

Finally, the approach uses a modified version of COP-$k$-means, Algorithm 6, to cluster data respecting only must-belong constraints. The time complexity of CBA algorithm is $O(nk^2Tds^2)$, where $n$ is the number of elements, $k$ is the number of clusters, $T$ is the number of iterations to $k$-means convergence, $d$ is the number of dimensions and $s$ is the number of elements without constraints.

**Algorithm 4** Association phase algorithm

**Require:** clustering $\pi_1$ and $\pi_2$, each with $k$ clusters
**Ensure:** map $\mathcal{M}$ of agreement associations among input clusterings
1: initialize $\mathcal{M} \leftarrow \emptyset$
   $J \leftarrow Jaccard(\pi_1, \pi_2)$
2: **for** $i$ in 1 to $k$ **do**
3: $(p, q) \leftarrow$ position of $J$ with highest value
4: $\mathcal{M}[q] = p$ {cluster id $q$ is associated to a cluster id $p$ on map $\mathcal{M}$}
5: $J[q, p] = \infty$
6: **end for**
7: **return** association map $\mathcal{M}$

## 3.2 bClust

bClust (10)[2] represents item 4.5 of taxonomy and uses the bootstrap approach producing multiple partitions and then combines them using a hierarchical algorithm. Algorithm 7, which describes bClust, first constructs $b$ bootstrap samples by drawing with replacement from the original dataset $\mathbf{X}$. Then a base clustering method $\mathcal{A}$ groups each sample producing $b \times k$ centers $c_{11}, c_{12}, \ldots, c_{1k}, \ldots, c_{bk}$ where $k$ is the number of centers used in the base method and $c_{ij}$ is the $j$-th center found using $\mathbf{X}_n^i$. Then it combines all produced

[1] Experiments indicate the algorithm is stable regarding the variation in the value of $h$.
[2] This technique is available in the `e1071` package for R.

**Algorithm 5** Constraint phase algorithm

**Require:** a map $\mathcal{M}$, clustering $\pi_1$ and $\pi_2$ and a distance matrix $dm$
**Ensure:** constraint map $\mathcal{R}$

initialize $\mathcal{R} \leftarrow \emptyset$ // every instance starts without constraint
**for** each instance $x$ in $\pi_1$ and $\pi_2$ **do**
  let $i$ be the cluster id to which $x$ belongs in $\pi_1$
  let $j$ be the cluster id to which $x$ belongs in $\pi_2$
  **if** i = $\mathcal{M}[j]$ **then**
    $\mathcal{R}[x] \leftarrow i$ // instance $x$ receives a constraint to be in cluster $c_i$
  **end if**
**end for**
let $s$ be the number of constraints in $\mathcal{R}$ equals to 0
**for** each constraint $r$ in $\mathcal{R}$ equals 0 **do**
  $p \leftarrow 1$
  **while** $p < s$ **do**
    let $pos$ be the position in $dm$ of the closest element to $r$
    $\mathcal{RM} \leftarrow \mathcal{R}[pos]$ // add an instance $pos$ to reallocation process
    $p \leftarrow p + 1$
    $d[pos, r] \leftarrow d[r, pos] \leftarrow \infty$
  **end while**
  let $ent = entropy(\mathcal{RM})$ // normalized entropy: Equation 10
  **if** $ent < h$ **then**
    let $mode$ be the most common cluster id constraint from $\mathcal{RM}$
    $\mathcal{R}[r] \leftarrow mode$
  **else**
    $\mathcal{R}[r] \leftarrow 0$ // do not set a constraint
  **end if**
**end for**
**return** $\mathcal{R}$

**Algorithm 6** Merge of clusterings algorithm

**Require:** number of clusters $k$, dataset $\mathbf{X}$ and constraints $\mathcal{R}$
**Ensure:** clustering $\pi_f$ of $\mathbf{X}$ into $k$ clusters

Let $\pi_f$ be the resulting clustering
Initialize $\pi_f \leftarrow \mathcal{R}$
Compute centroid for each cluster in $\pi_f$
**while** clusters are changing **do**
  **for** each instance $ds \in \mathbf{X}$ **do**
    Compute distance from $x$ to each centroid
    $c_i \leftarrow$ nearest centroid to $x$
    **if** $c_i$ follows constraint $\mathcal{R}[x]$ **then**
      $\pi_f[x] \leftarrow c_i$
    **end if**
  **end for**
  Recompute centroids
**end while**
**return** $\pi_f$

centers into a new centroid set $C^b$. They proposed an optional prune procedure which removes all centers of clusters with no elements. This procedure prunes the set $C^b$ by computing $\mathbf{X}_n$ partition with respect to $C^b$ using Equation 11 and removes all centers where the corresponding cluster is empty (or below a predefined threshold $\theta$), resulting in the new set $C^b_{prune}$. It also makes all members of $C^b_{prune}(k, \theta)$ unique by remove duplicates.

bClust uses a hierarchical clustering algorithm to create a dendrogram using centers $C^b$ or $C^b_{prune}$ and obtains an original data partition by cutting the dendrogram at a certain level, resulting in a partition $C^b_1, \ldots, C^b_m, 1 \leq m \leq bk$, of set $C^b$. Then, the algorithm assigns the cluster $c(x_i)$ to each point $x_i \in \mathbf{X}_n$ producing the final clustering, where $c(x_i) \in C^b$ is the closest center to $x_i$. bClust time complexity is $O(bQKn^2)$, where $b$ is the number of bootstrap samples, $Q$ is the cost of a base clustering algorithm, $K$ is the number of centers and $n$ is the number of elements.

$$C^b_{prune}(K, \theta) = \{c \in C^b(K) | \#\{x : c = c(x) > \theta\}\} \tag{11}$$

**Algorithm 7** bClust

**Require:** Dataset $\mathbf{X} = \{x_1, \ldots, x_n\}$
A base clustering algorithm $\mathcal{A}$
The number $b$ of bootstrap samples
**Ensure:** Clustered data
1: Construct $b$ bootstrap training samples $\mathbf{X}_1, \ldots, \mathbf{X}_b$ by drawing with replacement from the original sample $\mathbf{X}$
2: Run $\mathcal{A}$ over each sample dataset, resulting in a centers set
3: Combine all centers into a new data set $C^b = C^b(k) = \{c_{11} \ldots, c_{bk}\}$
4: (Optional) Execute prune procedure
5: Run a hierarchical cluster algorithm on $C^b$ (or $C^b_{prune}$), resulting in a dendrogram
6: Cuts the dendrogram and obtains the partition $C^b_1, \ldots, C^b_m, 1 \leq m \leq bk$
7: Assign the cluster $c(x_i)$ to each point $x_i \in \mathbf{X}_n$
8: **return** Clustered data

## 3.3 $k$-means Consensus Clustering

$k$-means Consensus Clustering (KCC) (12) is a $k$-means based algorithm which fits into item 3.6 of taxonomy in Figure 1. It works with the cluster labels of each basic partitioning instead of the data points itself, reducing a consensus clustering problem to a $k$-means clustering problem, where the data points become the partitionings labels and the centroids are the basis of the consensus partition.

In details, it produces a consensus partition by clustering a binary dataset $\mathbf{X}^{(b)}$ containing the concordance among cluster labels of each partitioning. This concordance denotes that the base partitionings assign the same cluster label to a data point. In the

paper, the authors use the utility concept to describe the level of concordance among partitionings.

Algorithm 8 describes in a simplified way each of KCC. As input, the algorithm receive a dataset $\mathbf{X}$, a set of basic partitionings $\Pi$, and the number of clusters $k$. First, KCC computes a binary dataset containing the concordance among the base partitionings. Then, it uses the $k$-means procedure as an optimization algorithm with an alternative point-to-centroid distance to cluster the binary data set, to obtain in the final consensus partition.

**Algorithm 8** Simplified $k$-means Consensus Clustering

**Require:** Dataset $\mathbf{X}$
The set of basic partitionings $\Pi$
The number of clusters $k$
**Ensure:** The consensus partitioning $\pi_c$
1: Compute the binary data set $\mathbf{X}^{(b)}$ from partitionings $\Pi = \{\pi_i, \ldots, \pi_r\}$
2: $\pi_c \leftarrow k\text{-means}(\mathbf{X}^{(b)}, K)$
3: **return** $\pi_c$

The KCC goal is to compute the consensus partition with maximum utility, that is, maximum concordance among base partitionings. Thus, the authors proved that it is possible to maximize the utility by minimizing the sum of distances between points to centroid in $k$-means, if certain $k$-means distance is chosen.

They defined this distance as a convex function, proving the equivalence of Equation 12 and implying that KCC always finds a consensus partition which have a maximum utility. KCC connects maximization of the utility to $k$-means clustering using Equation 12, where the left side represents the KCC utility function and the right side the $k$-means objective function.

In the left side the function $U(\pi, \pi_i)$ measures the utility of partition $\pi$ regarding $\pi_i$, where $w_i$ is the weight defined to $\pi_i$, $r$ is the number of basic partitions, $F$ is the set of possible consensus partitions. In the right side the function $f(\mathbf{x}_l^{(b)}, m_k)$ is a point-to-centroid distance function, where $k$ is the number of clusters, $\mathbf{x}_l^{(b)}$ is the binary concordance values for data point $\mathbf{x}_l$, and $m_k$ is the $k$-th centroid.

$$\max_{\pi \in F} \sum_{i=1}^{r} w_i U(\pi, \pi_i) \Leftrightarrow \min_{\pi \in F} \sum_{k=1}^{k} \sum_{\mathbf{x}_l \in C_k} f(\mathbf{x}_l^{(b)}, m_k) \tag{12}$$

Therefore, KCC clusters the $\mathbf{X}^{(b)}$ and produce the consensus partition $\pi_c$ using the $k$-means algorithm with a distance function calculated by differentiable convex functions. The goal is to maximize the utility consensus-function value $\Gamma$ using Equation 13, which is the left part of Equation 12.

$$\Gamma(\pi, \Pi) = \sum_{i=1}^{r} w_i U(\pi, \pi_i) \tag{13}$$

## 3.4 hkclustering

hkclustering[3] is an ensemble clustering algorithm which represents item 4.1 of taxonomy. Algorithm 9 describes hkclustering. The algorithm makes a mixture of the $k$-means and hierarchical clustering techniques to produce an improved set of initial clusters to re-cluster data. hkclustering first executes $u$ times the $k$-means algorithm building a set of centers $cr$. Then, it calls a hierarchical clustering method to clustering $cr$ and cuts the generated dendrogram to produce $k$ clusters of centers. The algorithm aggregates the centers in $cr$ using a mean function to find average centers for each cluster. Then it uses this average centers as initial centroids in another $k$-means execution, producing the final clustering. The algorithm has complexity time of $O(n^2)$, where $n$ is the number of elements.

**Algorithm 9** hkclustering algorithm

**Require:** Dataset $\mathbf{X}$;
Number of clusters $k$;
Number of iterations to find centroids $u$.
**Ensure:** Final partition $\pi_f$
1: Execute kmeans($\mathbf{X}$,$k$) $u$ times and build a set of centers $cr$
2: Execute the hierarchical clustering using centroid method in $cr$ and cut the generated dendrogram to produce $k$ clusters of centers
3: $ncr \leftarrow$ Aggregate the centers in $cr$ using a mean function to find $k$ average centers for each cluster
4: $\pi_f \leftarrow$ Execute kmeans($\mathbf{X}$,$ncr$), where $ncr$ are the average centers used as initial clusters
5: **return** $\pi_f$

## 3.5 Non-ensemble algorithms

Hierarchical Complete Link (HCL) produces a hierarchical dendrogram of elements and then makes a cut on dendrogram to produce the clustering. This cut is commonly based in the desired number of clusters. HCL first computes distances for each data points and sorts them to find the smallest distance to merge iteration phase in $O(n \log n)$, then it updates distances metrics in $O(n)$. The algorithm traverses $n$ sorted lists of distances, so it will have done $n^2$ traversal steps. Summing up, the algorithm has time complexity in the worst time of $O(n^2)$.

Hierarchical Single Link (HSL) computes distances and finds the smallest distance in $O(n^2)$ steps. It merges the two closest clusters and updates the distance matrix in $O(n)$. Finally, HSL updates the closest points array on each step using $O(n)$ computations. Thus, the final complexity is $O(n^2)$.

[3] This technique is available in the `hkclustering` package for R.

Common implementations of $k$-means algorithm aims to minimize the sum of square distances objective function, running at most $T$ iterations for $n$ points in $d$ dimensions and $k$ centers, giving the complexity time $O(nkTd)$. Usually, $T$ is small and $d$ and $k$ are constants, therefore its time complexity is approximated by $O(n)$. Lloyd's $k$-means algorithm works better on data sets where elements are naturally organized in the form of spheroids (44).

The density-based algorithm DBSCAN scrolls through each data point and its neighborhood defining small clusters and merging them by using two parameters, a distance $\epsilon$ and a minimum data points $minPts$. DBSCAN finds arbitrarily sized and shaped clusters in contrast to spherical clusters preferred by $k$-means. The time complexity is $O(n^2)$ because it computes distances for all points looking for cluster centers.

## 3.6 Comparison of algorithm complexities

Table 1 lists the time and space complexity of each algorithm, where $n$ is the number of elements, $b$ is the number of bootstrap samples, $k$ is the number of centers, $T$ is the number of iterations to $k$-means convergence, $Q$ is the cost of a base clustering algorithm, $r$ is the number of non-zero dimensions for each data point and $d$ is the number of dimensions and $s$ is the number of elements without constraints.

Table 1 – Time and space complexity, ensemble classification and main reference of each algorithm used in experiments.

| **Algorithm** | **Time complexity** | **Space complexity** | **Ensemble clustering** | **Reference** |
|---|---|---|---|---|
| HCL | $O(n^2 \log n)$ | $O(n^2)$ | No | (15) |
| HSL | $O(n^2)$ | $O(n^2)$ | No | (15) |
| $k$-means | $O(nkTd)$ | $O(nkd)$ | No | (45) |
| DBSCAN | $O(n^2)$ | $O(n)$ | No | (17) |
| CBA | $O(nk^2Tds^2)$ | $O(n^2)$ | Yes | (11) |
| bClust | $O(n^2kbQ)$ | $O(n^2 + bkn)$ | Yes | (10) |
| hkclustering | $O(n^2)$ | $O(n^2)$ | Yes | (13) |
| KCC | $O(nkTdr)$ | $O(n)$ | Yes | (12) |

We choice these algorithms according to the availability for execution and popularity of use, but the their costs/complexity limited the size of the datasets for systematic experimentation with hypothesis testing purposes.

Chapter 4

# Proposed Approach

The main goal of this work was to study the impacts of using random transformations to improve ensemble clustering quality. A random transformation is a procedure which applies a structural change in each dataset element before clustering. Blaser e Fryzlewicz (6) introduced a transformation called Random Rotations used to improve the results of Random Forest as described in Section 2.1.4. Based on that achievement, we considered the following hypothesis: applying random transformations to an unsupervised setting will lead to improvements on cluster quality?

