TÉCNICO
LISBOA

Clusterval: Python package for determining the number of clusters in longitudinal datasets

Nuno Miguel Canhoto da Silva

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor(s): Prof. Susana de Almeida Mendes Vinga Martins

Prof. Alexandra Sofia Martins de Carvalho

October 2021

Declaration:

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

To my family

Acknowledgments

I want to sincerly thank my supervisors, Prof. Alexandra Carvalho and Prof. Susana Vinga, for letting me be a part of this project even though I was outside of Portugal during the period of development. It meant a lot to me the trust that I received and the continuous support and advice that was given to me through out the development of this thesis.

I would also like to send a word of thanks to Kishan Rama for the ambitious and incredibly useful tool built, AliClu. I wish one day it will be used by professionals and helping patients have a better life.

Last but not least, I want to express my profound gratitude to my friends, collegues from Instituto Superior Técnico, my girlfriend and, especially, my family for supporting me, giving me strength and making me believe i could finish my thesis. This is the last step in my student life and i could not be more proud and happy to have arrived here. This accomplishment is thanks to all of you.

Resumo

O motivo para o uso de clustering é encontrar padrões nos dados. Idealmente, estes algoritmos devolvem os grupos de elementos corretos, porém, esse objectivo nem sempre é alcançado. Portanto, a estrutura de clustering escolhida pelo algoritmo deve ser avaliada, e um bom critério a seguir na validação de clusters é que objetos na mesma partição devem estar próximos. Por outro lado, diferentes clusters estão de preferência notavelmente distantes, respetivamente uns aos outros.

A informação que pode ser obtida da analise dos padrões nos dados tem o potencial para ser útil em virtualmente qualquer área. Nesta dissertação, o foco é numa ferramenta criada em 2019, chamada AliClu, que combina alinhamento de sequências médicas com clustering de forma a analisar dados longitudinais.

O nosso objetivo é aprimorar o aspecto de validação de clustering do AliClu, através do uso de métricas de avaliação. Adicionalmente, iremos trabalhar no sentido de automatizar a geração das sequencias de tratamentos, que são os elementos dos clusters extraidos do dataset Reuma.pt. Finalmente, fazemos uma contribuição para a comunidade de clustering e interessados em geral, por intermédio de uma biblioteca de Python a que chamamos *clusterval*, que permite o acesso ao processo de clustering de forma fácil.

O trabalho aqui apresentado segue dois critérios principais para validação de clusters (external e internal), que serão introduzidos com o respetivo conhecimento relacionado. Seguidamente, a biblioteca *clusterval* é introduzida e o seu funcionamento explicado. No mesmo capítulo, descrevemos a automatização da representação visual dos clusters finais. Adicionalmente, é feita uma experiência extensiva a partir dos indices descritos, usando conjuntos de dados sintéticos e reais.

Palavras-chave: clusterval, clustering, indices de validação de clusters, AliClu

Abstract

Clustering is concerned with finding patterns in a given data. Ideally, clustering algorithms output the correct clusters of elements, although not always that is achieved. The clustering structure chosen by the algorithm should then be evaluated, and a good criterion to follow when doing cluster validation is that objects in the same partition should be close to each other. On the other hand, different clusters are noticeably distant in respect to each other.

The knowledge that can be obtained from the data patterns has the potential to be useful in virtually any area. In this work we focus on a tool created in 2019, called AliClu, that combines alignment of medical sequences with clustering to analyse longitudinal data.

Our goal is to enhance the clustering validity aspect of AliClu through the use of evaluation metrics. Furthermore, we will work on automating the generation of the patients sequences. Finally, we make a contribution to the clustering community and broaden interest as well, by making the clustering process easily available, in the form of a Python library which we call *clusterval*.

The work we present follows two main criteria for clustering validation (external and internal), which will be presented with respective background. Following, the *clusterval* package is introduced and, its functionality explained. In the same chapter we explain the automation behind the clusters visual representation. Additionally, an extensive experiment is made with the described indices, making use of synthetic and real-world datasets.

Keywords: clusterval, clustering, cluster validation indices, AliClu

Contents

List of Tables

List of Figures

Nomenclature

μ_c	Mean value of partition P_c
μ_p	Mean value of partition P
$\mathbf{x}$	Pattern, representing a single data item
c_i	Cluster i
c_ij	(i, j) elements of a partition P_c
c_j	Cluster j
d	the dimensionality of the pattern $\mathbf{x}$
$D(c_i, c_j)$	Distance between c_i and c_j.
d_ij	(i, j) elements of a partition P
$J(C)$	Error square criterion for clustering C
$J(U, C)$	Error square criterion for fuzzy clustering C
M	Maximum number of pairs in a dataset
N	Number of points in the dataset
n_c	Number of clusters
n_i	Number of objects in cluster i
u_{ij}	Membership of data point x_j in cluster C_i
x_i	Attribute or feature of pattern $\mathbf{x}$
EMR	Electronic Medical Records

Chapter 1

Introduction

1.1 Motivation

The principal motivation for this thesis is the project started in [1], which presents a method that combines Temporal Needleman-Wunsch [2] and agglomerative clustering [3]. Although quality work was made, after reading and understanding what was done, it should be possible to build a stronger and more reliable algorithm with the goal of achieving the best results possible. One way that can be done is by developing the clustering validation aspect.

