SILESIAN UNIVERSITY OF TECHNOLOGY

FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND COMPUTER SCIENCE

PHD THESIS

DATA CLUSTERING WITH MIXTURES OF MULTIDIMENSIONAL DISTRIBUTIONS

mgr Mateusz Kania

promoter: prof. zw dr hab inż Andrzej Polański

The following thesis was financially supported by European Union funds project AIDA (Applied Integrative Data Analysis) POWR.03.02.00-00-I029.

Contents

1 Introduction

Unsupervised clustering is a group of algorithms that belong to scientific areas of data analysis, machine learning and artificial intelligence. They aim to solve problems of assigning a certain number of objects/items into groups, on the basis of some similarity/distance criterion/metrics between objects. Unsupervised clustering is a vital and fast developing area, with numerous applications in contemporary data science algorithms. In data scientific applications unsupervised clustering can be defined as an independent problem, with suitably specified quality criteria or as a part of some data analysis pipelines with many possible functions, e.g., data filtering, estimation of data structure, computing of some quality indexes of algorithms or their parts [1]. There are plethora of approaches to construction of unsupervised clusstering algorithms [2, 3], and also many surveys devoted to comparisons between different unsupervised algorithms [4].

Despite very intensive research already done in the area there are still problems requiring attention and deeper studies. One of the problem, which *is* very often encountered by data scientists in their research work is the choice of unsupervised clustering algorithm. With huge number of available methods often accompanied by software implementations the choice becomes difficult. The expensive and tedious solution is implementing and comparing large number of unsupervised clustering algorithms for a studied problem. The possibility, which can support decision on the choice of the algorithm is using results of studies comparing classes of algorithms. Two classes defined in[5] are model based clustering algorithms versus heuristic clustering algorithms. In this thesis we distinguish similar two classes of unsupervised clustering algorithms, which more rigorously are defined as follows:

- Probabilistic Model Based Algorithms: unsupervised clustering algorithms based on mixtures of multivariate distributions of feature vectors / observations vectors,

- Distance Function Based Algorithms: unsupervised clustering algorithms based on distance functions defined for pairs of feature vectors / observations vectors.

The above defined classed roughly correspond to classed of algorithms defined in [5]. The aim of the study in this thesis, as specifies below is comparison off algorithms for these two classes.

1.1 Aim

The aim of the PhD project, which was realized and described in this document was to derive, implement and compare models and related algorithms of unsupervised clustering. The work emphasises the usage of model-based algorithms using multivariate mixture distributions. We compare them with distance-based algorithms, k-means, k-medoids, agglomerative hierarchical clustering and fuzzy c-means.

To achieve this aim, we implemented two model based unsupervised clustering algorithms and four distance based unsupervised clustering algorithms. First model based algorithm, called Gaussian Mixture EM, is based on a multivariable mixture of normal distributions. The second one, Multinomial Mixture EM, is based on a mixture of multinomial distributions. The naming convention with EM stresses the fact that numerically these clustering algorithms rely on using expectation maximization (EM) algorithms for mixtures [6]. Distance based algorithms are: agglomeratvie hierarchical clustering, k-means, k-medoids, fuzzy c-means.

Along with implementing model-based and distance algorithms, we applied those algorithms to several data sets. Part of the data was simulated mixtures of multivariable distribution, both gaussian and multinomial. The other part, and the majority of data, consists of actual data downloaded from various, mostly publicly available sources. Having the results, we have used a few different metrics, like the Adjusted Rand Index, to quantify clustering results. Then, we presented our findings graphically, along with a brief description of the results.

1.2 Theses

1. Unsupervised clustering methods based on mixtures of distributions achieve their optimal performance when data statistics are consistent with true distributions.s of
2. Unsupervised clustering based on mixtures of distributions are competitive compared to distance based methods.
3. Applicability of clustering based on mixtures of distributions to practical problems relies on elaborating algorithmic implementation specialized for large sizes of datasets.

1.3 Original elements of the thesis and publications related to the thesis

The original elements and contributions of the submitted theses are as follows:

- Formulating algorithms for decomposing mixtures of multivariable Gaussian and multinomial distributions
- Elaborating software tools in an R language environment implementing unsupervised clustering algorithms based on mixtures of Gaussian and multinomial distributions. Op-

timizing the elaborated implementation such that it enables clustering of large datasets or order of hundreds of thousands of features/observations.

- Based on code sources available in the literature, implementing several distance-based clustering algorithms.
- Elaborating software tools implementing a collection of quality indexes of clustering in the R language environment
- Elaborating software tools for simulating multidimensional data of Gaussian or multinomial distributions
- Creating a collection of the real dataset for comparison study with possible variable structure and sizes of practical importance
- Performing comparison study for all analyzed clustering algorithms for the real and simulated dataset

Publications/conference presentation related with this thesis are:

Kania, M., Polański, A., Unsupervised clustering for detection of gene expression patterns in human cancers. 2022 , Recent Advances in Computational Oncology and Personalized Medicine, Volume 2, Silesian University of Technology Publishing House

The publication consist of the comparison of distance and model based algorithms in the gene expression data of different human cancers. We compared how various unsupervised algorithms can distinguish different cancer patterns.

Unsupervised clustering of gene data of TCGA patients by using mixtures of multidimensional Gaussian distributions,5th Advanced Online & Onsite Course on Data Science & Machine Learning | August 22-26, 2022, Castelnuovo Berardenga (Siena) Tuscany, Italy

This conference presentation describes the use of Gaussian Mixture Models, along with distance based algorithms, to compare and find patterns in various TCGA expressions.

In the papers below, unsupervised clustering techniques were used / implemented as parts of data analysis scenarios.

Mika, J., Tobiasz, J., Zyla, J., Papiez, A., Bach, M., Werner, A., Kozielski, M., Kania, M., Gruca, A., Piotrowski, D. and Sobala-Szczygieł, B., 2021. Symptom-based early-stage differentiation between SARS-CoV-2 versus other respiratory tract infections—Upper Silesia pilot study. Scientific reports, 11(1), pp.1-13.

Henzel, J., Tobiasz, J., Kozielski, M., Bach, M., Foszner, P., Gruca, A., Kania, M., Mika, J., Papiez, A., Werner, A. and Zyla, J., 2021. Screening Support System Based on Patient Survey Data—Case Study on Classification of Initial, Locally Collected COVID-19 Data. Applied Sciences, 11(22), p.10790.

Kania, M., Szymiczek, K., Labaj, W., Foszner, P., Gruca A., Szczęsna A., Polanski A., Computational methods for modelling cancer clonal evolution, 2022, (in press), POB2,

Artificial Intelligence and Data Processing, Silesian University of Technology Publishing House.

2 Algorithms based on mixtures of multivariate distributions

In this chapter, we describe model-based, unsupervised clustering algorithms, using mixtures of multivariable distributions, implemented in the thesis. In the beginning, we list probability distribution used in modeling. Then, we introduce models of mixtures and related concepts. Finally, we present algorithms constructed with the help of the expectation maximization (EM) method.

2.1 Foundation

2.1.1 Probability distribution models

Probability distributions more or less accurately reflects natural phenomenons around the world. If we want to study structure of mails to distinguish ham and spam, we can use properties of multinomial distributions [!citation]. To estimate shark attack across various seas, we can model it with Poisson distributions [7]. Then, when it comes to biological processes, like gene expression levels or bacteria lifespan, they tend to follow normal distributions.

The basis for unsupervised clustering is parametric, multivariate probability distribution models, which we describe in this subsection. There are two models suitable and often applied in multivariate distributions, multivariate Gaussian distribution and multinomial distribution.

2.1.2 Univariate and Multivariate Gaussian distribution

2.1.2.1 Univariate Gaussian (normal) distribution

Normal distribution has two parameters: μ which is mean value and σ which is a standard deviation. Mathematical notation of normal distribution is $X \sim N(\mu, \sigma^2)$. Independently of values of mean and standard deviation, all normal distributions have symmetric, bell-curved shape.

The standard normal distribution is related to the normal distribution and has mean equal 0 and standard deviation equal 1. Probability density function is given by the following formula[8]:

$$f(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \tag{2.1}$$

where:x is observation, μ is a mean, and σ is standard deviation

2.1.2.2 Multivariate Gaussian (normal) distribution

The multivariate normal distribution is a generalization of the univariate normal distribution. It may have n dimensions where $n \in \{0, \infty\}$. The multivariate normal distribution plays a fundamental role in a multivariate analysis, thanks to its various properties. While it is true that real data is never exactly multivariate normal, it is often useful to use normal density, because of its close approximation to the "true" population distribution.

Due to a central limit theorem, the sampling distributions of many multivariate statistics are approximately normal, despite of the form of the parent population. An n-dimensional random variable X with mean vector and covariance matrix Σ is said to have a non-singular multivariate normal distribution when its density function is of the form[8]:

$$f(x, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{2.2}$$

where:

$x = [x_1, x_2 ... x_M]$ is a vector of observations

$\mu = [\mu_1, \mu_2, \ldots, \mu_M]$ is vector of means

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1M} \\ \sigma_{12} & \sigma_{22} & \cdots & \sigma_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{M1} & \sigma_{M2} & \cdots & \sigma_{MM} \end{bmatrix}$$ is a covariance matrix

$|\Sigma|$ - denotes matrix determinant

x^T stands for vector x transposition.

2.1.2.3 Multivariable diagonal normal distribution

In the case of normal distributions an important aspect is high requirement for computational power and/or memory requirement for multidimensional case. If we have an observation given by a vector with $1000 = 10^3$ entries, the size of the covariance matrix will be dim(Σ) = $10^3 x 10^3$, which requires one million of records of memory space. This calls for more efficient approaches to handle this kind of data.

We define multivariable diagonal normal distribution as multivariable normal distribution with diagonal covariance matrix. Elements of observation vector x are uncorrelated so we use index "U" to distinguish it. Its probability density function is therefore defined as follows

$$f_U(x, \mu, \Sigma_U) = \frac{1}{(2\pi)^{n/2}|\Sigma_U|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma_U^{-1}(x-\mu)} \tag{2.3}$$

where:

$x = [x_1, x_2 ... x_M]$ is a vector of (uncorrelated) observations

$\mu = [\mu_1, \mu_2, \ldots, \mu_M]$ is vector of means

$\Sigma_U = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \sigma_M^2 \end{bmatrix}$ is a diagonal covariance matrix

$|\Sigma_U|$ - denotes matrix determinant

x^T stands for vector x transposition.

2.1.3 Bernoulli, binomial and multinomial distribution

In order to present multinomial distributions we begin from Bernoulli and binomial distributions. The reason is that binomial distribution is a generalization of Bernoulli and multinomial generalization of the binomial distribution. We can express the values of those distributions as non-negative integers.

2.1.3.1 Bernoulii distribution

If we consider a probabilistic experiment with two outcomes, it is called a Bernoulli trial. The result might a success with probability p or failure, with probability $1 - p$. An intuitive example of a Bernoulli trial might be a quality check of products in the factory. It might either be a success or a failure.

2.1.3.2 Binomial distribution

Binomial distribution describes results of repeating Bernoulli trials with probability of success p. The most common example is tossing a coin a finite number of times and more than one. We can assume that tail is a success and the head is a failure. The following formula gives binomial distribution probability function [8]:

$$Pr(k,n,p) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k}, \quad k = 0, 1, \ldots, n. \tag{2.4}$$

in the above $\left(\begin{array}{c} n \\ k \end{array} \right)$ is binomial coefficient, n is the number of trials, k number of successes, and p the probability of success.

2.1.3.3 Multinomial distribution

We can consider multinomial distribution as a multidimensional generalization of the binomial distribution. It inherits binomial properties and introduces new ones. The name "multi", suggests that we have more than two categories. The typical example of multinomial distribution is rolling a die fixed number of times. Whether the die is fair or not, each side, called category, has some probability p. As a different case, consider testing the durability of an intricate car component under crash conditions. The part may be damaged in different ways, each with distinct probabilities. We could apply the multinomial distribution to estimate the probability of a particular combination of failures.

The following equation describes the multinomial distribution probability function [9]:

$$Pr(N,x,p) = \frac{N!}{x_1!\cdots x_M!} p_1^{x_1}\cdots p_M^{x_M} = N!\prod_{i=1}^{M}\left(\frac{p_i^{x_i}}{x_i!}\right), \tag{2.5}$$

in the above, N is the number of trials

$x = [x_1, x_2 ... x_M]$ is one observation vector of recorded counts of categories,

$p = [p_1, p_2, ... p_M]$ is vector of probabilities of categories

$$x_1 + x_2 + ... + x_M = N \tag{2.6}$$

$$p_1 + p_2 + ... + p_M = 1 \tag{2.7}$$

2.1.4 Mixture distributions

To start with mixture distributions, it is worth to read the story of the biologist Raphael Weldon and mathematician Karl Pearson, which presents probably the first matematical approach to a mixture distributions [10].

The simple example of the mixture distribution might be shown upon different races of a dogs. As we already know, there is siginificant difference between e.g labrador and chiuwawa. Those differences accounts for weight, height, but also size of organs as well.

In general, mixture models provides broader spectrum of information than single distributions.

In the medicine they might be used to analyze gene expression data or in early drug development [11]. They are also successfully used to approximate specificity and sensitivity in the case of lack of golden standard. Albeit mixture models are not limited only to biology and medicine. Fields like astronomy, psychology or engineering, to name a few, also benefits from them. Their flexibility and usefulness is described in several books [12][11]

In our study we are focused on mixture models that are more precisely called finite mixture models.

A mixture distributions is a mixture of at least two distributions of the same type or, possibly of different types. As an example, the discrete case of distribution of weight in the population of adults might be expressed as follows:

$$w(weight) = p(man) * w_{man}(weight) + p(woman)w_{woman}(weight|woman)$$

Where the probabilities p(man), p(woman) are also called mixing probabilities w_{man} and w_{woman} are probability density functions of weights of man and woman.
For mixtures with arbitrary numbers of components and obsevation of the form of scalars or vestoer the definition of mixture distribution can be expressed as follows. If a random variable or random vector x, takes values in sample space, Ω, a K component mixture distribution $f(x)$ is represented as follows:

$$f(x) = \alpha_1 f_1(x) + \cdots + \alpha_K f_K(x) \qquad (x \in \Omega) \tag{2.8}$$

where:
$\alpha_j > 0, \quad j = 1, \ldots, K; \quad \alpha_1 + \cdots \alpha_K = 1$, are called mixing proportions or component weights,
$f_j(x), j = 1, \ldots, K$ are probability density functions of component distributions
We say that X has a finite mixture distribution and $f(x)$ is a finite mixture density function.
Most often component densities have forms dependent on paramters,

$$f_k(x) = f_k(x \mid \Theta_k)$$

thus we write

$$f(x|\Theta) = \alpha_1 f_1(x \mid \Theta_1) + \cdots + \alpha_K f_K(x \mid \Theta_K) = \sum_{k=1}^{K} \alpha_k f_k(x \mid \Theta_k) \tag{2.9}$$

where:
α_k- mixing proportion of k-th component
Θ_k- parameteres of of k-th component
$f_k(x \mid \Theta_k)$ - probability density function of data x given parameters Θ_k

2.1.4.1 Mixtures of normal distributions

Mixtures that consist of only normal distributions are either univariate or multivariate. In simple words, mixture of univariate distribution consist of one only feature, when multivariate has many. Sometimes we distingusigh also bivariate normal distributions.

Mixture of univariate normal distribution It is a mixture where each component that belongs to K follows its respective normal dsistribution.

The common example of mixture of univariate normal distribution is height of some population. Although the way of how we will create clusters and interpret them, depends on many circumstances. Let's consider situation, when we want to distinguish between height of a man and a woman. When our data is labeled, we can use statistical test to check if there are statistically significant differences. In the case of the unlabeled data, we are no longer able to relay on statistical tests. We could no longer ask about differences between two groups, as we are presented with one feature. At this point, the question we want to answer is whether there are two subgroups in the given data. Potential trap here is that we will find as many clusters, as we want, most of the time. We need further analysis to verify how significant those results are for our research.

$$f(x|\mu,\sigma,\alpha) = \sum_{k=1}^{K} \alpha_k f_k(x|\mu_k,\sigma_k) \tag{2.10}$$

where:

$f(x|\mu,\sigma,\alpha)$ is probability density function of normally distributed random variable X given parameters,

arguments of the density function on the left hand side are

$\mu = [\mu_1,\mu_2,...\mu_K]$,

$\sigma = [\sigma_1,\sigma_2,...\sigma_K]$,

$\alpha = [\alpha_1,\alpha_2,...\alpha_K]$,

parameters of components are

μ_k is mean of component k

$\sigma_k$is standard deviation of component k,

$\alpha_k$is a mixing proportion of component k

Mixture of multivariate normal distributions Each component is a multivariable normal density function. Probability density function of the mixture is as follows

$$f(x|\mu,\Sigma,\alpha) = \sum_{k=1}^{K} \alpha_k f_k(x|\mu_k,\Sigma_k) \tag{2.11}$$

where:

$f(x|\mu,\Sigma,\alpha)$ is probability density function of normally distributed random variable X given parameters,

arguments of the density function on the left hand side are

$\mu = [\mu_1,\mu_2,...\mu_K]$ matrix composed of vectors of means

$\Sigma = [\Sigma_1,\Sigma_2,...\Sigma_K]$ list of covariance matrices

$\alpha = [\alpha_1,\alpha_2,...\alpha_K]$, vector of mixing proportions

parameters of components are

$\mu_k = [\mu_{k1}, \mu_{k2}, ...\mu_{KM}]$ vector of means of component k,

$$\Sigma_k = \begin{bmatrix} \sigma_{k,11} & \sigma_{k,12} & \cdots & \sigma_{k,1M} \\ \sigma_{k,12} & \sigma_{k,22} & \cdots & \sigma_{k,2M} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{k,M1} & \sigma_{k,M2} & \cdots & \sigma_{k,MM} \end{bmatrix}$$ covariance matrix of component k,

$\alpha_k$is a mixing proportion of component k

Mixture of diagonal multivariate normal distributions For the reason already stated previously we focus on the special case of multivariable normal distributions - diagonal multivariable normal distribution. Probability density function of the mixture is as follows

$$f(x|\mu, \Sigma^U, \alpha) = \sum_{k=1}^{K} \alpha_k f_k(x|\mu_k, \Sigma_k^U) \tag{2.12}$$

where:

$f(x|\mu, \Sigma^U, \alpha)$ is probability density function of normally distributed random variable X given parameters,

arguments of the density function on the left hand side are

$\mu = [\mu_1, \mu_2, ...\mu_K]$ matrix composed of vectors of means

$\Sigma^U = [\Sigma_1^U, \Sigma_2^U, ...\Sigma_K^U]$ list of covariance matrices

$\alpha = [\alpha_1, \alpha_2, ...\alpha_K]$, vector of mixing proportions

parameters of components are

μ_k vector of means of component k,

$$\Sigma_k^U = \begin{bmatrix} \sigma_{k,1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{k,2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \sigma_{k,M}^2 \end{bmatrix}$$ is a diagonal covariance matrix of component k,

$\alpha_k$is a mixing proportion of component k

2.1.4.2 Mixtures of multinomial distributions

Mixtures of binomial distributions Binomial mixtures consist of more than one binomial distribution and can be described by the following probability function

$$Pr(l, N, p) = \sum_{k=1}^{K} \alpha_k \binom{N}{l} p_k^l (1 - p_k)^{N-l}, \quad l = 0, 1, \ldots, n, \tag{2.13}$$

where

$\begin{pmatrix} N \\ l \end{pmatrix}$ is a binomial coefficient

N is number of trials

l is number of successes

α_k is a mixing proportion of k-th component

p_k^l is probability of success

$(1-p_k)^{N-l}$ is probability of failure

The above formula described one observation (i.e., l successes) taken from mixture of k binomial distributions. We assume that we record D observations l_1, l_2,..., l_D.

Mixtures of multinomial distribution Below we present the formula for probability function of $k$component mixtures of multinomial distributions

$$Pr(N,x,p) = \frac{N!}{x_1!\cdots x_M!} p_1^{x_1}\cdots p_M^{x_M} = N!\prod_{i=1}^{k}\left(p_{kM}^{x_M}/x_{kM}!\right), \tag{2.14}$$

$$x_1 + x_2 + \ldots + x_M = N \tag{2.15}$$

$$p_1 + p_2 + \ldots + p_M = 1 \tag{2.16}$$

in the above, N is the number of observations

k is the number of components in the mixture

$x = [x_1, x_2 \ldots x_M]$ is vector of observed counts of observed categories,

$p = [p_1, p_2, \ldots p_M]$ is vector of probabilities of categories

2.2 EM algorithm

Clustering methods, that are based on a matrix of similarity or dissimilarity measures between objects. The purpose was to segregate data in such way, that objects in one group were similar to themselves but different from the other groups. However this approach still lefts many questions unanswered. Which clustering method should we use for certain data? What to do with outliers or objects that did not fall into any groups? How do we assess uncertainty about clustering results?

Statistical approach that uses probability distributions might address some of those questions. To use it for clustering problem, we can use finite mixture model (FMM). Inside this model, each mixture component is described by a probability distribution. One of the first successful method that used FMM and answered some of the questions was presented in 1950s by

Paul Felix Lazarsfeld. The model was called latent class model and it was using multivariate discrete data as input. The model was based on assumption, that within each group its characteristics was statistically independent [(Lazarsfeld, 1950a,c)]. In 1963, Wolfe introduced model for clustering continuous data, along with software NORMIX that he was developing. It allowed to analyze mixtures of multivariate normal distributions. In his proposal, estimation of model parameters was done by maximum likelihood using Expectation-Maximization algorithm. It was followed with relevant theory in the next years (Wolfe, 1965, 1967, 1970). [5].

Before going into the details of how the EM actually works, we can list its basic components upon which it is built.

2.2.1 Algorithm in general form

EM gained worldwide popularity after publishing it in 1977 by Arthur Dempster, Nan Laird, and Donald Rubin. The article "Maximum Likelihood from Incomplete Data via EM algorithm" presented a structured and general way to parameter estimation based on the maximum-likelihood method. The algorithm was called **EM** since each step consisted of an **expectation** step, followed by a **maximization** step.[13].

The Expectation-Maximization (EM) algorithm is an iterative method for finding maximum likelihood (ML) estimates in problems with latent or hidden variables. As already mentioned, latent variables are not directly observed but are thought to impact the observed data.

The EM algorithm alternates between two steps: the E-step (Expectation), where the expected value of the latent variables given the current estimates of the parameters is computed. Then is the M-step (Maximization), where the parameters are re-estimated based on the expected values of the latent variables calculated in the E-step. These two steps are repeated until convergence occurs, i.e., the parameters' estimates stop changing.

The EM algorithm has become popular in machine learning and statistics because it can be applied to many models. The list includes Gaussian mixture models, hidden Markov models, missing data, truncated distributions, censored or grouped data. As McLachlan points out, also in statistical models such as random effects, convolutions, log-linear models, latent class and latent variable structures. However, one of the limitations of the EM algorithm is that it can be sensitive to the initial parameter estimates and may converge to a local maximum of the likelihood function instead of the global maximum. [6].

First, all EM algorithms utilize probability distributions to carry out results. We already mentioned how important they are in describing natural phenomena. We are particularly interested in the normal and multinomial distribution, as they are widely observed. In addition they allow us to assess the uncertainty of clustering results.

The second principle, but not less important, is Bayes's Theorem. Although Expectation-Maximization is not using full Bayesian Inference, its expectation step is based on it. Precisely on the following equation:

$$p\left(k \mid x_M, \Theta^g\right) = \frac{\alpha_k^g p_k\left(x_M \mid \theta_k^g\right)}{p\left(x_M \mid \Theta^g\right)} = \frac{\alpha_k^g p_k\left(x_M \mid \theta_k^g\right)}{\sum_{k=1}^{M} \alpha_k^g p_k\left(x_M \mid \theta_k^g\right)} \tag{2.17}$$

where:

α_k which is prior probability, can be though of as mixing proportion or probability of each mixture component,

$p_k\left(x_M \mid \theta_k^g\right)$ is some probability function of x_M given guessed parameter θ_k^g

The third important thing is iterative nature of EM. Data will be fitted during n-th iterations until convergence occurs.

2.2.1.1 Hidden variable

The main idea behind EM algorithms is existing of a hidden or latent variable which is never directly observed. It is often referred to as a factor that combines several other variables and indicators. An example of such a hidden variable is the existence of groups among observations. Given a multidimensional data matrix that does not contain labels, we do not know to which group a given observation belongs. This information is hidden. However, we can reproduce this variable more or less accurately using various data grouping methods. Instead of precision, we can also look at it as a variable that sets trends in the data. It gathers around itself the most similar observations with a similar structure. Depending on the approach, we can use them in different ways. As in this paper, on the one hand, we can check how well we can separate the tumours into their original groups. On the other hand, a latent variable may suggest that certain types of cancer are more similar to each other.

2.2.1.2 Maximum-likelihood

Assume we have density function $p(x \mid \Theta)$ in which x is conditionally described by parameters Θ. Data set size is $N$and is described by $p(x \mid \Theta)$. x can be for example domestic cat's weight measure and Θ are mean $\mu$and standard deviation σparameters.

$$p(\mathscr{X} \mid \Theta) = \prod_{i=1}^{N} p\left(\mathbf{x}_i \mid \Theta\right) = \mathscr{L}(\Theta \mid \mathscr{X}) \tag{2.18}$$

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \mathscr{L}(\Theta \mid \mathscr{X}) \tag{2.19}$$

To run the algorithm, we start from initialization. Next, it iterates over two proceeding steps: Expectation step (E-Step) and Maximization step (M-step), until it reaches convergence [14].

2.2.1.3 General EM

By general EM we the A joint density function

$$p(\mathbf{z} \mid \Theta) = p(\mathbf{x}, \mathbf{y} \mid \Theta) = p(\mathbf{y} \mid \mathbf{x}, \Theta) p(\mathbf{x} \mid \Theta) \tag{2.20}$$

$$p(\mathbf{x} \mid \Theta) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{x} \mid \theta_i) \tag{2.21}$$

2.3 Multivariate Gaussian Mixture EM

2.3.1 Implementation

Pseudocode

Algorithm 2.1 Gaussian Mixture Model EM

```
> Input:
   k = int
   data = Matrix(m,n)
> Parameters initialization:
   temp_k = sample from 1:k times m
   alphas = sample 1:k from Unif(0.1,1)
   alphas = alphas/sum(alphas)
   for i in 1:k {
      means[i] = mean(data[temp_k == i])
      vars[i] = variance(data[temp_k == i])
   }
> Expectation:
   while {change > change_limit & itr <= itr_limit)
      alphas_old = alphas
      means_old = means
      vars_old = vars
      for i in 1:k{
         p1 = (data - means[i,])^2
         p2 = colSums(p1/vars[i,])/2
         p3 = sum(log((vars[i,])))/2 - (n/2)log(2pi)
         out = log(alphas[i]) - p3 - p2
      }
   }
   > Maximization:
```

2.3.2 Initialization

The initialization step requires first estimates of the parameters. In a multivariate normal distribution, it will be μ_k (mean), σ_k^2 (variance), π_k(mixing proportion), of the kth component. There are various ways to obtain those estimates. One of the popular ones is to use the k-means algorithm. It might also be done randomly by other clustering methods like hierarchical clustering using hierarchical.

2.3.3 Expectation step (E-step)

After initialization, we proceed to the expectation step of the EM-algorithm. Equation for the **E-step** can be written as follows:

$$p(k|n) = \frac{\alpha_k^i f(x_n, \mu_k^i, \sigma_k^i)}{\Sigma_{k=1}^K \alpha_k^i f(x_n, \mu_k^i, \sigma_k^i)} \tag{2.22}$$

where:
α_k^i- probability (mixing proportion) of k-th component
$f(x_n, \mu_k^i, \sigma_k^i)$ - normal density function of observation x_n and paramteres μ_k^i and σ_k^i
In the numerator, from the normal PDF function, we obtain the likelihoods of the values x_n (observations) over μ_k and σ_k. Next, we multiply it by the probability of occurrence (or mixing proportions). The results should be interpreted as the probability that values x_n belongs to the normal distribution with parameters μ_k and σ_k. We are repeating this step for all other parameters. The denominator is used to normalize the results so that they can sum up to 1. It consists of the sum of all posterior distributions.

2.3.4 Maximization step (M-step)

In the maximization step, posterior probabilities from the E-step are used to obtain new parameters

$$\mu_k^{i+1} = \frac{\sum_{n=1}^N p(k|n) x_n}{\sum_{n=1}^{N_k} p(k|n)} \tag{2.23}$$

$$\alpha_k^{i+1} = \frac{\sum_{n=1}^N p(k|n) x_n}{N} \tag{2.24}$$

$$\alpha_\ell^{new} = \frac{1}{N} \sum_{i=1}^N p\left(\ell \mid x_i, \Theta^g\right) \tag{2.25}$$

$$\mu_\ell^{new} = \frac{\sum_{i=1}^N x_i p\left(\ell \mid x_i, \Theta^g\right)}{\sum_{i=1}^N p\left(\ell \mid x_i, \Theta^g\right)} \tag{2.26}$$

$$\Sigma_{\ell}^{new} = \frac{\sum_{i=1}^{N} p\left(\ell \mid x_i, \Theta^g\right)\left(x_i - \mu_{\ell}^{new}\right)\left(x_i - \mu_{\ell}^{new}\right)^T}{\sum_{i=1}^{N} p\left(\ell \mid x_i, \Theta^g\right)} \tag{2.27}$$

where:

μ_k^{i+1}- new mean value of the kth component

$(\sigma_k^{i+1})^2$ - new variance of the kth component

π_k^{i+1} - new probability (mixing proportion) of the kth component

Expectation and maximization step are repeated until convergence occurs.

$$p\left(k \mid x_n, p^{\text{old}}\right) = \frac{\alpha_k^{\text{old}} f_k\left(x_n, p^{\text{old}}\right)}{\sum_{\kappa=1}^{K} \alpha_{\kappa}^{\text{old}} f_{\kappa}\left(x_n, p^{\text{old}}\right)} \tag{2.28}$$

$$\alpha_k^{\text{new}} = \frac{\sum_{n=1}^{N} p\left(k \mid x_n, p^{\text{old}}\right)}{N} \tag{2.29}$$

$$\mu_k^{\text{new}} = \frac{\sum_{n=1}^{N} x_n p\left(k \mid x_n, p^{\text{old}}\right)}{\sum_{n=1}^{N} p\left(k \mid x_n, p^{\text{old}}\right)}, \quad k = 1, 2, \ldots, K \tag{2.30}$$

$$(\sigma_k^{\text{new}})^2 = \frac{\sum_{n=1}^{N} \left(x_n - \mu_k^{\text{new}}\right)^2 p\left(k \mid x_n, p^{\text{old}}\right)}{\sum_{n=1}^{N} p\left(k \mid x_n, p^{\text{old}}\right)}, \quad k = 1, 2, \ldots, K \tag{2.31}$$

$$\alpha_k^{\text{new}} = \frac{\sum_{n=1}^{N} p\left(k \mid x_n^1, \ldots, x_n^M, p^{\text{old}}\right)}{N} \tag{2.32}$$

$$\mu_{k,m}^{\text{new}} = \frac{\sum_{n=1}^{N} x_n^m p\left(k \mid x_n^1, \ldots, x_n^M, p^{\text{old}}\right)}{\sum_{n=1}^{N} p\left(k \mid x_n^1, \ldots, x_n^M, p^{\text{old}}\right)}, \quad k = 1, 2, \ldots, K \tag{2.33}$$

$$\left(\sigma_{k,m}^{\text{new}}\right)^2 = \frac{\sum_{n=1}^{N} \left(x_n - \mu_k^{\text{new}}\right)^2 p\left(k \mid x_n^1, \ldots, x_n^M, p^{\text{old}}\right)}{\sum_{n=1}^{N} p\left(k \mid x_n^1, \ldots, x_n^M, p^{\text{old}}\right)} k = 1, 2, \ldots, K, m = 1, \ldots, M \tag{2.34}$$

2.3.5 Convergence

The algorithm finishes when convergence occurs. It might be done in several ways. One is to use a log-likelihood function. The log-likelihood function indicates the goodness of fit. Low changes in log-likelihood scores, like a few decimal numbers between iterations, might be the preferred point to stop the algorithm. The log-likelihood function is presented below:

$$L = \sum_{d=1}^{D} \left[\sum_{k=1}^{K} \ln \hat{\alpha}_k p(k|d) + \sum_{k=1}^{K} \sum_{m=1}^{M} n_{dm} \ln \hat{p}_{km} p(k|d) \right] \tag{2.35}$$

The log-likelihood function is also an indicator of the goodness of fit. With a higher score, we can suspect that data is classified correctly.

2.3.6 Optimization

Despite of R language that we implemented EM algorithm, our implementation is considerably fast. It is mostly because we are not calculating full covariance matrix, but only diagonal variances. It comes at the slight cost of the accuracy (we are not refering specificaly to the meteric here). We also tried to stay on the logarithmic scale as long as possible, to reduce numeric errors.

2.3.7 Various methods of initalization and EM parameters

We wanted to verify if initial conditions influence clustering. If yes, we wanted to investigate what kind of impact it is, like if the convergence is faster or if clustering yields better results overall.