Besides the Random Rotations, we have also evaluated other transforms: Mahalanobis (46) and density-based. Table 2 summarizes a description of three proposed random transformations: Random rotations, Mahalanobis and density-based. We detailed Mahalanobis and density-based transformations in the following sections.

Table 2 – A summary of data transforms.

| **Transformation** | **Description** |
|---|---|
| Random Rotations | Rotates the dataset by multiplying data matrix by a random rotation matrix. |
| Mahalanobis | Distorts the dataset by multiplying data matrix by a random covariance matrix. |
| Density based | Attracts or repulses data points using cluster centroids as reference. |

## 4.1 Random Mahalanobis transformation

Mahalanobis transformation change data elements by modifying the covariance among elements. It uses the Mahalanobis distance (46) to provide more possibilities to change dataset characteristics when compared to Random Rotations, making possible to change the scale among features, the features relation proportion and to rotate.

Mahalanobis distance, defined in Equation 14, is a generalized idea of how many standard deviations away a data point $\mathbf{x}$ is from a data point $\mathbf{y}$. It uses the $d \times d$ covariance

matrix $\mathbf{S}$ to compute the distance between points. Instead of using Mahalanobis distance to actually calculate distance between elements in the clustering process, we use it to change the elements position in data space. That is, we use these new distances among the elements to set new positions to each element and then use an ensemble clustering with their default configuration to this transformed dataset.

We picked two ways to generate the random covariance matrix. The first option is to use the normal distribution with mean 0 and standard deviation 1, and the second is to use the uniform distribution with minimum and maximum values 0 and 1. We also define the matrix diagonal values as 1 to not interfere on an element itself. We emphasize that any probability distribution can be used to generate the random covariance matrix.

A drawn covariance matrix transforms each element using Equation 15, where $\mathbf{x}_i$ is the original element, $\mathbf{x}'_i$ is the transformed element and $\mathbf{S}'$ is the random covariance values of column $j$.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{y})} \tag{14}$$

$$\mathbf{x}'_i = \mathbf{x}_i \mathbf{S}' \tag{15}$$

In the experiments, we also included variations of Mahalanobis random transformations which consider only some dimensions to transform. That is, giving a data set $\mathbf{X}$ with $d$ dimensions and the random covariance matrix $S$, the transformation variation selects a percentage of $S$ dimensions to keep the column values and then set remaining columns as 0, which means that the algorithm only transforms some characteristics of data set and discards the remaining dimensions. We select the percentages of 20%, 50% and 80% of $d$ to produce the variations. Removing some dimensions allows us to evaluate the impacts of random Mahalanobis transformation when applied progressively and if the transformation can supply the loss of information due the random selection. This could easily burden of high dimensionality for clustering. Figure 2 shows an example of transformed data set using Mahalanobis transformation with normal distribution.

## 4.2 Density-based transformation

We considered density attraction and density repulsion transformations. These transformations attract or repulse the data points from determined reference points. This kind of transformation is also found in "gravity"-inspired algorithms. First, it uses the $k$-means++ procedure to select $k$ initial centers, defined in Algorithm 10. The procedure samples a data point to be the first center using a $1/n$ probability and then it computes $d(\mathbf{x})$, which is the distance of each data point to its nearest existing center. To sample the next center, it updates the probabilities array $pr$, where each data point $\mathbf{x}$ has a probability proportional to $d(\mathbf{x})^2$.

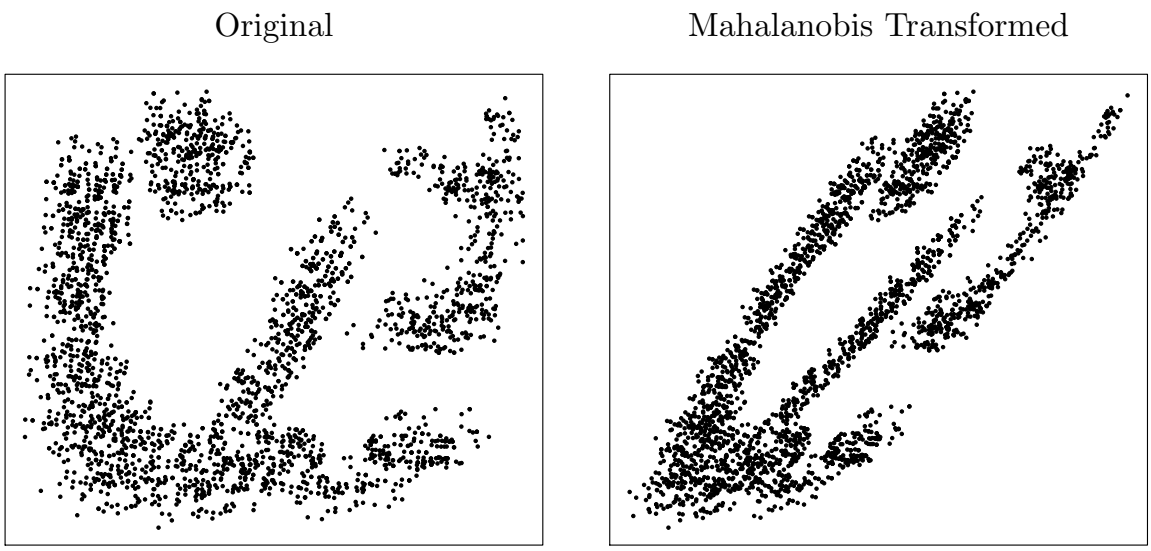


Figure 2 – Original and Mahalanobis-transformed Artificial Anchor data set using Equation 15 and a normal distribution. Section 5.1 provides more details about this dataset.

[†]This dataset is available at https://github.com/gabrielgdr/ArtificialAnchorDataset.

The second part uses the obtained initial set of centers to attract or repulse the data point $\mathbf{x}_i$. Thus, Algorithm 11 takes as reference the closest center $\mathbf{c}_i$ for $\mathbf{x}_i$ and applies Equation 16 to transform data point $\mathbf{x}_i$, where $\alpha$ is a scale factor.

$$\mathbf{x}'_i = \mathbf{x}_i + \alpha(\mathbf{x}_i - \mathbf{c}_i) \tag{16}$$

The scale factor $\alpha$ defines how much the procedure moves the elements. It is a random value sampled using a normal distribution with mean 3 or $-3$ and standard deviation 1. These values are chosen manually due to empirical observations. Using $\alpha < 0$ produces an attraction effect, the transformation attracts each element to a closest center reducing the distance between them using Equation 16. It causes the opposite effect when $\alpha > 0$, where $\mathbf{x}'_i$ is the transformed element, $\mathbf{x}_i$ is the natural element, and $c_i$ is the closest center to element $\mathbf{x}_i$.

We increased transformation diversity by changing initial clusters density, since Random Rotations and Mahalanobis transformation produce an element distortion on dataset. This diversity helps the study giving more information about the effects of random transformations on ensemble clustering.

**Algorithm 10** $k$-means++ centers selection procedure

**Require:** Dataset $\mathbf{X}$
Number $k$ of clusters
**Ensure:** Set of $k$ centers
1: Set probability $\mathbf{pr} \leftarrow 1/n$
2: **for** $i$ in 1 to $k-1$ **do**
3: $\mathbf{centers}[i] \leftarrow$ Sample the $i$-th center using $pr$
4: Compute the square of distances between each point to the $i$-th center
5: $\mathbf{pr} \leftarrow$ Update probability array for the next sampling.
6: **end for**
7: $\mathbf{centers}[k] \leftarrow$ Sample the $k$-th center using $\mathbf{pr}$
8: **return** Return the set of centers with lowest total within-cluster sum of squares.

**Algorithm 11** Density-based transformation algorithm

**Require:** Dataset $\mathbf{X}$
Number $k$ of clusters
Scale factor $\alpha$
**Ensure:** Transformed dataset $\mathbf{X}'$
$\mathbf{centers} \leftarrow$ Generate initial cluster centers using Algorithm 10
1: **for** $i$ in 1 to $n$ **do**
2: $\mathbf{c}_i \leftarrow$ closest center to data point $\mathbf{x}_i$
3: $\mathbf{x}'_i \leftarrow \mathbf{x}_i + \alpha(\mathbf{x}_i - \mathbf{c}_i)$
4: **end for**
5: **return** Transformed dataset $\mathbf{X}'$

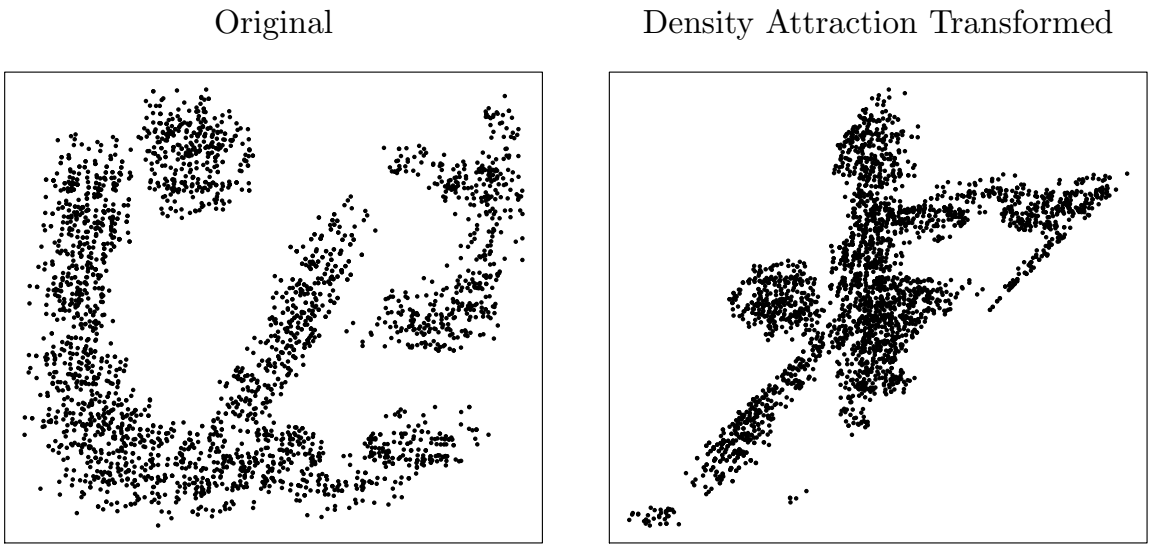


Figure 3 – Original Artificial anchor dataset and Density attraction transformed dataset using Equation 16 with $\alpha = 3$.

## 4.3 Concluding remarks

We considered two types of random transformations, based on Mahalanobis distance and based on density. These transformations, along with random rotations and their variations, allow a wide study of the effects of random transformations on ensemble clustering algorithms.

As Blaser and Fryzlewicz (6) achieved improvements for random forests, we also expect an increase in cluster quality using the transformations. The usability of the proposed transformations naturally is restricted to numerical datasets, because these methods are not directly applied to categorical data. Although, in a mixed dataset, it is possible to apply them on the numerical characteristics.

Chapter 5

# Tests and Experiments

Our experiments aim to provide a detailed view of the impacts caused by random transformations on clustering ensemble by evaluating if random transformations can improve cluster quality. Another way to assess the impacts, not used in this work, would be to evaluate qualitatively cluster shapes and cluster novelties obtained by the ensemble algorithms. Therefore, our work considers objective cluster quality because it is an important aspect of clustering and there is a wide quantity of clustering measures to perform this evaluation.

We used four complementary quality measures to evaluate the results of ensemble clustering algorithms: Accuracy, Adjusted Rand Index, Dunn Index and Silhouette Width Criterion. Furthermore, we combined these four measures to help us observe the impacts more generally. Each measure is detailed in Section 5.2.

In the experiments we considered the random transformations listed in Table 3. Additionally, we evaluated the effects of sequences of transformations to ensemble clustering. For example, Random Rotations followed by Mahalanobis transformation and random rotations followed by Uniform Mahalanobis transformation. We also considered the procedure to select randomly dimensions described in Section 4.1. In total, we evaluated 25 ways to transform a dataset.

## 5.1 Dataset details

Most datasets used in our experiments are found in the UCI Machine Learning Repository (42). We listed some details for each dataset in Table 4. We choose these 11 datasets aiming at a high diversity of attributes and elements number, but still considering only numeric datasets due to the nature of the clustering algorithms listed in Table 1.

Iris dataset has 3 classes of 50 instances, where each class refers to a species of iris plant. One class is linearly separable from the other two, which are not linearly separable from each other.

Table 3 – Random transformations used in experiments.

| Transformation | Acronym |
|---|---|
| Original (no transformation) using 100%, 80%, 50% and 20% of $d$ | **N100**,**N80**,**N50**,**N20** |
| Random Rotations using 100%, 80%, 50% and 20% of $d$ | **R100**, **R80**, **R50**,**R20** |
| Mahalanobis transformation using 100%, 80%, 50% and 20% of $d$ | **M100**, **M80**, **M50**, **M20** |
| Random Rotations followed by Mahalanobis transformation using 100%, 80%, 50% and 20% of $d$ | **RM100**, **RM80**, **RM50**, **RM20** |
| Uniform Mahalanobis transformation using 100%, 80%, 50% and 20% of $d$ | **UM100**, **UM80**, **UM50**, **UM20** |
| Random Rotations followed by Uniform Mahalanobis transformation using 100%, 80%, 50% and 20% of $d$ | **RUM100**, **RUM80**, **RUM50**, **RUM20** |
| Density attraction (average = $-3$ and standard deviation= 1) | **DA** |
| Density repulsion (average = 3 and standard deviation= 1) | **DR** |

Table 4 – Meta-data description of datasets used in experiments.

| Name | Number of attributes | Number of elements |
|---|---|---|
| Artificial Anchor | 2 | 3038 |
| Haberman's | 3 | 306 |
| Iris | 4 | 150 |
| Banknote | 5 | 1372 |
| Transfusion | 5 | 748 |
| Vertebral Column | 6 | 310 |
| Seeds | 7 | 210 |
| Pima | 8 | 768 |
| Heart Disease Data | 13 | 302 |
| Wine | 13 | 178 |
| Sonar | 60 | 208 |
| Hill-Valley | 101 | 606 |

Banknote authentication dataset contains coefficients from wavelets transformation to characterize images taken from genuine and forged banknote-like specimens.

Blood transfusion service center dataset is an organized database of Blood Transfusion Service Center donors in Hsin-Chu City in Taiwan, containing information about blood samples donated about every three months. The label, a binary field, indicates if the person has donated blood in March of 2007.

Seeds dataset contains data about kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian. There are 70 samples of each variety, randomly selected for the experiment.

Haberman's survival dataset describes cases from a study conducted on the survival of patients who had undergone surgery for breast cancer. It contains a label which indicates if the patient had died within 5 years, or survived 5 years or longer.

In the Hill-Valley dataset each example is a sequentially plot of 100 points into a two-dimensional space. These points represent either a Hill or a Valley observing the Y coordinate.