Something which is important to keep in mind when clustering data is that the model does not possess any information about patterns or categories in the data. The clustering algorithm will try to find the best partition of the data so that in each of data groups formed we get very similar data points, in terms of its attributes and features [4]. There is a variety of clustering algorithms currently available, each with input parameters that we can vary in order to obtain the best structure. As a consequence of experimenting with these variations, we are likely going to have to consider more than one possible partition. In case we want to choose only one of the possible clustering results, we have to understand which one represents better our problem context. At this step of the clustering process is where clustering validation measures can be introduced, and this is precisely the subject of this work.

In the literature, numerous clustering validity indices exist that we can use to evaluate the results of a clustering algorithm. Understanding the indices is of great value because some might be better suited for specific clustering algorithms, others can be very computationally expensive, and some work well with a particular type of data. The more knowledge we have on the many ways that are available to analyze the resulting clusters the more efficient will be the work done and will lead us to generating more accurate insights.

However, most of the clustering indices are not easily accessible for usage, which would be of immense value to the potential users, since many times they have to write these indices from scratch. Most libraries for clustering contain some indices to evaluate clustering, but in some cases having more insights could make a difference. Therefore, with this work, we expect to also address this lack of readily available clustering validity measures in the scientific programming world, more specifically in the Python language.

The audience for this thesis and tool here presented includes the pattern recognition and data analysis communities, who should view it as a summarization of current methods and presentation of a usable tool, and to the broader community of scientific professionals, who should view it as an introduction to a mature field that made many contributions in computations for a vast number of areas, with the addition of a tool that might be of great interest.

1.2 Objectives and Contributions

Our goal with this work is to improve the results and give more robustness and confidence to AliClu by increasing the number of clustering validation indices (CVIs) used. Furthermore, given the fact that the visualization of the final clusters is very important to end users, allowing them to implement the data analysis information in their work, we will automatize the representation of the resulting partitions through graphs. Moreover, we have the goal of providing the community with numerous metrics to evaluate clustering results, hence, we propose constructing a Python library with clustering validity indices that is readily available, easy to use and reliable. Available at `https://github.com/Nuno09/clusterval`.

1.3 Thesis Outline

We will go through the previous work done on clustering and validation methods as well as a presentation of the AliClu tool in Section 2. Section 3 presents in detail the contents of the library and how to use it, along with a description on the representation of the final clusters, resulting from AliClu. Section 4 shows the results obtained when using the libraries methods on synthetic, real-world datasets and the Reuma.pt dataset. Finally, in Section 5 we will share some conclusion on the work done.

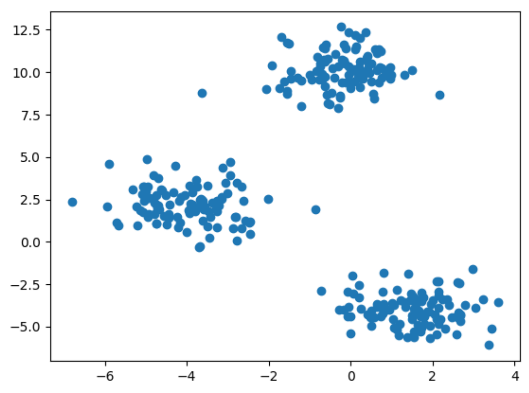
Chapter 2

Related Work

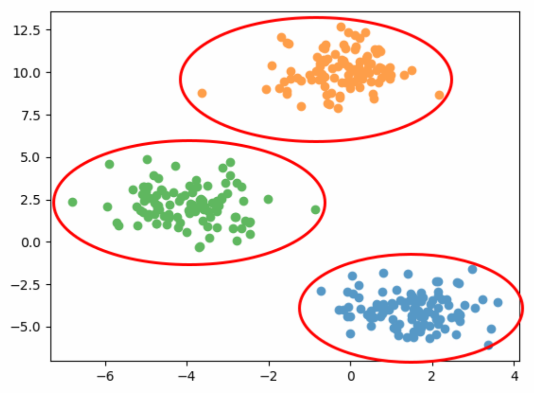
2.1 Overview of Clustering

Webster dictionary [5] defines “Cluster Analysis“ as “a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics“. An example of what clustering tries to achieve is shown in Figure 2.1. The objective is to develop an automatic algorithm that will discover the natural grouping (Figure 2.1b) in the unlabeled data (Figure 2.1a).

Therefore, clustering can be defined as the problem of determining the structure of clustered data, without prior knowledge of the number of clusters or any other information about their composition [6].



(a) Input data.



(b) Desired clustering.

Figure 2.1: **(a)** Clustering starts with a set of unlabeled points and **(b)** tries to detect the clusters in the data.

The development of data clustering methods has been a multidisciplinary feat [3]. Computer scientists, biologists, taxonomists, medical researchers, and others who gather data and analyse it have in some way contributed to clustering methodology.

Many books have been published about data clustering [7], a classic one is [8], which talks about the advances of electronic data processing and shows how to make use of the increasingly available data to compute similarities between taxa and, therefore, segregate them according to those calculations [9].

Other books [3] contain very deep description, analysis and discussion of many cluster analysis algorithms.

There are different approaches to clustering and can be best described with the help of the hierarchy shown in Figure 2.2 [10]

Clustering can broadly be divided into two groups: **Hierarchical** and **Partitional**.