We investigated different types of initialization:

- random
- k-means with a random subset
- k-means with ++ initialization

Moreover we have added one EM based o k-means and set maximum iteration to 15. We wanted to check, if and how much it will impact results and check if we should conduct an experiment on the larger set of algorithms.

2.3.8 Other EM implementations

In addition to different types of initialization, we have also checked some of the existing GMM algorithms' capabilities. One of the most popular is Mclust, an excellent implementation of the EM algorithm. However, it has a considerable drawback regarding multidimensional data. It computes the full covariance matrix, thus, is extremally computationally demanding.

Another attractive implementation is called GMM in the ClusterR package. The function itself is the wrapper around gmm_diag coming from the Armadillo library. It is worth mentioning that Armadillo is a C++ library for linear algebra & scientific computing. It consists of numerous helpful functions.

GaussEM based on random initialization. What our implementations have in common is that we used diagonal GMM. It might be a better choice than the relatively more complicated problem of estimating GMMs with complete covariance matrices.

We presented a few variants of GaussEM. They differ regarding initialization type or encoded number of iterations. The comparison only partially exhausts all the possibilities worth a

separate study. We chose only random initialization and two flavours of k-means because of speed and memory management. In another case, we limited the number of iterations to 15. We wanted to see if there is a difference in efficiency.

2.4 Multinomial Mixture EM

2.4.1 Implementation

Pseudocode

Algorithm 2.2 Multinomial Mixture Model EM

```
> Input:
    k = int
    data = Matrix(m,n)
> Parameters initialization:
    temp_k = sample from 1:k times m
    alphas = sample 1:k from Unif(0.1,1)
    alphas = alphas/sum(alphas)
    for i in 1:k{
       probs[i] = mean(data[temp_k == i])
       probs[i] = probs[i]/sum(probs[i])
     }
> Maximization:
> Expectation:
```

2.4.1.1 LSE trick

One of the challanges in machine learning are extremaly small numbers. Following statements were produces in R software.

```
1e-323 > 0; 1e-324 == 0
```

The first statement return TRUE, when the second statement return FALSE. In the example above we were checking if one after 323 decimal places is bigger than zero. The result was true, which seems correct. However in case when we tested whether one after 324 decimal places is bigger, the result was false. Another test, now for equality between this value and zero, returned true. This means that value 1e-324 was approximated to zero. Actually, R language uses IEEE 754-2008/IEC60559:2011 standard which is called 'binary64' format or double-precision floating point. Computers are using floating point to represent fractions of values.

The problem with very small numbers is common in computer sciences and especially in machine learning. It might occur, although it is not a rule, if the data has many dimensions,

like in multivariate datasets. Exemplary algorithm which has to address this kind of issue is multinomial Expectation-Maximization. Because of its nature it often has to deal with hundreds of categories. Assume that we have only ten categories and we evenly distribute probability p between them. It means that each feature will have a $p = 0.1$. In case of 100 features $p = 0.01$. Now, it does not seem to be the problem, when the probability is evenly distributed. However it is rarely the case. Depending on the software at some point, computers will return 0. This is not the correct result. One solution to this problem is to use logarithms and logarithm-based operations. Particularly useful is Log-Sum-Exp (**LSE**) function. It allows for logarithms standardization, avoiding zero error at the same time. The general formula for **LSE** is as below [15]:

$$\text{LSE}(x_1, ..., x_n) = \ln(exp(x_1) + ... + exp(x_n)) \tag{2.36}$$

where: x_n is observation, log is natural logarithm, exp is exponentiaion

2.4.2 Input data

For presented cases, we treat each observation as a row, and each variable as a column.

Data points belong to $\mathbb{N}$. The data shows counts of how many times observation was noted within each variable.

$$\begin{gathered} n_{11} + n_{12} + n_{1m} \ldots + n_{dM} \\ n_{21} + n_{22} + n_{2m} \ldots + n_{dM} \\ n_{d1} + n_{d1} + n_{dm} \ldots + n_{dM} \\ \vdots \\ n_{Dm} + n_{Dm} + n_{Dm} \ldots n_{DM} \end{gathered} \tag{2.37}$$

The usual requirement to start EM for multinomial mixture is to provide **k** number of clusters. Knowing the number of subgroups in the data, we are able to initialize parameters in the first step of an algorithm.

2.4.3 Initialization

During initialization, we need to create first guess of the parameters. In case of multinomial mixture we need to initialize **mixing proportions** (α) and **probabilities** (p) for each category/variable. This should be done for each mixture component $k \in \{1..K\}$.

Mixing proportions indicates how much of the mixture space belong to K. Depending on the number of components, K we need to provide equal number of α_K 2.38.

$$\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \cdots \quad \alpha_K \tag{2.38}$$

Assume that we have a mixture where $k = 3$. In that case we need to create 3 α parameters. We can use uniform distribution $\alpha_K \sim U(0.1, 1)$ to obtain initial alphas. We advise to keep the interval within [0.1, 1], because extremely low values might cause over dominance of larger α during estimation step. After choosing values from uniform distribution, they should be standardized.

$\hat{\alpha}_K = \frac{\alpha_1 + \alpha_2 + ... + \alpha_k}{\sum_{k=1}^{K} \alpha_k}$ and $\sum_{k=1}^{K} \hat{\alpha}_k = 1$

Number of probabilities are equal to the number of dimensions/categories in the data. If the dataset consist of 10 categories, we should provide probability for each category, for each element in K.

Probabilities depict how likely given variables will occur in the given group.

$$\begin{aligned} &p_{11} + p_{12} + \ldots + p_{kM} \\ &p_{21} + p_{22} + \ldots + p_{kM} \\ &p_{k1} + p_{k2} + \ldots + p_{kM} \\ &\vdots \\ &p_{Km} + p_{Km} + \ldots p_{KM} \end{aligned} \tag{2.39}$$

2.4.4 E-step

During the E-step we calculate likelihood for each observation. It is done by multiplying each mixing proportion parameter α_k by all probabiliby successes rised to their number of occurences in the data (respective observation). Then we standardize this value, by divinding numerator by the sum of calculated numerators for all of the components.

$$p(k|d) = \frac{\alpha_k \times p_{k_1}^{nd_1} \times p_{k_2}^{nd_2} \times ... \times p_{k_M}^{nd_M}}{\sum_{k=1}^{K} \alpha_k p_{k_1}^{nd_1} \times p_{k_2}^{nd_2} \times ... \times p_{k_M}^{nd_M}} = \frac{\alpha_k \prod_{m=1}^{M} p_{k_m}^{nd_m}}{\sum_{k=1}^{K} \alpha_k \prod_{m=1}^{M} p_{k_m}^{nd_m}} \tag{2.40}$$

where

α_k - is mixing proportion of k component

nd_M - is a count of successes for a given category

$p_{k_M}^{nd_M}$- is probability of success in 1 to M category of k component, rised to the power of nd_M

2.4.5 M-step

During maximization step, parameter α and probabilities are updated. We can update mixing proportion as in the equation below:

$$\hat{\alpha} = \frac{\sum_{d=1}^{D} p(k|d)}{D} \tag{2.41}$$

where:

$\hat{\alpha}$ - is vector of new mixing proportions

D - is the count of observations

In the case of probabilities, we have to calculate new probabilities for each mixture component k separatelyu. Then, we can standardize those probabilities over all mixture components, as presented in the following equasion:

$$\hat{p}_{km} = \frac{\sum_{d=1}^{D} n_{dm} p(k|d)}{\sum_{n=1}^{M} \sum_{d=1}^{D} n_{dm} p(k|d)} \tag{2.42}$$

where:

$\hat{p}_{km}$ - is a vector of new probabilities proportions

2.4.6 Convergence

Within each iteration, the log-likelihood function should increase. If we observe a low changes in the log-likelihood score, like after a few decimal places it might be good time to stop the algorithm. The log-likelihood function is as presented below:

$$L = \sum_{d=1}^{D} \left[\sum_{k=1}^{K} \ln \hat{\alpha}_k p(k|d) + \sum_{k=1}^{K} \sum_{m=1}^{M} n_{dm} \ln \hat{p}_{km} p(k|d) \right] \tag{2.43}$$

The log-likelihood function is also an indicator of the goodness of fit. In the perfect scenario, with a higher score, the parameters are closder to true parameters.

3 Algorithms based on distance functions

We often perceive and describe the distance between objects as we constantly use various metric measures. For example, we need to estimate the distance between cars to know if we can safely change lanes. When we are in a hurry, we might want to choose the shortest path to our destination.

In this chapter, we will present four algorithms based on distance functions. Those algorithms are based on various statistical distance metrics, like euclidean or manhattan.

3.1 Foundation

3.1.1 Input

Assume there are n objects, like trees, people, stocks etc. To gather them in groups, we need some way to compare how similar or not they are. Clustering applications typically work on either two types of structure.

In the first structure, we can describe objects with their attributes. Those might be, e.g., height, weight, count of words or value. We can also provide the real value for each: 165 cm, 55 kg, 15 words or 17 euros. The first structure can be arranged into an n by m matrix, where rows correspond to objects and columns to the attributes. Tucker (1964) calls such an objects-by-variables matrix two-mode because rows and columns differ.

The second structure is built up with the same set of objects in rows and columns. It contains two types of proximity measures between all pairs of objects. They are called similarity and dissimilarity. Similarity measures how much objects resemble each other, while dissimilarity estimates how far away two objects are from each other. This second structure, object-by-object matrix, is called one-mode.

3.1.2 Variables

3.1.2.1 Scale

Units of measures have a significant impact on clustering results in the case of vector data points. When expressed in different scales, some coordinates of data vectors can dominate results of clustering. Therefore, aiming at reducing / compensating for that, we apply scaling and standardization procedures presented below.

3.1.2.2 Different variables

Different variables, i.e., different entries of data vectors can exhibit variability of ranges on the basis of two factors. One factor will be the fact that they are of different type and are measured in different units. Another factor will be the fact that even when they are measured in the same system of units, they can exhibit physical differences leading to differences that need to be compensated.

3.1.2.3 Standardization

Standardization makes cluster structure more shallow, by reducing effect of large samples, since it lowers their contribution. Standardization is an attempt to achieve objectivity across different units. Standardization makes data independent of unit measures, we can put all units on the same scale.

In the procedure of standardization first we have to calculate the mean value of variable f given by:

$$\mu_f = \frac{1}{n}\left(x_{1f} + x_{2f} + \cdots + x_{nf}\right) \tag{3.1}$$

where: n is number of observations, μ is a mean, x_{nf} is observation

Next we have to calculate the measure of dispersion in the data. One of the most commonly used is standard deviation:

$$\sigma = \sqrt{\frac{1}{n-1}\left\{(x_1-\mu)^2 + (x_2-\mu)^2 + \cdots + (x_n-\mu)^2\right\}} \tag{3.2}$$

where: n is number of observations, μ is a mean, x_n is observation and σ is standard deviation

The equation for standardization is as follows

$$z_n = \frac{x_n - \mu}{\sigma} \tag{3.3}$$

where:x_n is observation, μ is a mean, and σ is standard deviation

Sometimes those standardized measurements are called z-scores. They are devoid of units measures, because both numerator and denominator are expressed in the same units.

3.1.3 Statistical distance

Statistical distance contains broad spectrum of methods in probability or information theory. Very different measures can be used to quantify the distance between two statistical objects. It can be a distance between two probability distributions, two random variables or samples. We

present definitions, properties of distance functions and will show most important function measures of distance that were applied in this thesis.

Statistical distances are important notions for developing applications of grouping algorithms. There are four properties, that will characterize them.

Property	**Description**
$d(x,y) \geq 0$ (non-negativity)	Distances are nonnegative
$d(x,y) = 0$ if and only if $x = y$	Distance from object to itself is equa
$d(x,y) = d(y,x)$ (symmetry)	Distance from object x to object y is
$d(x,z) \leq d(x,y) + d(y,z)$ (subadditivity / triangle inequality).	Distance from x to y is shorter, than

Table 3.1: Conditions to determine different types of metrics

In general, if all four conditions are met, it is a distance or metric. It is important to distinguish between them, because when only the first and second property is satisfied, the statistical distance is called a divergence. In algorithms that we are presenting, all four criteria are satisfied.

3.1.3.1 Distance between two points

Important step to quantify degree of dissimilarity between objects is the choice of distance metric. We need to compute such distance between each i, j pairs of objects. One of the commonly used metric to do that is an Euclidean or Manhattan distance.

Euclidean distance Pythagorean famous theorem, allow us to calculate hypotenuse of a perpendicular triangle. Since $AB^2 + BC^2 = AC^2$ then

$AC = \sqrt{(AB^2 + BC^2)}$. Basically, this straight line is the shortest distance between point A and C. Now, lets impose the triangle on the Cartesian plane and assume that two points are single objects. Each one is described by x and y coordinates so they are within two dimensions. Analogically as in Pythagorean theorem, we can calculate the distance between object A and C as $(A_{x1} - C_{x2})^2 + (A_{y1} - C_{y1})^2$.

The Euclidean distance is the straight-line distance between any two points. It is a non-negative real number such that $\mathbb{R}_{\geq 0} = \{x \varepsilon R | x \geq 0\}$.

If we consider two points from distribution $X = (x_1, x_2 \ldots x_n)$ and from distribution $Y = (y_1, y_2 \ldots y_n)$ in Cartesian coordinates, then distance, between those points are given by general equation:

$$d_{Euc}(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots (x_n - y_n)^2} = \tag{3.4}$$

$$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{3.5}$$

where**:**

$\mathbf{d_{Euc}}$ - Euclidean distance

$\mathbf{X}, \mathbf{Y}$ - variables

$\mathbf{x_i}, \mathbf{y_i}$ - realizations of X and Y

$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$- root of sum of squares between all x_i and y_i

Euclidean distance is a useful and straightforward tool when it comes to finding the nearest hospital, considering an emergency helicopter flight. It is increasingly used in a molecular conformation in bioinformatics, localization of sensor networks, or dimensionality reduction in statistics .

Minkowski distance Minowksi distance is a generalization of both Euclidean and Manhattan distance. It is also called Lp metric and is given by equation (8) where p is any nonnegative, real number.

$$d_{Mink}(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{3.6}$$

where**:** $\mathbf{d_{Mink}}$ - Minkowski distance

$\mathbf{X}, \mathbf{Y}$ - variables

$\mathbf{x_i}, \mathbf{y_i}$ - realizations of X and Y

$\sum_{i=1}^{n} |x_i - y_i|^p$ - sum of absolute difference between all x_i and y_i raised to the power of p.

Minkowski distance can be considered as a generalization of Euclidean and Manhattan distance. If $p = 2$ it is the same as Euclidean distance and with $p = 1$ it is equivalent to Manthattan distance

Minkowski distance is applied in fuzzy clustering and all applications from Euclidean distance and Manhattan distance.

Manhattan distance Manhattan distance, called also city block distance, owns its peculiar name thanks to following reasoning. Imagine a city, where streets are part of a grid, so we have only vertical and horizontal line segments. Suppose you want to get from the block A to block B. Then, the shortest possible distance might be described by following equation: |xA - xC| + |yA - yC|. It goes with an assumption, that we do not own a helicopter and we cannot fly. Otherwise, the shortest path will be an euclidean distance. L. Kaufman and P.J. Rousseeuw advise to use manhattan distance if for example a difference of 4 in first variable and 1 in second is the same as 2 in first variable and 3 in second . Euclidean distance with

emphasize larger value, thus increasing weight of a variable.

$$d_{Man}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.7}$$

where: $\mathbf{d_{Mink}}$ - Manhattan distance

$\mathbf{X},\mathbf{Y}$ - variables

$\mathbf{x_i},\mathbf{y_i}$ - realizations of X and Y

Applications of manhattan distance:

https://iq.opengenus.org/manhattan-distance/

3.2 Hierarchical clustering

Hierarchical clustering owns its name thanks to its structure and how it creates the clusters. It produces a series of partitions. Specifically, data points are gathered into clusters resembling an upside-down tree shape (pic!). The choice of a presented shape is arbitrary, as it might be circular or graph-like (pic!). What is shared by all three pictures is a visible hierarchy. Each data point is a part of a systematically growing cluster that incorporates all of the data at the end. To achieve this result, we can agglomerate or divide the data. In the first case, called the agglomerative or top-down method, individual data points are grouped into larger clusters until they cover the data set. In the divisive or bottom-down approach, data is divided into smaller groups up to single data points.

3.2.1 Agglomerative clustering

Agglomerative clustering is the most common approach in Hierarchical Clustering.

Before we perform clustering, we have to make two decisions. The first one is how we measure the distance or similarity between points. It might be euclidean, Manhattan distance or many others. The second decision is the linkage method. It dramatically influences how data points will be clustered in consecutive iterations. Some common examples are single, complete or average linkage. Following clustering, we can decide on any number of clusters without repeating the calculations. It is a characteristic feature of Hierarchical Clustering not found in other unsupervised algorithms.

In the first step, we need to measure how far points lie from each other. To do that, we can use similarity (e.g. Jaccard index) or distance (e.g. Euclidean) metric. The choice depends on scientific questions and the data itself. Next is the choice of linkage method. It will determine the way how the data will be clustered together. It has a tremendous impact on the results.

The last but important thing is deciding the number of clusters we would like to see. We are provided with two different approaches to that problem. The first one is to choose the number

of groups exactly. Another way is to cut the branches at a specific tree height. The tree's height depends on the largest distance between the two clusters in the data. Thus choosing this way will give us groups of data with alike similarity or distances between them.

3.2.2 Divisive clustering

Divisive hierarchical clustering reverse the question presented in the agglomerative hierarchical clustering. It asks how to divide trivial solution with one cluster to n singleton clusters. This approach is much less common than agglomerative methods, mostly due to the extensive use of computational power. In divisive methods one cluster is consecutively splitted into two smaller clusters. The challenge is how to find an optimal splitting space. Exploring all the possibilities is computationally expensive, as in the first step there exist 2^(n-1)-1 ways to split first cluster into two separate ones. One solution to this problem was presented in the DIvisive ANAlysis Clustering (**DIANA**).

3.2.2.1 DIANA

How it works (need tables and pictures) At the beginning, **DIANA** has to split up the data into two separate groups. It is done without considering all possible divisions, but rather iteratively. First, average distance between object and group of objects is calculated for each of them. Then, object with largest average dissimilarity is chosen to initiate "splinter group". Next, average dissimilarity is calculated between object and remaining objects in the larger group. Te results are compared with the splinter group. Object with the largest average dissimilarity difference is then moved to the splinter group, making together one cluster.

3.2.3 Linkage methods

When the distance metric was calculated, it showed us how similar or how close the objects are to each other. Linkage methods are used to join such objects or observations using distance measures as a base. In other words, it is a function that creates clusters from the objects based on their similarity. There are several linkage methods and each one of them provides slightly different results. We will present 8 most popular: single, complete, average, Ward's, Ward's 2, McQuitty, median and centroid.

3.2.3.1 Single linkage

In each step, the two clusters with the smallest minimum pairwise distance are merged.

$$l_s(X,Y) = \min_{x \in X, y \in Y} d(x,y) \tag{3.8}$$

where:

l_s - single linkage

X, Y - two sets of elements (clusters)

$\min_{x \in X, y \in Y} d(x,y)$ - minimal distance between the two elements $x \in X$ and $y \in Y$.

3.2.3.2 Complete linkage

In each step, the two clusters with the smallest maximum pairwise distance are merged.

$$l_c(X,Y) = \max_{x \in X, y \in Y} d(x,y) \tag{3.9}$$

where:

l_c - complete linkage

X, Y - two sets of elements (clusters)

$\max_{x \in X, y \in Y} d(x,y)$ - maximum distance between the two elements $x \in X$ and $y \in Y$.

3.2.3.3 Average linkage

It might be found under name Unweighted Pair Group Method with Arithmetic Mean or **UPGMA**. In this method, distance between points from both clusters is measured. Then, the average distance is calculated and clusters with smallest average distance are merged.

$$\frac{1}{|X| \cdot |\mathscr{Y}|} \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} d(x,y) \tag{3.10}$$

$$l_{(\mathscr{X} \cup Y),Z} = \frac{|\mathscr{X}| \cdot d_{\mathscr{X},Z} + |\mathscr{Y}| \cdot d_{\mathscr{Y},Z}}{|\mathscr{X}| + |\mathscr{Y}|} \tag{3.11}$$

where:

$l_{(\mathscr{X} \cup Y),Z}$ - average linkage

X, Y - two sets of elements (clusters)

$d(x,y)$ - distance between the two elements $x \in X$ and $y \in Y$.

3.2.3.4 Ward's

Ward's method shares some similarities with analysis of variance (ANOVA). It aims to minimize error sum of squares (ESS), when linking clusters together.

$$\mathrm{ESS}(X) = \sum_{i=1}^{N_X} \left| x_i - \frac{1}{N_X} \sum_{j=1}^{N_X} x_j \right|^2 \tag{3.12}$$

$$l(X,Y) = ESS(XY) - [ESS(X) + ESS(Y)] \tag{3.13}$$

where:

ESS(X) - error sum of squares of variable X

ESS(Y) - error sum of squares of variable Y

$d(x,y)$ - distance between the two elements $x \in X$ and $y \in Y$,

N_X - number of observations in set X

https://jbhender.github.io/Stats506/F18/GP/Group10.html

https://www.statistics.com/glossary/wards-linkage/

3.3 K-means

K-means algorithm is an iterative, distance based algorithm. It is relatively fast, simple and computationally effective. In addition, low memory usage makes it suitable for cluster detection in large data sets. This is major advantage over hierarchical clustering methods, that are based on dissimilarity matrix. Although they allows to explore any number of clusters, they might become huge and computationally demanding. Last but not least, k-means might be successfully used as a starting step in other algorithms. Concrete example of that will be the initialization part of Expectation-Maximization algorithm.

From mathematical point of view, k-means is an approximation of the normal mixture model. Parameters are estimated using maximum likelihood method. The main idea behind the k-means is that observations are gathered around artificially introduced centers, called **centroids. Centroid** can be perceived as a generalization of the mean and it is a geometric center of a convex object. In general, distance between centers and observations should be minimal. Data points closest to the particular center are part of its cluster.

The initial number of **centroids** is equivalent to the number of clusters **k** in the data. k is required to start the algorithm. If we have strong assumption regarding number of clusters, we can use it. Otherwise, there are few ways to determine number of clusters experimentally. Algorithm stops when it reaches stability, which depends on the convergence criteria. Example of that will be creating groups with highest similarity of points within given cluster and lowest between different clusters. In the commonly used Hartigan-Wang implementation, the stop criterion can be satisfied in two ways. One is by minimizing the total sum of variance within clusters (**WCSS**) as given in the following equation.

$$WCSS = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \left\| x_{ij} - c_i \right\|^2 \tag{3.14}$$

where**:** $WCSS$ - total sum of variance within clusters

$\mathbf{c_i}$ - centroid

x_{ij} - observation ij

There are few k-means algorithms: Lloyd, Forgy, MacQueen and the one already mentioned Hartigan-Wong. The last one is the default k-means algorithm in the R software, that we have used in our comparison.

3.3.1 Lloyd and Forgy

Lloyd and Forgy algorithms are quite similar. They are both batch algorithms, which means that transformation is applied to the whole data.

The key distinction between those two is that Llyod take into account discrete data distribution, when Forgy considers continuous. Consider random vector $X = \{x_1, x_2, \cdots, x_3\} \in \mathscr{R}^d$, where $\mathscr{R}^d$ is sample space of d dimensions. For X, algorithm tries to find **k number** of centroids $C = \{c_1, c_2, \cdots, c_k\} \in \mathscr{R}^d$ that solves minimization problem:

$E = \sum_{i=1}^{k} \sum_{j=1}^{n} d\left(c_i, x_{ij}\right)$, for discrete distribution (Llyod) density function and d is distance metric.

Apart from that difference, the following procedure is the same.

3.3.2 Hartigan-Wong

$$SSE2 = \frac{N_i \sum_j \left\| x_{ij} - c_i \right\|^2}{N_i - 1} < SSE1 = \frac{N_1 \sum_j \left\| x_{1j} - c_1 \right\|^2}{N_1 - 1} \tag{3.15}$$

where:

$SSE1$ - sum of the squared Euclidean distances of each point to first centroid

$SSE1$ - sum of the squared Euclidean distances of each point to second centroid

$\mathbf{c_i}$ - centroid

x_{ij} - observation ij

3.3.3 Initialization

We compared different two different types of initializations for kmeans:

- Hartigan wong, as implemented in R
- k-means++ from package ClusterR

From the results we can see that

3.4 Fuzzy clustering

Fuzzy k-means clustering, often called fuzzy c-means (**FCM**), can be considered a soft version of k-means. Each observation has some set of degree of belonging to a cluster. This

is different approach than in k-means, where membership is binary and given observation belong (1) or not (0) to particular group.

FCM centroids are the means of all observations weighted by their degree of membership to the cluster. They can be calculated using the formula:

$$c_k = \frac{\sum_i w_k (x_i)^m \times x_i}{\sum_i w_k (x_i)^m} \tag{3.16}$$

where: c_k - centroid of a cluster k,

w_k - degree of membership to the cluster,

m - fuzzifier coeficient

The FCM tries to minimize objective function:

$$\arg\min_C \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^m \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2, \tag{3.17}$$

where: w_{ij} indicates the degree of the belonging to the cluster. It allows to calculate the distance between the observation and the centroid k. Formula is given by :

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left\| \mathbf{x}_i - \mathbf{c}_j \right\|}{\left\| \mathbf{x}_i - \mathbf{c}_k \right\|} \right)^{\frac{2}{m-1}}} \tag{3.18}$$

where:

m - fuzzifier coeficient

The fuzzifier m playes important role in fuzziness of clusters. It can take values from [1, ∞]. A large value of m results in lower degree of belonging w_{ij}, resulting in fuzzier groups. Often for the data without any apriori knowledge, parameter m is commonly set to 2. When the m = 1, the cluster membership should converge to 0 or 1, as in the k-means algorithms.

Due to vast similarities to k-means, FMC suffers from alike problem, it does not guarantee convergence to the global optimum. It might get stuck in local minima. In addition, the results heavy depends on the initial choose of weights.

3.4.1 Initialization and distances

In the case of fuzzy c-means, we have checked different initialization but also distance metrics.

As in k-means, we checked random initialization and the ++ initialization.

Moreover, a few sources report that manhattan distance might be a better choice in the case of multidimensional data.[cite the soruces]. We made a simple comparison between c-means with manhattan metric and euclidean. The reason is that those are often the subject of discussion.

We divided the results by actual and simulated data.

3.5 k-medoids

The most significant advantage of the k-means algorithm is computational efficiency. Calculating averages and assigning them to the closest mean is algebraically fast. It is relatively easy to parallelize. Those merits make this algorithm applicable to large databases.

Unfortunately, it is based on the square of the Euclidean distance. However we are not always interested in this kind measure of similarity. Such a measure is also susceptible to large distances, and a single outlier can significantly affect the sum of squared distances.

PAM works similarly to k-means, except that the centres of the clusters are the cases in the data set (called centroids or clusters). Therefore, the set of possible cluster centres in the PAM method is much smaller than in the k-means method. Usually, the results of the PAM method are more stable. Moreover, the k-medoid algorithm will allow us to use other distance measures at the price of higher computational complexity. In other words k-medoids method is an extension of the k-means method.

The k-medoids method finds such objects representing clusters (k-medoids) among the observations, to minimize the sum of the distances of all non-medoid elements from their closest medoids.

Another difference between the k-means algorithm and the k-medoid method is how the distance between observations is defined. In the case of the k-means Euclidean distance is used, while the PAM uses mainly Manhattan distance. In addition, k-medoids are supposed to deal with the problem of outliers as well as noise in the data. It is considered more robust than k-means. However, its computational usage can be considered high even by today's standards. It performs slowly in large data sets.

We can distinguish two phases of PAM namely construction phase and swap phase.

Construction phase consist of:

1. Split the dataset into k k clusters with k k medoids assigned
2. Calculate the distance matrix between the medoids and the other observations
3. Assign each observation (non-medoid) to the closest cluster

Swap phase:

1. Using iteration, replace one of the medoids with one of the non-medoids and check that the distances of all non-medoids from their nearest medoids are smaller.
2. If at least one medoid swap has occurred, repeat step 3. Otherwise, end the algorithm.

4 Study pipeline

In this chapter we present full pipeline. We will start with chosing and selecting datasets. Then, we will proceded to the filtration and scaling methods that we used. After that, we will introduce cluster evaluation methods. It consist of methods of visualizing data as well as ways to present algorithms performance or efficiency.

We can divide whole datasets into two main subsets. The first one consists of artificially created data that will challenge algorithm performance in a more controlled manner. The second one consists of real data sets chosen with the multidimensionality criterion in mind.

4.1 Data preparation

4.1.1 Simulated data

Data simulation is a vast research field with plenty of practical applications. We can use it in data prediction and during testing new programs or algorithms, to name a few. Simulation of data is a way to generate random numbers from the stochastic process. A stochastic process is a time-ordered sequence of random variables or something described by probability distributions. For example, we can model height population data using a mixed normal-uniform distribution model. [7].

$h_i \sim \text{Normal}(\mu, \sigma)$

$\mu \sim \text{Normal}(160, 15)$,

$\sigma \sim \text{Uniform}(0, 10)$

where h_i is i-th observation,

Both μ and σ parameters are generated from the separate distributions. First parameter, μ is generated from the normal distribution with parameters $\mu = 160$, and $\sigma = 15$. Second parameter, σ is generated from the uniform distribution with minimal value of 0 and maximum value of 10. Here, our exemplary model doesn't describe the height of any particular population. It is a very rough example that is within reasonable limits. Extreme values of this model will barely go lower than 95 cm and exceed 240 cm. The reason is that the shortest man recorded was Chandra Bahadur Dangi, who measured 54.6 cm. ["Shortest Person Ever Declared: Chandra Bahadur Dangi". Huffington Post. 27 February 2012.] On the opposite, the record for the tallest man recorded belongs to Robert Wadlow, with 272 cm of height. [http://altonweb.com/history/wadlow/]

Data simulation can become pretty handy. The ground truth is fully known, contrary to the actual data, when we are sometimes only partially aware of the groups in the data. We know the exact model parameters and can compare them with the estimated values.

Some models might get exceptionally complicated by having many different parameters. It may become untraceable to predict their effect on the data. However, since we have complete control over model parameters, we can tune them accordingly and observe the result. In this manner, we can perceive data simulation as a reflection of some natural system and a controlled experiment [16].

Using data simulation techniques, we are no longer limited by actual measurements. We can generate an almost infinite number of data with the structure and parameters we want. It allows us to test different kinds of data. However, it might not reflect the real data accurately, giving us false estimations of model efficiency. It is based on a few limitations. The very special that should be taken into mind is Random Number Generation.

4.1.1.1 Simulated multivariate normal data

Mixtures were generated according to multivariate normal distribution properties. Each mixture component consisted of random vector μ, matrix Σ, mixing proportion α and a fixed number of dimensions d, across all mixture components. Each random vector μ was generated from uniform distribution such that $\mu : U(0,10)$ from the interval [0,10]. Σ was created assuming that the correlation between variables (elements of data vectors) equals 0, so they are independent. Variances were then generated from a uniform distribution from the interval [0.1, 1] to avoid 0 variances. Mixing proportions were obtained from a uniform distribution from the interval [0.1, 1] to prevent one component from dominating the mixture. The vector of mixing ratios was standardized, to sum up to 1.

Counting different filtration and scaling, I created over 15 000 files. They contained from 2 to 10 clusters and between 5 and 1600 dimensions. Step between dimenstion was taken arbitrally. The higher number of files than in multinomial data comes from the fact that we have used two different PRGN packages, namely MASS and MultiRNG.

4.1.1.2 Simulated multinomial data

Mixtures were generated according to multinomial distribution properties. Each mixture component consisted of: random vector p of probability successes, number of observations n, mixing proportion α and d fixed across all groups. p was generated from a uniform distribution from the interval $[0,1]$. Then all values were standardized so that all elements sum up to 1. n was generated from the interval $[3,n]$. Mixing proportions of all components were calculated using uniform distribution from the interval [0.1,1] and standardized, to sum up to 1.

With different types of filtration, I created over 8 000 files. Similarly, as in multivariate data, they contained from 2 to 10 clusters between 5 and 1600 dimensions with arbitrarily taken steps between dimensions.

4.1.1.3 Random Number Generator (RNG)

As of the time of writing the thesis, there is no genuinely random number generator. Instead, we have PRNG. The letter "P" stands for the "pseudo". Although some efforts exist to create such a system using quantum computers, we will focus on PRNG. It is a program, or part of it, that generates numbers based on some initial information. This base information is called a seed. It can use many different sources of initial data. Depending on the software, it can be actual time, active processor time or key inputs transformed into numerical values. It is why pseudo-random number generators don't produce random numbers. When a seed initializes PRNG, it can take an infinite number of different values. However, most of the time, it is restricted by software design or by computer memory.

4.1.1.3.1 Computation optimization One of many optimisation-related parts is how fast we can solve arithmetic equations. If we consider matrix multiplication or even more straightforward vector multiplication, it consists of a few steps to obtain the result. Using a high-level programming language, or even mid-level, like c++, we can use it out of the box without the need to code it manually. Those languages already offer such libraries.