Sonar dataset tries to identify rocks and mines. Each sample is a set of 60 sonar signals ranging from 0 to 1, and each signal represents the energy within a particular frequency band, integrated over a certain period of time. A label associated with each record contains the letter identification, "R" if the object is a rock and "M" if it is a mine.

Pima Indians Diabetes dataset contains diabetes-related information of female patients with Pima Indian heritage. They considered only females with at least 21 years of illness.

Heart Disease dataset lists patients with attributes related to heart diseases. The data collected are: age; sex; chest pain type; resting blood pressure; serum cholesterol; fasting blood sugar; resting electrocardiographic results; maximum heart rate achieved; exercise induced angina; ST depression induced by exercise relative to rest; slope of the peak exercise ST segment; number of major vessels colored by flouroscopy; and level of defect.

Wine dataset is the result of chemical analysis of Italian wines derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wine.

Vertebral Column dataset is a biomedical dataset built by Dr. Henrique da Mota during a medical residence period in the Group of Applied Research in Orthopedics (GARO) of the *Centre médico-chirurgical de réadaptation des Massues*, Lyon, France. Each row represents a patient using six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine: pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis. The dataset uses the following convention for the class labels: DH (Disk Hernia), Spondylolisthesis (SL), Normal (NO) and Abnormal (AB).

Additionally, we used the dataset Artificial Anchor (Figure 4), which is designed to be a difficult case for clustering algorithms to decide the boundaries and shapes of clusters. In this dataset, there are 2 dimensions and three clusters of quasi-spheroid (ranging in $([240, 325], [550, 650])$) and non-spheroid shapes (on lower-left and upper-right positions).

## 5.2 Evaluation measures

We evaluated clustering algorithms using the average correctly placed instances using class external information as Accuracy and Adjusted Rand Index (18) as external evaluation measures. Also, we used the following internal measures, Dunn Index (19) and Silhouette Width Criterion (SWC) (20). We chose these measures because they evaluate characteristics interesting to our work. Also they are standard in the literature and evaluate orthogonal characteristics of the clustering (47).

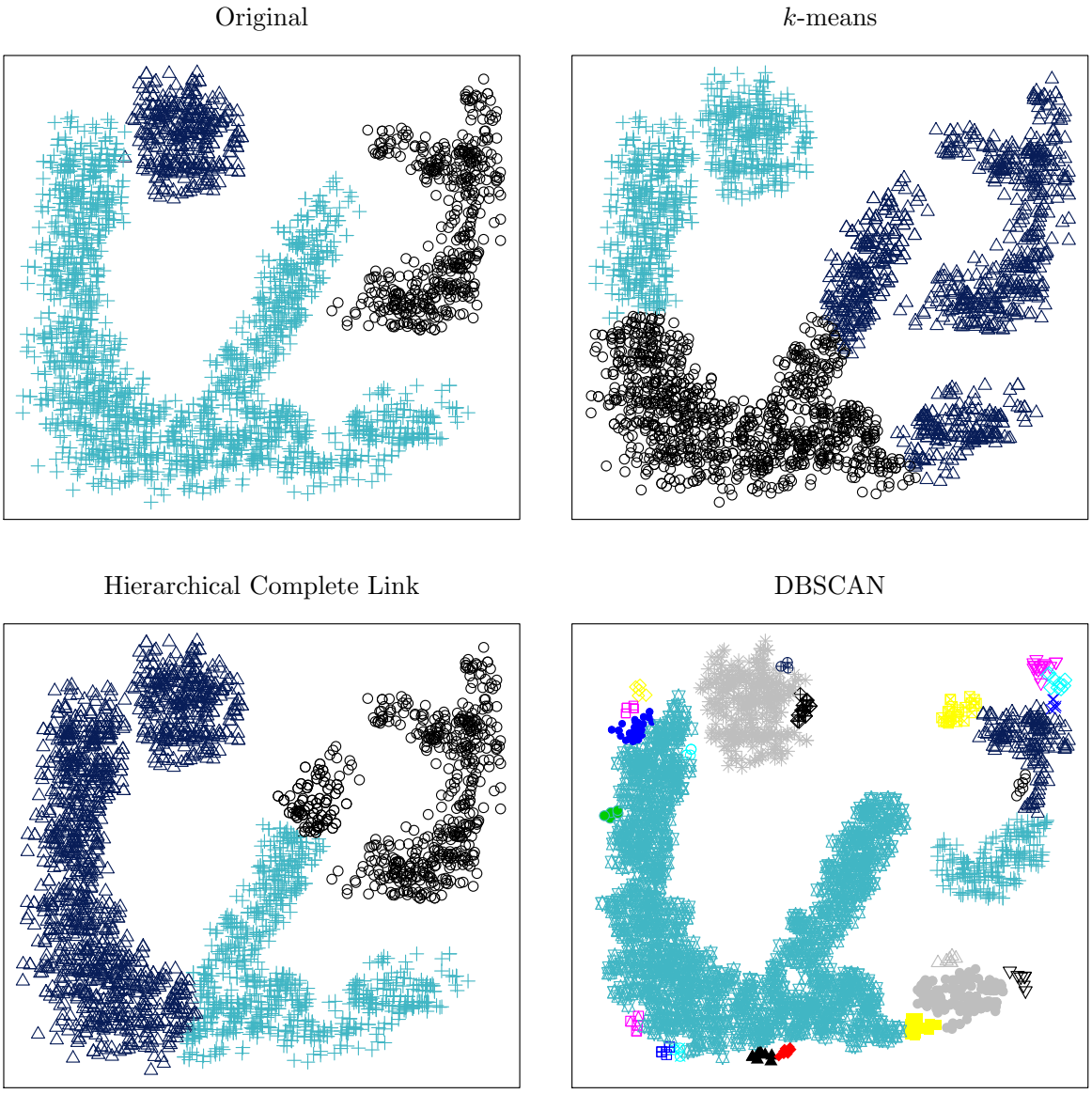


Figure 4 – Clustering of Artificial anchor dataset performed by three algorithms, $k$-means; Hierarchical Complete-Link; and DBSCAN ($\epsilon$ defined using kNN dist plot as suggested in DBSCAN R package manual).

Adjusted Rand Index, defined on Equation 17, is a measure which verifies the agreements and disagreements between two sets of partitions by using dataset external information, i.e. classes. Given the partition tables $R$ containing the classes labels and $Q$ being the partitions of clustering, $a$ is the number of pairs of elements belonging to the same class in $R$ and to the same cluster in $Q$, $b$ is the number of pairs of elements belonging to the same class in $R$ and to different clusters in $Q$, $c$ is the number of pairs of elements belonging to different classes in $R$ and to the same cluster in $Q$, $d$ is the number of pairs of data objects belonging to different classes in $R$ and to different clusters in $Q$ and $M = a + b + c + d$.

$$w = \frac{a - \frac{(a+c)(a+b)}{M}}{\frac{(a+c)(a+b)}{2} - \frac{(a+c)(a+b)}{M}} \tag{17}$$

Dunn Index is a cluster validity criterion based on geometrical measures which employs concepts directly related to within-group and between-group. It gives values between zero and infinity, where higher values represent dense and well-separated clusters. In the experiments we normalized the obtained values into $[0, 1]$ interval. Equation 18 computes Dunn Index where, $\delta_{p,q}$ is the *cluster distance*, defined as minimum distance between a pair of elements across clusters $p$ and $q$, $\Delta_l$ is the *diameter* of the $l$th cluster, defined as the maximum distance between a pair of objects within $l$-th cluster and $k$ is the number of clusters.

$$DN = \min_{p,q \in \{1,\ldots,k\}, p \neq q} \left\{ \frac{\delta_{p,q}}{\max_{l \in \{1,\ldots,k\}} \Delta_l} \right\} \tag{18}$$

Silhouette Width Criterion ($SWC$) measures compactness and separation of clusters. Equation 19 defines the silhouette of an element, where $a_{p,i}$ is the average distance of the element $\mathbf{x}_i$ to all other elements in cluster $p$, $d_{q,i}$ is the average distance of this element to another cluster $q$, $q \neq p$ and $b_{p,i}$ is $\min_{q \in \{1,\ldots,k\}, p \neq q} d_{q,i}$. Equation 20 gives the average silhouette of all $n$ elements. This measure returns high values for well-formed clusters due to their reduced values of intra-group distance and higher inter-group distance. We normalized the $SWC$ values into $[0, 1]$ interval because originally it returns values between $-1$ and 1.

$$s_{\mathbf{x}_i} = \frac{b_{p,i} - a_{p,i}}{\max\{a_{p,i}, b_{p,i}\}} \tag{19}$$

$$SWC = \frac{1}{n} \sum_{i=1}^{n} s_{\mathbf{x}_i} \tag{20}$$

In order to summarize all four orthogonal characteristics highlighted by the previous evaluated measures, we used a cluster quality measure $CQ_{A,T}$ which aggregates the results of all four measures using algorithm $A$ with transformation $T$ applied on all datasets, as defined in Equation 21. In this equation, $Accuracy_{A=k\text{-means},T=\text{Mahalanobis},D=\text{Iris}}$ represents

the average accuracy of 30 iterations of the algorithm $k$-means on Iris dataset using the Mahalanobis transformation. The same idea is applied to the other measures, $ARI_{A,T,D}$, $Dunn_{A,T,D}$ and $SWC_{A,T,D}$. Thus, we obtained a summarizing value $CQ_{A,T}$ for evaluating a given algorithm and a given random transformation for all datasets, as shown in Figure 5 and Figure 6.

$$CQ_{A,T} = \frac{1}{D}\sum_{D}(Accuracy_{A,T,D} + ARI_{A,T,D} + Dunn_{A,T,D} + SWC_{A,T,D}) \tag{21}$$

## 5.3 Experiment design

We executed the experiments using 25 configurations of random transformations and the original dataset. Due to the non-applicability of dimension reduction on low dimensional datasets, we divided the experiments in two parts. One for high dimensional datasets, with 8 or more dimensions, and another for low dimensional datasets.

For high dimensional datasets, we applied the following transformations: N100, N80, N50, N20, RM100, RM20, RM50, RM80, M100, M20, M50, M80, RUM100, RUM20, RUM50, RUM80, UM100, UM20, UM50, UM80, R100, R80, R50, R20, DA, and DR. For low dimensional datasets, we applied the following transformations: N100, RM100, M100, RUM100, UM100, R100, DA, and DR.

The algorithms used in the experiments were: Hierarchical Complete Link (HCL)[1] and Single Link (HSL)[1], Hartigan and Wong $k$-means[1], Constraint based algorithm (CBA)[2], bClust[3], hkclustering[4], $k$-means Consensus Clustering (KCC)[5] and DBSCAN[6].

Each experiment instance comprises 30 executions of each algorithm-dataset pair. In total, we used 5580 dataset variations for 8 algorithms.

Demšar (22) suggested using the Friedman test (21) to compare algorithms by ranking $m$ different clustering algorithms using $l$ clustering measures, evaluating and verifying whether there is an algorithm whose performance is consistently different than the others. Therefore, in order to verify statistically whether random transformations had significantly improved cluster quality we used the Friedman test and the *post hoc* Nemenyi's procedure (22).

Friedman test is a non-parametric statistical test proposed by Milton Friedman (21) which aims to detect differences among multiple experiment instances by using a rank system. First, it verifies whether the null hypothesis holds, in which there are only random differences among the results in different conditions, by calculating a Friedman statistic $Fr$ defined in Equation 22 where, $r_i^j$ is the rank of $j$-th algorithm of $i$-th evaluation.

[1] From stats R package.
[2] From cba R package, available on github.
[3] From e1071 R package.
[4] From hkclustering R package.
[5] Source code provided by authors via e-mail.
[6] From dbscan R package.

Taking $l$ clustering measures, $m$ clustering algorithms, and confidence level $1-\alpha$, if $Fr > \chi^2_{m-1,1-\alpha}$, then Friedman test rejects the null hypothesis and applies a *post hoc* test, like Nemenyi's procedure (22).

$$Fr = \frac{12l}{m(m+1)}\left[\sum_j R_j^2 - \frac{m(m+1)^2}{4}\right], R_j = \frac{1}{l}\sum_i r_i^j \tag{22}$$

Nemenyi's procedure compares the average of each experiment instance with each other average, pair for pair, trying to identify if the observed averages are equivalent. Similarly, it uses the null hypothesis of Friedman test. The procedure is useful to identify the difference between each pair of algorithms, providing a detailed statistical comparison. Also, Conover and Iman (48) highlights the conservatism of Nemenyi's when it handles with false-positive errors (Type 1 errors), leading to avoid them. Therefore, it favors the maintenance of null hypothesis over rejecting it.

To find the corresponding $p$-value from the table of normal distribution, it uses the critical value $z$. Equation 23 defines $z$, where $R_i$ is the ranking value of the first algorithm, $R_j$ is the ranking value of the second algorithm, $m$ is the number of different clustering algorithms, and $n$ is the number of elements.

$$z = \frac{R_i - R_j}{\sqrt{\frac{m(m+1)}{6n}}} \tag{23}$$

## 5.4 Experimental results

We organized the experimental results in two parts. Due to the usage of dimensionality reduction transformations, we show first the results of low dimensional datasets ($d < 8$) and after, the results of high dimensional datasets ($d \geq 8$).

### 5.4.1 Low dimensional datasets results

In the first part, Figure 5 summarizes the average $CQ_{A,T}$ results of each algorithm-transformation pair for low dimensional datasets, presented as a heat map, where we observed consistent better results on bottom-right of the table. We show in Table 5 all quality measures values used to draw the heat map with the greater values for each measure highlighted in bold.

Nevertheless, it is not possible to identify if there are significant differences between the results by only using the $CQ_{A,T}$ measure. Thus, we used the Friedman ranking table followed by Nemenyi's procedure to present a statistical value comparison, considering 95% of confidence level ($\alpha = 0.05$) for all statistical tests.