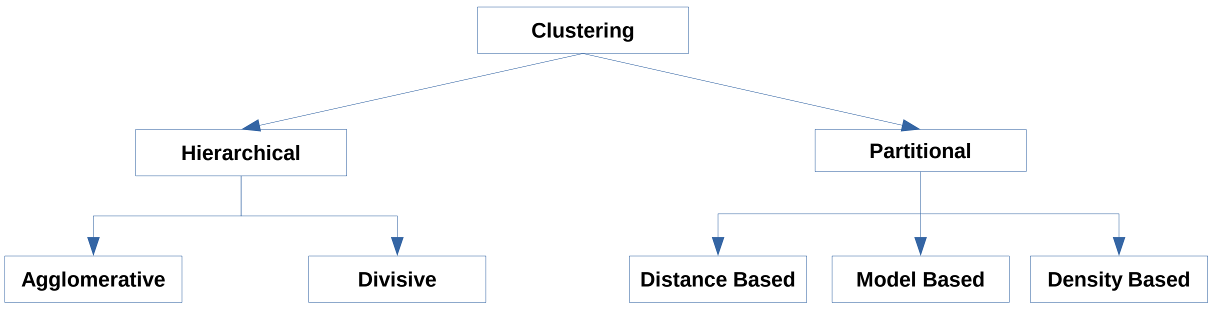


Figure 2.2: Hierarchy of clustering approaches.

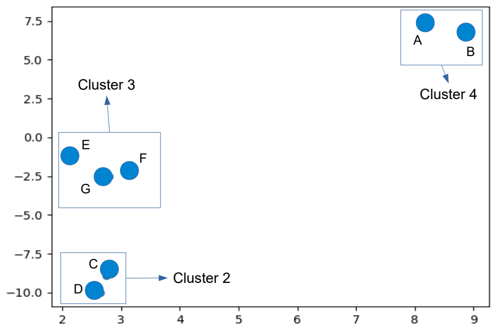
Furthermore, we can also analyse it from the perspective of membership level, clustering can be classified as hard or fuzzy. In hard clustering (traditional approach), each object belongs to one and only one cluster, while in fuzzy clustering, each object has some degree of membership in each partition. The latter method arrives from the notion that in some real-world contexts, there is not precisely defined criteria of membership and ambiguity arises in some cases. Despite being a very interesting method to study it will not be covered in this work, but the reader is encouraged to read [11].

2.1.1 Hierarchical Clustering

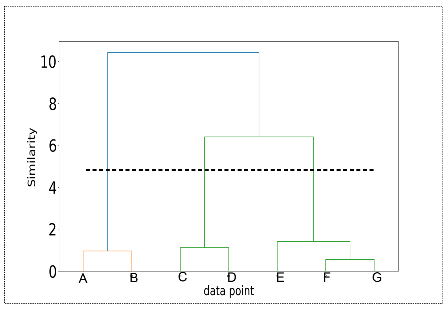
Hierarchical clustering algorithms produce a nested series of partitions. The clustering structure obtained can be represented as a dendrogram, like the one shown in Figure 2.3b. The algorithms that produce the clustering structure follow one of two approaches: **Agglomerative** clustering, which follows a bottom-up approach, in the sense that the algorithm begins with partitions of single objects that are successively merged until a termination condition is met, or only one cluster remains. Alternatively, we can use **Divisive** clustering, which will start with all objects belonging to one unique clustering that will be split at every iteration of the algorithm until a stopping criterion is satisfied, or what is left is single object clusters.

Regardless of the approach taken, merging and splitting of elements follows the similarity between each of the clusters. Lets take the set D that contains n objects, $\mathbf{x}_0 ... \mathbf{x}_n$. From it we can produce a proximity matrix P with n rows and n columns that indexes the degree of similarity between any two objects [12, 13].

When considering the calculation of the similarities, it is natural to view it as a distance problem. Considering that each data point possesses several continuous attributes, the most similar objects will be the ones that have the closest values when comparing their attributes. Consequently, we also need to define a way of measuring the attributes distances, for instance, we can use metrics from the Minkowski family, like the simple Euclidean distance defined in Eq. 2.1.



(a) Data points form 3 clusters.



(b) The dendrogram obtained with single link hierarchical clustering.

Figure 2.3: Hierarchical clustering measures the similarity between each point and builds a structure based on calculations that represent the relation between each data point or group of points. Following an observation of the dendrogram **(b)**, the same can be cut in 3 clusters (the cut is represented by the dashed line).

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_j) &= (\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2)^{1/2} \\ &= \|\mathbf{x}_i - \mathbf{x}_j\|_2. \end{aligned} \tag{2.1}$$

Euclidean distance works well for compact or isolated clusters and tends to favor hyperspherical-shaped clusters of equal size [14]. The drawback to the use of Minkowski metrics is that large-scaled features tend to dominate the others. One solution for this problem is the normalization of the continuous features. Another example of a metric to compute distances between objects is the Mahalanobis distance, but because it requires the computation of the inverse of the sample covariance matrix, this metric can turn out to be computationally expensive, although a way to use it while not being too demanding on resources is presented in [14]. Note that, the calculation of distances is an element that belongs to both hierarchical and partitional clustering.