The first to mention is BLAS. The shortcut refers to Basic Linear Algebra Subprograms. In general, it provides classic interfaces for linear algebra calculations. Inside the BLAS library, we can distinguish three different BLAS levels. BLAS1 (level 1) allows for vector-vector operations. BLAS2 is responsible for matrix-vector operations. Finally, BLAS3 makes matrix-matrix operations possible. In summary, BLAS is the computational kernel for linear algebra and other scientific applications. If we can make BLAS optimise its libraries, the whole application written with it will benefit.

Another essential library is LAPACK. LAPACK stands for Linear Algebra PACKage and is used along with BLAS. The actual language that LAPACK is written in is Fortran 90. It consists of routines for solving least-squares, eigenvalues or singular value decomposition. LAPACK also handles routines for solving lower–upper (LU) decomposition, QR factorisation or Cholesky decomposition, to name a few. It handles dense and some sparse matrices, precisely banded ones. However, it does not have routines for sparse matrices in general. In terms of numbers, similar functionality is provided for real and complex matrices in all areas. Both double and single precision is supported.

The relation between BLAS and LAPACK is simple. LAPACK heavily depends on BLAS as it is built upon it. [https://netlib.org/lapack/ ; https://www.openblas.net/]

https://csantill.github.io/RPerformanceWBLAS/ provides some results that compare mentioned

alternatives to BLAS. In this comparison, each library has its fortes. However, Intel MKL has the highest number of the best results in all comparisons.

In our computations, we have used OpenBLAS libraries, which in comparison, performed slightly worse. However, there are some debates about which library is better. Intel MKL was primarily optimised under intel processors, and some users historically reported worse performance on AMD processors.

Nevertheless, despite the library choice, alternatives offer better optimisation and gains in performance.

R specific optimisation

There is a common idea that R is slow, which is valid to some extent. The R is high-level programming language. Second is tahat is was designed with statistical analysis in mind. Many functions are not created with speed and efficiency but with stability and reliability.

That being said, I will analyse mean function. First, it consists of many sanity checks, guaranteeing that the user provided the correct input. Secondly, it handles different data types, like time, date, and date-time classes. All of those checks require additional time. It stacks up the more we are executing the function. As in the presented example, the mean withouh all of the sanity checks and internall validtions, will acheieve better calculation time.

The presented exercise concludes that deep R knowledge allows for writing efficient functions. We can strip many functions from things that we do not use and need. However, it is still a high-level language and optimised C code might be more efficient. As a rule of thumb, the more we rely on some function, and the more specific data type it contains, it might be a good idea to use lower-level language.

4.1.2 Real data

4.1.2.1 Preparation

Having the data, the first step was initial processing. All data was parsed to fixed matrix format. In each case, matrix $n \times d$ consisted of observations n placed in rows, and variables d in columns, to maintain consistency across all analyzed data sets. In the next step, data was processed to select analyzed features, using different filtration methods. After this point, each data set was clustered using unsupervised algorithms. Because of the fact that in unsupervised clustering we do not have any labels, to prepare summary of results, we had to determine membership of clusters. To achieve that, we have used Hungarian algorithm, which optimizes assignment of cluster membership by maximizing value of matches. We presented results, using and comparing various visualization tools for dimensionality reduction like: umap, tSNE and PCA. To compare algorithms efficiency, we have used common statistics like mean, median, but also more sophisticated indexes, like Dice, Jaccard or Rand. Metrics was interpreted and wrapped in comments.

The most important part of our comparative analysis of unsupervised clustering algorithms is computational experiments concerning some publicly available data sets. Nowadays, extensive collections of real-world datasets can be used for comparative experiments of unsupervised clustering algorithms, e.g., [cytaty]. There are also many publications focusing on empirical comparisons of unsupervised clustering methods. When choosing data sets for clustering, the following criteria were used. The first criterion was dimensionality. Datasets chosen for analyses are multivariate with a large number of attributes. The second criterion was that features used for clustering are numerical (real or integer). The third criterion was that data come from different areas. Two datasets (data on frequencies of somatic mutations in genes in cancer samples, data on levels of gene expressions in cancer samples) come from the TCGA database of mutations in cancer tissues [cytat], two datasets (data codons in different species, data on sports activities) come from UCI database [cytat],. The fourth criterion is the availability of information on classes in the data, establishing the ground truth.

4.1.2.1.1 Datasources

TCGA The TCGA project started in 2005, with the aim of identification of complex genomic interactions in majors cancers. Five year later, the TCGA was made available to public. Then, in December 2013, TCGA concluded sample collection with over 20 000 biospecimens across 32 types of cancer. The data become publicitly available for academic purposes worldwide. .

UCI The UCI repository is a vast collection of databases and data generators. It is an open access repository of data, which people worldwide donate. The machine learning community generally uses it for empirically studying various ML algorithms. UCI repository was firstly created in 1987 as an ftp archive by David Aha and students of UC Irvine. As UCI site reports, since that time, it was cited over 1000 times, placing it among the top 100 most quoted "papers" in all computer science.

4.2 Data processing

In the filtration part, we either did not change the data or used the variance decomposition method to reduce the “noise” in the data.

4.2.1 No filtration

In this case, we left the datasets unchanged. Datasets contained all of the original variables. We used it in the in the scaling step as well and in the clustering.

4.2.2 Variance decomposition.

For each variable, we have calculated its respective variance. As a result, we received a vector of n variances with varying values of n, depending on the used dataset. We assumed that obtained vector is, in fact, a one-dimensional mixture that contains essential variables and additional noise. To separate them, we used a mixture decomposition technique. In this scenario, we have used the mclust package. We tried to align mixtures containing from two to even twenty-five components. The Bayesian Information Criterion decided upon the final number of groups.

4.2.2.1 Bayesian Information Criterion (BIC)

Bayesian Information Criterion is a usefull tool in model selection. Given by following equation:

$$\mathrm{BIC} = k\ln(n) - 2\ln(\widehat{L}). \tag{4.1}$$

where:

$\hat{L}$= the maximized value of the likelihood function of the model M i.e.$\hat{L} = p(x \mid \widehat{\theta}, M)$, where $\widehat{\theta}$ are the parameter values that maximize the likelihood function;

x = the observed data;

n = the number of data points in x, the number of observations, or equivalently, the sample size;

k = the number of parameters estimated by the model.

It penalizes model for using more parameters and is scalable because it takes into equation number of observations. I have used it during variance mixture decomposition, to group variances into k groups.

4.2.3 Scaling

We have used cenetering of variables, along with standardization. In this way, variables had similar siginificance in the

4.3 Clusters evaluation

Cluster evaluation might take place before and after clustering. Evaluation done before is mainly done for calculating or making expertise regarding the potential number of clusters. It can be done visually or by brute-force clustering. If we have a one to the three-dimensional dataset, it is relatively easy to visualize. Visualization is becoming more challenging with more than three dimensions.

4.3.1 Cluster assigment - Hungarian algorithm

The increased number of clusters poses a few challenges. One of them is deciding the clusterization labels. Because, in principle, unsupervised clustering is done without them. Eventually, we must decide which cluster should be assigned to which group.

One potential solution to this problem is The Hungarian matching algorithm, also called the Kuhn-Munkres algorithm.

The typical example of KM's usefulness is as follows. Imagine that we have three workers that offer prices for three services: cleaning, washing, and ironing. Showing their prices together in a single matrix, we will have the following:+

We can assign only one job to one worker, and our goal is to minimize the total cost. If we give washing to "worker 2" (which is the cheapest service), we will have to assign ironing to "worker 1" and cleaning to "worker 3". The total cost will be 65$. However, if we resist the temptation and assign washing to worker 3, we can give the cleaning to "worker 2" and ironing to "worker 3". To total cost will be equal to 55$ instead of the initial 65$. Although in the label assignment, we should maximize group observations instead of minimizing them. We can easily do that using the same algorithm by multiplying the matrix by -1. In this way, the highest values will become the lowest ones.

4.3.2 Clusters validation

There is a plethora of various metrics that allow us to compare the similarity or dissimilarity between two clustering methods. We applied a few different to describe algorithms' performance by comparing created clusters with the ground truth.

4.3.2.1 Rand Index and Adjusted Rand Index

Rand index is a popular metric to measure the results of clustering. In we are presented with two groups, we will see the following confusion $2x2$ matrix.

X	Y_1	Y_2	sums
X_1	n_{11}	n_{12}	a_1
X_2	n_{21}	n_{22}	a_2
sums	b_1	b_2	

(4.2)

In the above matrix: X are ground truth labels, Y are clustering labels, a is a sum of all observations that belongs to X_n, b is sum of all observations that belongs to Y_n. Then, RI is given by following formula:

$$Rand\ Index = \frac{TP+TN}{TP+FP+FN+TN} = \frac{n_{11}+n_{22}}{n_{11}+n_{21}+n_{12}+n_{22}} \tag{4.3}$$

where:**TP**and n_{11}- is the number of true positives**; FP**and$n_{21}$is the number of false positives**; FN**and n_{12} is the number of false negatives**; TN**and n_{22} is the number of true negatives.

The RI index takes ranges between 0 and 1, including themselves. In the perfect aligment, the term $n_{21}+n_{12}=0$ and $Rand\,Index=1$.

This scenario is sufficient only for two-class problems. If we have to estimate clustering quality or similarity of more than two classes, the table remains squared but its dimensionalty equals to k. Matrix below is similar to the matrix from the chapter 2. However, it shows groups of clusters. Bold characters indicates correctly assigned observations, that are located on the diagonal line. We will use this assumption for the rest of the thesis. Note that directly after the clustering, groups often will be scattered on the matrix. However, we ensured that this statement is true by assigning labels to clusters, using the Hungarian algorithm. Then, sorting is easily done internally in the R, when creating confusion matrix.

X	Y_1	Y_2	Y_m	$\cdots$	Y_M	sums
X_1	$\mathbf{n_{11}}$	n_{12}	n_{1m}	$\cdots$	n_{1M}	a_1
X_2	n_{21}	$\mathbf{n_{22}}$	n_{2m}	$\cdots$	n_{2M}	a_2
X_d	n_{d1}	n_{d2}	$\mathbf{n_{dm}}$	$\cdots$	n_{dM}	a_d
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_D	n_{D1}	n_{D2}	n_{Dm}	$\cdots$	$\mathbf{n_{DM}}$	a_D
sums	b_1	b_2	b_m	$\cdots$	b_M	

(4.4)

In the above matrix: X are ground truth labels, Y are clustering labels, a_D is a sum of all observations that belongs to X_D, b_M is sum of all observations that belongs to Y_M

In this case, Adjusted Rand Index (ARI) was proposed[17]:

$$ARI=\frac{\Sigma_{ij}\binom{n_{ij}}{2}-\left[\Sigma_i\binom{a_i}{2}\Sigma_j\binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\Sigma_i\binom{a_i}{2}+\Sigma_j\binom{b_j}{2}\right]-\left[\Sigma_i\binom{a_i}{2}\Sigma_j\binom{b_j}{2}\right]/\binom{n}{2}} \tag{4.5}$$

4.3.2.2 Jaccard and Weigted Jaccard Index

In simplified form for two-class case, Jaccard index takes a following formula:

$$Jaccard=\frac{TP}{TP+FP+FN}=\frac{n_{11}}{n_{11}+n_{21}+n_{12}} \tag{4.6}$$

where: **TP** is the number of true positives**; FP** is the number of false positives**; FN** is the number of false negatives

The basic difference from the Rand Index, is that it does not contain TN counts [18].

We implemented **Weighted Jaccard** index, based on the two class equation. First, we substituded denominator as in following

$$Weighted\,Jaccard = \frac{n_{11}}{a_1 + b_1 - n_{11}} \tag{4.7}$$

where: a_1 and b_1 are respective sums of FP and FN values of the first group

We are substracting term n_{11}, since both a_1 and b_1 have and we need only one. Then, we can generalize the equation for all of the groups

$$Weighted\,Jaccard = \sum \frac{n_{DM}}{a_D + b_M - n_{DM}} /k \tag{4.8}$$

where: a_D - is the sum of FP values in any group, b_M- is the sum of FN values in any group, n_{DM} - is the value of TP in any group, k is the number of components and

4.3.2.3 Accuracy Index

In the simplified, two class case, the accuracy index is given by following equation.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{n_{11} + n_{22}}{n_{11} + n_{21} + n_{12} + n_{22}}. \tag{4.9}$$

where: **TP** is the number of true positives**; FP** is the number of false positives**; FN** is the number of false negatives

In the equasion above, Accuracy index is the same as Rand index, thus sometimes it is called Rand accuracy. In our comparison we used its variation, called Balanced Accuracy.

We used this metric to create **Median Accuracy Assigment** plot. It shows how on median given algorithm assigned observations to the correct group. The higher the bar, the better. For the bigger picture, we have added the mean as a black dot and the standard deviation from the mean. From the perfect assignemnt we expect a mean of size 1 and 0 variance.

4.3.2.4 Simple Matching Coefficient

The SMC is easy statistic, which calculates efficiency of the clustering. To obtain this metric, we have to divide sum of all true postivie values by all observations. As one may already knew, this metric might not be suitable choice, when we have significalntly different number of observations.

$$SMC = \frac{TP}{TP + FP + FN + TN} = \frac{n_{11} + n_{22}}{n_{11} + n_{21} + n_{12} + n_{22}}$$

where:**TP** is the number of true positives**; FP** is the number of false positives**; FN** is the number of false negatives**; TN** is the number of true negatives

In the SMC metric we calculate numerator in a different way, than it was in the case accuracy index.

When we calculate the value of TN it contains

4.3.2.5 Weighted Simple Matching Coefficient

The WSMC is balanced version of SMC. It accounts for different number observation in the groups, thus might be more accurate, where the classes are not balanced.

$$WSMC = \sum \frac{n_{DM}}{a_D} / k$$

where **TP** is the number of true positives**; FP** is the number of false positives**; FN** is the number of false negatives**; TN** is the number of true negatives and **k** is the number of classes

4.3.2.6 Beta-binomial conjugate distribution

The beta distribution is a plot created based on the beta-binomial conjugate prior. Thanks to conjugation, we can mathematically update parameters α and β with respective successes and failures counts. Let us consider $Beta(1,1)$, as shown in the picture below. It shows our prior belief about algorithm performance. That means we are assuming an equal probability for its possible efficiency. If the algorithm assigns all observations incorrectly, we will observe a single, high line at 0. For all correct answers, it will be one line at one. Then, for everything in between., we will observe the distribution with different shapes, locations and standard deviations.

The nice thing about this approach is the possibility of so-called bayesian updating. Once we calculate the distribution parameters, we can use them as our prior distribution when new results arrive. Thus we do not have to recalculate model efficiency over and over again. The other helpful thing about this approach is its crystal clear structure. It is almost impossible to cheat with this kind of metric. One thing is its robustness. Even if we give some model the upper hand in terms of a higher probability of better performance, it will be quickly neglected after seeing the evidence. However, if we put a substantial initial prior without any evidence, it will be immediately visible to the reader.

4.3.3 Visualization

Dimension reduction methods allow the representation of observations in space with fewer dimensions than in the original data. Dimension reduction is often an intermediate step in classification, cluster analysis, or regression. In certain situations, it improves the effectiveness of these methods, increases stability and sometimes allows the inclusion of many variables in the analysis. The data is reduced to a two-dimensional space, making it easy to present

them on a graph. Methods from this group are also called feature extraction methods. Due to dimension reduction, new features might be used for other issues. It is also a popular method for visualising multidimensional variables, which we used in our study.

4.3.3.1 PCA

Its purpose is to extract the essential data from the matrix. It is then represented as a set of new orthogonal variables called principal components to display the pattern of similarity of the observations and the variables as points in two or three-dimensional space.

The principal component analysis defines new variables by combining their smallest possible subset. They are linear, weighted combinations of the original data set. The aim of creating new variables is that they will explain the total variability in the data set as much as possible. The new variables will form an orthogonal basis in the feature space. We should select variables so that the first variable reflects as much variability in the data as possible. After projecting the observations onto this vector, we want the variance of the projections to be the highest. After determining the first variable, the second is orthogonal to the first and explains as much of the remaining variability as possible. Another variable should be orthogonal to the first two, and the procedure continues until no variable remains.

4.3.3.2 SVD and TSVD

Singular Value Decomposition (SVD) is a factorization of a real or complex matrix into three matrices: U, Σ, and V*. The matrix U and V* are unitary matrices and Σ is a diagonal matrix containing the singular values of the original matrix. Truncated Singular Value Decomposition (TSVD) is a variation of Singular Value Decomposition (SVD) that only retains a subset of the singular values and corresponding singular vectors of a matrix. The idea behind TSVD is to reduce the computational cost of SVD by only considering the largest singular values that capture the most significant information of the original matrix. TSVD is commonly used in dimension reduction, denoising, and compression tasks.

We have used TSVD to visualize datasets.

4.3.3.3 tSNE

t-SNE stands for t-Distributed Stochastic Neighbor Embedding. It is a non-linear, unsupervised technique of dimensionality reduction. Its primary purpose is to explore and visualise multidimensional data. The t-SNE algorithm computes a measure of similarity between pairs in large- and small-dimensional spaces. For the next step, it tries to optimise both measures of similarity. In simple terms, t-SNE tries to simplify how multidimensional data is distributed in euclidean space. t-SNE is not designed to cluster data but primarily to explore and visualise

data. However, we can assess how many groups may be hidden in the data by visualising data in two or three-dimensional space [19].

Several R packages can create a t-SNE map: *tsne*, *Rtsne* or *cuda.ml.*

We used package cuda.ml because it utilises graphic card capabilities to produce results. It is efficient and swift. Using hundreds of cores is a huge advantage when dealing with large datasets. The major drawback is the VRAM storage. The VRAM capacity in the average graphic card oscillates between 4GB-8GB. The 6GB we used was insufficient for some of the datasets, so we had to resort to RAM/CPU calculations.

4.3.3.4 Random projection

Random projection is another computational technique used for dimensionality reduction. It is a method of representing high-dimensional data in a lower-dimensional space by using random linear projections.

In random projection, a random matrix is generated and multiplied with the original high-dimensional data points to produce the lower-dimensional projections. The idea behind random projection is that, for many applications, the high-dimensional data lies on or near a lower-dimensional subspace. The random projection provides a way to estimate this lower-dimensional subspace. One of the critical advantages of random projection is that it is fast and computationally efficient. Unlike dimensionality reduction techniques like principal component analysis (PCA) or singular value decomposition (SVD), random projection does not require eigendecomposition or singular value decomposition, which can be time-consuming.

5 Results

In the following chapter I presents results of clustering simulated multivariate and multinomial data, then I proceed to the real dataset. Each real dataset clustering is preceded by its description

5.1 Simulated data analysis

5.1.1 Multivariate normal mixtures

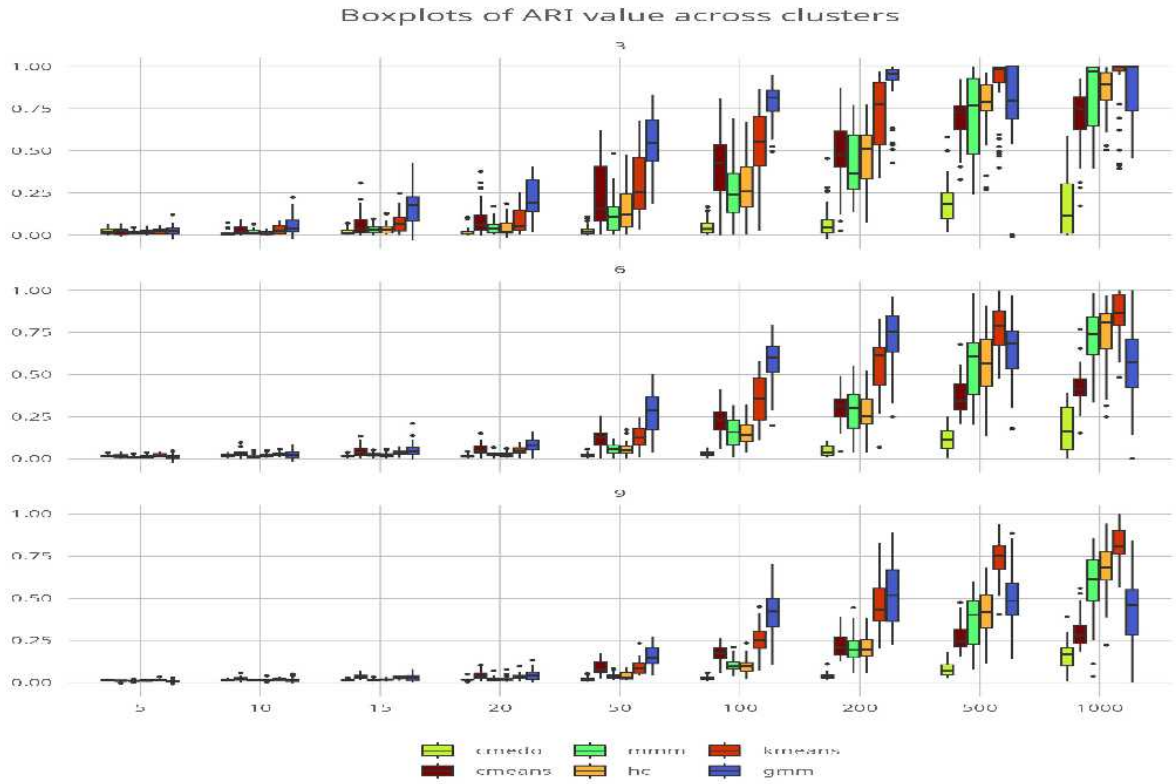

Figure 5.1: ARI index accross clusters in Simulated Multivariate Normal Distributions

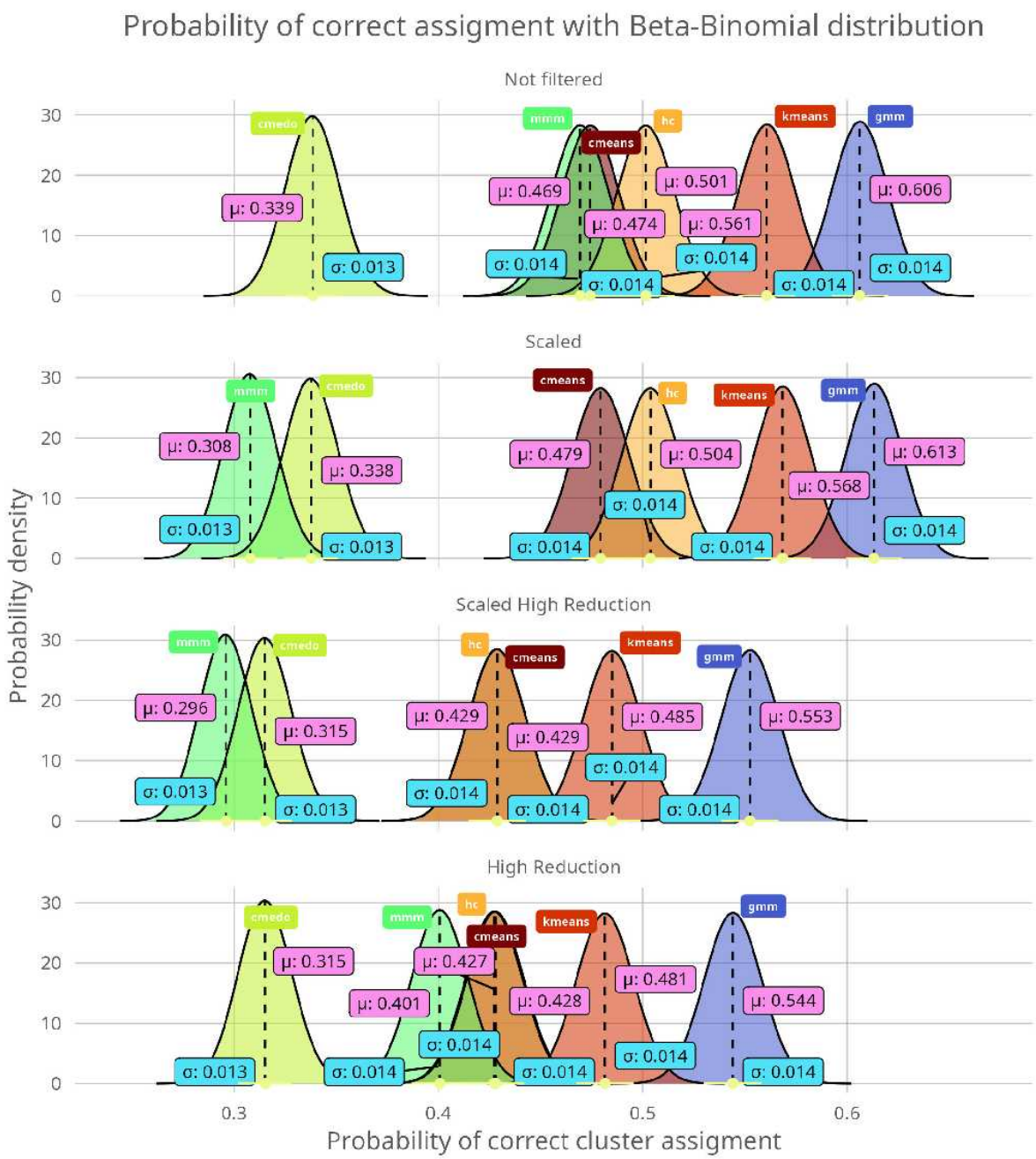

Figure 5.2: Various indexes accross clusters in Simulated Multivariate Normal Distributions

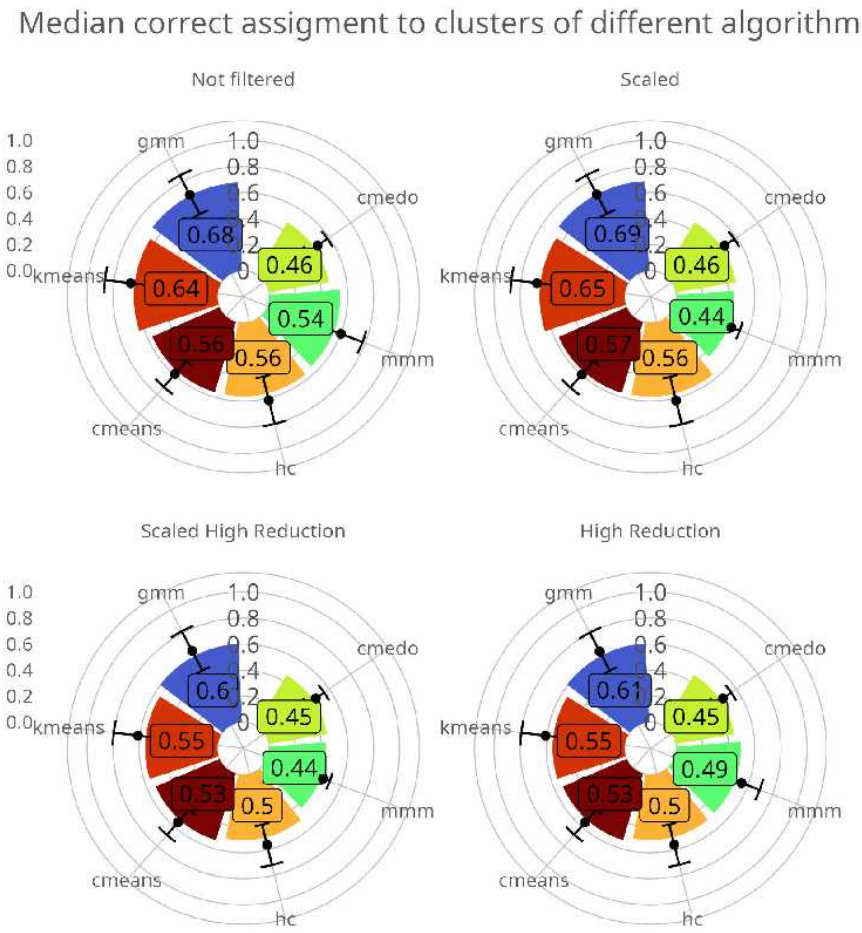

Figure 5.3: Various indexes accross clusters in Simulated Multivariate Normal Distributions

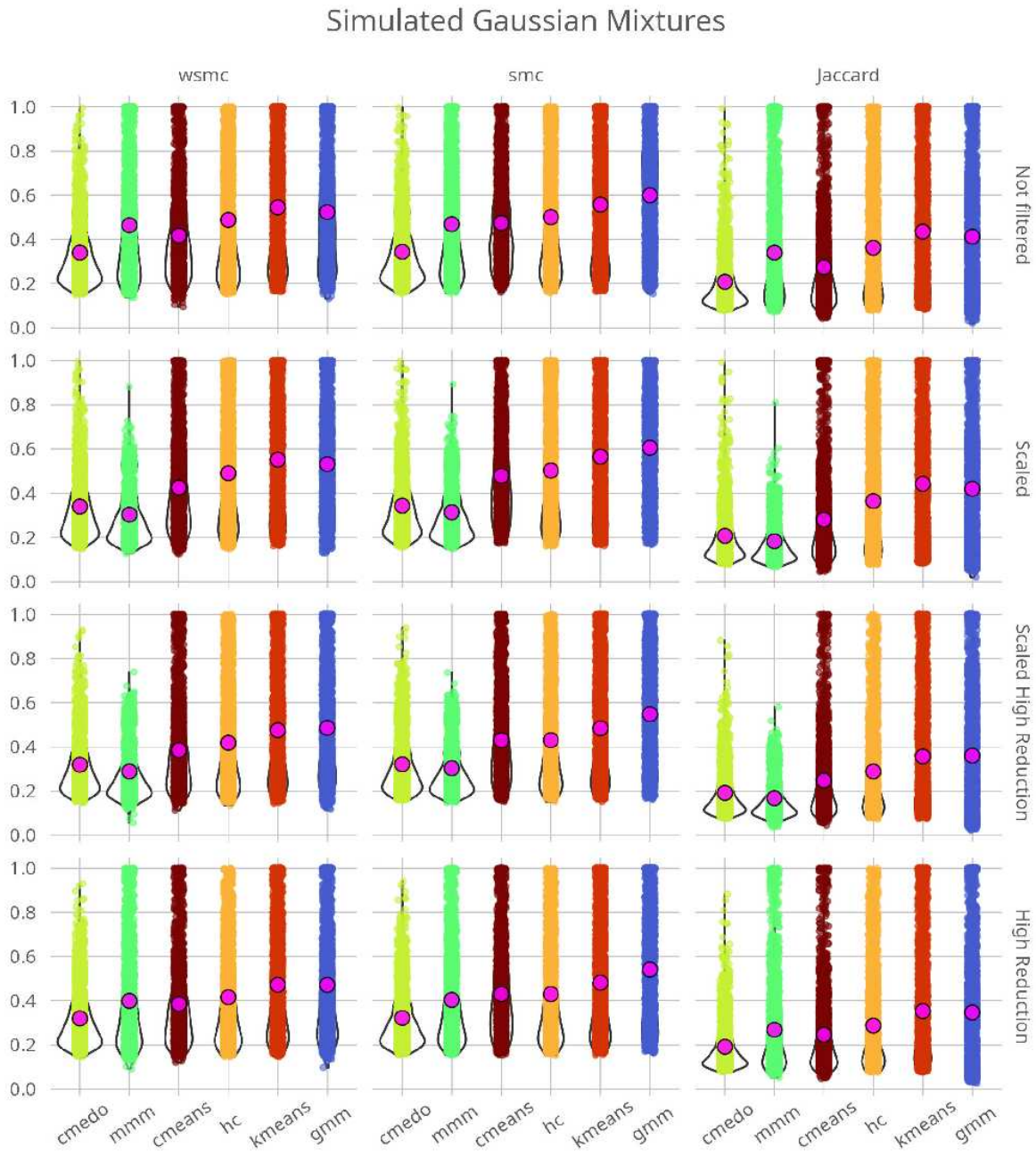

Figure 5.4: Various indexes accross clusters in Simulated Multivariate Normal Distributions

The metrics are highly correlated which indicates an agreement between them.