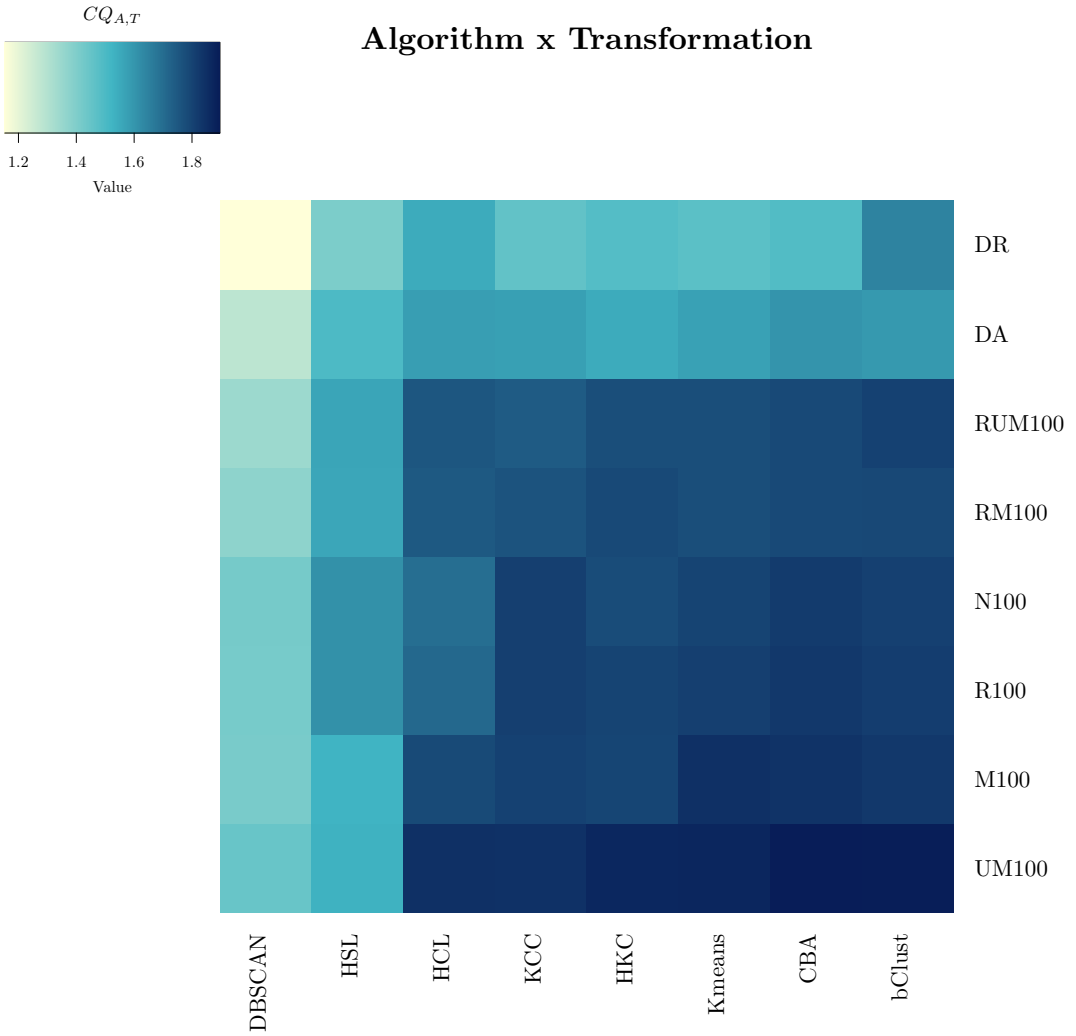


Figure 5 – Heat map of the cluster quality measure $CQ_{A,T}$ for datasets with low number of dimensions. Higher values are in darker colors indicating better performance of an algorithm-transformation pair.

Table 5 – All clustering measure values for each Algorithm-Transformation pair considering low dimensional datasets.

| **Algorithm** | **Transformation** | **Accuracy** | **ARI** | **Dunn Index** | **SWC** | **$CQ_{A,T}$** |
|---|---|---|---|---|---|---|
| N100 | CBA | 0.708 | 0.375 | 0.0128 | 0.725 | 1.82 |
| N100 | HCL | 0.655 | 0.328 | 0.0164 | 0.7 | 1.7 |
| N100 | HSL | 0.634 | 0.287 | 0.0374 | 0.656 | 1.61 |
| N100 | Kmeans | 0.685 | 0.406 | 0.00732 | 0.701 | 1.8 |
| N100 | bClust | 0.691 | 0.341 | 0.0269 | 0.751 | 1.81 |
| N100 | HKC | 0.693 | 0.418 | 0.00441 | 0.665 | 1.78 |
| N100 | KCC | 0.694 | 0.434 | 0.00699 | 0.677 | 1.81 |

*Continued on next page*

Table 5 – *Continued from previous page*

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| N100 | DBSCAN | 0.388 | 0.348 | **0.0529** | 0.632 | 1.42 |
| RM100 | CBA | 0.676 | 0.342 | 0.0139 | 0.757 | 1.79 |
| RM100 | HCL | 0.665 | 0.323 | 0.017 | 0.744 | 1.75 |
| RM100 | HSL | 0.624 | 0.249 | 0.032 | 0.658 | 1.56 |
| RM100 | Kmeans | 0.666 | 0.376 | 0.00484 | 0.729 | 1.78 |
| RM100 | bClust | 0.673 | 0.333 | 0.0214 | 0.764 | 1.79 |
| RM100 | HKC | 0.682 | 0.39 | 0.00427 | 0.712 | 1.79 |
| RM100 | KCC | 0.671 | 0.394 | 0.00487 | 0.694 | 1.76 |
| RM100 | DBSCAN | 0.4 | 0.296 | 0.0344 | 0.643 | 1.37 |
| M100 | CBA | 0.703 | 0.382 | 0.00803 | 0.748 | 1.84 |
| M100 | HCL | 0.679 | 0.35 | 0.0162 | 0.74 | 1.79 |
| M100 | HSL | 0.619 | 0.23 | 0.0303 | 0.648 | 1.53 |
| M100 | Kmeans | 0.694 | 0.421 | 0.00505 | 0.727 | 1.85 |
| M100 | bClust | 0.689 | 0.362 | 0.019 | 0.759 | 1.83 |
| M100 | HKC | 0.681 | 0.407 | 0.00386 | 0.707 | 1.8 |
| M100 | KCC | 0.681 | 0.424 | 0.00513 | 0.695 | 1.8 |
| M100 | DBSCAN | 0.426 | 0.331 | 0.0308 | 0.629 | 1.42 |
| RUM100 | CBA | 0.675 | 0.337 | 0.015 | 0.763 | 1.79 |
| RUM100 | HCL | 0.666 | 0.323 | 0.0176 | 0.751 | 1.76 |
| RUM100 | HSL | 0.622 | 0.243 | 0.0322 | 0.668 | 1.57 |
| RUM100 | Kmeans | 0.669 | 0.369 | 0.00509 | 0.732 | 1.78 |
| RUM100 | bClust | 0.676 | 0.332 | 0.0228 | **0.775** | 1.8 |
| RUM100 | HKC | 0.678 | 0.371 | 0.00468 | 0.723 | 1.78 |
| RUM100 | KCC | 0.664 | 0.382 | 0.00505 | 0.692 | 1.74 |
| RUM100 | DBSCAN | 0.387 | 0.305 | 0.0166 | 0.637 | 1.35 |
| UM100 | CBA | **0.721** | 0.419 | 0.00949 | 0.747 | **1.9** |
| UM100 | HCL | 0.702 | 0.394 | 0.0147 | 0.737 | 1.85 |
| UM100 | HSL | 0.619 | 0.23 | 0.0287 | 0.656 | 1.53 |
| UM100 | Kmeans | 0.693 | **0.445** | 0.00492 | 0.73 | 1.87 |
| UM100 | bClust | 0.713 | 0.403 | 0.0169 | 0.759 | 1.89 |
| UM100 | HKC | 0.699 | 0.445 | 0.00435 | 0.722 | 1.87 |
| UM100 | KCC | 0.693 | 0.444 | 0.00508 | 0.704 | 1.85 |
| UM100 | DBSCAN | 0.446 | 0.352 | 0.0341 | 0.616 | 1.45 |
| R100 | CBA | 0.716 | 0.379 | 0.012 | 0.721 | 1.83 |
| R100 | HCL | 0.664 | 0.328 | 0.0175 | 0.702 | 1.71 |
| R100 | HSL | 0.634 | 0.287 | 0.0374 | 0.656 | 1.61 |

*Continued on next page*

Table 5 – *Continued from previous page*

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| R100 | Kmeans | 0.69 | 0.411 | 0.00749 | 0.701 | 1.81 |
| R100 | bClust | 0.69 | 0.341 | 0.0319 | 0.754 | 1.82 |
| R100 | HKC | 0.703 | 0.404 | 0.00609 | 0.688 | 1.8 |
| R100 | KCC | 0.693 | 0.434 | 0.00699 | 0.677 | 1.81 |
| R100 | DBSCAN | 0.388 | 0.348 | 0.0528 | 0.629 | 1.42 |
| DA | CBA | 0.611 | 0.261 | 0.0294 | 0.706 | 1.61 |
| DA | HCL | 0.598 | 0.253 | 0.0298 | 0.702 | 1.58 |
| DA | HSL | 0.588 | 0.184 | 0.0473 | 0.681 | 1.5 |
| DA | Kmeans | 0.598 | 0.28 | 0.0137 | 0.683 | 1.57 |
| DA | bClust | 0.595 | 0.236 | 0.0357 | 0.728 | 1.6 |
| DA | HKC | 0.594 | 0.275 | 0.0119 | 0.669 | 1.55 |
| DA | KCC | 0.607 | 0.289 | 0.0133 | 0.667 | 1.58 |
| DA | DBSCAN | 0.383 | 0.217 | 0.0329 | 0.652 | 1.28 |
| DR | CBA | 0.613 | 0.24 | 0.0103 | 0.627 | 1.49 |
| DR | HCL | 0.61 | 0.238 | 0.0221 | 0.679 | 1.55 |
| DR | HSL | 0.567 | 0.141 | 0.0371 | 0.664 | 1.41 |
| DR | Kmeans | 0.572 | 0.263 | 0.00613 | 0.631 | 1.47 |
| DR | bClust | 0.627 | 0.271 | 0.0265 | 0.724 | 1.65 |
| DR | HKC | 0.6 | 0.279 | 0.00506 | 0.603 | 1.49 |
| DR | KCC | 0.572 | 0.265 | 0.00587 | 0.616 | 1.46 |
| DR | DBSCAN | 0.359 | 0.211 | 0.0217 | 0.561 | 1.15 |

Table 6 presents statistical rank for algorithms produced by Friedman test for low dimensional datasets, where lower values indicates better results. The algorithm CBA has the best results, and DBSCAN the worst results. Considering the performance of clustering algorithms, the test obtained the statistic $Fr = 45.833$ distributed according to a $\chi^2$ distribution with 7 degrees of freedom. Thus $\chi^2_{7,0.95} = 14.067$ and $Fr > \chi^2_{7,0.95}$ indicates statistical differences between the algorithms used in the test.

We also applied the Friedman test to produce a ranking for the quality of transformations, as presented in Table 8. This test shows statistical significance among the quality of transformations. The best random transformation is Uniform Mahalanobis 100% (UM100), and the worst is Density Repulsion (DR).

Using the post hoc Nemenyi's procedure we compared transformations pairwise as presented in Table 9, which shows only rejected null hypothesis for low dimensional datasets.

| **Algorithm** | **Ranking value** |
|---|---|
| CBA | 1.6 |
| bClust | 1.9 |
| $k$-means | 3.8 |
| HKC | 4.4 |
| KCC | 4.5 |
| HCL | 4.9 |
| HSL | 7.0 |
| DBSCAN | 8.0 |

Table 6 – Friedman-test ranking considering the results of all random transformations for low dimensional datasets. Lower values indicate better average clustering quality $CQ_{A,T}$ of algorithm $A$ for all transformations $T$.

| **Algorithms** | | $p$**-value** |
|---|---|---|
| **CBA** | **DBSCAN** | **0.000** |
| **bClust** | **DBSCAN** | **0.000** |
| **CBA** | **HSL** | **0.001** |
| **HSL** | **bClust** | **0.001** |
| **$k$-means** | **DBSCAN** | **0.015** |
| HKC | DBSCAN | 0.086 |
| KCC | DBSCAN | 0.119 |
| CBA | HCL | 0.223 |
| HSL | $k$-means | 0.223 |
| HCL | DBSCAN | 0.300 |
| HCL | bClust | 0.401 |
| CBA | KCC | 0.529 |
| CBA | HKC | 0.693 |
| HSL | HKC | 0.898 |
| bClust | KCC | 0.898 |
| HSL | KCC | 1.154 |
| bClust | HKC | 1.154 |
| CBA | $k$-means | 2.316 |
| HCL | HSL | 2.316 |
| $k$-means | bClust | 3.522 |
| HCL | $k$-means | 10.033 |
| HSL | DBSCAN | 11.598 |
| $k$-means | KCC | 15.128 |
| $k$-means | HKC | 17.075 |
| HCL | HKC | 19.127 |
| HCL | KCC | 21.265 |
| CBA | bClust | 23.471 |
| HKC | KCC | 25.724 |

Table 7 – $p$-values of Nemenyi's procedure for comparing pairs of algorithms ($\alpha = 0.05$) for low dimensional datasets. The procedure rejected null hypothesis in bold.

| **Transformation** | **Ranking value** |
|---|---|
| UM100 | 1.5 |
| R100 | 2.9 |
| M100 | 3.1 |
| N100 | 3.6 |
| RM100 | 4.8 |
| RUM100 | 5.1 |
| DA | 7.1 |
| DR | 7.9 |

Table 8 – Ranking values provided by Friedman test considering the results of all algorithms for low dimensional datasets. Lower values indicate better cluster quality $CQ_{A,T}$.

| **Algorithms** | | $p$**-value** |
|---|---|---|
| **UM100** | **DR** | **0.000** |
| **UM100** | **DA** | **0.000** |
| **R100** | **DR** | **0.002** |
| **M100** | **DR** | **0.003** |
| **N100** | **DR** | **0.012** |
| **R100** | **DA** | **0.018** |
| **M100** | **DA** | **0.031** |
| RUM100 | UM100 | 0.086 |
| N100 | DA | 0.102 |
| RM100 | UM100 | 0.223 |
| RM100 | DR | 0.300 |
| RUM100 | DR | 0.693 |
| RM100 | DA | 1.469 |
| RUM100 | R100 | 2.074 |
| N100 | UM100 | 2.581 |
| M100 | RUM100 | 2.869 |
| RUM100 | DA | 2.869 |
| RM100 | R100 | 3.889 |
| RM100 | M100 | 5.168 |
| M100 | UM100 | 5.168 |
| N100 | RUM100 | 5.657 |
| UM100 | R100 | 6.734 |
| N100 | RM100 | 9.303 |
| DA | DR | 15.128 |
| N100 | R100 | 17.075 |
| N100 | M100 | 20.186 |
| RM100 | RUM100 | 21.265 |
| M100 | R100 | 24.593 |

Table 9 – Nemenyi's procedure indicates statistical difference ($\alpha = 0.05$) for comparing pairs of transformations for low dimensional datasets.

### 5.4.2 High dimensional datasets results

The second part summarizes the average $CQ_{A,T}$ of each algorithm-transformation pair for high dimensional datasets using a heat map showed in Figure 6. We show in Table 10 all quality measures values used to draw the heat map with the greater values for each measure highlighted in bold. Then, we used the Friedman ranking table followed by Nemenyi's procedure to present a statistical value comparison.