Firstly, the hierarchical algorithm produces a proximity matrix that will represent the data. At each iteration, the algorithm performs merging or division of clusters (depending on the approach adopted) and afterward updates the proximity matrix. The goal of clustering is to form partitions of the data, which implies that at some moment, we are not just considering the distance between data points but also from one cluster to another. Cluster distance calculation can still use the similarity measures we talked about before, but first, we need to choose the linkage criteria. In this choice lies one of the aspects where algorithms differentiate.

Most hierarchical clustering algorithms are variations of the advances made in the 60's with single-link clustering [15], complete-link clustering [16], and minimum-variance [17]. What differentiates them is the method used to calculate the distance between a newly formed cluster c_1 and another cluster c_2. Following, we show each method we will consider in our work.

The **single link** method takes the similarity between two clusters as the *minimum* distance between all pairs of patterns drawn from the two clusters:

$$single(c_1, c_2) = \min_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2), \tag{2.2}$$

where D is the distance between elements x_1 and x_2 in clusters c_1 and c_2, respectively. This notation is the same for the following equations.

In the **complete link** the distance between two clusters takes the *maximum* of all pairwise distances between patterns in the two clusters:

$$complete(c_1, c_2) = \max_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2). \tag{2.3}$$

The **average link** considers the distance from one cluster to another as the *mean* distance between elements of each cluster:

$$average(c_1, c_2) = \frac{1}{\|c_1\|} \frac{1}{\|c_2\|} \sum_{x_1 \in c_1} \sum_{x_2 \in c_2} D(x_1, x_2). \tag{2.4}$$

Centroid link measures the distance between the centroid of one cluster and the centroid of another cluster:

$$centroid(c_1, c_2) = D\left(\frac{1}{\|c_1\|} \sum_{x_1 \in c_1} x_1, \frac{1}{\|c_2\|} \sum_{x_2 \in c_2} x_2 \right). \tag{2.5}$$

Ward's method uses the *Ward variance minimization* algorithm, which measures how much the sum of squares will increase if we merge two clusters.

$$ward(c_1, c_2) = \sum_{x_1 \in c_1} (x_1 - r_1)^2 + \sum_{x_2 \in c_2} (x_2 - r_2)^2 - \sum_{y \in c_{12}} (y - r_{12})^2, \tag{2.6}$$

where r_1, r_2 are the centroids of c_1 and c_2 , respectively, and r_{12} is the centroid of the two clusters merged.

Thus far, we have reviewed the basis of hierarchical clustering, which can have some limitations with dataset sizes, resources and outliers. Most recent advances can handle large datasets much better, examples being CURE [18], which is able to explore more sophisticated cluster shapes and also with reduced computational complexity, BIRCH [19] for robustness to outliers and ROCK [20] based on the idea of links between objects, instead of distance.

2.1.2 Partitional Clustering

A partitional clustering algorithm obtains a single partition of the data instead of a clustering structure, as described in the previous subsection. Hierarchical techniques are popular in biological, social, and behavioral sciences because of the need to construct taxonomies. Partitional are more frequently used in engineering applications where single partitions are important [3]. Moreover, partitional algorithms have advantages when dealing with large datasets for which the building of a dendrogram by a hierarchical algorithm would be computationally expensive and even hard to read by the human eye. One restriction that partitional algorithms impose is the choice of the number of desired output clusters.

Partitional algorithms assign the data to k clusters without any hierarchical structure by optimizing some criterion function. Looking at the clustering taxonomy (Figure 2.2), there is at least three types of partitional algorithms.

The most intuitive and frequently used criterion function is the *Error Square*, based on the use of the distance between objects. The general idea of these types of algorithms is to obtain the partition which, for a fixed number of clusters, minimizes the square error. Suppose we want to organize a set of objects $\mathbf{x}_j, j = 1, ..., N$, into k subsets $C = C_1, ..., C_k$. The error squared criterion J then is defined as:

$$J(C) = \sum_{i=1}^{k} \sum_{j=1}^{N} \|x_j - c_i\|^2. \tag{2.7}$$

The most popular algorithm that uses *Error Squared* is **K-means** [21]. The basic steps for K-means are as follows:

1. Initialize k centroids randomly or based on previous knowledge, each centroid representing a cluster;
2. Assign each object in the dataset to the nearest cluster C_k;
3. Recalculate the positions of the k centroids, based on the mean position value of the current partitions;
4. Repeat steps 2-3 until there is no change in the centroids or convergence is achieved for the error squared function.

K-means is popular much thanks to being very easy to implement, and its time complexity is $O(n)$, where n is the number of objects in the dataset. Despite its wide array of applications, K-means is not exempt of drawbacks. One of them, already mentioned earlier, is that the algorithm requires the input of the number of clusters k beforehand, which is not always possible to know accurately in real-world applications and in turn will require some parameter experimentation. Another problem is that it is very influenced by the starting conditions (randomly chosen centroids and initial partitions), the convergence centroids vary with different initial points. A solution to this is to run the algorithm many times with random initial partitions.

Many other algorithms have been proposed, each tackling one or more of the weak points of *K-means*. Going in detail over all would require single focus on *K-means*, which is not the purpose of this work, hence the reader can look up ISODATA [22], PAM [23], and K-medoids algorithm [24] for more information.