5.1.2 Multinomial mixtures

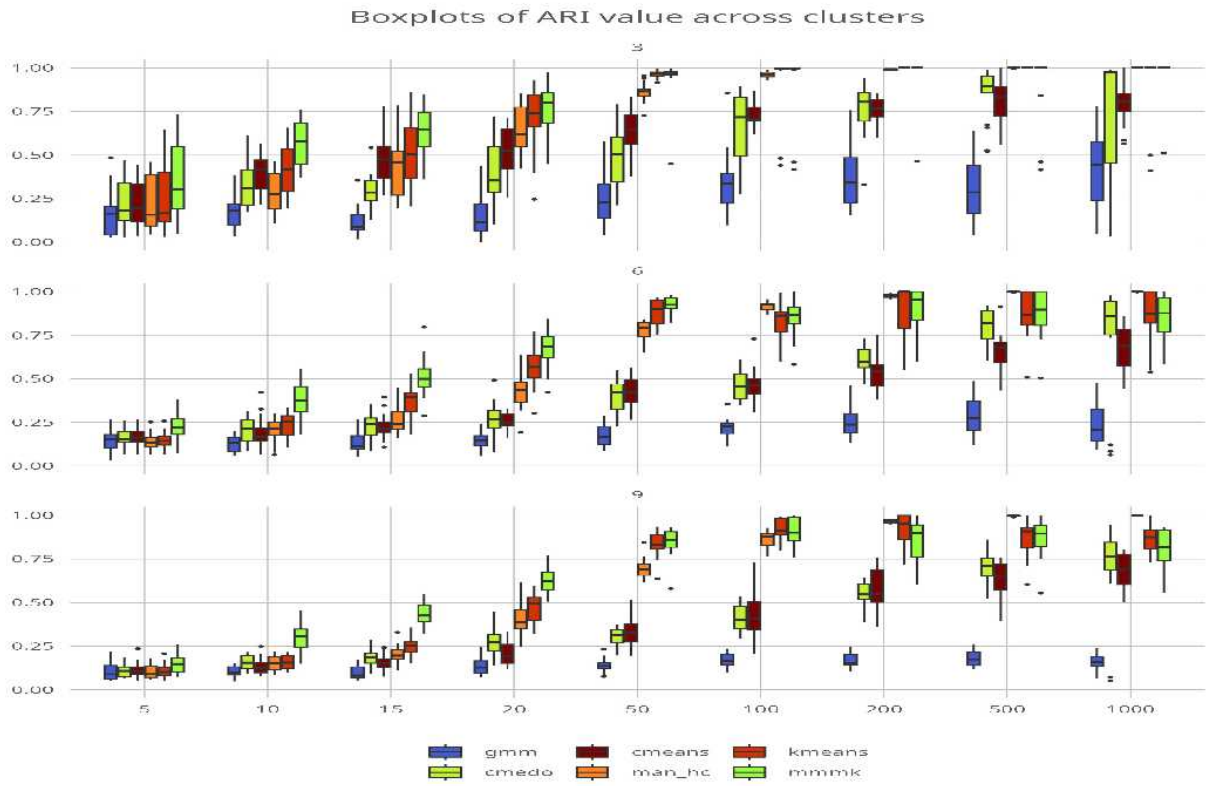

Figure 5.6: ARI index accross clusters in Simulated Multinomial Distributions

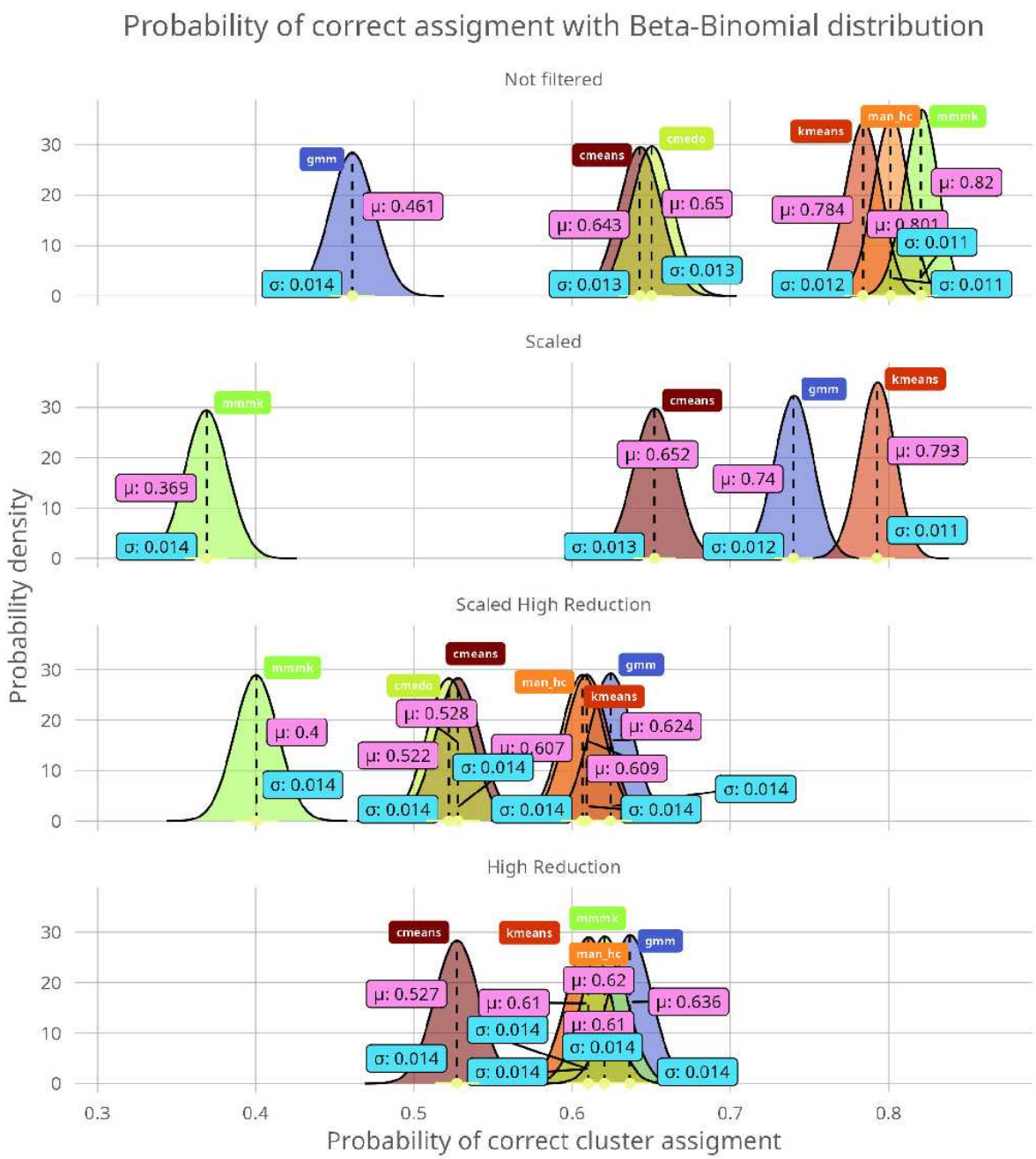

Figure 5.7: Various indexes accross clusters in Simulated Multinomial Distributions

In the figure 7 we can see thatthe higest result belongs to **MMM-K** when the data was not filtered. In addition its probability density of mean value is the highest, altought only slightly higher than in the case of **HC-MAN**. In

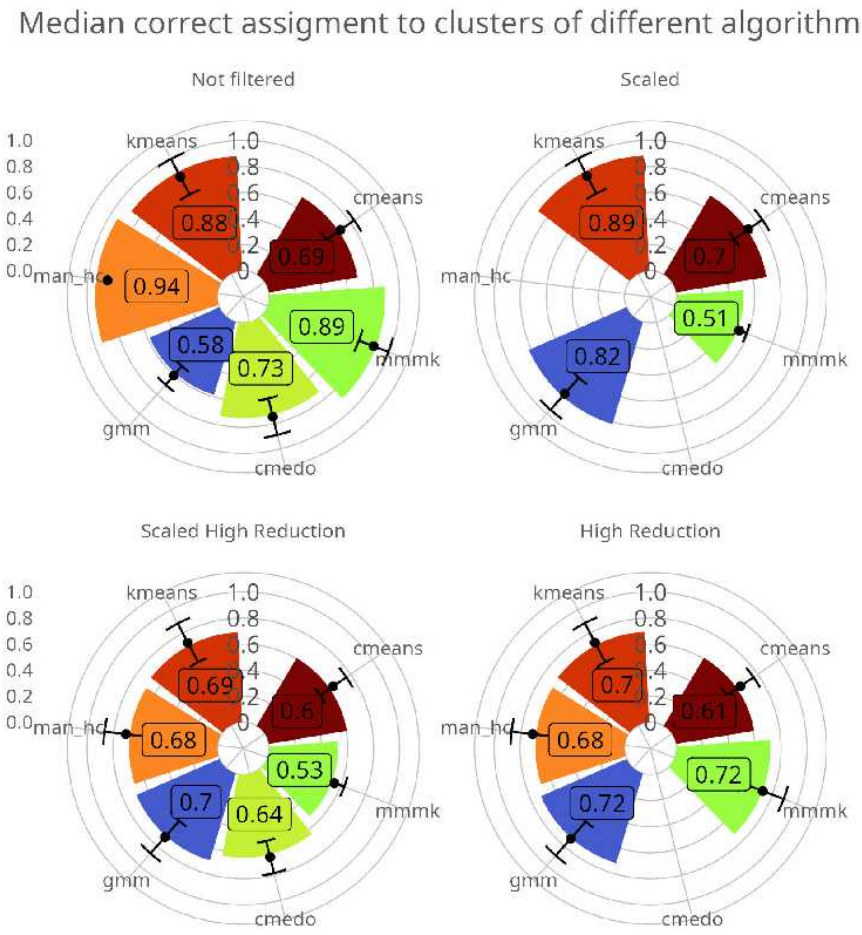

Figure 5.8: Various indexes accross clusters in Simulated Multinomial Distributions

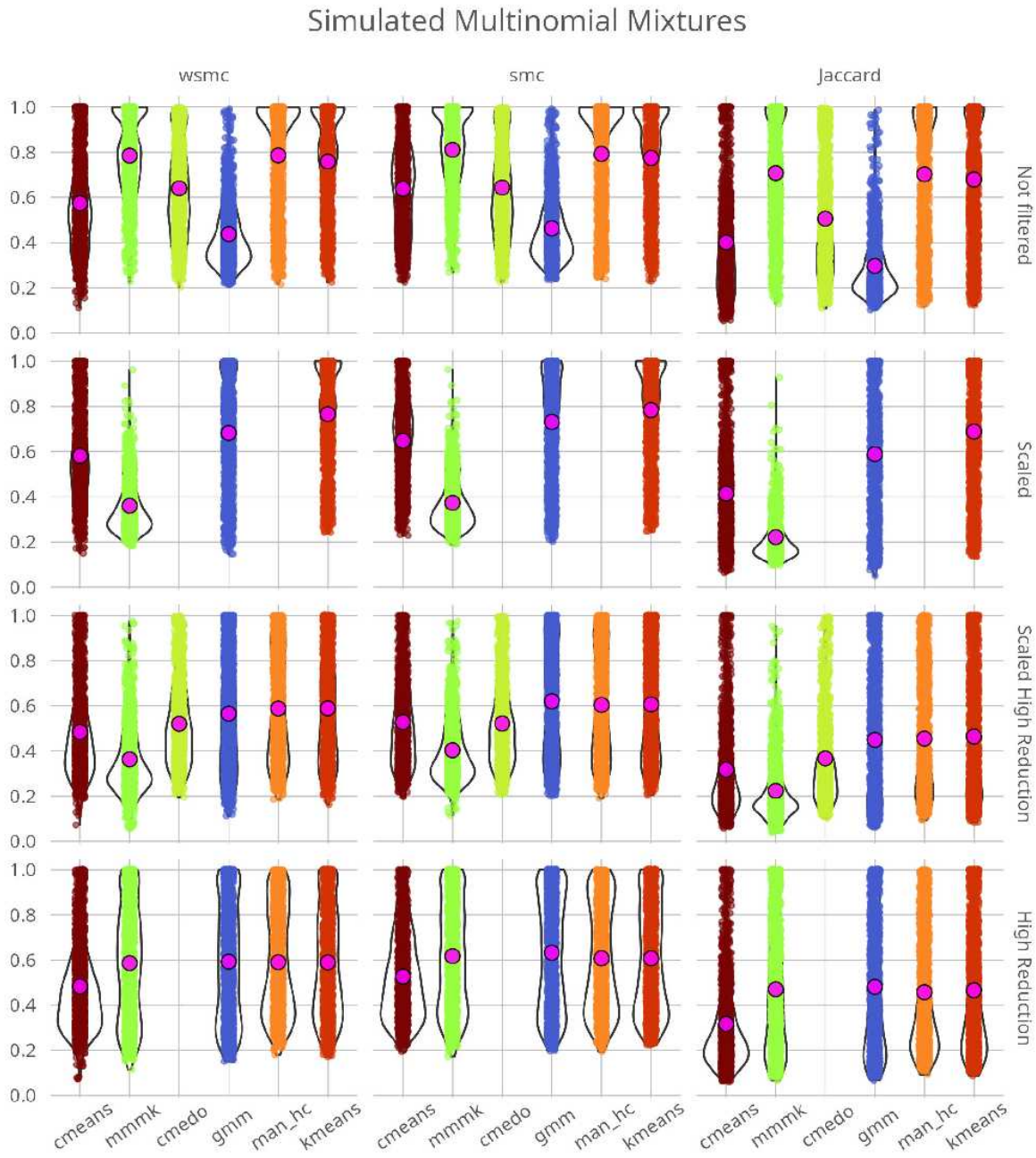

Figure 5.9: Various indexes accross clusters in Simulated Multinomial Distributions

Extremaly hight correlation of various metrics indicates, that they are in agreement regarding the results.

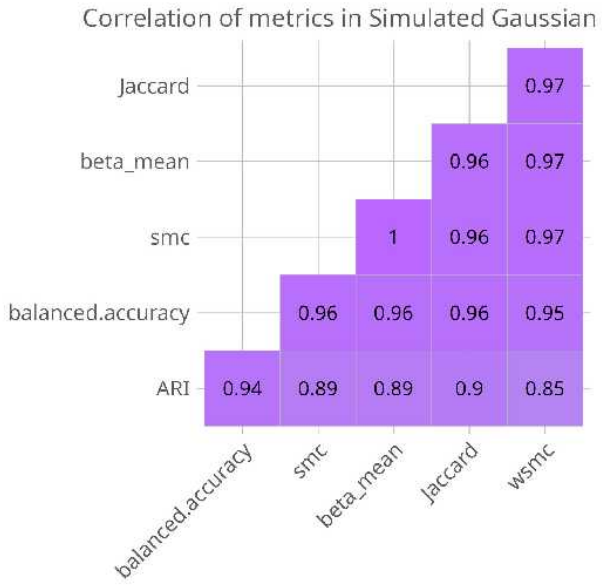

Figure 5.5: Metrics correlation in Simulated Multivariate Normal Distributions

5.2 Real data analysis

5.2.1 Somatic mutation counts

The first dataset concerned somatic mutations in DNA in cancer patients. Somatic mutations in cancers is an important and extensively researched topic. Somatic mutations can be caused by exposure to exogenous or endogenous mutagens or during DNA replication errors. Two major types of somatic mutations are distinguished: driver and passenger. Driver mutations are considered to confer cell growth, and related to cancer development, while passenger mutations have less or no impact on cancer growth in the organism. Very roughly, difference between driver and passenger mutations can be casused by their positions in DNA or their impact on transcripts (mRNAs, proteins) produced. If a mutation hits exon it is more likely to become a driver. Contrary mutations in introns would rather be passengers. In the aspect of protein coding, substitution of the nucleotide in the DNA can be synonymous or non-synonymous. Synonymous somatic mutations leave the resulting amino-acid unchenged, while non-synonymous reault in modifying the resulting amino-acid or results in producing no amino-acid at all. However, the problem of distinguishing between driver and passenger mutations in much more complex. There are numerous studies trying to estimate the impact of mutations on the risk of cancer development [20].

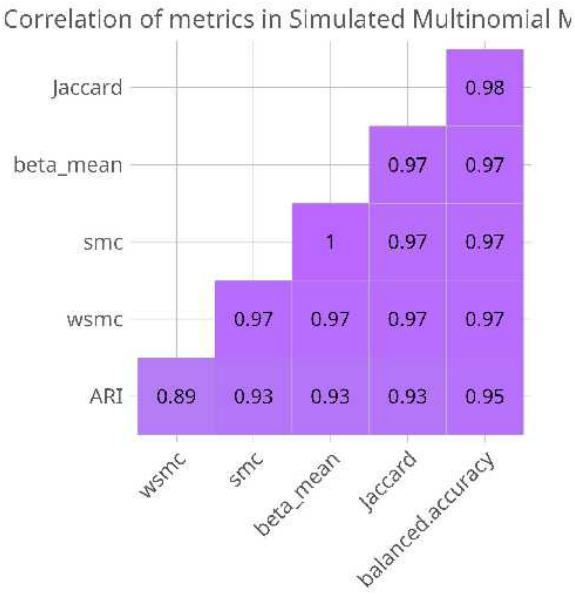

Figure 5.10: Metrics correlation in Simulated Multinomial Distributions

We wanted to cluster patients diagnosed with different cancer types, based on counts of somatic mutations in genes. Due to the complicated problem of distinguishing between driver and passenger mutations, in our computational experiments we take numbers (counts) of mutation occuring in genes as our observational vector data. The hypothesis behind this computational experiment is that the recorded information (mutation counts in genes) can be used to distinguish between various types of cancer. The second goal is to determine which of the clustering algorithms will perform best in the clustring task, when the quality criterion is agreement between unsupervised clustering results and the ground true.

Original, raw data was taken from The Cancer Genomic Atlas (TCGA), https://portal.gdc.cancer.gov/. For the initial analysis, BAM files that were submitted to TCGA, were converted to FASTQ format. The alignment of DNA sequences was done using BWA-MEM or BWA-aln, depend-

ing on the read length. Human genome used as reference was in version **GRCh38.d1.vd1**. In parallel to BWA algorithm, GATK was also used to improve quality of alignment. In the next step, five Somatic Variant Callers were used: MuSE, MuTect2, VarScan2, SomaticSniper, Pindel.

The data that we used in the experiment, was annotated using somatic variant caller **Mutect2**. It is one of few popular tools to detect **SNVs** (Single Nucleotide Variations) and indels (insertions and deletions in the DNA) [21]. According to Mutect2 probabilistic model, we filtered predicted somatic mutations. The data that we extracted included following information: sample number, which reflects single, anonymized patient, name of the gene in which mutation occurred and frequency of mutations. For the experiment, we chosen 10 different types of cancer, semi-arbitrally

For each of **SVC, Variant Effect Predictor** was used. Ensembl database is a comprehensive source of genomic data, and the Variant Effect Predictor (VEP) is a part of it. It is a multipurpose tool to annotate possible mutation effects. It can predict whether the mutation was synonymous, missense, or stop gained. VEP also estimates the potential impact on the organism, which ranges from low to moderate up to high. The vital characteristic of VEP is that it provides many predictions regarding possible gene variants or transcripts. It means that numerous different variants might describe one mutation. In rare cases, we can obtain contracting suggestions, like missense and synonymous variants of the exact mutation location. It also provides information on possible gene mutations that occurred in the desired format. We choose Ensembl genes over common HGNC names. The structure presented by Ensembl seems to be more consistent and not prone to drastic changes, as in the case of HGNC - when one gene might have had a few different aliases.

We took two different approaches to this study. In the first case, given that patient in a specific location had more than one variant, we sum them up. Then we took the variant which occurred the highest number of times, along with the respective gene. Assume mutation in chromosome 1 in nucleotide 144. As for the second case, we have filtered all variants with missense, synonymous or stop gained consequence. Then, similarly to the first case, we took the variant which appeared the highest number of times, together with its gene.

Sample ID	HMCN1	HSPG2	HTT
TCGA-2Y-A9GU-10A-01D-A3	0	0	0
TCGA-2Y-A9GV-10A-01D-A3	1	0	0
TCGA-2Y-A9GW-10A-01D-A3	2	0	0

Table 5.1: Part of somatic mutation frequency table for one type of cancer

Next, we mixed mutations frequency information from all selected cancer types, in all possible combinations without repetitions. In total, we created 57 mixtures consisting of 2, 3, 4, 5 and 6 components. Each set consisted of almost 35 000 variables and number of observations ranged between Since the data is categorical and data points belongs to $\mathbb{I}$, we assumed that

this type of data might be described by mixture of multinomial distributions.

In the first scenario, we took only the first gene, disregarding the others. In the second one, we took all possible genes that might be transcribed from the sequence.

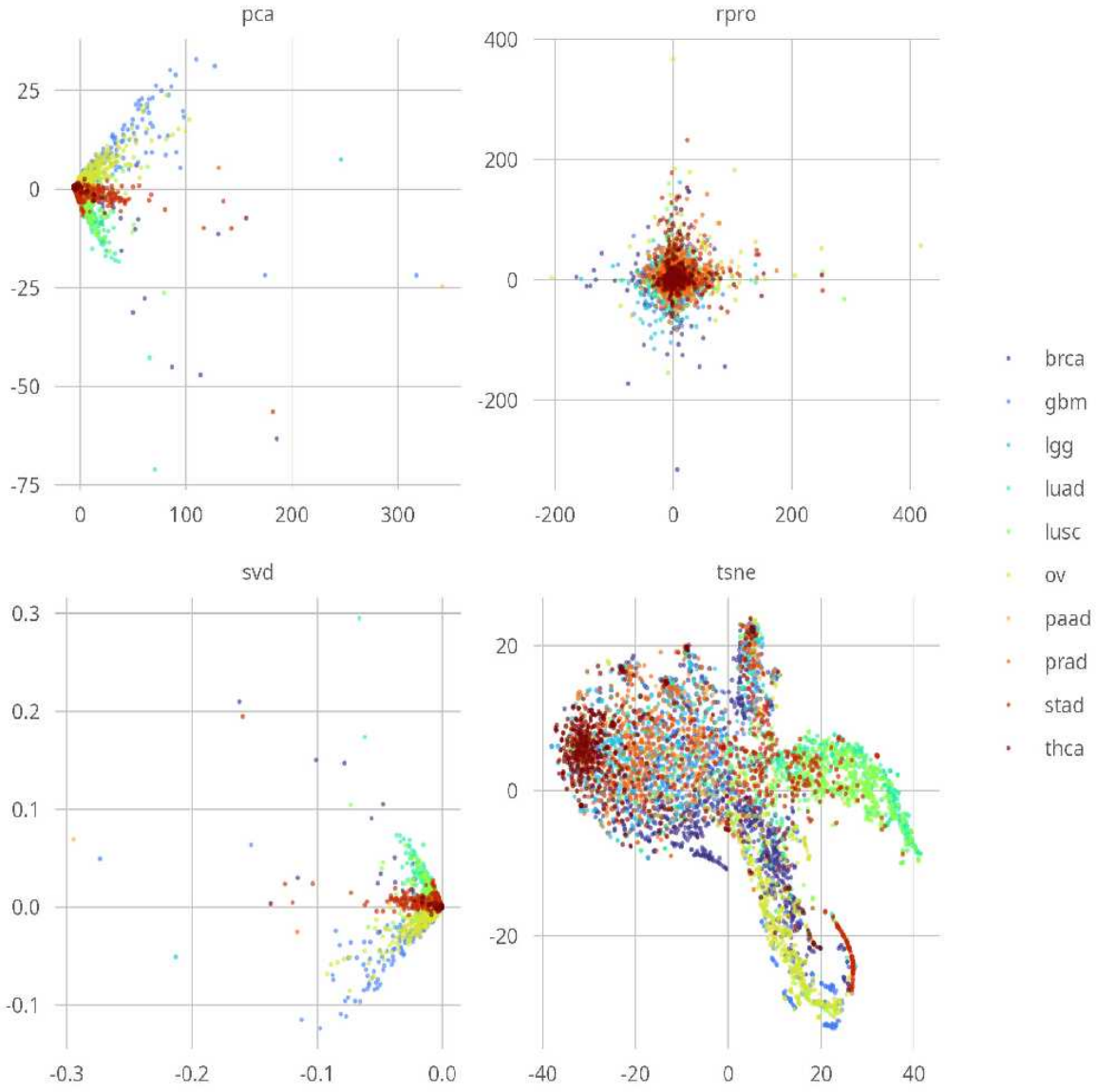

Figure 5.11: t-SNE plot of Somatic Mutations Count with all variants

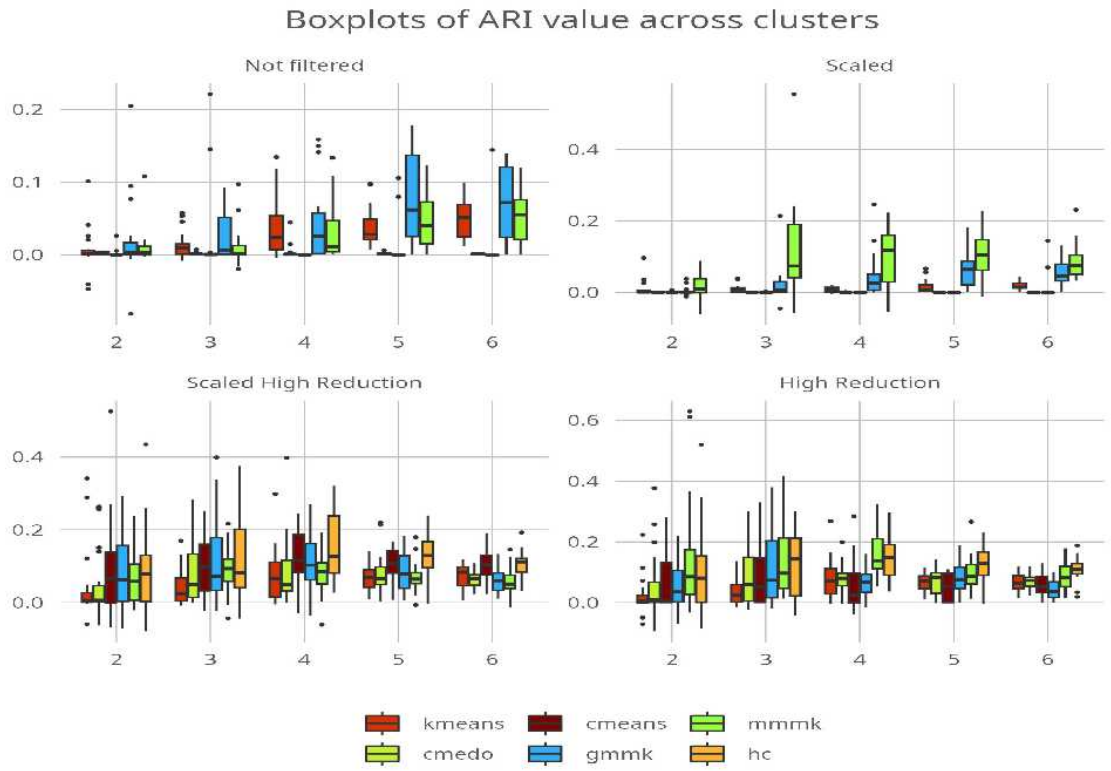

Figure 5.12: ARI index accross clusters in Somatic Mutations Counts

Results Looking at the chart presenting the ARI index, with varying components, we can see that scores tend to be more condensed around the median with the increased number of clusters. However, the slightest such differences can be seen in the case of the k-means algorithm. Its scores are very condensed around the median from the beginning, however low to zero values.

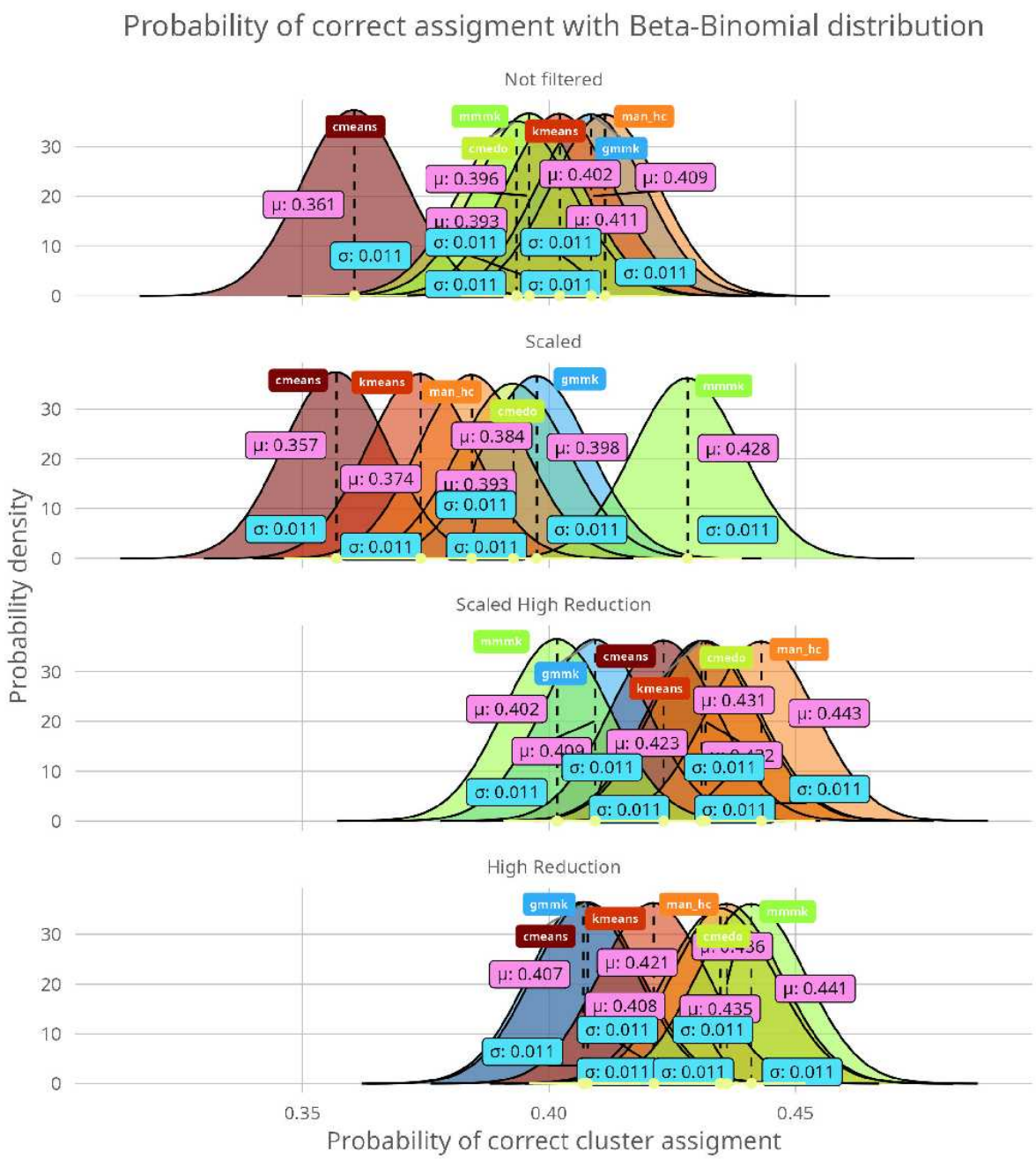

Figure 5.13: Various indexes accross clusters in Somatic Mutations Counts

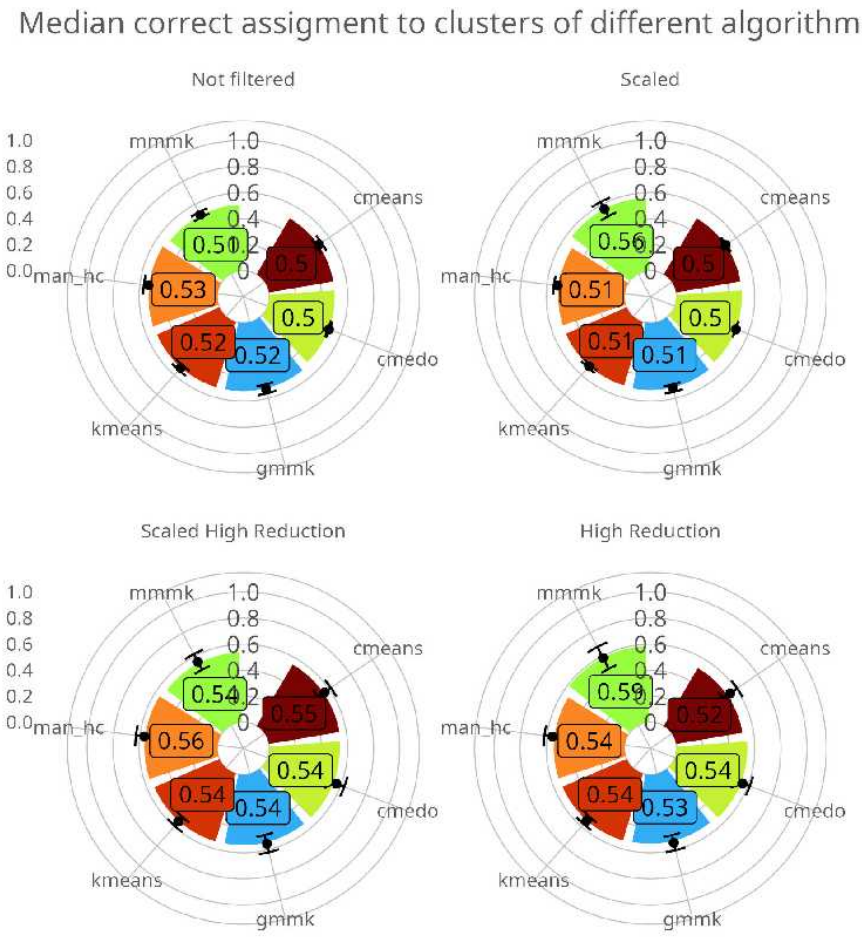

Figure 5.14: Various indexes accross clusters in Somatic Mutations Counts

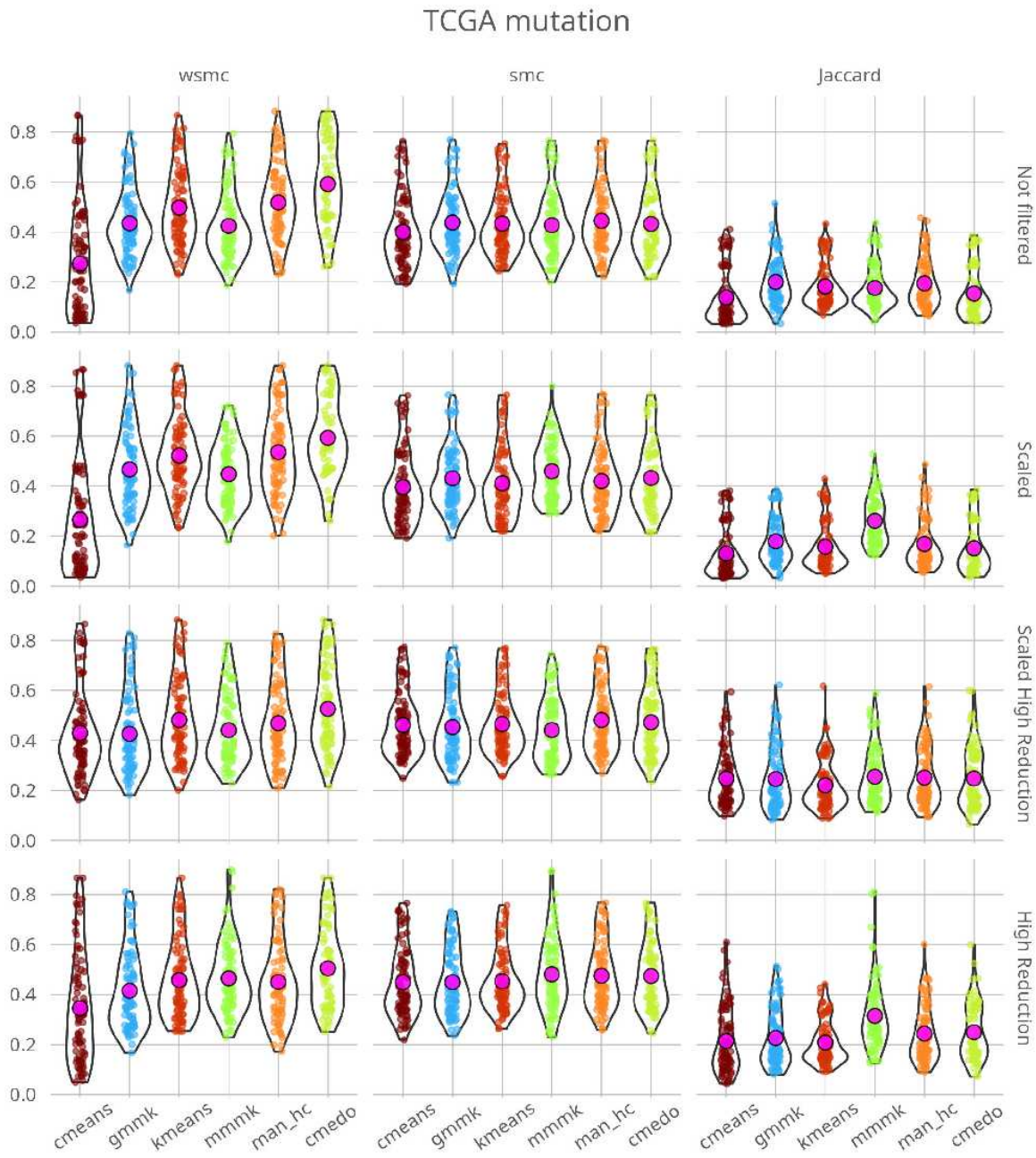

Figure 5.15: Various indexes accross clusters in Somatic Mutations Counts

In the case of somatic mutation counts, we already saw that observations seem to mix, rendering finding patterns a complex process.