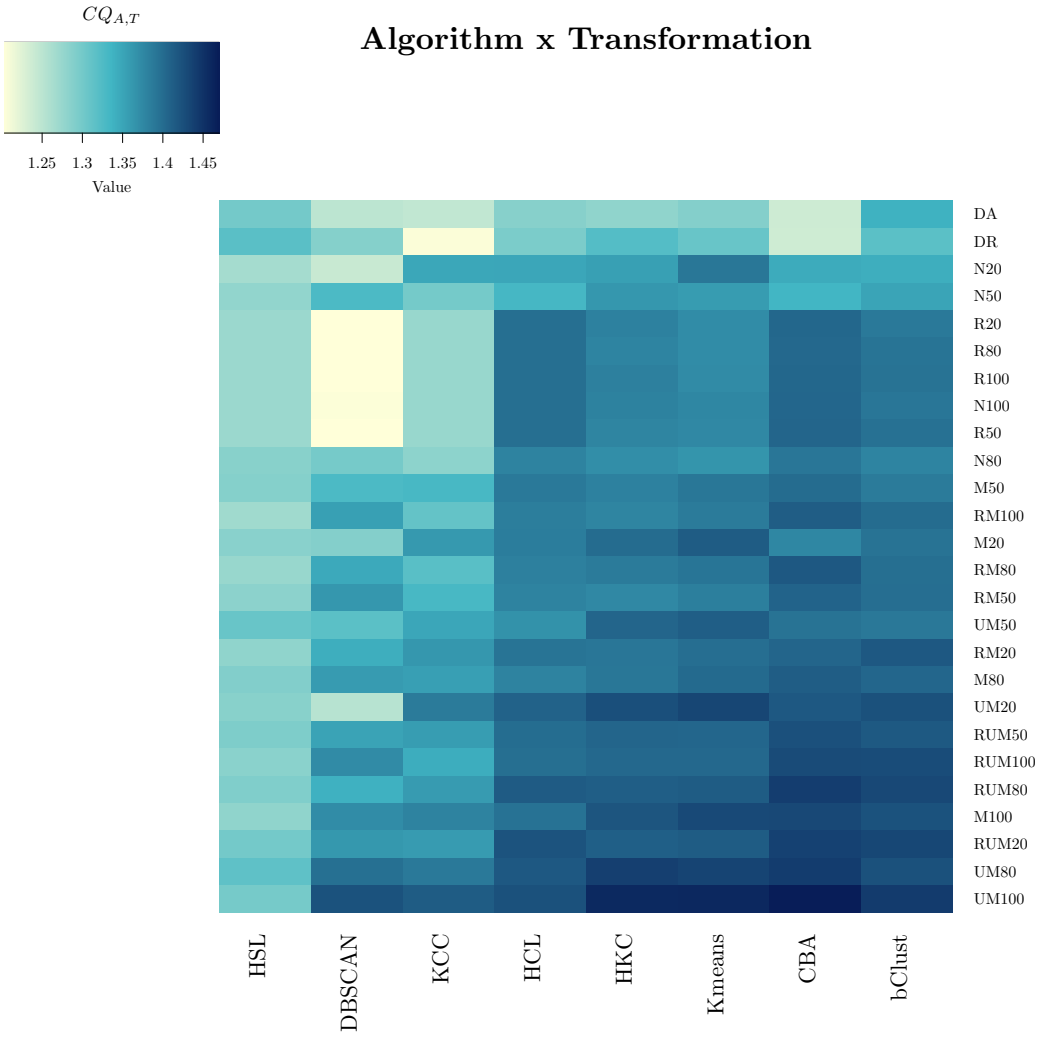


Figure 6 – Heat map of the cluster quality measure $CQ_{A,T}$ for datasets with high number of dimensions. Higher values are in darker colors indicating better performance of an algorithm-transformation pair.

Table 10 – All clustering measure values for each Algorithm-Transformation pair considering high dimensional datasets.

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| N100 | CBA | 0.562 | 0.163 | 0.00462 | 0.676 | 1.4 |
| N100 | HCL | 0.554 | 0.159 | 0.0077 | 0.677 | 1.4 |
| N100 | HSL | 0.534 | 0.0799 | 0.0127 | 0.647 | 1.27 |
| N100 | Kmeans | 0.548 | 0.173 | 0.00355 | 0.653 | 1.38 |
| N100 | bClust | 0.546 | 0.149 | 0.00841 | 0.689 | 1.39 |
| N100 | HKC | 0.605 | **0.287** | 0.0032 | 0.485 | 1.38 |
| N100 | KCC | 0.51 | 0.157 | 0.00312 | 0.605 | 1.28 |
| N100 | DBSCAN | 0.467 | 0.0851 | 0.0123 | 0.64 | 1.2 |
| N80 | CBA | 0.561 | 0.157 | 0.00455 | 0.669 | 1.39 |
| N80 | HCL | 0.553 | 0.152 | 0.00744 | 0.667 | 1.38 |
| N80 | HSL | 0.529 | 0.08 | 0.0129 | 0.664 | 1.29 |
| N80 | Kmeans | 0.542 | 0.168 | 0.00344 | 0.651 | 1.36 |
| N80 | bClust | 0.544 | 0.143 | 0.00797 | 0.683 | 1.38 |
| N80 | HKC | 0.602 | 0.274 | 0.00329 | 0.491 | 1.37 |
| N80 | KCC | 0.516 | 0.156 | 0.00347 | 0.607 | 1.28 |
| N80 | DBSCAN | 0.537 | 0.0999 | 0.0115 | 0.651 | 1.3 |
| N50 | CBA | 0.541 | 0.143 | 0.00401 | 0.648 | 1.34 |
| N50 | HCL | 0.535 | 0.139 | 0.00626 | 0.653 | 1.33 |
| N50 | HSL | 0.529 | 0.0796 | 0.0128 | 0.658 | 1.28 |
| N50 | Kmeans | 0.536 | 0.162 | 0.00363 | 0.657 | 1.36 |
| N50 | bClust | 0.533 | 0.131 | 0.00818 | 0.68 | 1.35 |
| N50 | HKC | 0.598 | 0.26 | 0.00301 | 0.503 | 1.36 |
| N50 | KCC | 0.523 | 0.156 | 0.0032 | 0.618 | 1.3 |
| N50 | DBSCAN | 0.535 | 0.112 | 0.00976 | 0.673 | 1.33 |
| N20 | CBA | 0.549 | 0.151 | 0.00335 | 0.641 | 1.35 |
| N20 | HCL | 0.54 | 0.145 | 0.00635 | 0.658 | 1.35 |
| N20 | HSL | 0.528 | 0.0814 | 0.0133 | 0.643 | 1.27 |
| N20 | Kmeans | 0.54 | 0.17 | 0.00375 | 0.678 | 1.39 |
| N20 | bClust | 0.523 | 0.123 | 0.00832 | 0.689 | 1.34 |
| N20 | HKC | 0.566 | 0.198 | 0.00378 | 0.588 | 1.36 |
| N20 | KCC | 0.532 | 0.165 | 0.00321 | 0.649 | 1.35 |
| N20 | DBSCAN | 0.502 | 0.1 | 0.00591 | 0.633 | 1.24 |
| RM100 | CBA | 0.56 | 0.168 | 0.00386 | 0.681 | 1.41 |
| RM100 | HCL | 0.547 | 0.151 | 0.00626 | 0.679 | 1.38 |

*Continued on next page*

Table 10 – *Continued from previous page*

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| RM100 | HSL | 0.532 | 0.0796 | 0.0109 | 0.647 | 1.27 |
| RM100 | Kmeans | 0.549 | 0.171 | 0.00313 | 0.665 | 1.39 |
| RM100 | bClust | 0.549 | 0.152 | 0.00696 | 0.694 | 1.4 |
| RM100 | HKC | 0.55 | 0.173 | 0.00312 | 0.653 | 1.38 |
| RM100 | KCC | 0.524 | 0.158 | 0.00273 | 0.626 | 1.31 |
| RM100 | DBSCAN | 0.542 | 0.12 | 0.0112 | 0.682 | 1.36 |
| RM20 | CBA | 0.553 | 0.165 | 0.00275 | 0.685 | 1.41 |
| RM20 | HCL | 0.543 | 0.152 | 0.00497 | 0.694 | 1.39 |
| RM20 | HSL | 0.532 | 0.0796 | 0.00965 | 0.66 | 1.28 |
| RM20 | Kmeans | 0.539 | 0.168 | 0.00253 | 0.69 | 1.4 |
| RM20 | bClust | 0.549 | 0.153 | 0.00534 | 0.711 | 1.42 |
| RM20 | HKC | 0.547 | 0.172 | 0.0023 | 0.67 | 1.39 |
| RM20 | KCC | 0.536 | 0.165 | 0.00199 | 0.66 | 1.36 |
| RM20 | DBSCAN | 0.537 | 0.112 | 0.00786 | 0.685 | 1.34 |
| RM50 | CBA | 0.56 | 0.167 | 0.00318 | 0.677 | 1.41 |
| RM50 | HCL | 0.545 | 0.15 | 0.00581 | 0.679 | 1.38 |
| RM50 | HSL | 0.532 | 0.0802 | 0.0104 | 0.661 | 1.28 |
| RM50 | Kmeans | 0.542 | 0.17 | 0.00289 | 0.669 | 1.38 |
| RM50 | bClust | 0.551 | 0.149 | 0.00553 | 0.693 | 1.4 |
| RM50 | HKC | 0.547 | 0.172 | 0.00274 | 0.653 | 1.38 |
| RM50 | KCC | 0.532 | 0.162 | 0.00251 | 0.636 | 1.33 |
| RM50 | DBSCAN | 0.54 | 0.12 | 0.00995 | 0.694 | 1.36 |
| RM80 | CBA | 0.56 | 0.171 | 0.00346 | 0.684 | 1.42 |
| RM80 | HCL | 0.544 | 0.151 | 0.00605 | 0.682 | 1.38 |
| RM80 | HSL | 0.531 | 0.0799 | 0.0106 | 0.653 | 1.27 |
| RM80 | Kmeans | 0.548 | 0.171 | 0.0031 | 0.67 | 1.39 |
| RM80 | bClust | 0.548 | 0.148 | 0.00595 | 0.695 | 1.4 |
| RM80 | HKC | 0.552 | 0.174 | 0.00321 | 0.658 | 1.39 |
| RM80 | KCC | 0.528 | 0.161 | 0.0026 | 0.629 | 1.32 |
| RM80 | DBSCAN | 0.534 | 0.115 | 0.011 | 0.687 | 1.35 |
| M100 | CBA | 0.569 | 0.175 | 0.00274 | 0.686 | 1.43 |
| M100 | HCL | 0.553 | 0.156 | 0.00489 | 0.682 | 1.4 |
| M100 | HSL | 0.535 | 0.0805 | 0.00879 | 0.657 | 1.28 |
| M100 | Kmeans | 0.563 | 0.184 | 0.00324 | 0.682 | 1.43 |
| M100 | bClust | 0.562 | 0.157 | 0.00476 | 0.7 | 1.42 |
| M100 | HKC | 0.572 | 0.195 | 0.00296 | 0.651 | 1.42 |

*Continued on next page*

Table 10 – *Continued from previous page*

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| M100 | KCC | 0.55 | 0.174 | 0.00265 | 0.654 | 1.38 |
| M100 | DBSCAN | 0.541 | 0.128 | 0.00941 | 0.695 | 1.37 |
| M20 | CBA | 0.545 | 0.154 | 0.00224 | 0.676 | 1.38 |
| M20 | HCL | 0.539 | 0.146 | 0.00508 | 0.695 | 1.39 |
| M20 | HSL | 0.527 | 0.0843 | 0.0108 | 0.663 | 1.29 |
| M20 | Kmeans | 0.537 | 0.163 | 0.00244 | 0.712 | 1.41 |
| M20 | bClust | 0.535 | 0.135 | 0.00612 | 0.719 | 1.39 |
| M20 | HKC | 0.547 | 0.17 | 0.00251 | 0.681 | 1.4 |
| M20 | KCC | 0.527 | 0.158 | 0.00201 | 0.674 | 1.36 |
| M20 | DBSCAN | 0.513 | 0.104 | 0.0049 | 0.667 | 1.29 |
| M50 | CBA | 0.559 | 0.156 | 0.00269 | 0.684 | 1.4 |
| M50 | HCL | 0.55 | 0.148 | 0.00473 | 0.687 | 1.39 |
| M50 | HSL | 0.53 | 0.0797 | 0.00982 | 0.668 | 1.29 |
| M50 | Kmeans | 0.542 | 0.161 | 0.00279 | 0.686 | 1.39 |
| M50 | bClust | 0.54 | 0.134 | 0.00537 | 0.708 | 1.39 |
| M50 | HKC | 0.553 | 0.175 | 0.00247 | 0.651 | 1.38 |
| M50 | KCC | 0.525 | 0.153 | 0.00233 | 0.651 | 1.33 |
| M50 | DBSCAN | 0.53 | 0.106 | 0.0068 | 0.686 | 1.33 |
| M80 | CBA | 0.563 | 0.167 | 0.00263 | 0.68 | 1.41 |
| M80 | HCL | 0.549 | 0.148 | 0.00494 | 0.677 | 1.38 |
| M80 | HSL | 0.532 | 0.0803 | 0.00942 | 0.669 | 1.29 |
| M80 | Kmeans | 0.552 | 0.173 | 0.00299 | 0.675 | 1.4 |
| M80 | bClust | 0.557 | 0.147 | 0.00497 | 0.697 | 1.41 |
| M80 | HKC | 0.563 | 0.183 | 0.0028 | 0.643 | 1.39 |
| M80 | KCC | 0.542 | 0.167 | 0.00258 | 0.644 | 1.36 |
| M80 | DBSCAN | 0.54 | 0.119 | 0.0089 | 0.692 | 1.36 |
| RUM100 | CBA | 0.557 | 0.167 | 0.0031 | 0.702 | 1.43 |
| RUM100 | HCL | 0.54 | 0.148 | 0.00532 | 0.704 | 1.4 |
| RUM100 | HSL | 0.532 | 0.0799 | 0.00933 | 0.663 | 1.28 |
| RUM100 | Kmeans | 0.543 | 0.168 | 0.00277 | 0.69 | 1.4 |
| RUM100 | bClust | 0.552 | 0.151 | 0.00538 | 0.72 | 1.43 |
| RUM100 | HKC | 0.548 | 0.17 | 0.00263 | 0.683 | 1.4 |
| RUM100 | KCC | 0.527 | 0.158 | 0.00215 | 0.657 | 1.34 |
| RUM100 | DBSCAN | 0.544 | 0.12 | 0.00928 | 0.699 | 1.37 |
| RUM20 | CBA | 0.557 | 0.165 | 0.00225 | 0.714 | 1.44 |
| RUM20 | HCL | 0.548 | 0.153 | 0.00433 | 0.717 | 1.42 |