Previously in this chapter we mentioned fuzzy clustering, and inside this approach we want to make a notable reference to the *Fuzzy C-means algorithm*, or in other words, the fuzzy version of hard clustering method, *K-means*. This is particularly useful when the boundary between the clusters is ambiguous. The procedure of fuzzy *C-means* [25] is similar to *K-means*, with the addition of a partition matrix $U = u_{ij}$, $u_{ij} \in [0,1]$ and $\sum_{i=1}^{K} u_{ij} = 1$ that is calculated using the following equation:

$$u_{ij} = 1/\sum_{l=1}^{k}(\frac{\|x_j - c_i\|}{\|x_j - c_l\|}), \tag{2.8}$$

where u_{ij} is the membership of data point x_j in cluster C_i. The error square function J is updated from Eq. (2.7) to the following:

$$J(U,C) = \sum_{i=1}^{k}\sum_{j=1}^{N} u_{ij}\|x_j - c_i\|^2, \tag{2.9}$$

where the points membership u_{ij} is considered.

The overall steps for *Fuzzy C-means* (FCM) are:

1. Set a value for the number of clusters k and initialize the centroids $\mathbf{C}$;
2. Calculate or update the membership matrix U using Eq. (2.8);
3. Compute the new centroids for the clusters using the updated matrix U;
4. Repeat steps 2-3 until there is no change in the centroids or convergence is achieved for the error squared function.

FCM, like k-means, suffers from initial partition dependence, as well as sensibility to noise and outliers. There is in the literature methods addressing this issue. For instance, in [26] an estimation is made on the centroids of the initial partition. [27] studies a method for carrying out fuzzy clustering without a priori assumptions on the number of clusters in the dataset. Finally, [28] introduces a modification that increases robustness to outliers.

From a probabilistic perspective, as described in [29], data objects are assumed to be generated according to several probability distributions. Data points in different clusters were generated by different probability distributions. They can be derived from different types of density functions (e.g., multivariate Gaussian or t-distribution), or the same families, but with different parameters. If the distributions are known, finding the clusters of a given data set is equivalent to estimating the parameters of several underlying models. Suppose the so called mixing probability $P(C_i)$ for cluster $C_i, i = 1, ..., k$ and the conditional probability density $p(x|C_i, \theta_i)$, where θ_i is the unknown parameter vector. Then, the mixing probability density for the whole dataset is expressed as:

$$p(x|\theta) = \sum_{i=1}^{k} p(x|C_i, \theta_i)P(C_i), \tag{2.10}$$

where $\theta = (\theta_1, ..., \theta_k)$, and $\sum_{i=1}^{k} P(C_i) = 1$. As long as parameter θ is decided, we can calculate the posterior probability for assigning a data point to a cluster, using Bayes's theorem. Parameter θ can be estimated using maximum likelihood estimation (ML).

Unfortunately, since the solutions of ML cannot be obtained analytically in most circumstances, an iteratively sub optimal approach is required, the most popular one being the Expectation-Maximization (EM) algorithm [30]. EM divides the dataset into two parts, for each data point x_j, x_j=$\{{x_j}^g, {x_j}^m\}$, in which ${x_j}^g$ represents the observable features and ${x_j}^m = ({x_{j1}}^m, ..., {x_{jK}}^m)$ is the missing data, where ${x_{ji}}^m$ chooses a value of 0 or 1 according to weather or not x_j belongs to component (cluster) i. Thus, the complete data log-likelihood is:

$$l(\theta) = \sum_{j=1}^{N} \sum_{i=1}^{k} x_{ji}^m \log[P(C_i)p(x_j^g|\theta_i)]] \tag{2.11}$$

The EM algorithm will generate a series of parameter estimates for θ. One of the disadvantages of EM is its sensitivity to the selection of initial parameters.

2.2 Overview of Clustering Evaluation Metrics

In this section, our focus will be on discussing methods to validate the clustering resulting from applying a clustering algorithm to a dataset. We can find in the literature two main methods: **internal** and **external** validation [31], [32]. Although in some previous work there is also distinction of a third approach, relative validation [33], but seems like nowadays the community leans more to the first two methods and for the purpose of this work we will do the same.

At the turn of the 60s, coming from the work done in [34], some methods for comparison of dendrograms start to arise, for instance, the cophenetic index [35] measures the connectness of data points in the clusters by representing their linkage by "levels", the index increases the less "levels" it takes to connect all points. With the goal of addressing the dependence of the results on the clustering algorithm used, in [36] some statistical criteria is presented for that analysis, such as entropy, average distance from nearest cluster center and coefficient of belongingness. In [37], it is presented a mathematical criteria and related statistical theory for finding the "best" partition, for instance, some of the criteria is the minimum pairwise distance considering all partitions, and a measure for scatterness using a total scatter matrix of the N dataset points.

Very important advances in the area of clustering evaluation came in the 70s. Firstly, [38] proposed the first objective method for clustering validation, with the Rand index, that like many new indicies after it, compares two different clustering results based on the similarity between pairs of points. A couple of years after the Dunn index [39] follows, expanding the idea of pairwise comparison to evaluate two clusterings. Davies and Bouldin [40] introduced a new idea and from it a new index (DB index) which considers the dispersion inside the clusters and also the distance between the clusters, the advantage of this approach is that it is less dependent on parameters given, and like we mentioned before, this is one of the biggest drawbacks of unsupervised learning and clustering in general.

From the developments we talked about so far, many other indices were derived, each trying to improve the quality of the clustering produced and reducing the drawbacks from previous work.