In the case of the other algorithms, we notice more diverse scores when we compare only two or three types of cancers. In other words, when compared together, some cancers might show a more robust pattern of somatic mutations than others. Although sometimes it might be caused simply by class imbalance (create a such experiment with simulated results). For example, when we compare breast cancer to lung adenocarcinoma, the ARI index equals almost 0.67. Albeit both classes are well balanced.

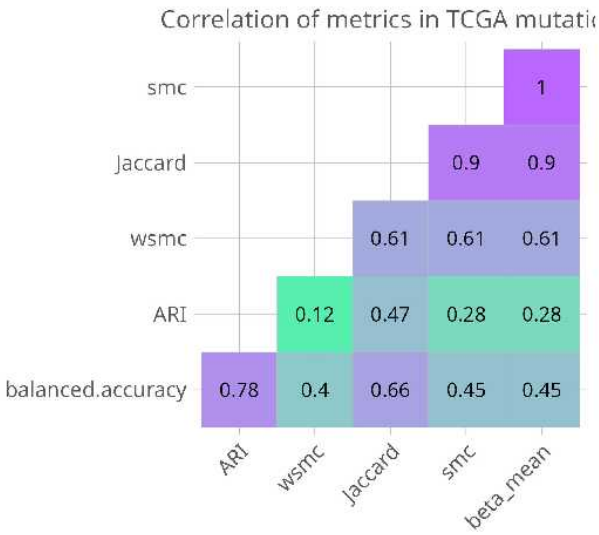

Figure 5.16: Metrics correlation in TCGA mutations

The other interesting situation is when we receive a negative ARI index. In some cases, it might be the fault of incorrect implementation. If that is not the case, we might say that results are, in fact, less than expected. We can witness such a situation when we compare Glioblastoma and prostate adenocarcinoma. The ARI index is below 0. If we exclude the probability of wrong implementation (checking a few different ones), the results are less than expected. Some remarks were made to indicate that results might be complementary or orthogonal [https://stackoverflow.com/questions/42418773/how-can-we-interpret-negative-adjusted-rand-index].

If we mix more cancer types, those differences are gradually less observable. This situation might be related to the homogeneity of the somatic mutation in different cancers. Taking a large number of observations, along with the lack of an explicit probability model for selecting an appropriate variant, we are receiving a mixture of similar mutations. https://www.nature.com/article

Apart from comparing the two clusters, we can observe that the general median is similar for all groups.

GaussEM, a model-based algorithm, scored the highest of all algorithms. Our educated guess was that mixtures of multinomial distributions might be the best model to describe somatic mutation count data. However, the presented results show that in all cases, GaussEM performed better than MultinomialEM. The third best algorithm was Hierarchical Clustering. The ARI index picture shows a negative ARI index in the frame where we looked for patterns in two cancer groups. There are two possible reasons for that. The first one is that implementation might contain errors. We rejected that possibility, as we tested a few implementations,

including one from the mclust package. The results were the same. The second explanation is that random clustering results were less than expected. We do not observe a similar phenomenon in the rest of the clustering groups (3-6).

Three out of four presented indexes show little differences in model-based algorithms. The score difference was less than 1% in the ARI index, 0.2% from the WJAC and 0.6% in SMC. The WSMC index showed that GaussEM performed better than MultinomialEM by almost 2.1%. At the same time, the maximum difference between the lowest and the highest-performing algorithm was about 10%. A similar situation is seen in the case of the ARI index, at almost 13%, and WJAC, at 14.5%. The lowest distinction in performance can be seen in the SMC index, only 4.1%. Binomial test The majority of the results are statistically significant. In the picture, the lower score comes with better results. The picture shows that MultinomialEM had the lowest p-value scores, with a median below 0.25. The second one, with a median result of about 0.3, was Gaussian Mixture EM.

5.2.2 Gene expression

Gene expression is a series of events leading to the display of the information in the cell. Roughly speaking, we can say that gene expression lets arise phenotype, something we can observe, from genotype.

We can consider the genome as information storage. However, it cannot pass the information to cells on its own. To use biological information encodedsdf in the genome, enzymes and various proteins must participate in complex biochemical reactions that lead to Genom expression. The first product of Genom expression is transcriptome, a group of RNA particles. They come from those protein-coding genes that the cells need the most. The transcriptome is created during the process called transcription when genes are rewritten as RNA particles.

Some of those particles are called mRNAs or messenger RNA. The primary role of mRNA is to function as a template for translation. During this process, their sequences are first translated to amino acids, which then build functional proteins. During various events, like gene mutation, the expression level of mRNA might be increased, decreased or even halted. It can be related to multiple diseases, including cancer.

cBioPortal is an interactive interface to the resources such as TCGA. It provides open access to molecular profiles and clinical attributes of different cancer genomic studies.

The data is primarily multidimensional and contains, but is not limited to, data on DNA methylation, mRNA and microRNA expression or phosphoprotein level data (RPPA). We used mRNA (messenger RNA) expression data for our analysis. First, we wanted to confirm if such kind of data contains enough information to distinguish between different types of cancer. In other words, if various cancers show different expression patterns.

The format of gene expression of cancers was already suitable for our purposes. We only transposed the matrix to present sample numbers as observations and genes as variables. The

example is shown in the table.

We chose the data that consisted of the median expression level of RNA sequencing data. We have chosen the same cancer types as in the case of somatic mutation frequency. However, in the case of gene expression data, we had slightly different numbers of observations and features, as shown in table 4. In total, we again choose ten types of cancer. To create mixtures, we randomly selected 2, 3, 4, 5 and 6 components and joined them together. Each component represented one cancer gene expression data. In this way, we created 10 two components mixtures, 10 three components mixtures, and up to ten six components mixtures. Altogether, we made 50 mixtures in all possible combinations without repetitions. In the case of data expressions, all data points belong to R, which is quite different than in the case of "mutation counts". Our assumption here was that mixtures of normal distributions might well approximate data.

Sample ID	**HMCN1**	**HSPG2**	**HTT**
TCGA-2Y-A9GU-10A-01D-A3	24.32	0	1.5
TCGA-2Y-A9GV-10A-01D-A3	124.12	25.2	0
TCGA-2Y-A9GW-10A-01D-A3	42.24	12.1	52.32

Table 5.2: Part of somatic mutation frequency table for one type of cancer

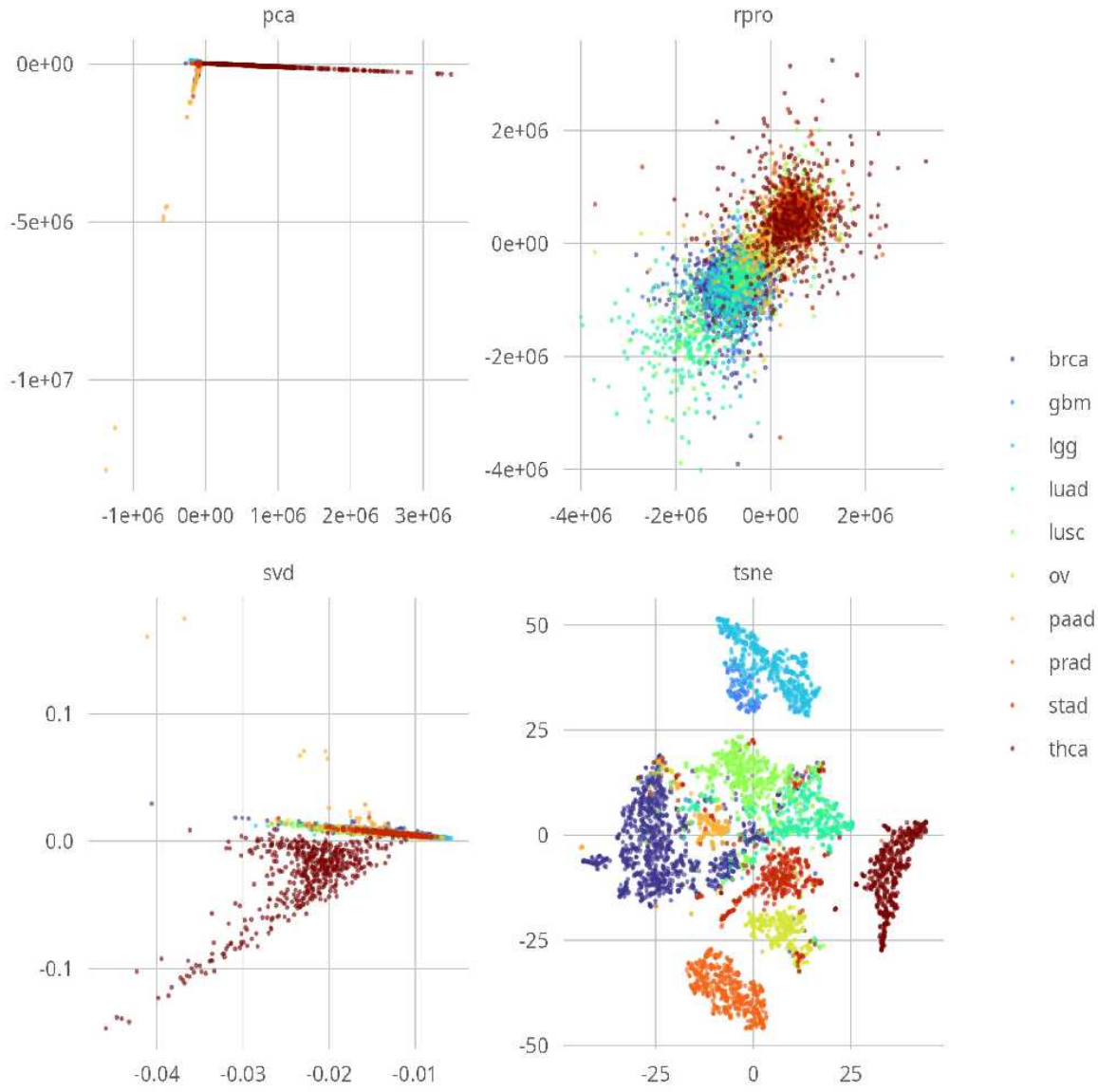

Figure 5.17: t-SNE plot of Gene Expressions

Results

The following result shows only four out of 6 algorithms, because HC and k-medoids methods were not able to complete the tasks without numerical and memory errors.

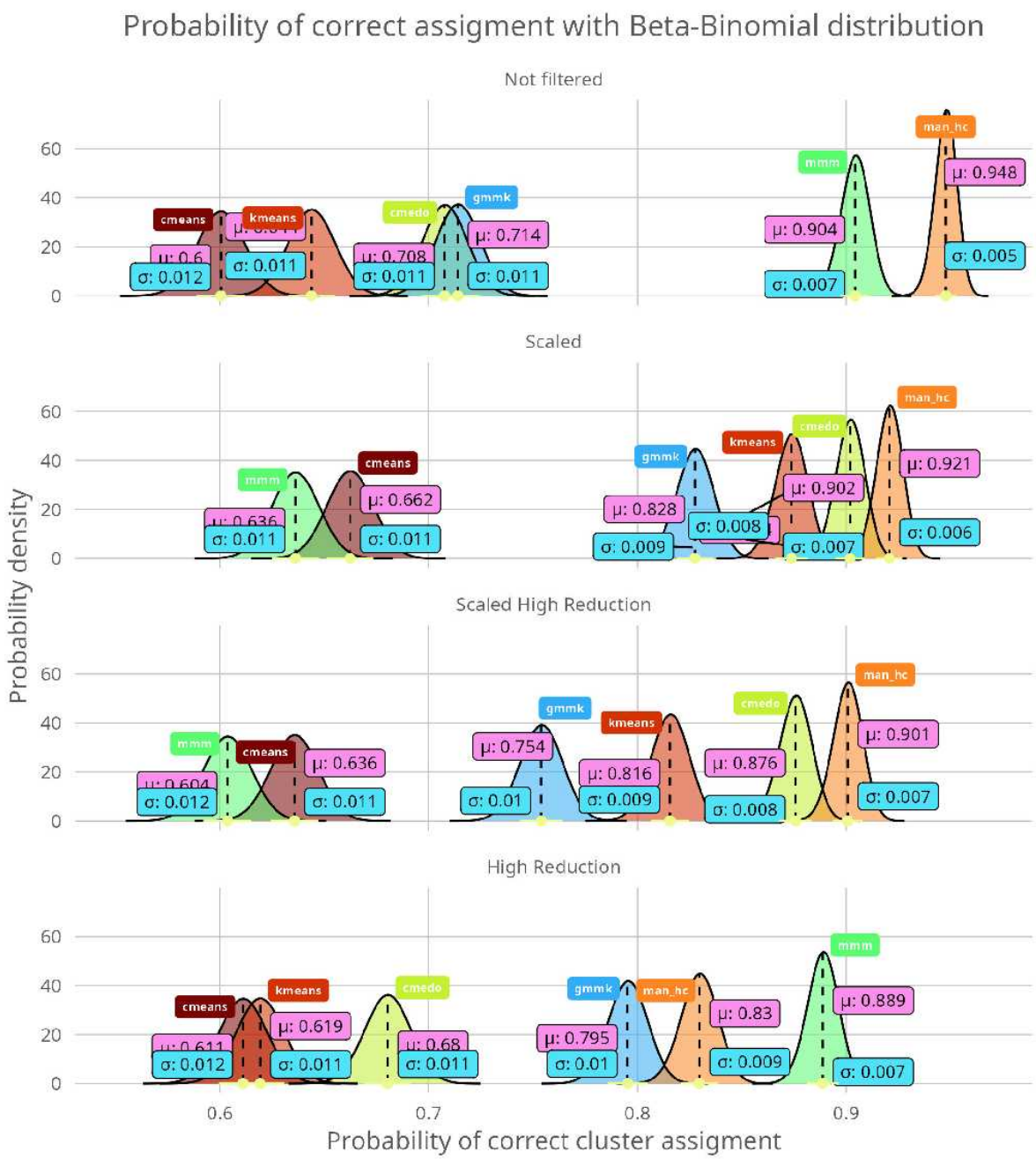

Figure 5.18: Various indexes accross clusters in expressions

In the figure showing BB metric, we see that the most rightside pitch belongs to mmmk. Moreover, it has the highest probability density, so we can be more conviced in its probability of correct assigment, than in the case of other algorithms. In addition, it has the lowest standard deviation.

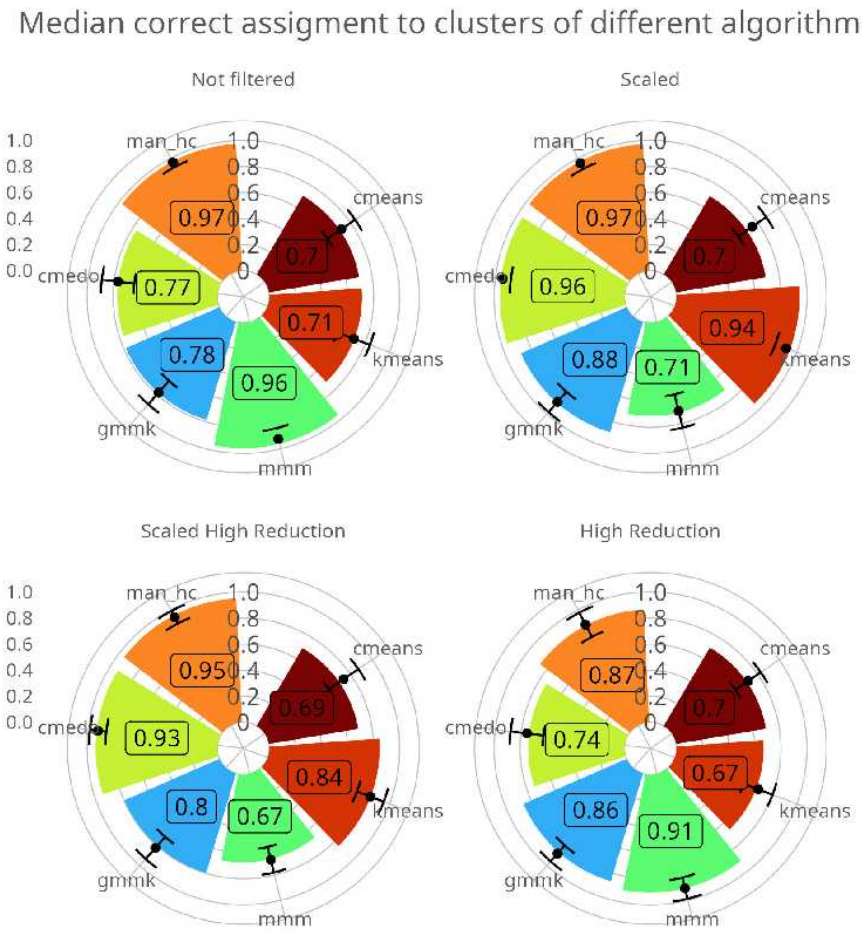

Figure 5.19: Various indexes accross clusters in expressions

In the MCA , we see that although mmmk perormed better overall, gmmk had advantage when the data was scaled.

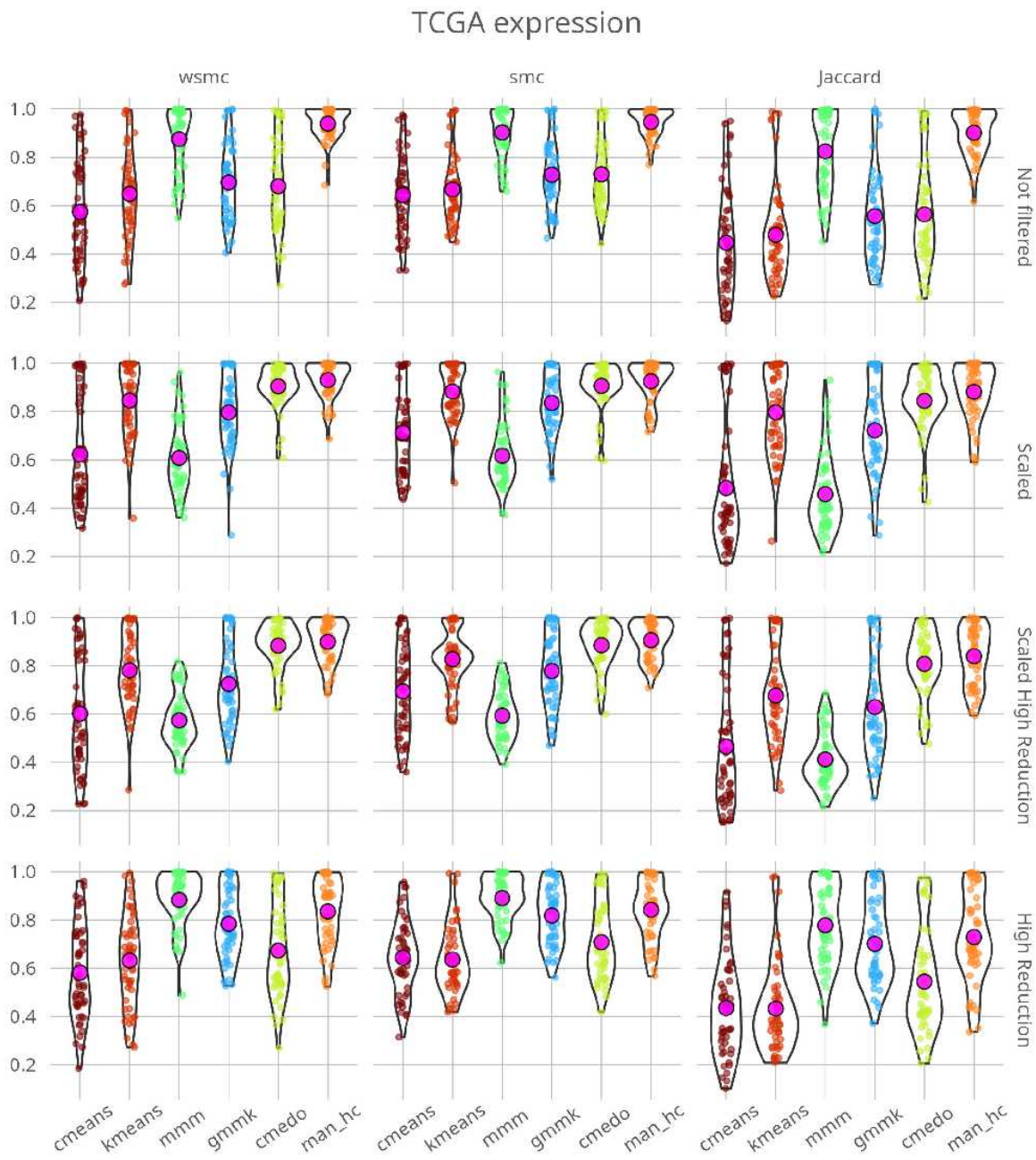

Figure 5.20: Various indexes accross clusters in TCGA expressions

Similarly to the somatic mutation counts, we can observe an initial increase in the median ARI index with its peak at the four component mixtures. It follows a decrease in scores for all algorithms but with varied impacts. However, the average score difference is much more visible in the case of cancer gene expressions than somatic mutations, as all of the algorithms started with a relatively high score.

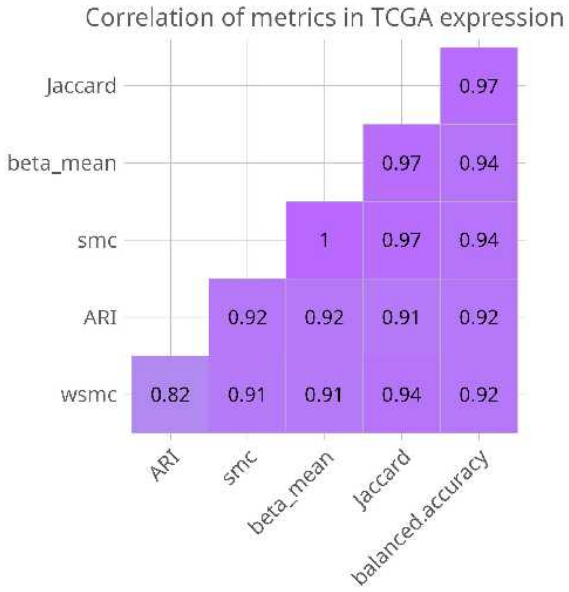

Figure 5.21: Metrics correlation in TCGA expressions

Contrary to the somatic mutation counts, single scores in the first three cluster groups covered a much broader spectrum between 0 and 1. We also noticed that almost all algorithms performed the best when only four mixture components were present. Here, "the best" means that scores gathered around the median took the highest place across all groups of datasets. In the case of k-means and fuzzy c-means, the median black band lies around 0.25. It is much above 0.5 in the others. However, it might be a feature of selected datasets. (check the statistical significance of this)

Where five or six groups were present, HC scored significantly lower than GaussianEM and MultinomialEM. Moreover, in 6 component mixtures, both algorithms keep on a similar level to the median score of the ARI index. However, scores of MultinomialEM were more condensed around a slightly lower median.

In the general ARI index perspective, both model-based algorithms and HC scores reached values above 0.75. In the case of fuzzy c-means, we noted only one value above 0.75, and k-means scored none. Apart from MultinomialEM, each algorithm scored some values below 0, with the k-means having the highest count.

Looking at the shape of violin plots, we notice that for the distance-based algorithms, the bottle is wider near the lower values. In model-based algorithms, it is the other way around. The bottle points towards lower values, being broader near the higher values. We cannot say the exact thing about the AJAC index. It looks like the former index additionally emphasised differences. For example, in k-means, the band near the value of 0.25 is much broader than in the ARI index. In fuzzy c-means, we can observe the enlargement of the bottle located between 0.5 - 0.75 value bands. At the same time, its broader part in the ARI plot near the value of 0.25 became thinner. The shape of the Hierarchical algorithm became more streamlined than in the case of the ARI index. Also, GaussEM became less streamlined

and lost its pointer toward lower values. In fact, according to the AJAC, it did not have any values near 0. The thicker part of MultinomialEM went between bands 0.5 and 0.75. At the same time, its pointer became thinner, showing fewer values near 0. For both ARI and AJAC indexes, we observe a significant difference in average performance in favour of the model-based algorithms. It is more than two times better for the ARI index and nearly twice for the Averaged Jaccard. In addition, looking at the AJAC, GaussianEM's lowest values were near the band 0.25. Although MultinomialEM scored 0, it might be related to the random initialisation stuck in a local minimum. Overall, its mean performance was higher than the GaussianEM above. As previously, the SMC and WSMC metrics showed better results. Metrics from both scores seem to be more consistent, which is most likely caused by well-balanced classes. In distance-based algorithms, we notice a steady step of about 0.04 units, starting from k-means up to HC. Then, the measure between distance and model-based algorithms increased up to almost 0.1.

Finally, the step between MultinomialEM and GaussianEM was about 0.04 again.

5.2.3 Codons frequency usage in different species

We have 20 amino acids that are coding proteins built from codons. A single codon consists of three DNA bases. Historicaly, DNA code was considered universal among all the species. It meant that three nucleotides called a codon, code the same amino acid for most of them. Currently, we know that it is not an entirely valid statement. Although the genetic code shown in the table is used in most of the genes across almost all species, we can find many exceptions. In particular, the mitochondria genomes' are often using non-standard codons.

It is suggested, that the codons frequency usage may vary in different organisms.[22].

The dataset we used, was shared by Bohdan Khomtchouk, from Section of Computational Biomedicine and Biomedical Data Science, University of Chicago. The codon frequency set, was build using CUTG (Codon Usage Tabulated from GenBank, which is available on the site (https://www.kazusa.or.jp/codon/). The data contains frequencies of codon usage by a number of diverse organisms. Each organism was assigned to its respective kingdom, as shown in the table.

The "Kingdom" is an abbrevation code consisting of 3-letter, that corresponds to the names from CUTG database. Data from UCI slightly differs from original data, as author describes that they manually changed class of bacteria into archaea, plasmids, and bacteria proper.

The DNAtype is coded by an integer that represents genomic composistion in the given species. "SpeciesID" is an unique interger number, that differentiates various species, along with the "SpeciesName". Codons' number in the column "Ncodons" was obtained by summing the codons for different species found in the CUTG database. Then, the number of codons was normalized by dividing each codon (like UUU, UUA, etc.) by the codons species sum, as listed in the "Ncodons" column. That is how frequencies of codons were obtained. All codons columns are floats, with 5 decimal digits.

The goal of the original paper "Codon Usage Bias Levels Predict Taxonomic Identity and Genetic Composition", that used the data, was to build machine learning classifier to distinguish between species. Meanwhile we want to determine, if we can restore the species structure, by dividing them on their respective kingdoms basing on codon frequency with different unsupervised learning algorithms.

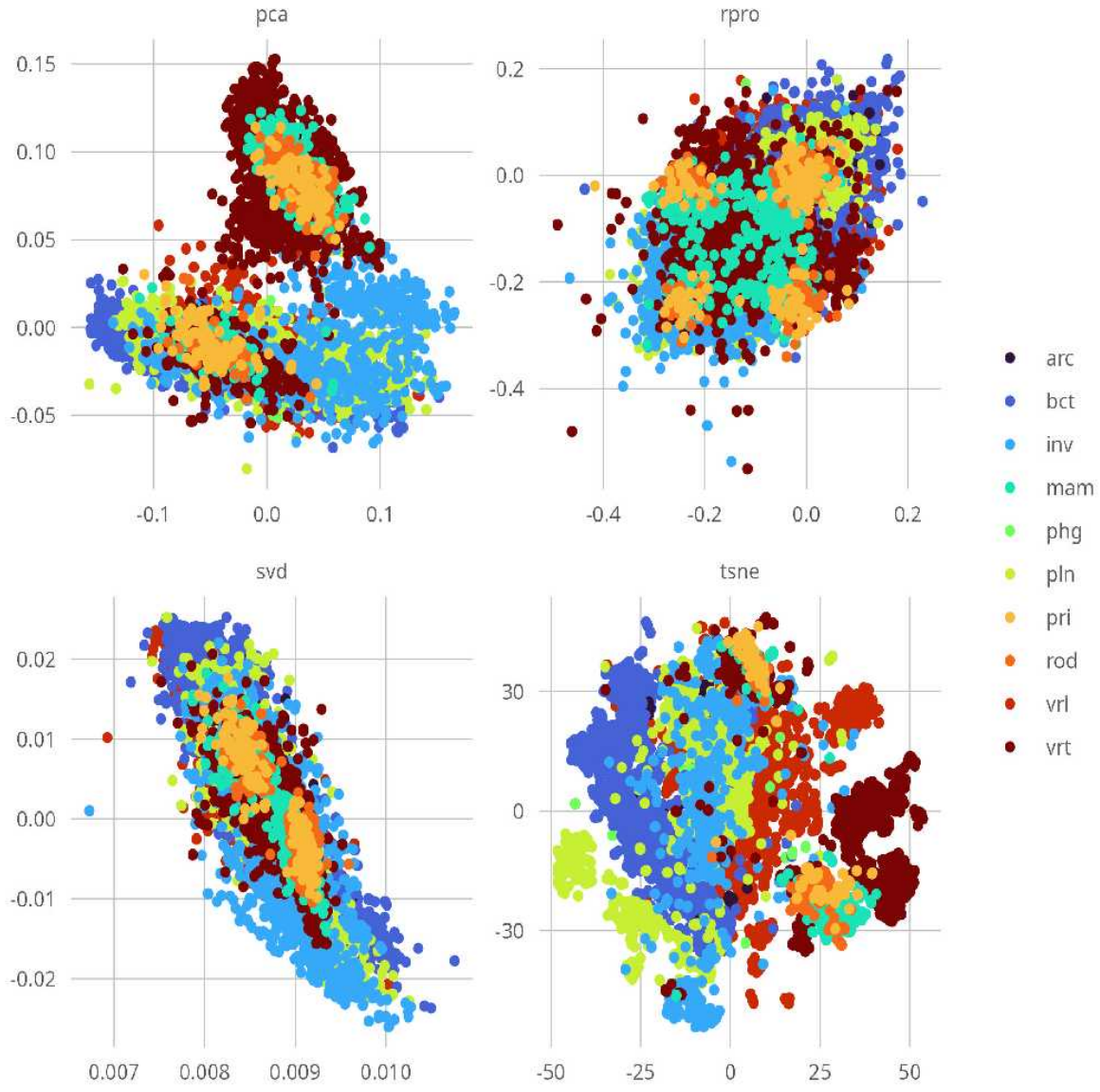

Figure 5.22: t-SNE of standardized Codons by animal Kingdoms

In our study, we have slightly modified the original data. We dropped plasmids as they represented the smallest group, comprising 18 of over 13026 observations. We used the column "Ncodons" to create denormalized data closer to the original data from the CUTG. Our exper-

iment consisted of two dichotomic parts, for which we used original and denormalized data. In its central position, we tried to retrieve the structure of different Kingdom types. Here, we proceed as in the general pipeline. In the second part, we have used the "DNAtype" information to check if there is a visible structure. In this case, we did not create any additional data.