*Continued on next page*

Table 10 – *Continued from previous page*

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| RUM20 | HSL | 0.533 | 0.0799 | 0.00838 | 0.68 | 1.3 |
| RUM20 | Kmeans | 0.54 | 0.166 | 0.00223 | 0.705 | 1.41 |
| RUM20 | bClust | 0.552 | 0.152 | 0.00409 | 0.726 | 1.43 |
| RUM20 | HKC | 0.545 | 0.169 | 0.00205 | 0.695 | 1.41 |
| RUM20 | KCC | 0.527 | 0.16 | 0.00165 | 0.671 | 1.36 |
| RUM20 | DBSCAN | 0.532 | 0.114 | 0.00776 | 0.709 | 1.36 |
| RUM50 | CBA | 0.556 | 0.167 | 0.00278 | 0.7 | 1.43 |
| RUM50 | HCL | 0.544 | 0.152 | 0.00486 | 0.699 | 1.4 |
| RUM50 | HSL | 0.533 | 0.0802 | 0.00889 | 0.671 | 1.29 |
| RUM50 | Kmeans | 0.544 | 0.17 | 0.00252 | 0.689 | 1.41 |
| RUM50 | bClust | 0.549 | 0.148 | 0.0047 | 0.716 | 1.42 |
| RUM50 | HKC | 0.549 | 0.172 | 0.00243 | 0.683 | 1.41 |
| RUM50 | KCC | 0.535 | 0.164 | 0.00193 | 0.657 | 1.36 |
| RUM50 | DBSCAN | 0.541 | 0.116 | 0.00843 | 0.688 | 1.35 |
| RUM80 | CBA | 0.561 | 0.167 | 0.00312 | 0.71 | 1.44 |
| RUM80 | HCL | 0.548 | 0.152 | 0.00522 | 0.71 | 1.42 |
| RUM80 | HSL | 0.532 | 0.0797 | 0.00886 | 0.671 | 1.29 |
| RUM80 | Kmeans | 0.548 | 0.171 | 0.00256 | 0.694 | 1.41 |
| RUM80 | bClust | 0.552 | 0.151 | 0.00509 | 0.724 | 1.43 |
| RUM80 | HKC | 0.55 | 0.173 | 0.00241 | 0.687 | 1.41 |
| RUM80 | KCC | 0.534 | 0.163 | 0.00206 | 0.66 | 1.36 |
| RUM80 | DBSCAN | 0.532 | 0.118 | 0.00877 | 0.683 | 1.34 |
| UM100 | CBA | 0.568 | 0.175 | 0.00157 | 0.725 | **1.47** |
| UM100 | HCL | 0.551 | 0.152 | 0.00429 | 0.717 | 1.42 |
| UM100 | HSL | 0.535 | 0.08 | 0.00703 | 0.677 | 1.3 |
| UM100 | Kmeans | 0.56 | 0.177 | 0.00269 | 0.72 | 1.46 |
| UM100 | bClust | 0.557 | 0.149 | 0.00463 | 0.733 | 1.44 |
| UM100 | HKC | 0.566 | 0.182 | 0.00257 | 0.708 | 1.46 |
| UM100 | KCC | 0.551 | 0.173 | 0.00192 | 0.69 | 1.41 |
| UM100 | DBSCAN | 0.549 | 0.128 | 0.00787 | **0.738** | 1.42 |
| UM20 | CBA | 0.548 | 0.161 | 0.00173 | 0.707 | 1.42 |
| UM20 | HCL | 0.542 | 0.153 | 0.00364 | 0.711 | 1.41 |
| UM20 | HSL | 0.524 | 0.0816 | 0.00858 | 0.672 | 1.29 |
| UM20 | Kmeans | 0.542 | 0.168 | 0.00241 | 0.722 | 1.43 |
| UM20 | bClust | 0.541 | 0.144 | 0.00492 | 0.734 | 1.42 |
| UM20 | HKC | 0.543 | 0.167 | 0.0023 | 0.713 | 1.43 |

*Continued on next page*

Table 10 – *Continued from previous page*

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| UM20 | KCC | 0.531 | 0.164 | 0.00179 | 0.691 | 1.39 |
| UM20 | DBSCAN | 0.494 | 0.104 | 0.00371 | 0.652 | 1.25 |
| UM50 | CBA | 0.549 | 0.153 | 0.00153 | 0.69 | 1.39 |
| UM50 | HCL | 0.534 | 0.136 | 0.00358 | 0.695 | 1.37 |
| UM50 | HSL | 0.531 | 0.0804 | 0.00858 | 0.689 | 1.31 |
| UM50 | Kmeans | 0.544 | 0.164 | 0.00236 | 0.703 | 1.41 |
| UM50 | bClust | 0.534 | 0.13 | 0.00496 | 0.721 | 1.39 |
| UM50 | HKC | 0.545 | 0.165 | 0.00222 | 0.694 | 1.41 |
| UM50 | KCC | 0.527 | 0.154 | 0.00172 | 0.667 | 1.35 |
| UM50 | DBSCAN | 0.524 | 0.102 | 0.00593 | 0.686 | 1.32 |
| UM80 | CBA | 0.561 | 0.168 | 0.00165 | 0.712 | 1.44 |
| UM80 | HCL | 0.549 | 0.152 | 0.00406 | 0.714 | 1.42 |
| UM80 | HSL | 0.534 | 0.0806 | 0.00762 | 0.693 | 1.31 |
| UM80 | Kmeans | 0.548 | 0.17 | 0.00257 | 0.716 | 1.44 |
| UM80 | bClust | 0.553 | 0.14 | 0.00438 | 0.727 | 1.42 |
| UM80 | HKC | 0.557 | 0.174 | 0.00238 | 0.707 | 1.44 |
| UM80 | KCC | 0.541 | 0.163 | 0.00179 | 0.683 | 1.39 |
| UM80 | DBSCAN | 0.542 | 0.12 | 0.00688 | 0.728 | 1.4 |
| R100 | CBA | 0.561 | 0.162 | 0.00462 | 0.676 | 1.4 |
| R100 | HCL | 0.554 | 0.159 | 0.0077 | 0.677 | 1.4 |
| R100 | HSL | 0.534 | 0.0799 | 0.0127 | 0.647 | 1.27 |
| R100 | Kmeans | 0.546 | 0.172 | 0.00355 | 0.652 | 1.37 |
| R100 | bClust | 0.55 | 0.152 | 0.00789 | 0.685 | 1.39 |
| R100 | HKC | 0.554 | 0.178 | 0.00321 | 0.648 | 1.38 |
| R100 | KCC | 0.51 | 0.157 | 0.00312 | 0.605 | 1.28 |
| R100 | DBSCAN | 0.467 | 0.0849 | 0.0123 | 0.639 | 1.2 |
| R80 | CBA | 0.561 | 0.162 | 0.00461 | 0.676 | 1.4 |
| R80 | HCL | 0.554 | 0.159 | 0.0077 | 0.677 | 1.4 |
| R80 | HSL | 0.534 | 0.0799 | 0.0127 | 0.647 | 1.27 |
| R80 | Kmeans | 0.545 | 0.171 | 0.00354 | 0.653 | 1.37 |
| R80 | bClust | 0.55 | 0.151 | 0.00816 | 0.685 | 1.39 |
| R80 | HKC | 0.552 | 0.177 | 0.00325 | 0.647 | 1.38 |
| R80 | KCC | 0.51 | 0.157 | 0.00312 | 0.605 | 1.28 |
| R80 | DBSCAN | 0.467 | 0.0849 | 0.0123 | 0.639 | 1.2 |
| R50 | CBA | 0.563 | 0.163 | 0.00464 | 0.676 | 1.41 |
| R50 | HCL | 0.554 | 0.159 | 0.0077 | 0.677 | 1.4 |

*Continued on next page*

Table 10 – *Continued from previous page*

| Algorithm | Transformation | Accuracy | ARI | Dunn Index | SWC | $CQ_{A,T}$ |
|---|---|---|---|---|---|---|
| R50 | HSL | 0.534 | 0.0799 | 0.0127 | 0.647 | 1.27 |
| R50 | Kmeans | 0.548 | 0.172 | 0.00354 | 0.653 | 1.38 |
| R50 | bClust | 0.549 | 0.152 | 0.00814 | 0.687 | 1.4 |
| R50 | HKC | 0.554 | 0.177 | 0.00333 | 0.645 | 1.38 |
| R50 | KCC | 0.51 | 0.157 | 0.00312 | 0.605 | 1.28 |
| R50 | DBSCAN | 0.467 | 0.0849 | 0.0123 | 0.639 | 1.2 |
| R20 | CBA | 0.562 | 0.163 | 0.00462 | 0.676 | 1.4 |
| R20 | HCL | 0.554 | 0.159 | 0.0077 | 0.677 | 1.4 |
| R20 | HSL | 0.534 | 0.0799 | 0.0127 | 0.647 | 1.27 |
| R20 | Kmeans | 0.545 | 0.171 | 0.00356 | 0.653 | 1.37 |
| R20 | bClust | 0.548 | 0.147 | 0.00829 | 0.686 | 1.39 |
| R20 | HKC | 0.554 | 0.177 | 0.00332 | 0.647 | 1.38 |
| R20 | KCC | 0.51 | 0.157 | 0.00312 | 0.605 | 1.28 |
| R20 | DBSCAN | 0.467 | 0.0849 | 0.0123 | 0.639 | 1.2 |
| DA | CBA | 0.542 | 0.113 | 0.0119 | 0.571 | 1.24 |
| DA | HCL | 0.538 | 0.114 | 0.0141 | 0.622 | 1.29 |
| DA | HSL | 0.536 | 0.0935 | **0.0205** | 0.65 | 1.3 |
| DA | Kmeans | 0.507 | 0.114 | 0.0112 | 0.658 | 1.29 |
| DA | bClust | 0.547 | 0.115 | 0.0156 | 0.662 | 1.34 |
| DA | HKC | 0.58 | 0.199 | 0.0101 | 0.492 | 1.28 |
| DA | KCC | 0.495 | 0.112 | 0.0112 | 0.628 | 1.25 |
| DA | DBSCAN | 0.537 | 0.107 | 0.00648 | 0.599 | 1.25 |
| DR | CBA | 0.543 | 0.115 | 0.0055 | 0.574 | 1.24 |
| DR | HCL | 0.534 | 0.112 | 0.00813 | 0.641 | 1.3 |
| DR | HSL | 0.529 | 0.079 | 0.0145 | 0.697 | 1.32 |
| DR | Kmeans | 0.534 | 0.128 | 0.00417 | 0.642 | 1.31 |
| DR | bClust | 0.545 | 0.108 | 0.00973 | 0.655 | 1.32 |
| DR | HKC | **0.613** | 0.248 | 0.00348 | 0.458 | 1.32 |
| DR | KCC | 0.51 | 0.117 | 0.00395 | 0.576 | 1.21 |
| DR | DBSCAN | 0.535 | 0.0943 | 0.0113 | 0.648 | 1.29 |

Table 11 presents the algorithm statistical rank produced by Friedman test for high dimensional datasets, where lower values indicates better results. The best ranked algorithm is CBA, and the worst ranked algorithm is HSL.

Table 13 presents the transformation statistical rank produced by Friedman test for high dimensional datasets, where lower values indicates better results. The best ran-

| **Algorithm** | **Ranking value** |
|---|---|
| CBA | 2.2 |
| bClust | 2.8 |
| $k$-means | 3.2 |
| HKC | 3.6 |
| HCL | 3.8 |
| KCC | 6.5 |
| DBSCAN | 6.8 |
| HSL | 7.2 |

Table 11 – Friedman-test ranking of all random transformations for high dimensional datasets. Lower values indicate better average clustering quality $CQ_{A,T}$ of algorithm $A$ for all transformations $T$.

| **Algorithms** | | $p$**-value** |
|---|---|---|
| **CBA** | **HSL** | **0.000** |
| **CBA** | **DBSCAN** | **0.000** |
| **HSL** | **bClust** | **0.000** |
| **CBA** | **KCC** | **0.000** |
| **HSL** | **$k$-means** | **0.000** |
| **bClust** | **DBSCAN** | **0.000** |
| **bClust** | **KCC** | **0.000** |
| **$k$-means** | **DBSCAN** | **0.000** |
| **HSL** | **HKC** | **0.000** |
| **HCL** | **HSL** | **0.000** |
| **$k$-means** | **KCC** | **0.000** |
| **HKC** | **DBSCAN** | **0.000** |
| **HCL** | **DBSCAN** | **0.000** |
| **HKC** | **KCC** | **0.001** |
| **HCL** | **KCC** | **0.003** |
| CBA | HCL | 0.418 |
| CBA | HKC | 1.014 |
| HCL | bClust | 3.949 |
| CBA | $k$-means | 3.949 |
| bClust | HKC | 7.211 |
| HSL | KCC | 7.211 |
| HCL | $k$-means | 9.403 |
| CBA | bClust | 9.403 |
| $k$-means | HKC | 14.937 |
| HSL | DBSCAN | 14.937 |
| $k$-means | bClust | 17.090 |
| KCC | DBSCAN | 17.090 |
| HCL | HKC | 20.555 |

Table 12 – $p$-values of Nemenyi's procedure for comparing pairs of algorithms ($\alpha = 0.05$) for high dimensional datasets. The procedure rejected null hypothesis which are highlighted in bold.

| Transformation | Ranking value |
|---|---|
| UM100 | 1.6 |
| UM80 | 2.5 |
| RUM20 | 4.8 |
| RUM80 | 6.0 |
| M100 | 7.3 |
| UM20 | 7.4 |
| RUM50 | 7.9 |
| RUM100 | 8.9 |
| M80 | 11.3 |
| RM20 | 11.4 |
| M20 | 13.3 |
| UM50 | 13.3 |
| RM80 | 13.8 |
| RM50 | 14.6 |
| M50 | 15.3 |
| RM100 | 15.6 |
| R50 | 17.6 |
| N100 | 17.9 |
| R100 | 18.0 |
| R20 | 18.8 |
| R80 | 19.3 |
| N80 | 19.3 |
| N20 | 20.9 |
| N50 | 21.0 |
| DR | 21.5 |
| DA | 22.3 |

Table 13 – Ranking values provided by Friedman test considering the results of all algorithms for high dimensional datasets. Lower values indicate better cluster quality $CQ_{A,T}$.

dom transformation is Uniform Mahalanobis 100% (UM100), and the worst is Density Attraction (DA).

We also applied the post hoc Nemenyi's procedure to compare transformations pairwise ($\alpha = 0.05$) as presented in Table 14, which shows only rejected null hypothesis for high dimensional datasets.