The foundation of most of the clustering validation indices (CVI) is the comparison of clusterings, one way to reach the conclusions is by using statistical tests, consequently, there is the need of producing many results in order to get an accurate estimate of the "best" clustering, and most of the methods presented use some kind of random sampling of results, particularly the Monte Carlo method is very useful and an interesting experiment with it can be found in [41], where 30 different indices were compared.

A Monte Carlo experiment is every experiment that relies on repeated random sampling to obtain numerical results. In a very broad way, for our problem context a monte carlo experiment will follow this steps:

1. For a given dataset, perform data clustering;
2. Set a maximum number of iterations for the experiment;
3. Produce a random sample from the original dataset and perform clustering;
4. Using a CVI, compare the original clustering with the new one and save the value;
5. Repeat steps 3-4 until the monte carlo maximum iterations are reached.

Variations for an experiment like this include comparisons of clustering algorithms, of different initial parameters or using several different CVIs for comparison.

The first group of CVIs we can choose from is called **external validation** measures, and it is based on the comparison of partitions, the partitions that are used for the comparisons are the one generated by the clustering algorithm, a partition generated from running the clustering algorithm on a random sample of the dataset, or a given partition of the data (or a subset of the dataset). A second approach is called **internal validation**, which is based on calculating properties of the resulting partitions, such as compactness of the clusters, separation and roundness. In the sequel we will describe the fundamentals for each of the two cluster validity approaches and list the several indices that will be part of our work.

2.2.1 External Clustering Validation Measures

The idea behind this approach is to test against the randomness of the clustering results. In order to evaluate the resulting structure, we use statistical tests, more specifically the average. The metrics that we use for this approach will evaluate the resulting clustering C, by comparing it to the clustering of a random subset the data, P.

Let us introduce the contingency matrix that is represented in Table 2.1, which contains information on the clusterings overlap. Each entry n_{ij} indicates the number of elements that are common to cluster C_i and P_j

		Partition C				
		C_1	C_2	...	$C_{k'}$	$\sum$
Partition P	P_1	n_{11}	n_{12}	...	$n_{1k'}$	$n_{1\bullet}$
	P_2	n_{21}	n_{22}	...	$n_{2k'}$	$n_{2\bullet}$
	.	.	.	...	.	.
	P_k	n_{k1}	n_{k2}	...	$n_{kk'}$	$n_{k\bullet}$
	$\sum$	$n_{.1}$	$n_{.2}$	...	$n_{.k'}$	n

Table 2.1: The contingency matrix

The information in the contingency matrix can be transformed into a pairwise mismatch agreement between the clusters, shown in Table 2.2.

		Partition C		
	Number of pairs	In the same cluster	In different clusters	Sums
Partition P	In the same cluster	a	b	$a + b$
	In different clusters	c	d	$c + d$
	Sums	$a + c$	$b + d$	M

Table 2.2: The mismatch matrix

The entries a, b, c and d represent counts of pairs among all distinct pairs of the clusters overlap. We can interpret a and d as agreements between the two configurations, and b and c as disagreements between the two clusterings C and P. Following, we present twelve external vaildation indices for which higher values correspond to better clustering structures, unless the oposite is mentioned. Moreover, most of the indices use the terms from the mismatch matrix in Table 2.2 and new terms will be properly described.

- **Adjusted Rand index** [42] :

  The adjusted Rand index is the “corrected for chance“ version of the Rand index [38]. The index is given by:

$$AR = \frac{a + d - n_c}{M - n_c}, \tag{2.12}$$

  where, using the contingency matrix from Table 2.1, n_c is given by:

$$n_c = \frac{n(n^2+1) - (n+1)\sum n_{i\cdot}^2 - (n+1)\sum n_{\cdot j}^2 + \sum\sum \frac{n_{ij}^2}{n}}{2(n-1)}. \tag{2.13}$$

  This index gives the overall concordance between two partitions, taking into account that the agreement between partitions could arise from chance alone.

- **Jaccard index** [33]:

  The index is calculated using:

$$J = \frac{a}{a + b + c}. \tag{2.14}$$

  It is very similar to Rand index, however, it disregards the pairs of elements that are in different clusters (d in the mismatch matrix) for both clusterings.

- **Fowlkes and Mallows index** [43]:

  The index is given by:

$$FM = \sqrt{\frac{a}{a+b} . \frac{a}{a+c}}. \tag{2.15}$$

  Fowlkes and Mallows proposed to calculate partitions dissimilarities based on the geometric mean between precision ($\frac{a}{a+b}$) and recall ($\frac{a}{a+c}$).

- **F-Measure** [44]:

  The following expression defines the index:

$$F = \frac{2a}{2a + b + c}. \tag{2.16}$$

  F-measure provides a combination of precision and recall in one metric, similarly to Fowlkes and Mallows.

- **Adjusted Wallace index** [45]

  Firstly, lets define the Wallace coefficients [46], based on the terms from Table 2.2:

$$AW_{C \to P} = \frac{a}{a+b}, \tag{2.17}$$

  which gives the probability of two pairs being in the same cluster of P, when they belong to the same cluster of C.