Moreover, we left only four types of DNA, as they accounted for over 98% of data observations. In addition, we repeated both parts but with modified data. Then, we tested all of the algorithms.

Results

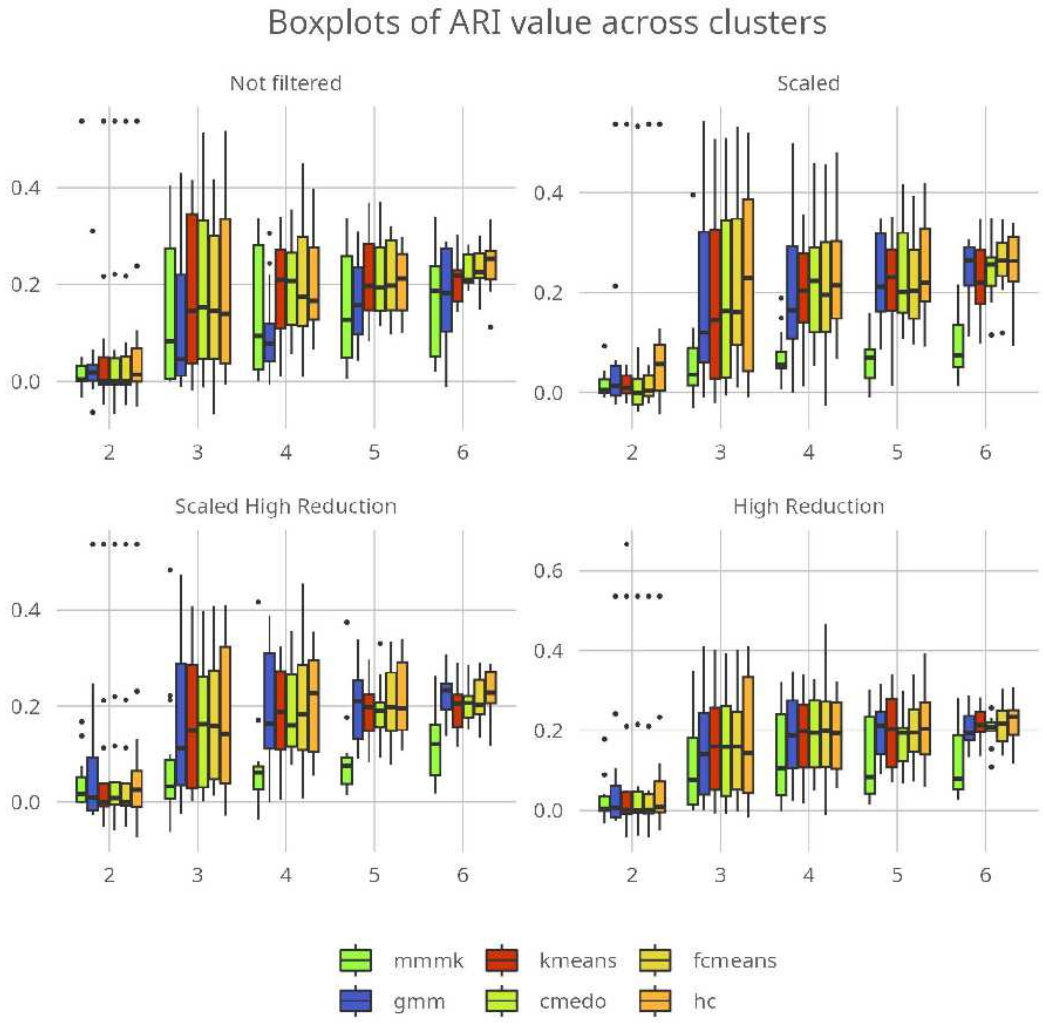

Figure 5.23: ARI index accross clusters in Codons

From the ARI index perscpective, both gmm and mmm algorithm had almost the lowst scores. In this comparison, fuzzy algorithm as well as hc had the highest results. Apart not filterted files, median scores were at similiar level with increased number of clusters. It was not the case for mmm, which had the lowest scores across the algorithms. Mixtures with two components scored very low for all types of filtration and scaling. Then, mixtures with 3 components had the highest spectrum of scores. Apart from two components mixture, with increased number of clusters, spectrum of results was smaller, however median was higher.

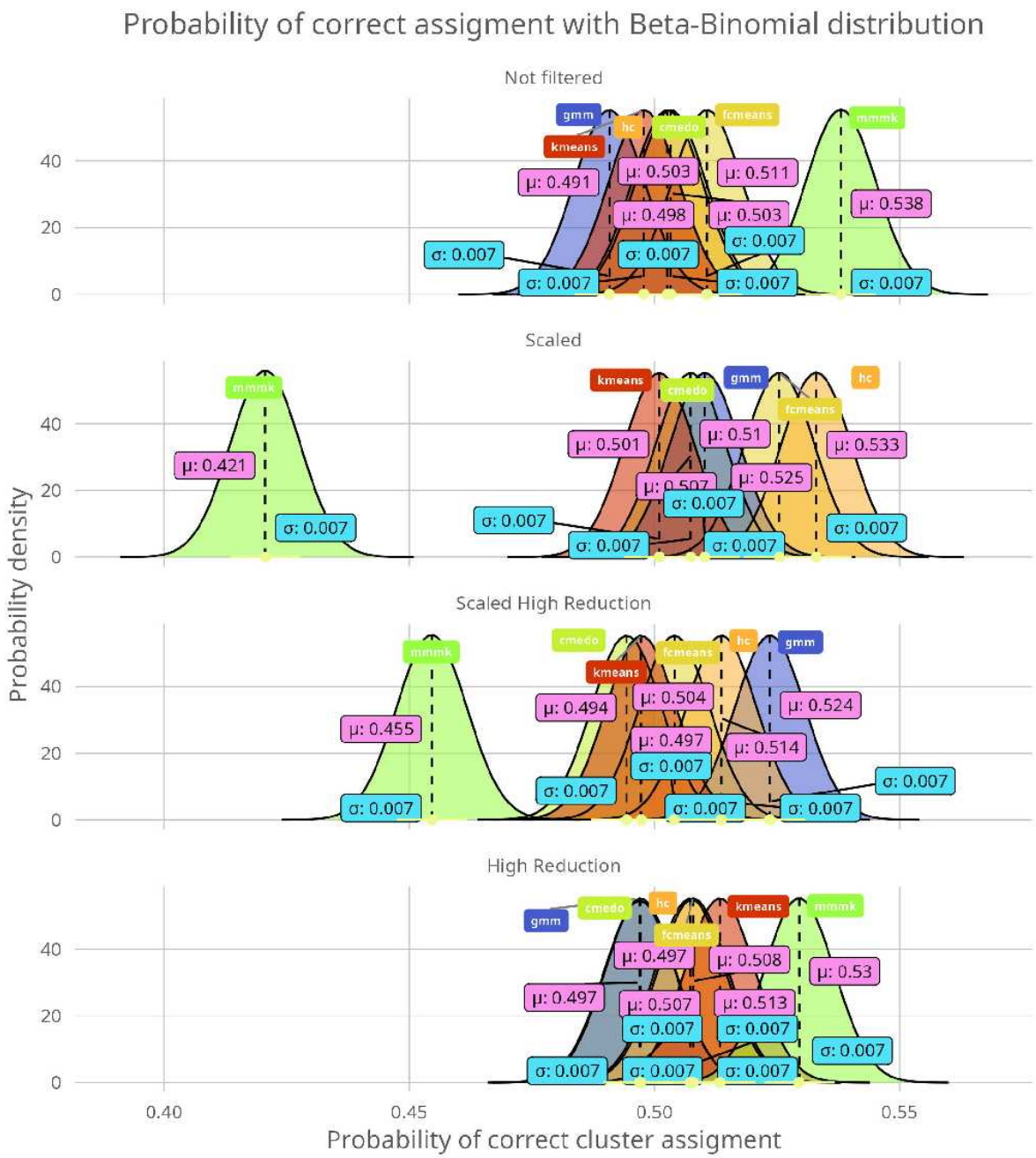

Figure 5.24: Beta-binomial distribution in Codons

In the case Beta-binomial distribution shows, results are quite different. We see, that the most right peak belongs to mmmk, but only for original data. In addition, it is only slightly overlapped with the scores from other algorithms. In the case of scaling, mmmk was again the last. GMM had the highest mean in the case of scaled and reduced data. However, it is much more overlaped with other distribution. Finaly, in the case when the variables were only

reduced, mmmk had again the highest mean, however it is more overlaped than in the not filtered data.

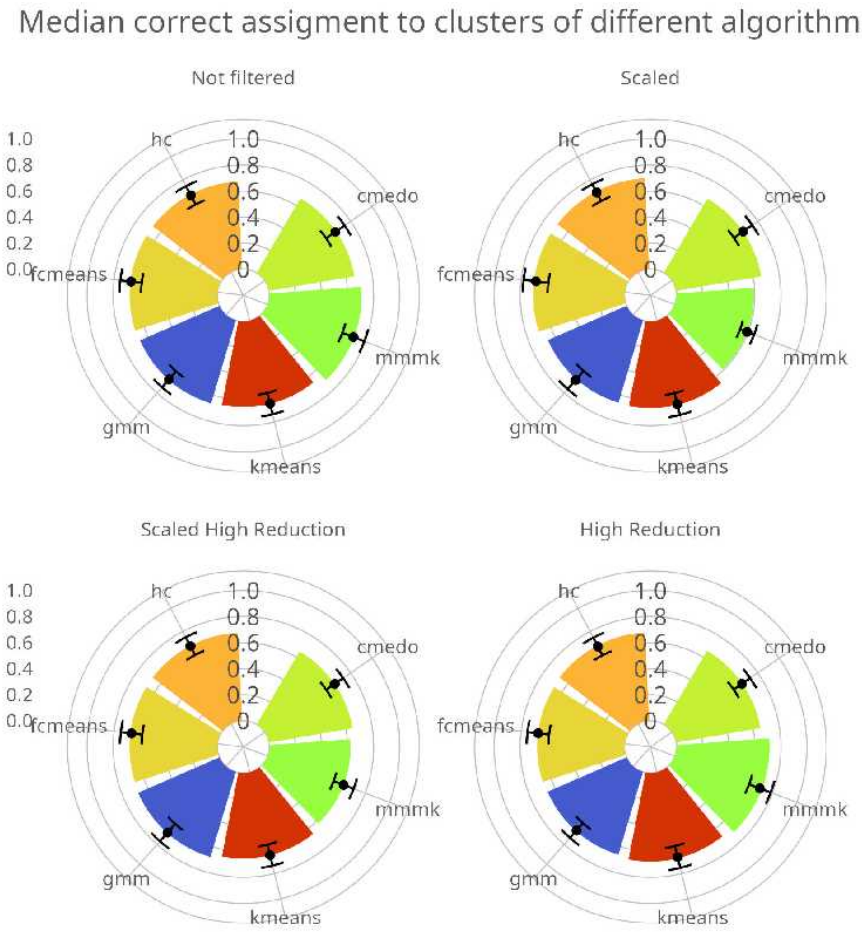

Figure 5.25: Accuracy index in Codons

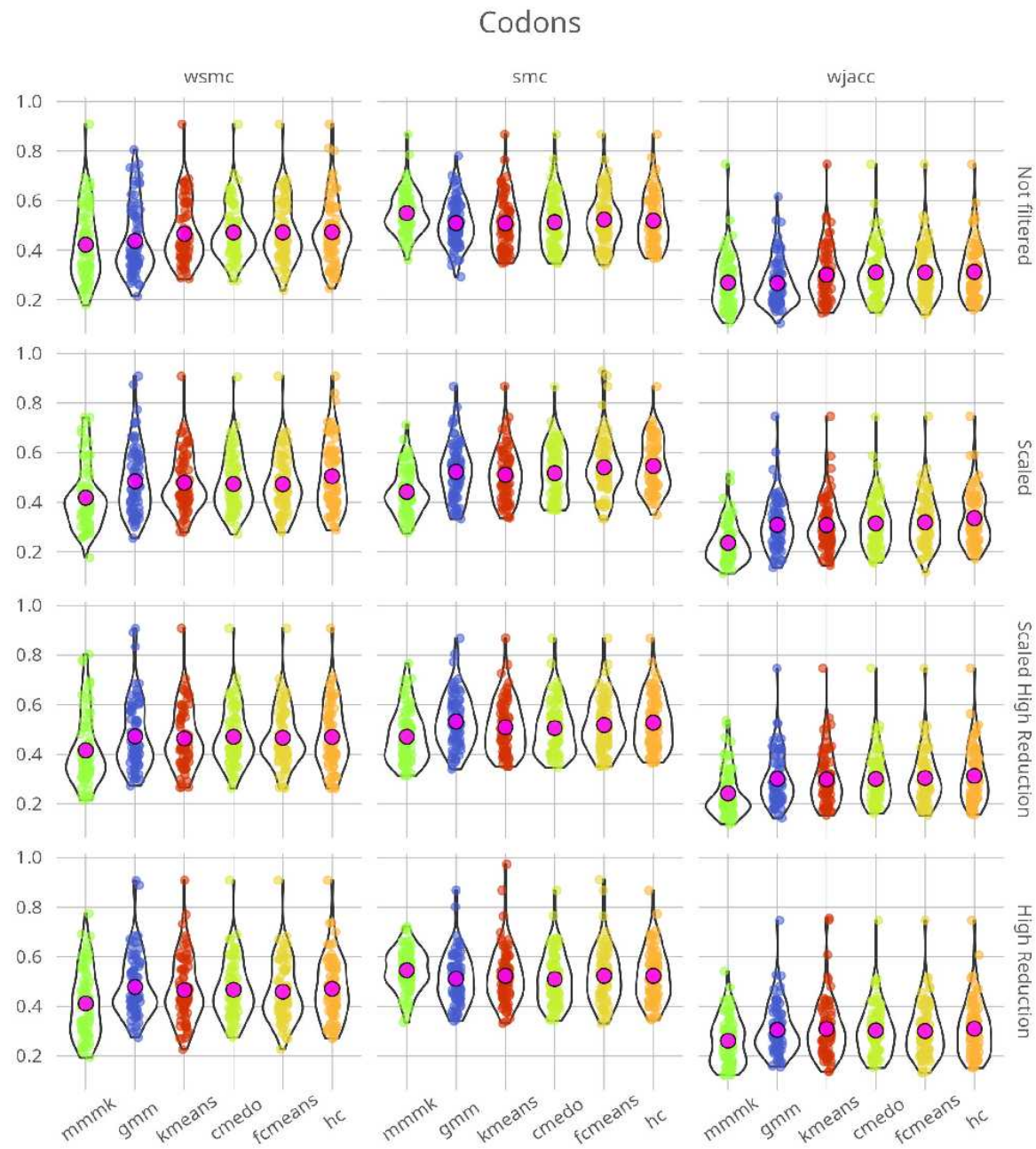

Figure 5.26: Various indexes accross clusters in Codons

The corealtion between metrics varies greatly. The ARI index and accuracy are only partialy correlated with other metrics.

Codon's pattern might be more complex to catch, as we already saw in the tSNE plot. In the first frame of the ARI metric, index medians range from 0 to nearly 0.16. GaussEM and HC are slightly above k-means and fuzzy c-means. Interestingly, most of the MultinomialEM results are at zero. We observe only one point score above nil and one close to 0.4. In the case of other algorithms, they received only a few scores above 0.4, which is relatively low. The highest score in these datasets was 0.8, which belongs to GaussEM.

Figure 5.27: Metrics correlation in Codons

The general tendency of the median scores seems to flatten for all algorithms. It was not visible in the case of somatic mutation count and cancer gene expressions. In contrast to mentioned analysis, scores were lower in mixtures with four groups than in three for all algorithms but HC. Notably, the HC algorithm scored highest in the three mixture components. Coming to the five groups' mixtures, we observe a slight increase in the median ARI scores for all algorithms. The results are better for k-means and Gauss EM, but with no difference for fuzzy c-means and lower for HC. Finally, median ARI scores are similar in the six component groups, highest in k-means. However, GaussEM scored. Similarly to our previous analysis, scores tend to be more concentrated around the median with increased components number.

SMC Interestingly, multinomialEM scored the highest across algorithms when class balance does not matter and alignments are optimised. Its average performance was 0.565, which was w 0.025 higher than the second-best algorithm in this comparison - GaussEM. HC

5.2.4 Sport activities

In recent years, the terms "smart" and especially "smartwatch" are gaining more popularity. A simple indicator of that is a trend presented by Google. The charts below show that during the last 18 years, the terms "smart" and "smartwatch" have increased. In the picture, a value of 100 means that the word was the most popular, while a 50 indicates that the term was twice as small. The value of zero means that there is insufficient data.

Smartwatches offer various functions, like phone calls, messaging, or playing music. However, they also allow measuring heart rate, calorie burn or daily steps count. Many of them can also gather various statistics about sports activities, including distance and speed, to name

a few. More sophisticated smartwatches can automatically determine the person's action and record appropriate metrics accordingly.

In this comparison, we have used the data that Billur Barshan shared on the UCI platform. He is a Yale professor from the Department of Electrical and Electronics Engineering, specialising in wearables, machine/deep learning and robotics. The data that he gathered is about various sports activities. In experiment took part eight people. Four were males, and four were females, all between the ages of 20 and 30. They were asked to perform 19 activities, like jumping, cycling, running or walking. What is essential, they were not instructed on how to do any of those exercises beforehand. Because of that, it might introduce some individual variation among all subjects. The data was recorded using various sensors. The single unit consisted of 9 accelerometers, gyroscopes, and magnetometers in x, y, and z coordinates. In total, five units, one placed on the torso, both arms and legs. Sensors were calibrated to gather the data with a 25 Hz sampling frequency. The time for each different activity was 5 minutes. In total, we had 45 attributes and 1140000 observations.

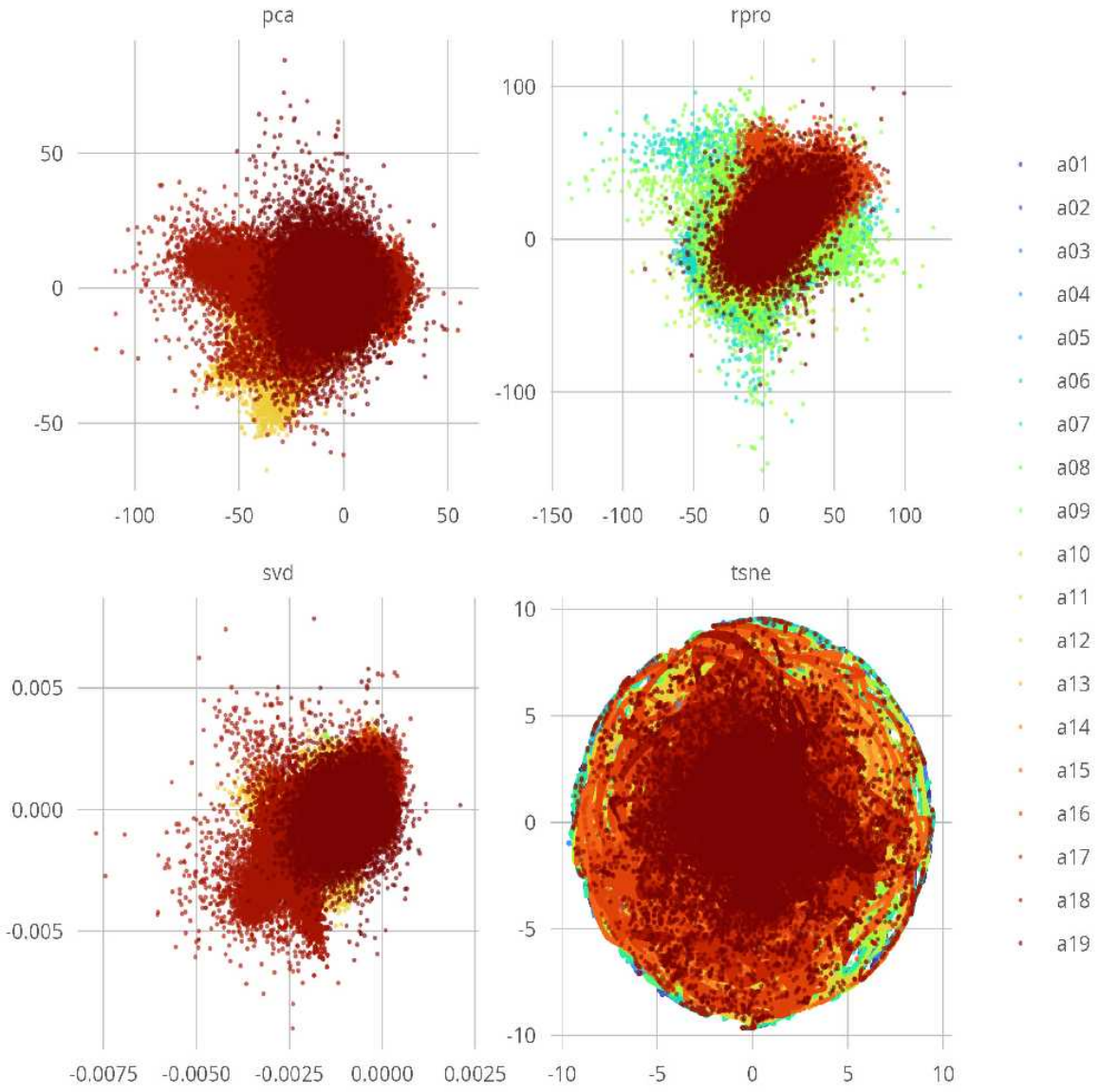

Figure 5.28: Various dimensionality reduction techniques in Sport Activities

Results

In the case of daily sports activities, we were not able to include Hierachical Clustering and k-medoids methods. Since even two mixture components consited of **over 100 thousands** observations, minimal RAM memory allocation was over **50 GB**. With up to 6 components, it increased to even **1000 GB**. Algorithms that overcomes limitation of RAM exists. However time of completion was significantly higher than any other presented algorithms.

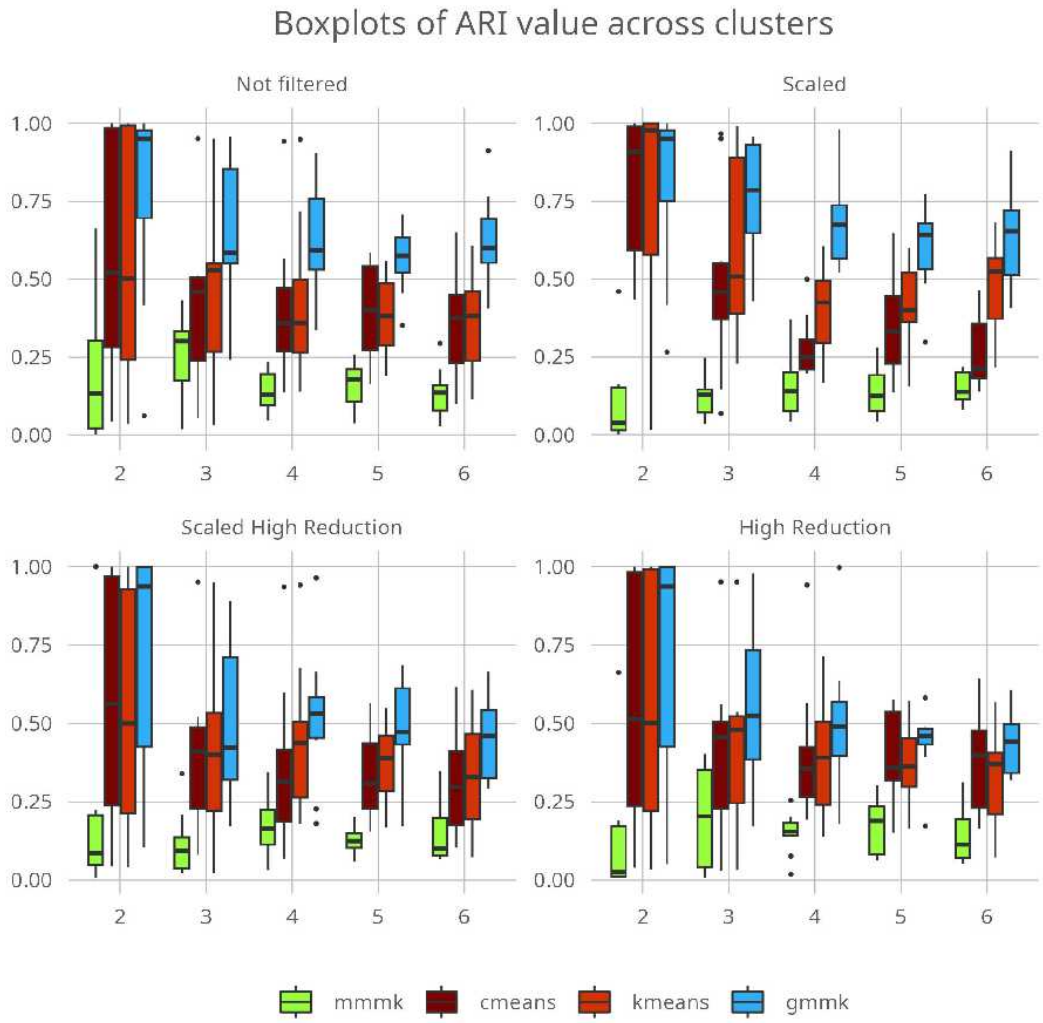

Figure 5.29: ARI index accross clusters in Sport Activities

In the case of **the ARI** index we can see that the **gmmk** has the best scores of all the algorithms. What is notable, is the fact that even with increased number of clusters, it maintained high ratio of correct assignments, comparing to the others. Scaling seemed to benefit all of the algorithms. We can see, that the results are more focused around the median, which is in addition higher. In case of variable reduction, algorithms performed worse, as if valuable information was lost in the process. Again, scalling had some positive impact, but results remained relatively low.

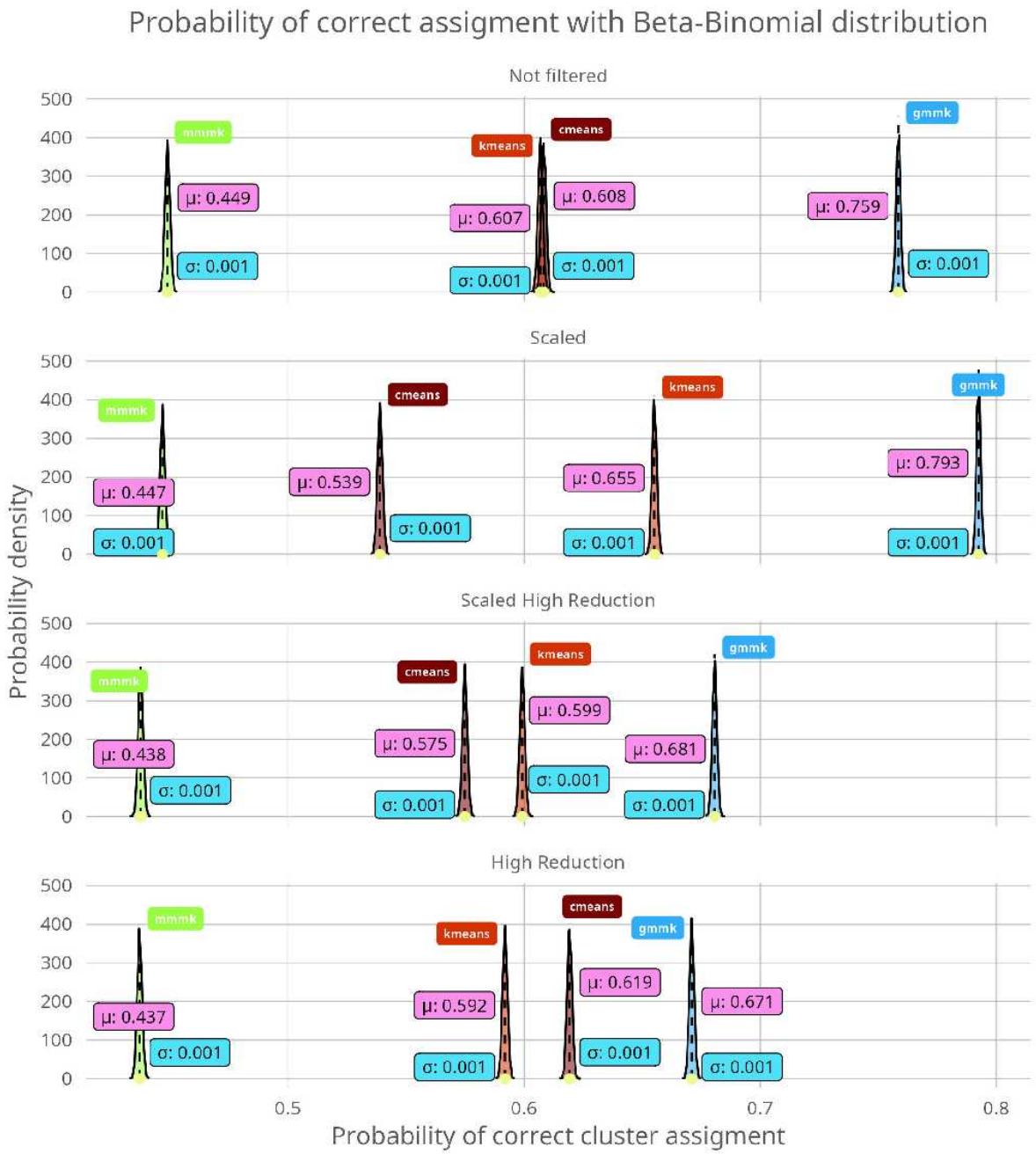

Figure 5.30: Various indexes accross clusters in Sport Activities

The highest and most rightside peak belongs to **gmmk**. When the full data was scaled, the result was even higher, taking more advantage of kmeans initialization. In addition, its probability density was also higher than in others. It give us slighlty more credability of the result. On the opposite, **mmm** algorithm had the lowest scores.

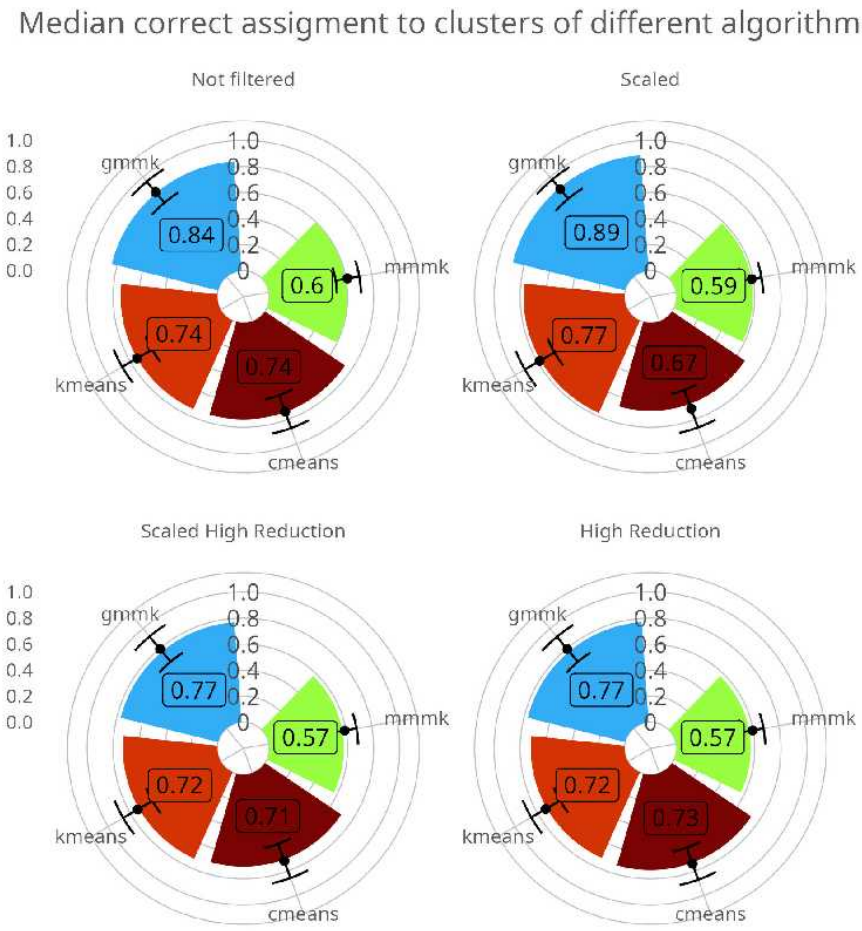

Figure 5.31: Various indexes accross clusters in Sport Activities

Median accuracy also shows, that gmmk performed the best across all of the compared algorithms.