## 5.5 Result Analysis

In our analysis, we observed the average $CQ_{A,T}$ values, our measure for overall quality, for algorithms and transformations using the heat maps in Figure 5 and Figure 6. The best performing algorithm for all datasets was bClust, obtaining the highest average $CQ_{A,T}$ value for high and low dimensional datasets. The best performing transformation for all datasets was Mahalanobis 100% (UM100) transformation, which obtained the highest

| Transformation | | $p$-value |
|---|---|---|
| UM100 | DA | 0.000 |
| UM100 | DR | 0.000 |
| UM80 | DA | 0.000 |
| N50 | UM100 | 0.000 |
| N20 | UM100 | 0.000 |
| UM80 | DR | 0.000 |
| N50 | UM80 | 0.000 |
| N20 | UM80 | 0.001 |
| N80 | UM100 | 0.001 |
| UM100 | R80 | 0.001 |
| RUM20 | DA | 0.002 |
| UM100 | R20 | 0.002 |
| N80 | UM80 | 0.004 |
| RUM20 | DR | 0.004 |
| UM80 | R80 | 0.004 |
| UM100 | R100 | 0.006 |
| *N100 | UM100 | 0.007 |
| N50 | RUM20 | 0.007 |
| RUM80 | DA | 0.007 |
| UM80 | R20 | 0.007 |
| N20 | RUM20 | 0.008 |
| UM100 | R50 | 0.009 |
| RUM80 | DR | 0.016 |
| UM80 | R100 | 0.016 |
| *N100 | UM80 | 0.019 |
| UM80 | R50 | 0.025 |
| N50 | RUM80 | 0.029 |
| M100 | DA | 0.029 |
| N20 | RUM80 | 0.033 |
| UM20 | DA | 0.033 |
| N80 | RUM20 | 0.049 |
| RUM20 | R80 | 0.049 |

Table 14 – Nemenyi's procedure indicates statistical difference for comparing pairs of transformations ($\alpha = 0.05$) for high dimensional datasets.

average $CQ_{A,T}$ value for high and low dimensional datasets. The overall best performing combination of an algorithm with a transformed data set was the CBA algorithm with the Uniform Mahalanobis transformation 100% (UM100), which obtained the highest $CQ_{A,T}$ for high and low dimensional datasets.

Although DBSCAN is designed to find clusters of arbitrary shapes, it was the worst performing algorithm for low dimensional datasets, obtaining the lowest average $CQ_{A,T}$ value. We believe the main reason for bad performance of DBSCAN is its high parameter sensitivity and in our experiments we set them by an automatic procedure suggested by the authors of the package. Another reason is that transformations can indirectly change the density of clusters, impacting in the core of the algorithm.

For high dimensional datasets, the worst performing algorithm was HSL. Possibly, its quality on high-dimensional data set is affected by the usage of the single-linkage rule (distance to closest points in different clusters) to merge clusters.

The worst performing transformations for low and high dimensional datasets were density-based transformations, which obtained the lowest average $CQ_{A,T}$ values.

Although similar to $k$-means, KCC, a sophisticated algorithm, using author's source code and default parameters, is the worst performing of all ensemble algorithms. It even

obtained worse results than Hartigan and Wong $k$-means for all transformations and datasets, observing the heat maps.

These results indicate a better flexibility of the ensemble algorithms CBA and bClust to handle random transformations and their resulting arbitrary data shapes. We believe that this flexibility relies on which base algorithm the ensemble algorithm used. In both cases, the ensemble used a hierarchical algorithm complete link as base, and for CBA, it additionally used a $k$-means algorithm. Regarding non-ensemble algorithms, $k$-means and HCL obtained good results, which are comparable with ensemble algorithms like KCC and HKC.

Although the focus of this dissertation is on random transformations, we also tested the performance of the clustering algorithms, providing another point of view, which provides information on which algorithm to choose. In the rankings of algorithms (Tables 6 and 11), CBA is the best performing algorithm.

Focusing on the random transformations for low dimensional datasets, three random transformations obtained better ranking values than the original datasets (N100, Friedman, $\alpha = 0.05$) as showed in Table 8, which in descending order are: Uniform Mahalanobis 100% (UM100), Random Rotation 100% (R100), and Mahalanobis 100% (M100).

The ranking of transformations is significant, however no random transformation, considering the aggregate quality measure $CQ_{A,T}$ and low dimensional datasets, has statistically improved results when pairwise compared to the original dataset in Nemenyie's procedure, as showed in Table 9. For comparison purposes, we also executed the Bonferroni-Dunn's and the Hommel's pairwise comparisons, and the results of both tests agrees with Nemenyie's results. Thus, we do not include them in the experimental results and analysis.

For high dimensional datasets we observed seventeen random transformations which obtained better ranking values than the original datasets (N100, Friedman, $\alpha = 0.05$), as showed in Table 13. The four best transformations in descending order of quality are: Uniform Mahalanobis 100% (UM100), Uniform Mahalanobis 80% (UM80), Rotation followed by Uniform Mahalanobis 20% (RUM20), and Rotation followed by Uniform Mahalanobis 80% (RUM80).

Considering the aggregate quality measure $CQ_{A,T}$ and high dimensional datasets, only two of these seventeen transformations have shown statistical differences when pairwise compared to the original dataset. These transformations are: Uniform Mahalanobis 100% (UM100) and Uniform Mahalanobis 80% (UM80), as showed in lines marked by * in Table 14.

Among these seventeen transformations, we observed improvements provided by the dimension reducing procedure, where all variations of Mahalanobis and Uniform Mahalanobis obtained better results when compared to the original datasets. The best transformations also improved cluster quality for non-ensemble algorithms, indicating their

application is not restrict to ensemble clustering algorithms.

Specifically considering the transformation that motivated this work, Random Rotations (6) achieved similar results to original dataset, and no statistical differences were observed. We believe that the main reason for this behavior is that the transformation only rotates data points, keeping the relative position among the data points. This occurs because the use of the Euclidean distance measure, which is rotation invariant. Thus, this transformation does not have significantly positive or negative effects.

Random Rotation variations which was followed by a Mahalanobis transformation obtained good results when used in high dimensional datasets. However, a special case demonstrates the transformations which consider the sequence of transformations can produce degenerated datasets, as seen in Figure 7.

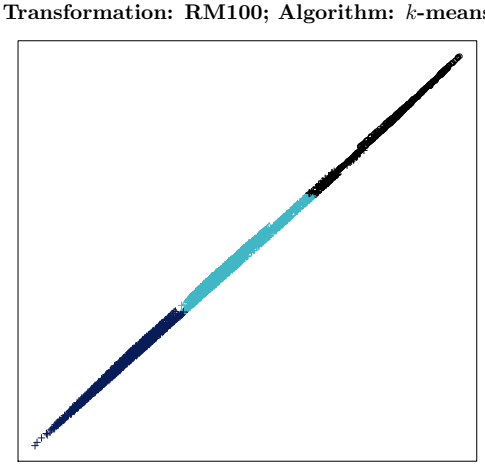


Figure 7 – Example of a degenerated dataset produced by Random Rotations followed by Mahalanobis transformation 100%.

Considering only density-based transformations, Density Attraction and Density Repulsion, did not provide good clusters. We believe this occurs mostly because it attracts the elements and mixes them producing one big cluster or repulsing producing a sparse space. This is a serious problem, specially for density algorithms like DBSCAN. In the artificial anchor dataset, which has only two dimensions, we observed Density Attraction can produce a transformed dataset with very dense and well-separated clusters, as seen in Figure 8. Clustering algorithms easily work with this transformed dataset and results in higher values for internal measures like Dunn Index and SWC, leading to the false conclusion that it was a good clustering. However, when evaluated with external measures, like Accuracy, it is possible to see the misleading conclusion.

In order to visually analyze the clustering results of the random transformations, we plotted the clustering results of the 8 transformations ordered by Accuracy and SWC. Figure 9 shows clustering results in decreasing order of quality by the external measure

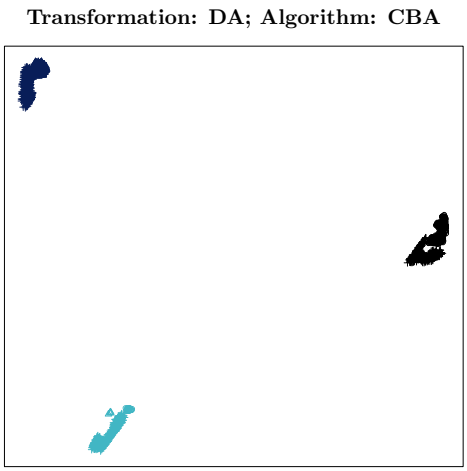


Figure 8 – Example of a misleading dataset produced by Density Attraction transformation.

Accuracy, where the first image is the best result, and the last image is the worst result. Figure 10 shows clustering results ordered by the external measure SWC.

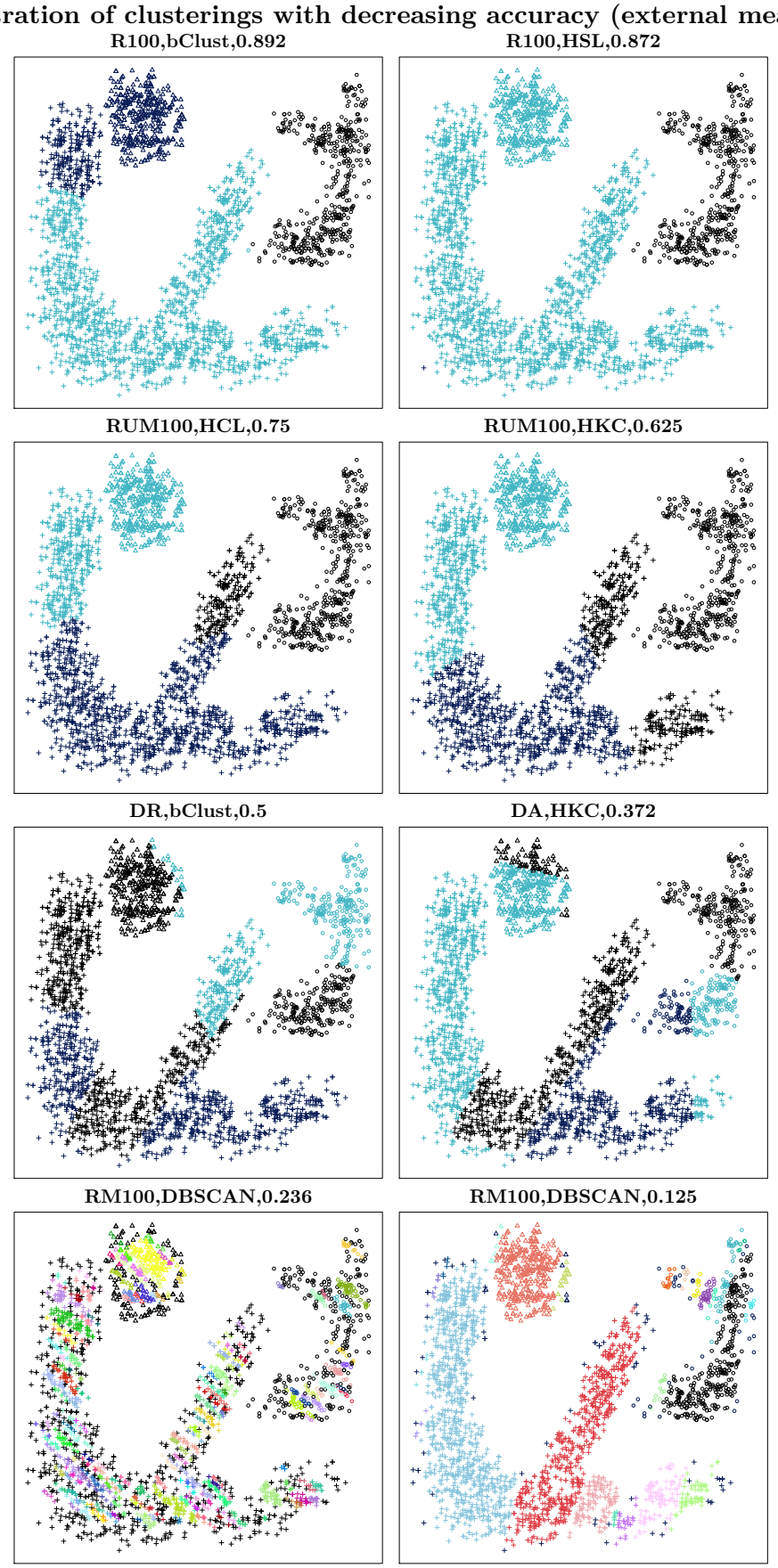


Figure 9 – The clusterings produced by 8 transformations on Artificial Anchor dataset, ordered by Accuracy. Each clustering configuration is represented with the triple: Transformation, Algorithm, and Accuracy value.

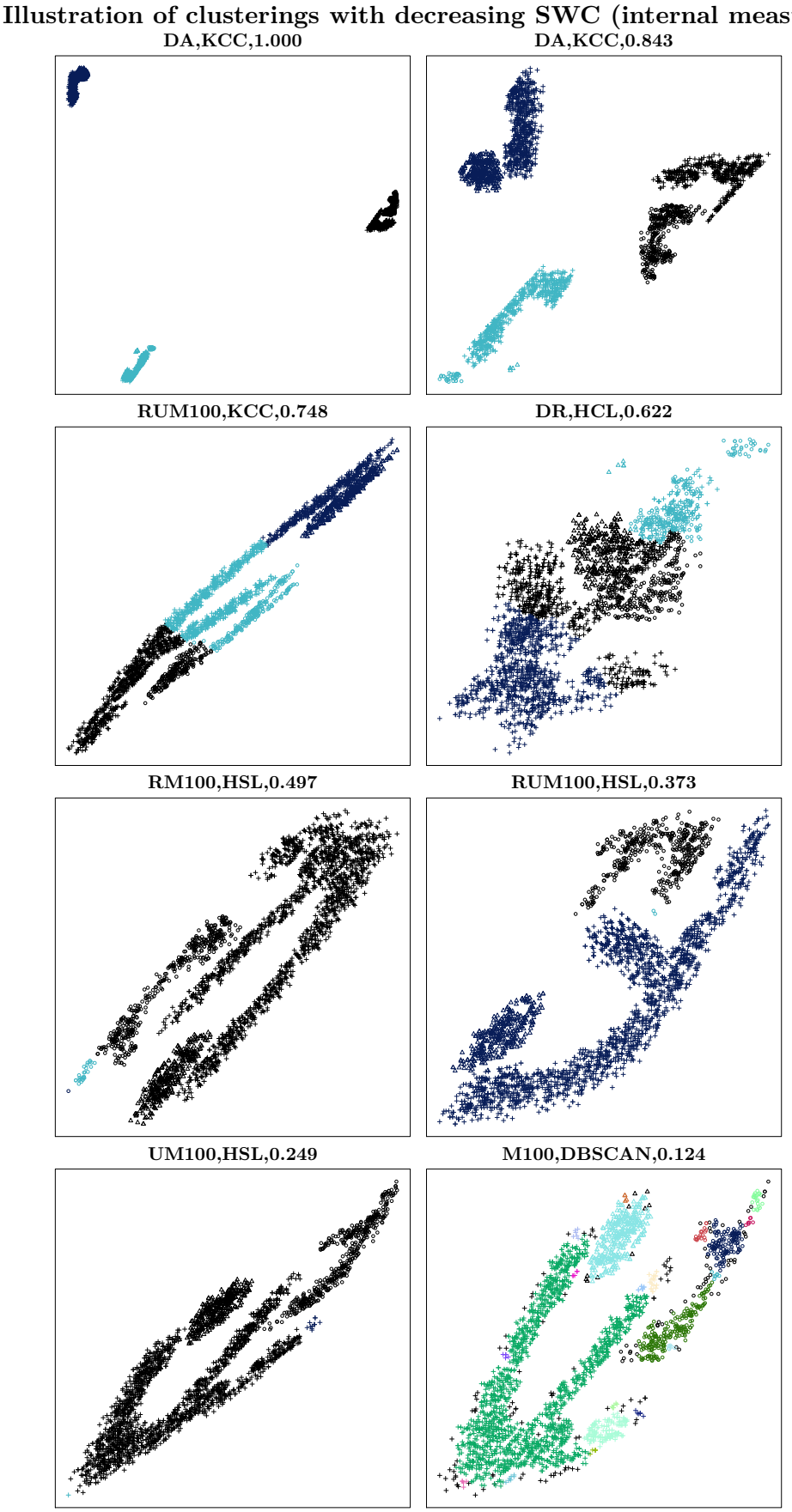


Figure 10 – The clusterings produced by 8 transformations on Artificial Anchor dataset, ordered by SWC. Each clustering configuration is represented with the triple: Transformation, Algorithm, and SWC value. Degenerated transforms favor high values for internal measures.