  Whereas the probability of two pairs being in the same cluster of C, when they belong to the same cluster of P is given by:

$$AW_{P \to C} = \frac{a}{a+c}, \tag{2.18}$$

  The Adjusted Wallace index is defined using:

$$AW_{C \to P} = \frac{W_{C \to P} - W_{i(C \to P)}}{1 - W_{i(C \to P)}}, \tag{2.19}$$

  where $W_{i(C \to P)}$ is the expected Wallace coefficient under independence and is computed as:

$$W_{i(C \to P)} = 1 - SID_P, \tag{2.20}$$

  where SID_P is the Simpson's index of diversity of the clustering P given by:

$$SID_P = 1 - \frac{1}{n(n-1)} \sum_{j=1}^{K} n_{\cdot j}(n_{\cdot j} - 1), \tag{2.21}$$

  where the terms come from the contingency matrix in Table 2.1.

- **Kulczynski index** [47]

  The Kulczynski dissimilarity score is given by:

$$K = \frac{1}{2}\left(\frac{a}{a+c} + \frac{a}{a+b}\right). \tag{2.22}$$

- **Phi index** [48]

  Phi coefficient is a classical measure of association between two variables, in the case of this problem, two partition. The index is given by:

$$Phi = \frac{ad - bc}{(a+b)(a+c)(b+d)(c+d)}. \tag{2.23}$$

- **Rogers-Tanimoto index**

  Another popular measure of similarity, which is described as:

$$RT = \frac{a+d}{a+d+2(b+c)}. \tag{2.24}$$

- **Sokal-Sneath index** [48]:

  One version of Sokal-Sneath metrics is given by:

$$SS = \frac{a}{a+2(b+c)}. \tag{2.25}$$

- **Hubert index** [42]:

  The Hubert's index used is the normalized version of the original metric, represented by Γ and can be defined as:

$$\Gamma = \frac{[(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(X(i,j)-\mu_x)(Y(i,j)-\mu_y)]}{\sigma_x\sigma_y}, \tag{2.26}$$

  where $\mu_x, \mu_y, \sigma_x, \sigma_y$ are the respective means and variances of X, Y matrices. The index takes values in the interval of -1 and 1.

  Rewriting to use the mismatch matrix we get:

$$\Gamma = \frac{Ma-(a+b)(a+c)}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}. \tag{2.27}$$

- **Variation of Information index** [49]:

  This measure makes use of the concepts of entropy and mutual information, both presented in detail in [50]. It is defined as:

$$VI = [\mathcal{H}(C) - \mathcal{I}(C,P)] + [\mathcal{H}(P) - \mathcal{I}(C,P)], \tag{2.28}$$

  where $\mathcal{H}(C), \mathcal{H}(P)$ are the entropy of clustering C and entropy of partition P, respectively, and $\mathcal{I}(C,P)$ is the mutual information between C and P.

  Using the contigency matrix we get:

$$VI = -\sum_i p_i \log p_i - \sum_j p_j \log p_j - 2\sum_i\sum_j p_{ij}\log\frac{p_{ij}}{p_i p_j}, \tag{2.29}$$

  where $p_{ij} = n_{ij}/n$, $p_i = n_{i.}/n$, $p_j = n_{.j}$.

  VI measures the amount of information that is lost or gained from C to P, hence a lower value for the index represents a better clustering.

- **Van Dongen index** [51, 52]:

  The measure is based on the maximum intersections of clusters. To find intersections between C and P we make use of the contingency table from Table 2.1.

  Van Dongen measure is then given by:

  $$VD = \left(2n - \sum_i \max_j n_{ij} - \sum_j \max_i n_{ij}\right) \Big/ 2n. \tag{2.30}$$

  Logically, a good clustering is one where intersections between clusters is minimal, which is indicated by a lower value of Van Dongen.

2.2.2 Internal Clustering Validation Measures

As the goal of clustering is to make objects within the same cluster similar and objects in different clusters distinct, internal validation measures are often based on the following two criteria:

- **Compactness.** This measures how closely related the objects in a cluster are. Some metrics measure this based on variance. Lower variance indicates better compactness. In addition, numerous measures estimate the cluster compactness based on distance, such as maximum or average pairwise distance, and maximum or average center-based distance.
- **Separation.** This measures how distinct or well-separated a cluster is from other clusters. For example, the pairwise distance between cluster centers and the pairwise minimum distances between objects in different clusters are widely used as a measure of separation. Also, measures based on density are used in some indices.

Following, we will define the eight internal validation indices we will use for our work.

- **Dunn index [53], [39]**

  Dunn calculates the minimum distance between clusters to measure the intercluster separation and the maximum diameter among all clusters to measure the intracluster compactness. The index is defined for a given number of clusters:

$$D(n_c) = \min_{i=1,\ldots,n_c} \left\{ \min_{j=i+1,\ldots,n_c} \left(\frac{diss(c_i, c_j)}{\max_{k=1,\ldots,n_c} diam(c_k)} \right) \right\}, \tag{2.31}$$

  where $diss(c_i, c_j)$ is the dissimilarity function between two clusters c_i and c_j defined as:

$$diss(c_i, c_j) = \min_{x \in c_i, y \in c_j} d(x, y), \tag{2.32}$$

  where x and y are elements in cluster c_i and c_j, respectively.

  In Eq. (2.31), $diam(c)$ is the diameter of a cluster, which may be considered as a measure of clusters' dispersion. The diameter of a cluster C can be defined as follows:

$$diam(C) = \max_{x, y \in C} d(x, y), \tag{2.33}$$

  where x and y are elements belonging to cluster C.