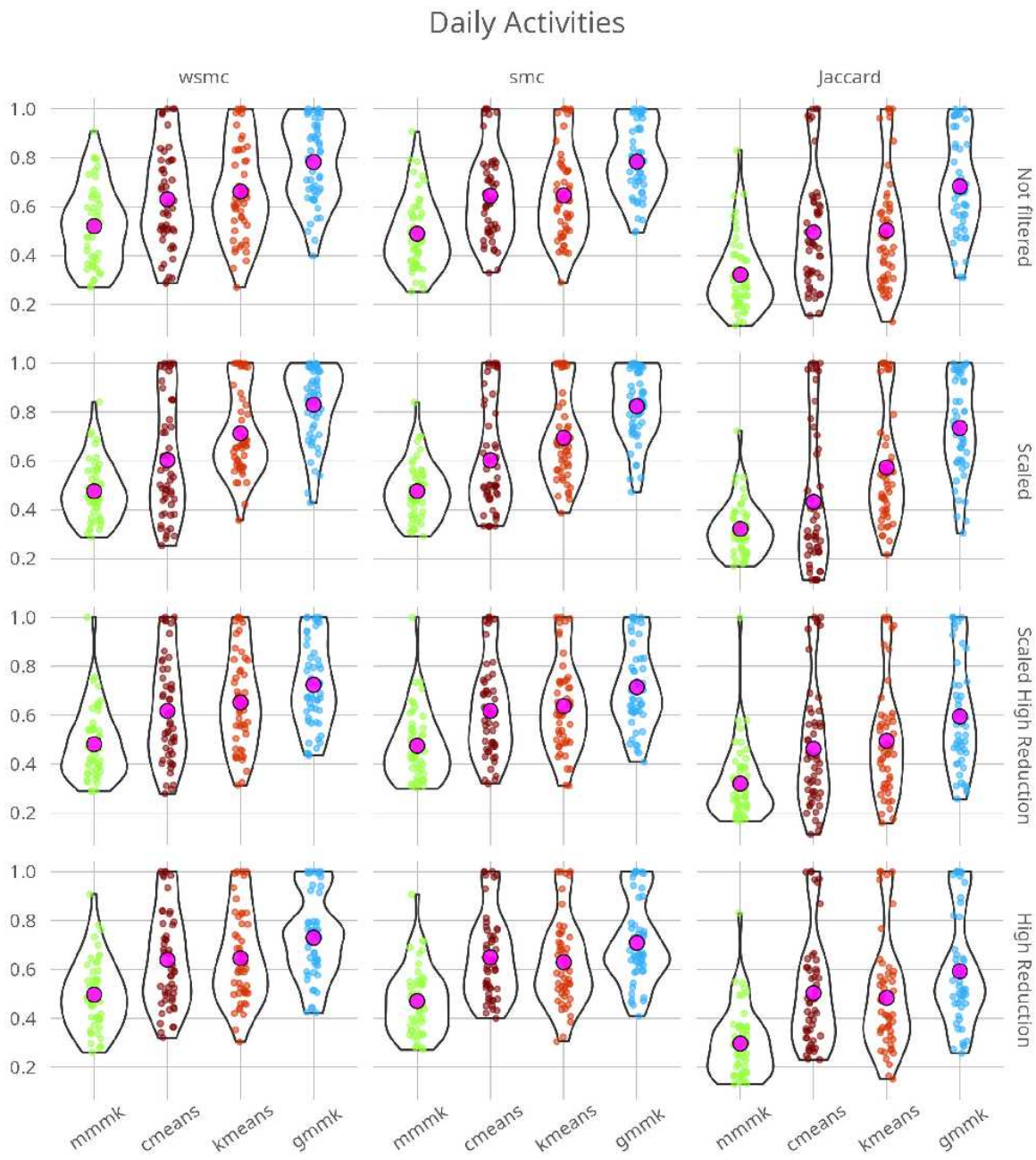

Figure 5.32: Various indexes accross clusters in Sport Activities

Looking at the t-SNE dimensionality reduction map, we see that although sports activities are incredibly mixed, their distribution has some visible pattern. The immense cloud of points in the middle is probably the initial position that has been recorded. A few activities like running, walking, and climbing stairs share the same initial position - standing still. However, our study does not filter any potential noise and uses all the available data.

Sports activities, due to a large number of rows, were the first time that a different HC method was tested. The built-in function for HC calculation took too much RAM space to work correctly. For example, creating a table of floats 100 000 columns by 100 000 rows was supposed to occupy almost 80 GB of RAM space [!!!check this!!!]

From the perspective of the ARI index, two-component mixtures have the broadest spectrum of outcomes across all datasets. It applies to all of the algorithms, excluding multinomialEM. The multinomial model-based algorithm performed poorly but better than in the Codons dataset.

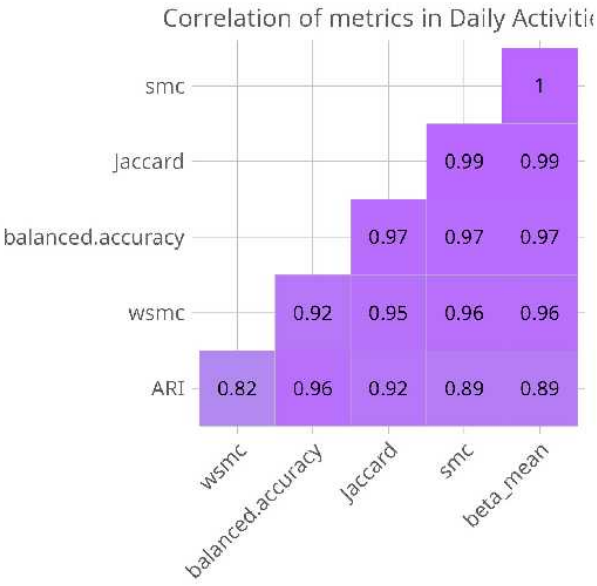

Figure 5.33: Metrics correlation in Sport Activities

In the three mixture components, the median score value of the distance-based algorithms increased. In addition, it became denser around the median, contrary to multinomialEM, in which the range of results became scattered. GaussEM median was slightly lower, but it maintains a similar level across all mixtures. Its peek median, however, was observed in the two-component mixtures. In the case of four component mixtures, we observe another increase in the median value score along with higher condensation of scores around it.

Finally, where the six groups were present, we noticed that the median score of the GaussEM was the only one above 0.5 value, but only slightly. The second-best algorithm was fuzzy c-means, with the median value slightly below 0.5. K-mean had the third highest median value, but its maximum scores were on the same line as Gauss EM. Then, scores of multinomialEM are between 0 and 0.25 without crossing these values. These scores make it the lowest-performing algorithm according to the ARI index.

Looking at different indexes, results from multinomialEM are still the lowest. On the other hand, indices based on the Hungarian algorithm show differences in the highest median scores and the distribution of values. The fuzzy c-mean is minimally better than GaussEM in all three indices, but the difference is not statistically significant. In the case of SMC and WSMC, we can see a sizeable band for GaussEM near the values of 0.75. For the Multinomial Mixture EM, there are two visible bands near the values of 0.50. The results of the fuzzy algorithm are also denser around the value of 0.75, but they are smoothly distributed. K-means results are also softly distributed but focused near a value of 0.5. However, its range and mean values are better than those of Multinomial Mixture EM. The averaged Jaccard results are in the middle

of ARI and the rest of the indices. In general, values are smoothly spread compared to other indices.

The majority of the results are statistically significant. However, we observe that some scores of multiEM and k-means exceed the pvalue of 0.05.

Binomial test

Most of the results were statistically significant, with a few values above 0.05. Interestingly, Multinomial Mixture EM scored the lowest across all algorithms. It also does not contain values above different significance levels.

5.2.5 The Free Music Archive

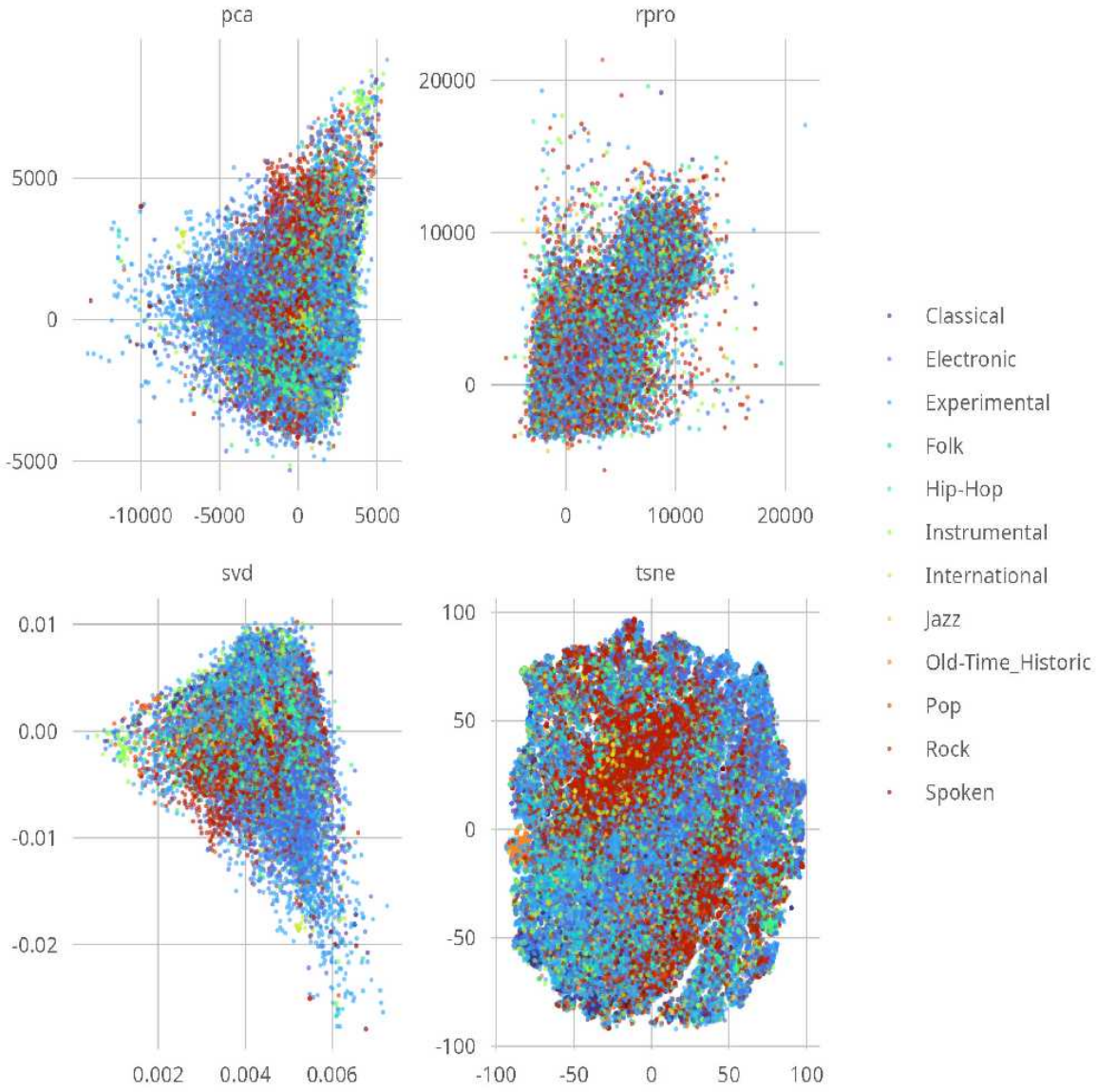

Figure 5.34: Various dimensionality reduction techniques in The Free Music Archive

The Free Music Archive is a large music analysis project. The data was made publicly available around the end of 2016. Its vast library consists of 106 577 songs covered by 16,341 artists. All of that is inside 14,854 albums. The FMA delivers pre-computed audio features jointly with user-level and track-level metadata.

Moreover, it is also possible to use full-length, high-quality audio—a complete archive weight of around 879 GiB. However, it comes with different packages, limited to selected genres.

Over 500 features describe each song. They were pre-computed with a package librosa, a rich python library for audio and music analysis. The package allows for low-level feature extraction, such as chromagrams, Mel spectrogram, Mel Frequency Cepstral Coefficient (MFCC), and other spectral and rhythmic qualities. The library site https://librosa.org/doc/latest/index.html offers more information like tutorials and documentation to create your analysis. All tracks in the archive are organised in a hierarchical taxonomy of 161 genres like rock, jazz, pop, or classical music.

For our analysis, we decided to reduce the data significantly for a few reasons. The first was that roughly half of the songs were assigned to more than one genre. Because of that, mentioned observations did not have the leading genre. We decided to exclude those songs from our analysis as it greatly simplifies the estimation of actual values during clustering. On top of that, we reduced our research to the twelve most frequent genres mentioned in the table. Our cutoff value was arbitrally. The last genre in our data, "Spoken", had 423 observations; the rest we did not include were below 194.

Our final dataset consisted of 49598 observations with 518 features. From this point, we proceeded as in our general pipeline.

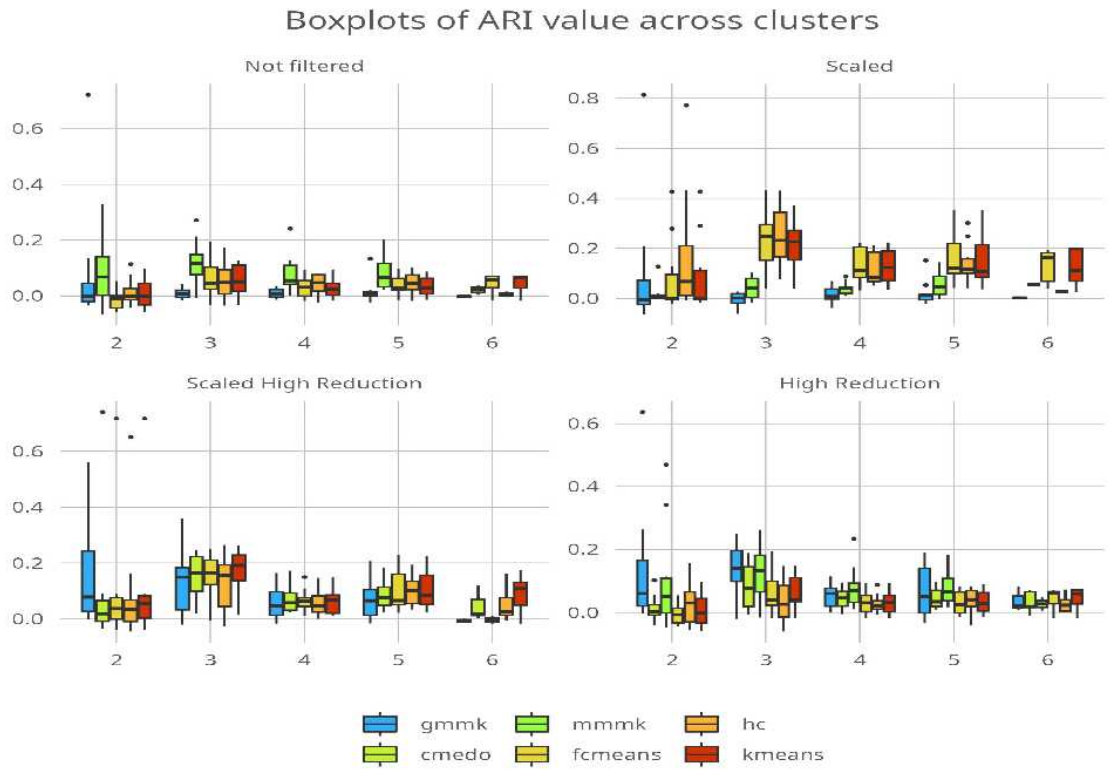

Figure 5.35: ARI index accross clusters in The Free Music Archive

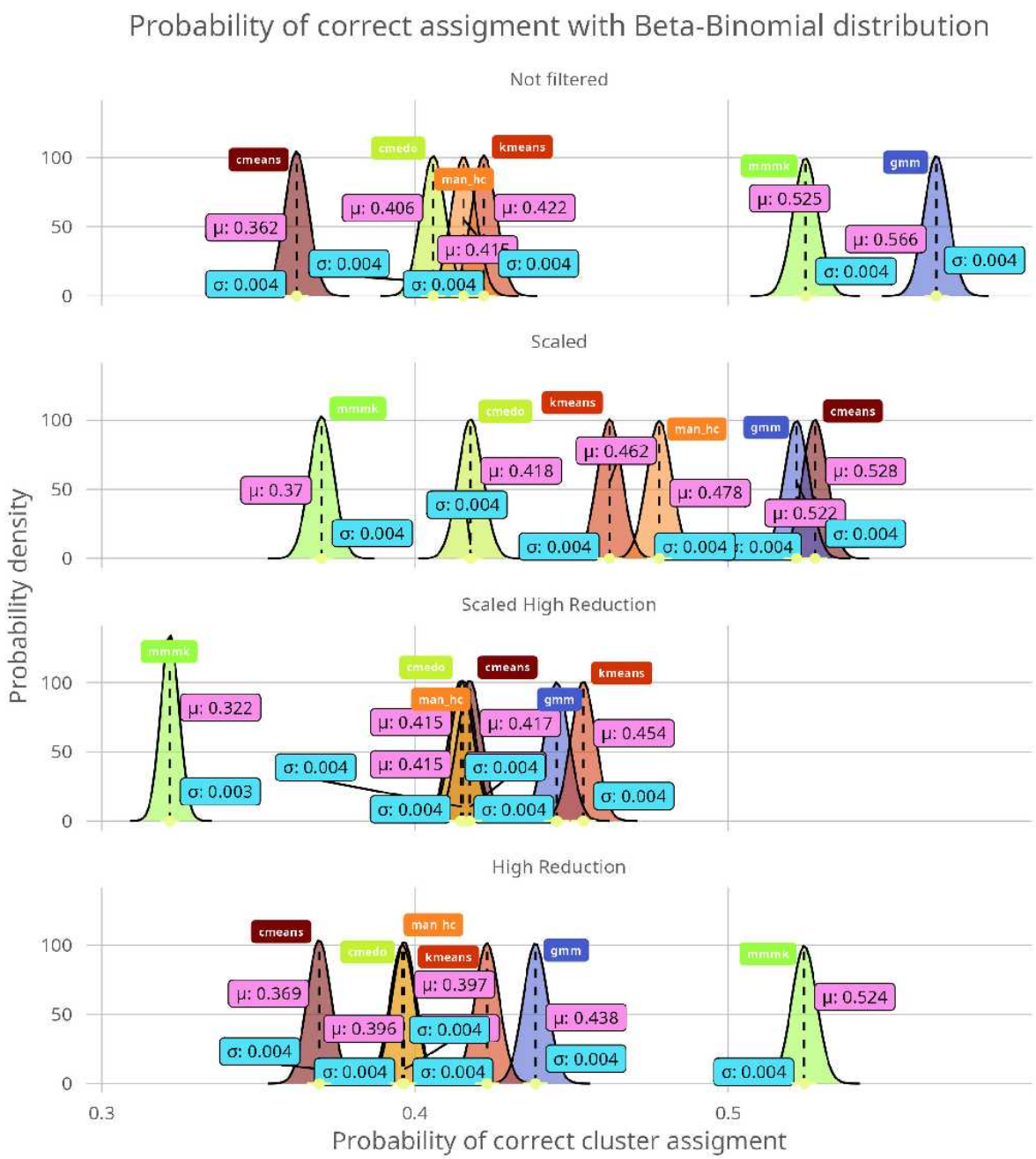

Figure 5.36: Various indexes accross clusters in The Free Music Archive

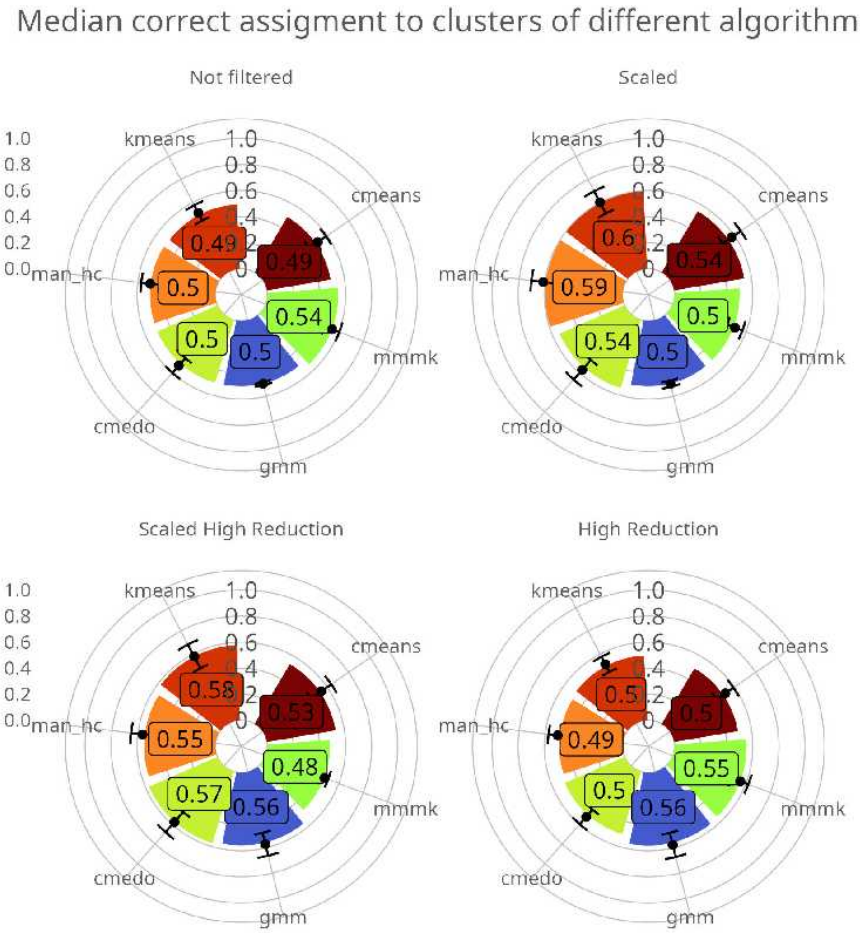

Figure 5.37: Various indexes accross clusters in The Free Music Archive

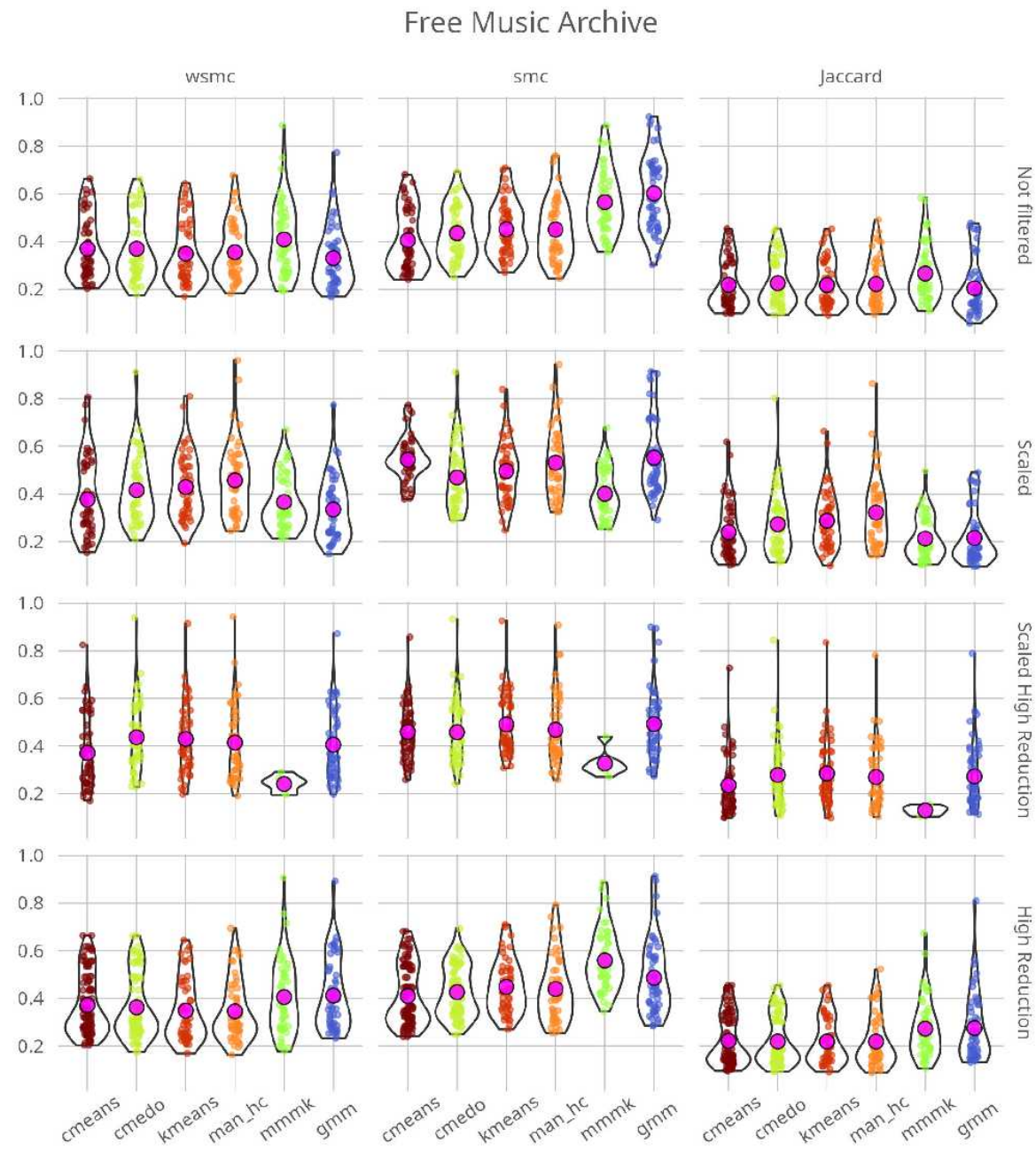

Figure 5.38: Various indexes accross clusters in The Free Music Archive

The presented metrics are slightly correlated. The lowest correlation is observed between WSMC and Accuracy. There is medium correlation between Adjusted Rand Index and the others.

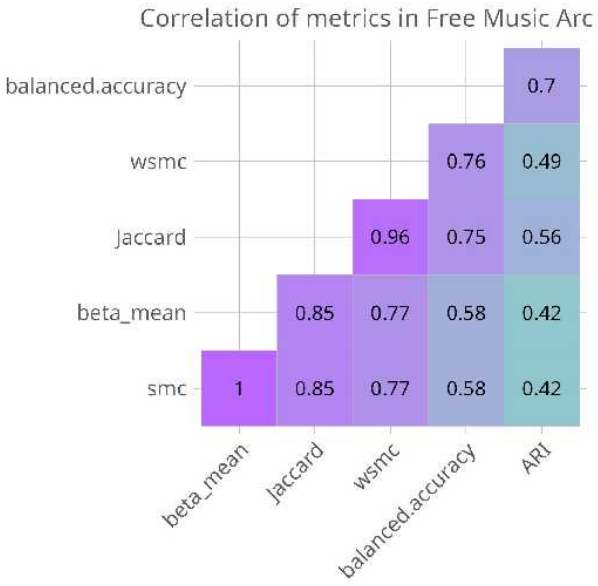

Figure 5.39: Corr plot of metrics in The Free Music Archive

5.2.6 NASA Kepplers

On 6 March 2009, The Delta II rocket took NASA's Kepler Space Telescope and carried it into space. The telescope was focused on the area with about 150,000 stars, like the sun within our solar system. Its ultimate purpose was to identify other habitable planets, excluding our own. Kepler's discoveries contain planets that orbit in so called habitable areas. The habitable area means, that it orbits within sufficient enough distance from a star. Sufficient is when surface temperature may be fit for life-giving liquid water.

The first discovery important discovery was Kepler-22b. It is an example of a habitable zone planet found during the mission. However, because it is almost 2.4 times the size of Earth, it is considered too large to be solid and life-supporting. However, scientists are convinced that different habitable zone planets found by the Kepler mission might be rocky, such as Kepler-62f, which is 40% larger than Earth. A twin to Earth that has the same temperature and size as Earth was not yet discovered. Still, the analysis is far from over as scientists continue to search the Kepler data for the tiny signature of such a planet. Other Kepler discoveries include hundreds of star systems hosting multiple planets and have established a new class of planetary systems where planets orbit more than one sun.

The mission ended its science observations after a faulty reaction wheel affected the telescope's ability to point precisely. However, the telescope reamined in service due to its next-generation mission proposal, called K2.