# Chapter 6

# Conclusion

This work considered the following hypothesis: applying random transformations to an unsupervised setting will lead to improvements on cluster quality? Thus, we studied ensemble techniques and their approaches in supervised and unsupervised settings.

Among the studied techniques, Blaser and Fryzlewicz (6) obtained improvements in random forest results by applying random rotations on input data. Following their successful results, we look up to slightly transform the data so that the original groups are not overly modified. Thus, we adapted their random transformation to the unsupervised setting and introduced two other types of transformations, based on Mahalanobis transformation and based on density.

We proceeded with empirical evaluation of the viability of applying these three types of random transformations on input data, aiming to produce ensemble cluster quality improvements.

In the empirical experiments we applied these three types of random transformations on 11 data sets obtained in UCI repository and 1 artificial data set. To evaluate the impacts of random transformations on clustering process, we selected 8 algorithms and introduced a mixed internal and external quality measure $CQ_{A,T}$. The $CQ_{A,T}$ measure evaluates each transformation $T$ per algorithm $A$ by summarizing 4 clustering evaluation measures for all data sets.

The experimental results were composed by heat maps containing the average $CQ_{A,T}$ and Friedman ranks for algorithms and transformations, separated in results for low dimensional data sets and high dimensional data sets. These results are easily reproducible using the randomClustering R package, which is freely available in online repository https://github.com/gabrielgdr/randomClustering.

Considering both low and high dimensional data sets, the best performing algorithm was bClust, the best performing transformation was Uniform Mahalanobis 100%, and the best overall performing combination of an algorithm with a transformed data set was the CBA algorithm with the Uniform Mahalanobis transformation 100%. Moreover, the statistical tests confirmed our research hypothesis and indicated seventeen random

transformations which obtained better ranking results when compared to non-transformed datasets. We also observed that density-based and Random Rotations followed by Mahalanobis transformations may produce degenerated data sets, leading to misleading cluster quality values.

Considering our result analysis, we have seen base algorithms being highly influential in the ensemble results. Therefore, we suggest a study on the influence of choosing certain algorithms as base for ensemble methods. Also, in order to prevent degenerated data sets, we believe that a pre-evaluation on transformed data using an internal measure could identify these degenerated data sets and discard it. Another gap is the use of different random transformations in each ensemble iteration instead of using the same transformed data set for all ensemble process.

# Bibliography

1 Minaei-Bidgoli B and Parvin H and Alinejad-Rokny H and Alizadeh H and Punch WF. Effects of resampling method and adaptation on clustering ensemble efficacy. **Artificial Intelligence Review**, v. 41, n. 1, p. 27–48, 2014. ISSN 1573-7462. Disponível em: <http://doi.org/10.1007/s10462-011-9295-x>.

2 Michalski RS and Carbonell JG and Mitchell TM. **Machine learning: An artificial intelligence approach**. San Francisco, CA, USA: Springer Publishing Company, Incorporated, 2013. ISSN 1431-1402. Disponível em: <http://doi.org/10.1007/978-3-662-12405-5>.

3 Schapire RE. The strength of weak learnability. **Machine learning**, Springer, v. 5, n. 2, p. 197–227, 1990. ISSN 1573-0565. Disponível em: <http://doi.org/10.1023/A:1022648800760>.

4 Witten IH and Frank E and Hall MA and Pal CJ. **Data Mining: Practical machine learning tools and techniques**. San Francisco, CA, USA: Morgan Kaufmann, 2016.

5 Breiman L. Bagging predictors. **Machine Learning**, Springer, v. 24, n. 2, p. 123–140, 1996. ISSN 1573-0565. Disponível em: <http://doi.org/10.1023/A:1018054314350>.

6 Blaser R and Fryzlewicz P. Random rotation ensembles. **Journal of Machine Learning Research**, v. 17, p. 1–26, 2016.

7 Siersdorfer S and Sizov S. Restrictive clustering and metaclustering for self-organizing document collections. In: ACM. **Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval**. New York, NY, USA, 2004. p. 226–233. ISBN 1-58113-881-4. Disponível em: <http://doi.org/10.1145/1008992.1009032>.

8 Brijnesh JJ. Condorcet's Jury Theorem for Consensus Clustering and its Implications for Diversity. **arXiv preprint arXiv:1604.07711**, 2016.

9 Topchy A and Jain AK and Punch W. A mixture model for clustering ensembles. In: SIAM. **Proceedings of the 2004 SIAM International Conference on Data Mining**. Florida, 2004. p. 379–390. Disponível em: <http://doi.org/10.1137/1.9781611972740.35>.

10 Leisch F. Bagged clustering. **SFB Adaptive Information Systems and Modelling in Economics and Management Science**, SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, 1999.

11 Silva GR and Albertini MK. Using multiple clustering algorithms to generate constraint rules and create consensus clusters. In: IEEE. **2017 Brazilian Conference on Intelligent Systems (BRACIS)**. Uberlandia, Brazil, 2017. p. 312–317. Disponível em: <http://doi.org/10.1109/BRACIS.2017.78>.

12 Wu J and Liu H and Xiong H and Cao J and Chen J. K-means-based consensus clustering: A unified view. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 27, n. 1, p. 155–169, 2015. ISSN 1041-4347. Disponível em: <http://doi.org/10.1109/TKDE.2014.2316512>.

13 Stoyanov K. **Hierarchical K-means clustering and its application in customer segmentation**. Tese (Doutorado) — University of Essex, UK, 2015.

14 Ronan T and Anastasio S and Qi Z and Sloutsky R and Naegle KM and Tavares PHSV. OpenEnsembles: A Python Resource for Ensemble Clustering. **Journal of Machine Learning Research**, v. 18, p. 1–6, 2018.

15 Jain AK and Murty MN and Flynn PJ. Data clustering: a review. **ACM Computing Surveys (CSUR)**, ACM, 31, n. 3, p. 264–323, 1999.

16 Lloyd S. Least squares quantization in PCM. **Information Theory, IEEE Transactions on**, 28, n. 2, p. 129–137, 1982.

17 Ester M and Kriegel H and S Jorg and Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Conference on Knowledge Discovery and Data Mining**. Portland, Oregon, USA: [s.n.], 1996. v. 96, n. 34, p. 226–231.

18 Hubert L and Arabie P. Comparing partitions. **Journal of Classification**, Springer, 2, n. 1, p. 193–218, 1985. ISSN 1432-1343. Disponível em: <http://doi.org/10.1007/BF01908075>.

19 Dunn JC. Well-separated clusters and optimal fuzzy partitions. **Journal of Cybernetics**, Taylor & Francis, v. 4, n. 1, p. 95–104, 1974. Disponível em: <http://doi.org/10.1080/01969727408546059>.

20 Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. **Journal of Computational and Applied Mathematics**, Elsevier, v. 20, p. 53–65, 1987. ISSN 0377-0427. Disponível em: <http://doi.org/10.1016/0377-0427(87)90125-7>.

21 Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the American Statistical Association**, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937.

22 Demšar J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, v. 7, n. Jan, p. 1–30, 2006.

23 Fred ALN and Jain AK. Data clustering using evidence accumulation. In: IEEE. **Pattern Recognition, 2002. Proceedings. 16th International Conference on**. Quebec, Canada, 2002. v. 4, p. 276–280. ISSN 1051-4651. Disponível em: <http://doi.org/10.1109/ICPR.2002.1047450>.

24 _____. Combining multiple clusterings using evidence accumulation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, USA, v. 27, n. 6, p. 835–850, 2005.

25 Minaei-Bidgoli B and Topchy A and Punch WF. Ensembles of partitions via data resampling. In: IEEE. **Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on**. Las Vegas, NV, USA, 2004. v. 2, p. 188–192. Disponível em: <http://doi.org/10.1109/ITCC.2004.1286629>.

26 Topchy A and Jain AK and Punch WF. Combining multiple weak clusterings. In: IEEE. **Data Mining, 2003. ICDM 2003. Third IEEE International Conference on**. Melbourne, Florida, USA, 2003. p. 331–338.

27 Strehl A and Ghosh J. Cluster ensembles-a knowledge reuse framework for combining multiple partitions. **The Journal of Machine Learning Research**, JMLR. org, 3, p. 583–617, 2003.

28 Efron B. Bootstrap methods: another look at the jackknife. **The annals of Statistics**, JSTOR, p. 1–26, 1979.

29 Moreau JV and Jain AK. The bootstrap approach to clustering. In: **Pattern Recognition Theory and Applications**. Berlin, Heidelberg: Springer, 1987. p. 63–71. Disponível em: <http://doi.org/10.1007/978-3-642-83069-3_6>.

30 Dudoit S and Fridlyand J. Bagging to improve the accuracy of a clustering procedure. **Bioinformatics**, Oxford University Press, v. 19, n. 9, p. 1090–1099, 2003.

31 Ben-Hur A and Elisseeff A and Guyon I. A stability based method for discovering structure in clustered data. In: **Pacific symposium on biocomputing**. Hawaii: [s.n.], 2001. v. 7, p. 6–17.

32 Barthélemy J and Leclerc B. The Median Procedure for Partitions. **Mathematics Subject Classification**, v. 19, p. 3–34, 1991.

33 Fern XZ and Brodley CE. Random projection for high dimensional data clustering: A cluster ensemble approach. In: **Proceedings of the 20th International Conference on Machine Learning (ICML-03)**. Washington, DC: [s.n.], 2003. p. 186–193.

34 Dempster AP and Laird NM and Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. **Journal of the Royal Statistical Society. Series B (methodological)**, JSTOR, p. 1–38, 1977.

35 Strehl A and Ghosh J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. **Journal of Machine Learning Research**, v. 3, n. Dec, p. 583–617, 2002.

36 Frossyniotis D and Likas A and Stafylopatis A. A clustering method based on boosting. **Pattern Recognition Letters**, Elsevier, v. 25, n. 6, p. 641–654, 2004.

37 Z, Y. et al. Incremental semi-supervised clustering ensemble for high dimensional data clustering. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 28, n. 3, p. 701–714, 2016.

38 Freund Y and Schapire RE. A desicion-theoretic generalization of on-line learning and an application to boosting. In: Springer. **European conference on computational learning theory**. London, UK, 1995. p. 23–37.

39 Diaconis P and Shahshahani M. On the eigenvalues of random matrices. **Journal of Applied Probability**, JSTOR, p. 49–62, 1994. ISSN 00219002. Disponível em: <http://doi.org/10.2307/3214948>.

40 Householder AS. Unitary triangularization of a nonsymmetric matrix. **Journal of the ACM (JACM)**, ACM, v. 5, n. 4, p. 339–342, 1958. ISSN 0004-5411. Disponível em: <http://doi.org/10.1145/320941.320947>.

41 Dunn JC. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Taylor & Francis, 1973.

42 Lichman M. **UCI Machine Learning Repository**. 2013. Accessed 19-July-2017. Disponível em: <http://archive.ics.uci.edu/ml>.

43 Topchy A and Minaei-Bidgoli B and Jain AK and Punch WF. Adaptive clustering ensembles. In: IEEE. **Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on**. Cambridge, UK, 2004. v. 1, p. 272–275.

44 Ostrovsky R and Rabani Y and Schulman LJ and Swamy C. The Effectiveness of Lloyd-Type Methods for the k-Means Problem. In: **Foundations of Computer Science, 2006. FOCS '06. 47th Annual IEEE Symposium on**. Washington, DC, USA: [s.n.], 2006. p. 165–176. ISSN 0272-5428.

45 JA, H.; MA, W. Algorithm as 136: A k-means clustering algorithm. **Journal of the Royal Statistical Society. Series C (Applied Statistics)**, [Wiley, Royal Statistical Society], v. 28, n. 1, p. 100–108, 1979. ISSN 00359254, 14679876. Disponível em: <http://www.jstor.org/stable/2346830>.

46 Mahalanobis PC. On the generalised distance in statistics. **Proceedings of the National Institute of Sciences of India, 1936**, p. 49–55, 1936.

47 Vendramin L and Campello RJGB and Hruschka ER. Relative clustering validity criteria: A comparative overview. **Statistical Analysis and Data Mining**, Wiley Online Library, v. 3, n. 4, p. 209–235, 2010. ISSN 1932-1864. Disponível em: <https://doi.org/10.1002/sam.10080>.

48 Conover WJ and Iman RL. On multiple-comparisons procedures. **Los Alamos Sci. Lab. Tech. Rep. LA-7677-MS**, p. 1–14, 1979.

# Appendix

APPENDIX A

# R packages for experimental results

This chapter shows the R packages to reproduce the experiments. We created a package aiming to facilitate the comparisons among various algorithms and transformations.

## A.1 CBA R code examples

Installing CBA algorithm package:

```
library(devtools)
install_git("https://github.com/gabrielgdr/CBA")
```

Run example of CBA algorithm with IRIS dataset:

```
library(cba)
cba(dataset=iris[,1:4], kmeans(iris[,1:4],centers=3)$cluster,
cutree(hclust(dist(iris[,1:4])),k=3),K=3)
```

The parameters are: *dataset* is the dataset which you want to clustering, *partition*1 is the cluster labels of first partitioning for other algorithm, *partition*2 is the the cluster labels of second partitioning for other algorithm and $K$ is the desired number of clusters for the ensemble partition.

## A.2 randomClustering R code examples

Installing randomClustering algorithm package:

```
library(devtools)
install_git("https://github.com/gabrielgdr/randomClustering")
```

Run example of randomClustering algorithm with IRIS dataset. The heat function produces the heat map.

```
library(randomClustering)
data <- randomclust(dataset=iris[,1:4],K=3)
heat(data)
```

The available algorithms are: “hclust”, “kmeans”, “dbscan”,“hdbscan”,“bclust”, “cba” and “hkclustering”. The available transformations are: “N100”,“RM100”,“RM20”,“RM50”,“RM80”, “M100”,“M20”, “M50”,“M80”,“RUM100”,“RUM20”,“RUM50”,“RUM80”,“UM100”, “UM20”,“UM50”,“UM80”,“R100”,“R20’,“R50’,“R80’,“DA”,“DR".

The default values is all algorithms and transformations. To proceed external evaluation, set the parameter *classes* with the classes ids.