By analysing the information gathered by Kepler, scientists' community have recognised over 3,600 candidates considered to be planets. They confirmed that 961 are indeed planets, many as small as Earth. Findings using Kepler Space Telescope currently account for over half of all the known exoplanets. We used the dataset that is a cumulative Activity Table of Kepler Objects of Interest (KOI). The KOI table contains information about the single KOI activity tables. It represents the actual results of different findings of the Kepler light curves. The goal of the cumulative table is to gather suitable qualities and stellar and

planetary data for all KOIs, in one place. All of the data presented originates in other KOI activity tables. The last status update was on 27 September 2018 and is considered complete. https://www.kaggle.com/datasets/keplersmachines/kepler-labelled-time-series-data

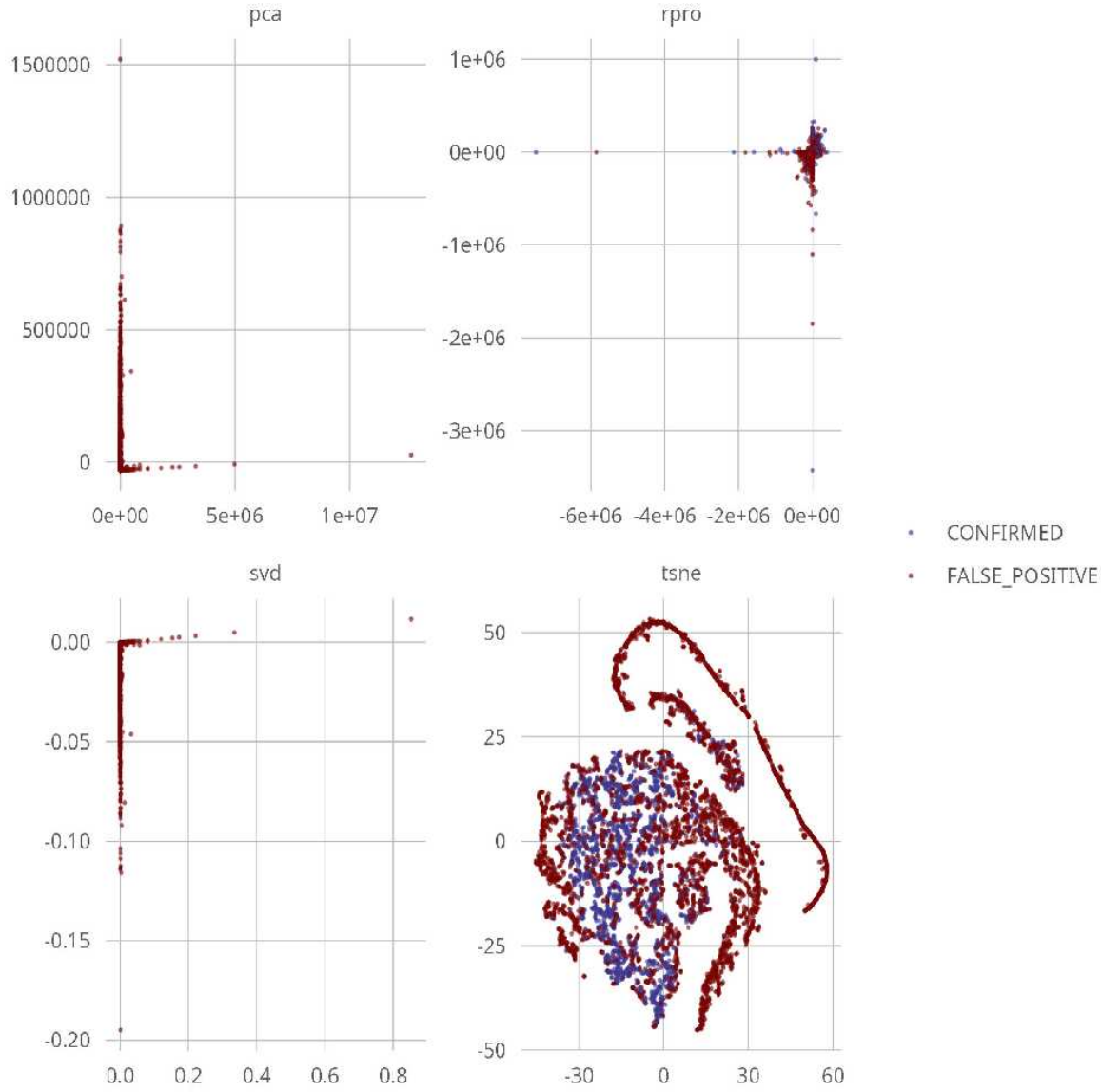

Figure 5.40: Various dimensionality reduction techniques in Kepplers

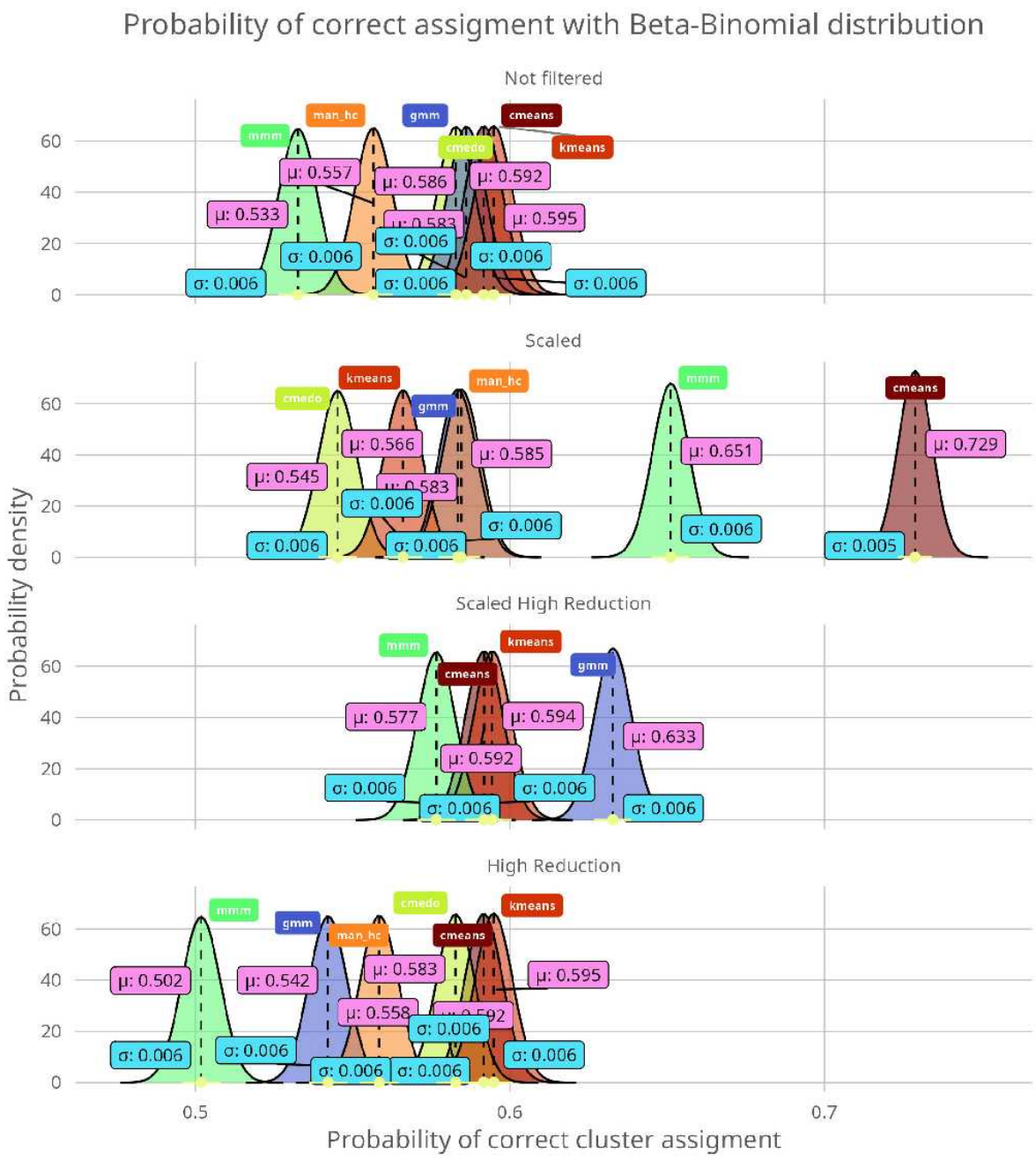

Figure 5.41: Various indexes accross clusters in Kepplers

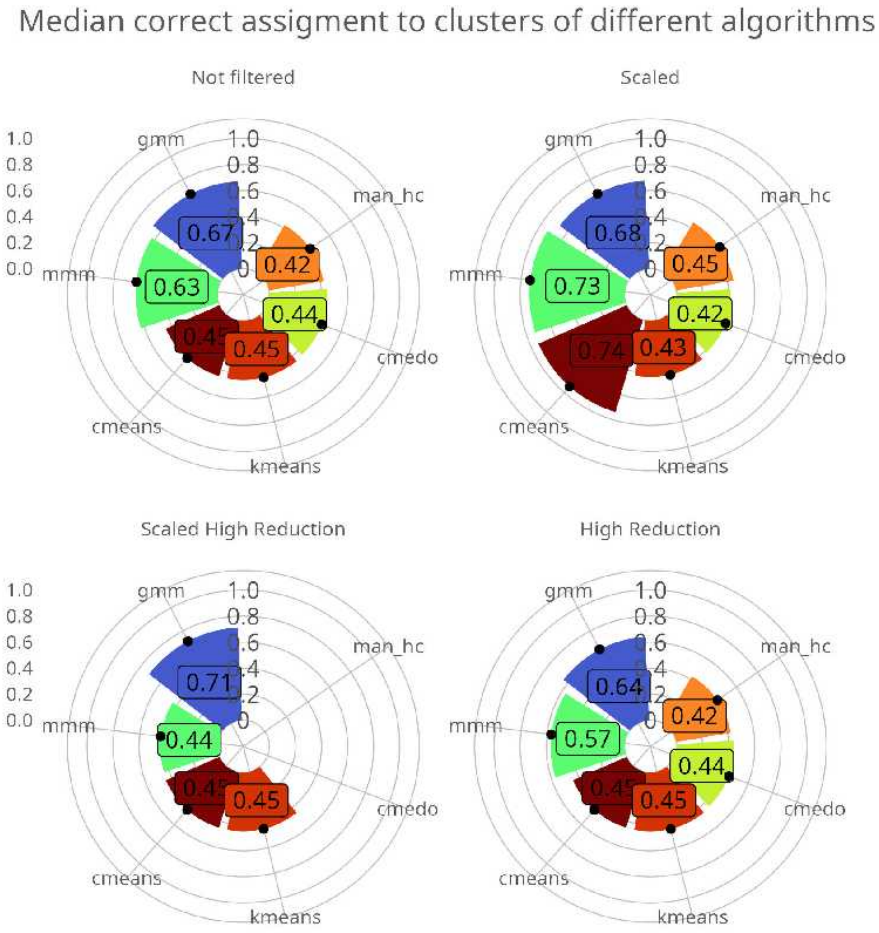

Figure 5.42: Various indexes accross clusters in Kepplers

5.2.7 Arrhythmia

One of the most specific heart rhythm disorders is called atrial fibrillation. Moreover, patients with such heart conditions have five times more increased risk of stroke. At the same time, atrial fibrillation causes almost 20% to 30% of strokes. In addition, strokes caused by atrial fibrillation are much more severe and fatal. They led to death much more often than strokes due to other causes. According to a study (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5467327/), in 2016, almost 7.6 million people in the European Union had atrial fibrillation. Studies estimate that this number will increase by 89% to 14.4 million by 2060. The current prevalence will rise by 22%, from 7.8% to 9.5%. Last but not least, yearly treatment consumes from 0.28% to 2.6% of European funds spent on healthcare.

The data we will explore in our thesis comes from the study whose original purpose was to differentiate between the presence and absence of cardiac arrhythmia. After that, observations were organized into one of the sixteen groups. The first class, 01, refers to "normal" ECG

classes. Then, the number from 02 to 15 refers to various arrhythmia categories. Finally, category 16 refers to the rest of the unclassified ones. For the time being, a computer program exists that classifies the data. However, there are differences between the cardiologist and the grouping done by the program. In our study, we will use a few approaches to this issue. The first one is to proceed as in our main pipeline. We will create many datasets and compare how well they cluster separately from two up to six clusters. Then, we will take all the groups and check how accurately unsupervised algorithms can differentiate between classes. Finally, we will take all the groups but look for only two clusters. In the last part, we are creating two class problems to check whether unsupervised methods can find a clear pattern in the presence and absence of arrhythmia.

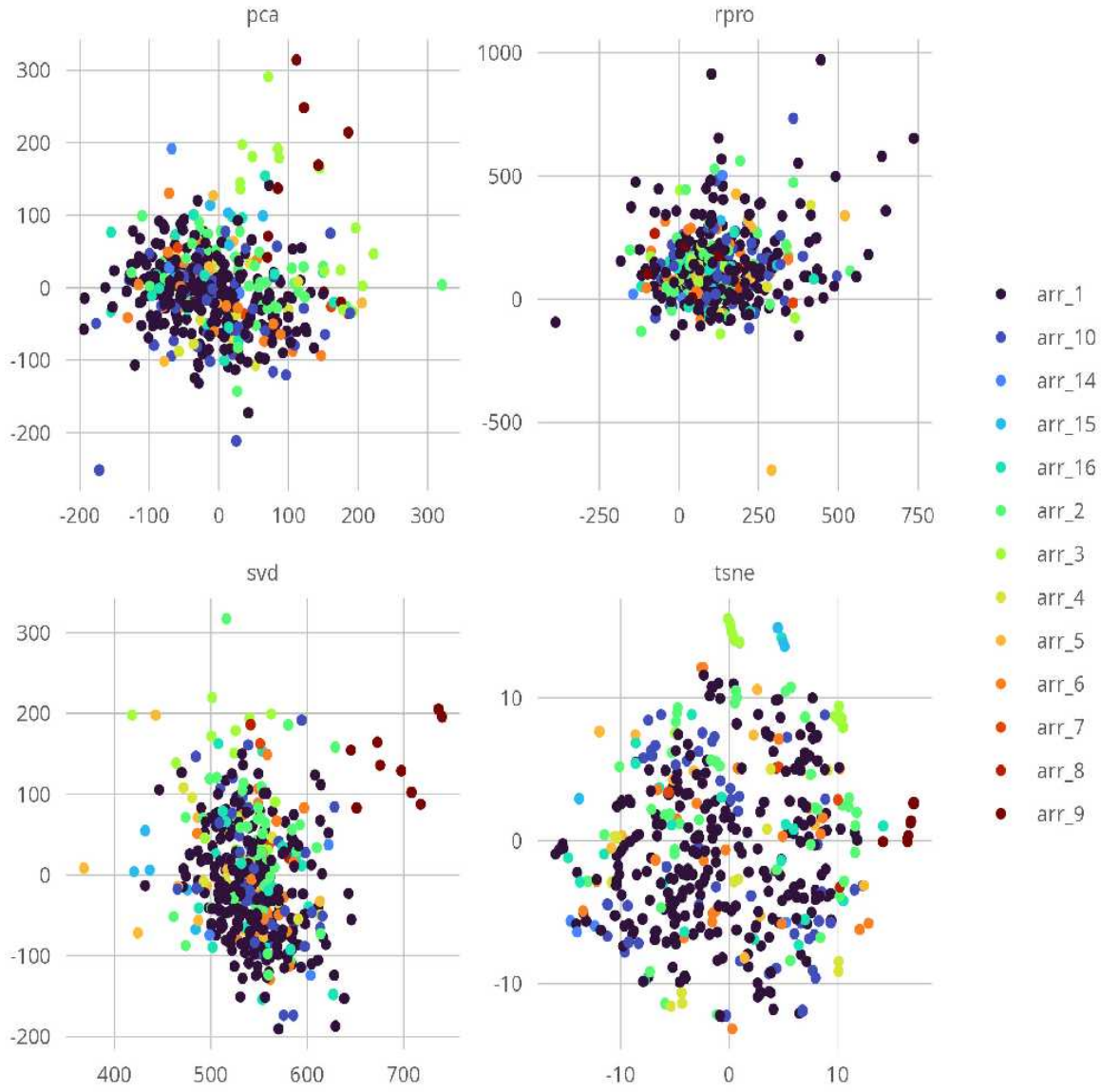

Figure 5.43: Various dimensionality reduction techniques in Arrhythmia

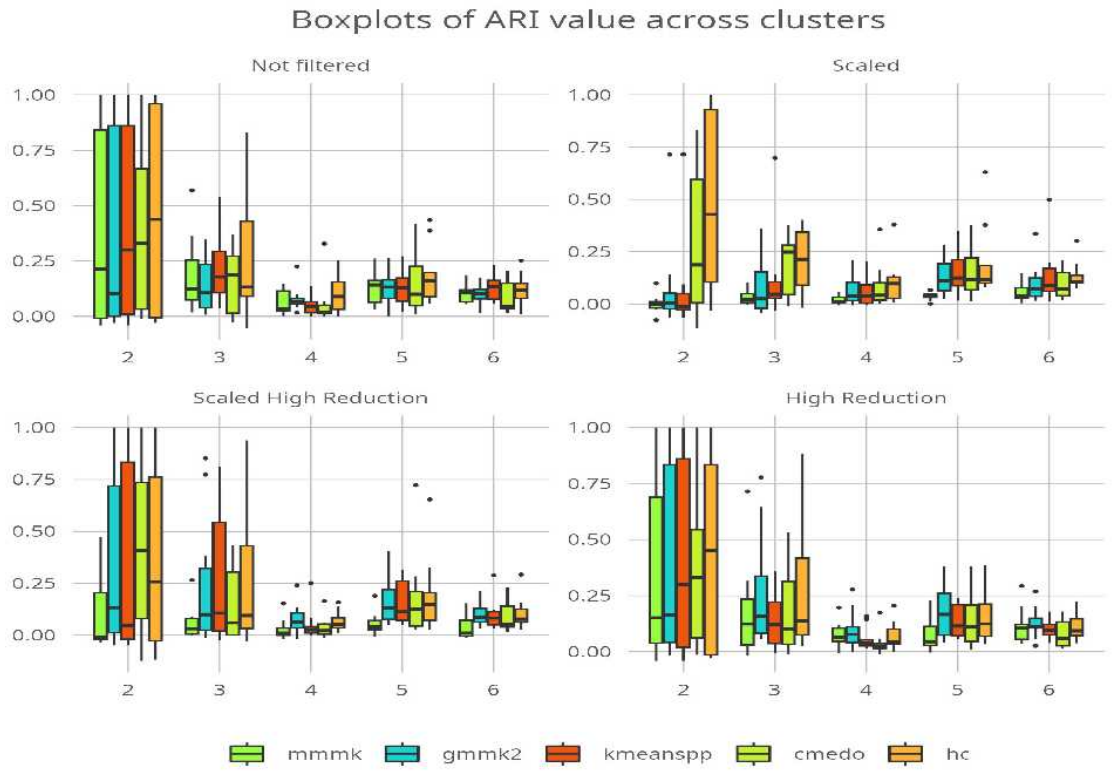

Figure 5.44: ARI index accross clusters in Arrhythmia

The ARI index presented GMMK as having the second highest median score in comparison with other algorithms.

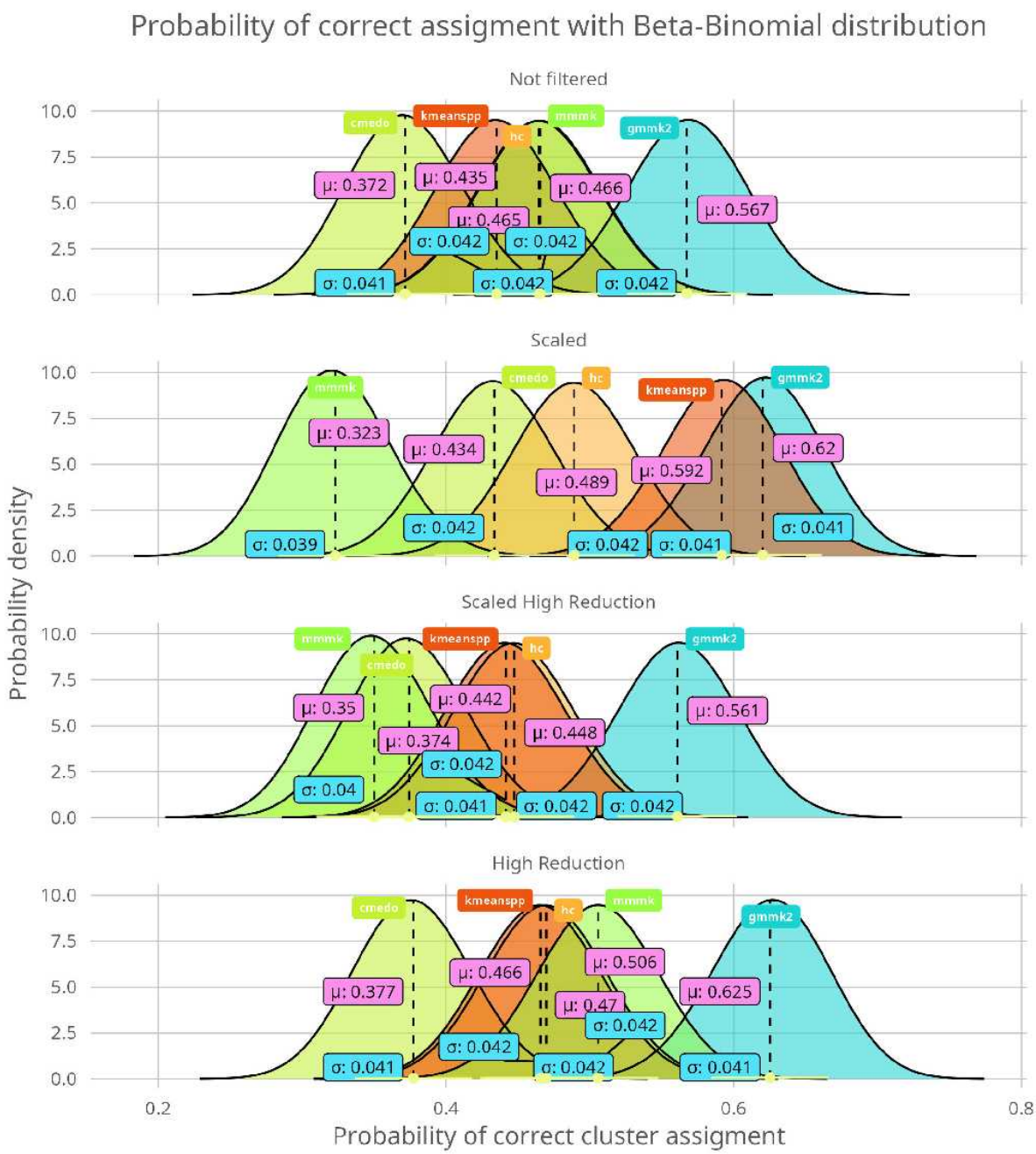

Figure 5.45: Various indexes accross clusters in Arrhythmia

In the case of beta distribution, GMMK scored the best results across other algorithms in the 3 filtering types. However, in case of scalling, cmeans perfomed slighly better. However, overlap between the two is huge.

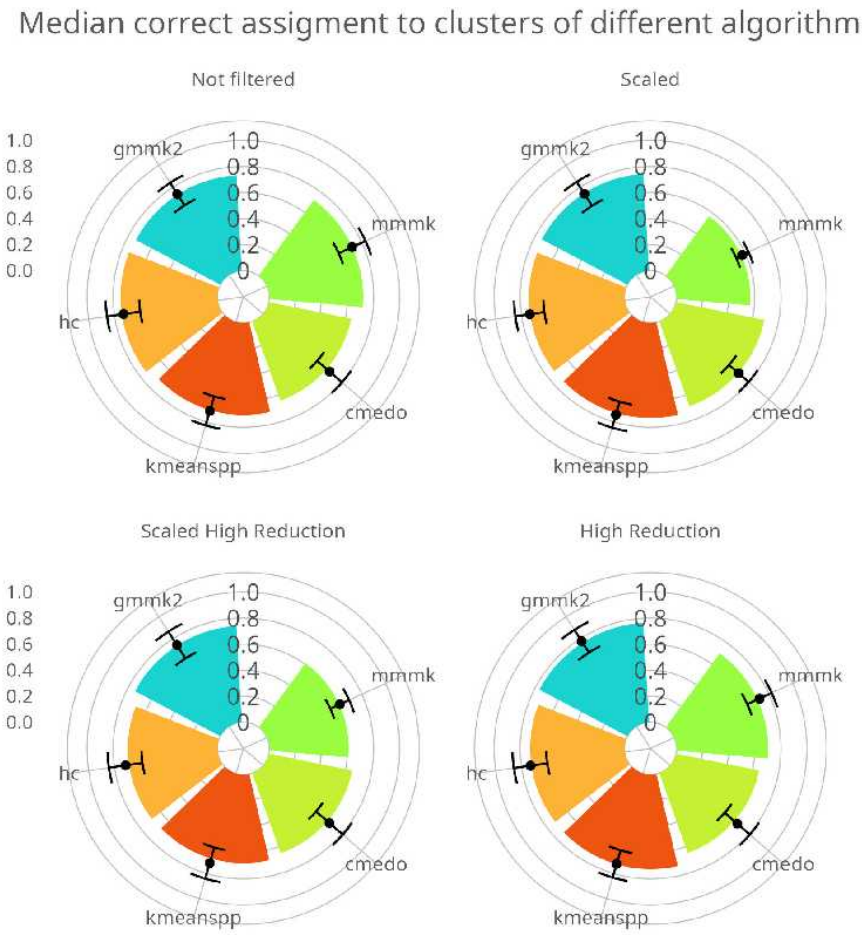

Figure 5.46: Various indexes accross clusters in Arrythmia

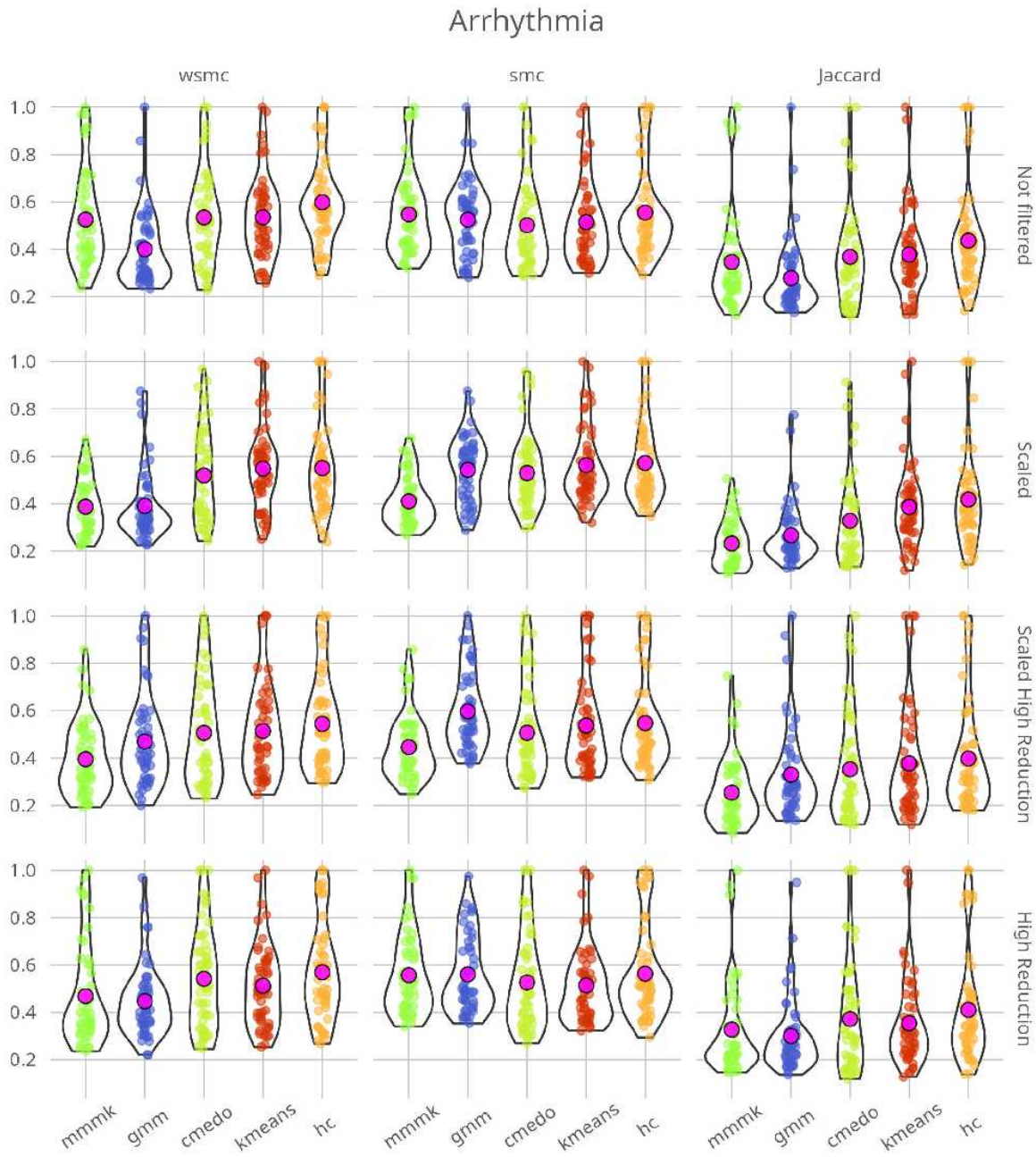

Figure 5.47: Various indexes accross clusters in Arrhythmia

Metrics are highly correlated. Even ARI index, which shown very low correlation with other metrics, in this case was more correlated than accuracy.

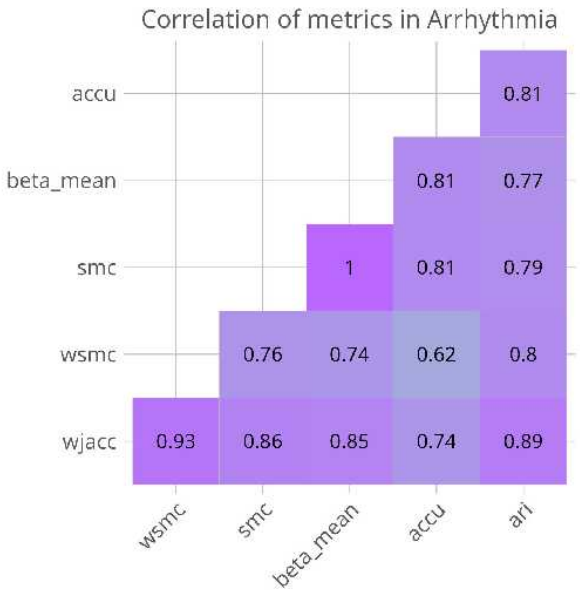

Figure 5.48: Corr plot of metrics in Arrhythmia

6 Conclusions

Based on appropriate versions of the EM algorithm, we formulated the algorithms for the decomposition of Gaussiance Multivariable Mixtures and Multinomial Mixtures. We ensured its stability by switching to the logarithmic scale whenever possible. Otherwise, we used minor hard-coded numeric corrections to avoid division by zero.

We have created an R implementation for Multinomial Mixture Models (MultinomEM) and Gaussian Mixture Models (GaussEM) that is openly available on the GitHub platform. One can install it as a package. However, it still requires to undergo rigorous package testing. Because of that, we cannot guarantee that it will work right out of the box in all machines and configurations. More sanity and dependency checks are required. However, the algorithm should be stable the majority of the time. We are regularly updating it, preferably to the thoroughly tested package. We might rewrite some parts of the code in Rcpp or Armadillo for efficiency in future implementations.

Based on the available source code, we have implemented a few distance-based algorithms for the comparison test. We also have tested a few tested different scenarios for them. It includes an experiment with manhattan/euclidean distance and different initialization of k-means.

We have implemented several metrics to compare algorithm performance and efficiency. In some cases, presented metrics are highly correlated. It is especially true for metrics with a common origin, like those derived with the help of the Hungarian algorithm. As we can see, each metric has its advantages and shortcomings. Some of them, like the ARI index, are easy to implement without additional steps. However, it may only be suitable in some cases.

On the other hand, if we want to compare if there was any correct assignment and how big it was, we should use different metrics. We should also consider unbalanced classes, which might significantly impact metrics. The proposed metric provides additional information about probability density based on beta distribution. We can call it confidence in the calculated probability. However, it requires more mathematical work to balance it properly.

We prepared an extensive simulation study that consists of thousands of Multivariate Gaussian Mixtures and Multinomial Mixtures. To do that, we prepared an R script that allows the creation of selected mixtures with any desired number of observations, dimensions, and components. It allows us to compare the algorithms with a controlled number of parameters and observe the difference in their performance with increased dimensions and clusters.

We have prepared a curated set of real datasets from various publicly available sources. Although the data comes from various science fields, the most significant part consists of the

genomic/medical data. Based on those datasets, we prepared hundreds of different components. Inside the same set, each group combination could occur only once. It allowed us to check the algorithm's performance with a controlled and differing number of components.

We have created R implementation for multinomialEM and GaussEM that is openly available on the GitHub platform. One can install it as a package. However, it did not undergo rigorous package testing, yey. Because of that, we cannot guarantee that it will work right out of the box in all machines and configurations. More sanity and dependency checks are required. However, the algorithm should be stable most of the time. We will update over time, preferably to the thoroughly tested package. In future implementations, we shall rewrite some parts of the code in Rcpp or Armadillo for efficiency.

Based on the available source code we have implemented a few distance-based algorithms that we haves used for the comparison test.

We have implemented several different metrics that allow us to compare algorithm performance and efficiency. In general, presented metrics are highly correlated. It is especially true for metrics with a common origin, like those derived from the Hungarian algorithm. As we can see, each metric has its advantages and shortcomings. The ARI index does not need additional information about labels. However, it can show us negative numbers, which are difficult to interpret.

The model-based algorithms presented in the thesis are a powerful tool in unsupervised clustering methods. In all of the datasets, they performed better than algorithms based on distance. However, there is a pitfall. In rare cases, they might fall into local minima. We have also shown that by using simple heuristics, we might slightly improve the results of the Gaussian Mixture EM. The working example of that heuristic was adding new variables based on the correlation between features. Surprisingly that sometimes Gaussian Mixture EM performed better than Multinomial Mixture EM and vice versa. It might be an indicator that there might be another distribution that will yield better clustering results.

Bibliography

[1] Chandan K Reddy, *Data Clustering: Algorithms and Applications*, Chapman and Hall/CRC, 2018.

[2] Brian S Everitt, et al., "Cluster analysis 5th ed", , 2011.

[3] Leonard Kaufman and Peter J Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, vol. 344, John Wiley & Sons, 2009.

[4] Christian Hennig, et al., *Handbook of cluster analysis*, CRC Press, 2015.

[5] Charles Bouveyron, et al., *Model-based clustering and classification for data science: with applications in R*, vol. 50, Cambridge University Press, 2019.

[6] Geoffrey J McLachlan and Thriyambakam Krishnan, *The EM algorithm and extensions*, vol. 382, John Wiley & Sons, 2007.

[7] Richard McElreath, *Statistical rethinking: A Bayesian course with examples in R and Stan*, Chapman and Hall/CRC, 2020.

[8] Nick T Thomopoulos, "Statistical distributions", *Applications and Parameter Estimates Cham, Switzerland: Springer International Publishing*, 2017.

[9] Merran Evans, et al., *Statistical distributions*, John Wiley & Sons, 2011.

[10] Walter Frank Raphael Weldon, "I. Certain correlated variations in crangon vulgaris", *Proceedings of the Royal Society of London*, 51(308-314), 1892, pp. 1–21.

[11] Peter Schlattmann, *Medical applications of finite mixture models.*, Springer, 2009.

[12] Geoffrey J McLachlan, et al., "Finite mixture models", *Annual review of statistics and its application*, 6, 2019, pp. 355–378.

[13] Arthur P Dempster, et al., "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1977, pp. 1–22.

[14] Jeff A Bilmes et al., "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models", *International Computer Science Institute*, 4(510), 1998, p. 126.

[15] Pierre Blanchard, et al., "Accurately computing the log-sum-exp and softmax functions", *IMA Journal of Numerical Analysis*, 41(4), 2021, pp. 2311–2330.

[16] Marc Kéry and J Andrew Royle, *Applied hierarchical modeling in ecology: Analysis of distribution, abundance and species richness in R and BUGS: Volume 2: Dynamic and advanced models*, Academic Press, 2020.

[17] Lawrence Hubert and Phipps Arabie, “Comparing partitions”, *Journal of classification*, 2(1), 1985, pp. 193–218.

[18] Raimundo Real and Juan M Vargas, “The probabilistic basis of Jaccard’s index of similarity”, *Systematic biology*, 45(3), 1996, pp. 380–385.

[19] Laurens Van der Maaten and Geoffrey Hinton, “Visualizing data using t-SNE.”, *Journal of machine learning research*, 9(11), 2008.

[20] Eleonore Lebeuf-Taylor, et al., “The distribution of fitness effects among synonymous mutations in a gene under directional selection”, *eLife*, 8, 2019, p. e45952, URL `https://doi.org/10.7554/eLife.45952`.

[21] David Benjamin, et al., “Calling somatic SNVs and indels with Mutect2”, *Biorxiv*, 2019, p. 861054.

[22] Bohdan B Khomtchouk, “Codon usage bias levels predict taxonomic identity and genetic composition”, *bioRxiv*, 2020.