  If the dataset contains compact and well-separated clusters, the distance between clusters is expected to be large and the diameter of the clusters is expected to be small. Thus, based on the definition in Eq. (2.31), we may conclude that larger values indicate a better clustering.

- **Davies-Bouldin index** [40]

  This measure computes the average similarity between each pair of clusters, therefore a lower value indicates a high dissimilarity between clusters, which is desirable.

  The similarity metric is defined as:

$$R_{ij} = (s_i + s_j)/d_{ij}, \tag{2.34}$$

  where s_i, s_j are the measures of dispersion of clusters C_i and C_j, respectively, and d_{ij} the distance between the two centroids.

  Then the index is defined as:

$$DB(n_c) = \frac{1}{n_c} \sum_{i=1}^{n_c} \max_{i=1,\dots,n_c, i \neq j} R_{ij}. \tag{2.35}$$

- **SD validity index** [54]

  It's based on the concept of average scattering, which indicates the compactness *within* the clusters and defined as:

$$Scat(n_c) = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{\|\sigma(c_i)\|}{\|\sigma(X)\|}, \tag{2.36}$$

  where c_i is the center of cluster i, X is a data set and $\sigma(c_i), \sigma(X)$ are the variance of cluster i and variance of dataset X, respectively.

  Additionally, SD considers the total separation of clusters, which indicates the separation *between* the clusters defined by the following equation:

$$Dis(n_c) = \frac{D_{max}}{D_{min}} \sum_{k=1}^{n_c} \left(\sum_{z=1}^{n_c} \|c_k - c_z\| \right)^{-1}, \tag{2.37}$$

  where $D_{max} = max(\|c_i - c_j\|) \forall_{i,j} \in 1, 2, 3, ..., n_c$ is the maximum distance between cluster centers. $D_{min} = min(\|c_i - c_j\|) \forall_{i,j} \in 1, 2, 3, ..., n_c$ is the minimum distance between cluster centers.

  Now we can define the index based on Eq. (2.36) and Eq. (2.37) defined as:

$$SD(n_c) = \alpha Scat(n_c) + Dis(n_c), \tag{2.38}$$

  where α is a weighting factor to normalize the two terms of the equation calculated using:

$$\alpha = Dis(max\ n_c). \tag{2.39}$$

  The n_c that minimizes the index can be considered the optimal value for the number of clusters in the dataset.

- **SDbw validity index** [55]

  Like SD index, it evaluates both criteria for “good“ clustering (i.e., compactness and separation) and can be calculated using:

$$SDbw(n_c) = Scat(n_c) + Dens\_bw(n_c), \tag{2.40}$$

where the first term, which measures compactness, is equal to Eq. (2.36). The second term evaluates the inter-cluster density and a low value indicates a better separation. The inter-cluster density can be defined as follows:

$$Dens\_bw(n_c) = \frac{1}{n_c(n_c-1)} \sum_{i=1}^{n_c} \left(\sum_{j=1, i\neq j}^{n_c} \frac{density(u_{ij})}{max\{density(c_i), density(c_j)\}} \right), \tag{2.41}$$

where c_i, c_j are, respectively, the center of clusters i and j, u_{ij} the middle point of the line segment defined by the clusters' centers c_i and c_j. The term density is defined as:

$$density(u) = \sum_{l=1}^{n_{ij}} f(x_l, u), \tag{2.42}$$

where n_{ij} = number of tuples that belong to clusters c_i and c_j i.e., $x_1 \in c_i \cup c_j$ represents the number of points in the neighbourhood of u. Such neighbourhood is calculated using:

$$f(x,u) = \begin{cases} 0, & \text{if } d(x,u) > stdev \\ 1, & \text{otherwise} \end{cases}, \tag{2.43}$$

where $f(x,u)$ is a hyper-sphere with center u and radius as the average standard deviation of the clusters, $stdev$.

The n_c that minimizes the above index can be considered as an optimal value for the number of clusters present in the dataset.

- **CVNN index** [56]

  The Clustering Validation index based on Nearest Neighbors (CVNN), is a validation measure based on the notion of nearest neighbours, and complements some of the already existing measures.

  Similar to previous internal metrics, it is generally based on **1)** inter-cluster separation and **2)** intra-cluster compactness. For the separation step the idea is that, if an object is located in the center of a cluster and is surrounded by objects in the same cluster, then it is well separated from other clusters and thus contributes little to the inter-cluster separation. If an object is located at the edge of a cluster and it is surrounded mostly by objects in other clusters, then contributes a lot to the inter-cluster separation. The measurement is given by the following equation:

$$Sep(n_c, k) = \max_{i=1,2,\ldots,n_c} \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{q_j}{k}, \tag{2.44}$$

  where k is the number of nearest neighbours, n_i is the number of objects in ith cluster C_i, and q_j is the number of nearest neighbours of the jth object in C_i that are not in C_i. Note that a lower value of Sep indicates a better clustering separation.

  To measure the compactness within the clusters, we compute the average pairwise distance between objects in the same cluster, which is given by:

$$Comp(n_c) = \sum_i \left[\frac{2}{n_i \cdot (n_i - 1)} \sum_{x,y \in C_i} d(x,y) \right], \tag{2.45}$$

  where n_i is the number of objects in the ith cluster C_i, and x and y are two different objects in C_i. Note that a lower value of $Comp$ indicates a better clustering separation.

  Based on the above equations we can define the CVNN index as:

$$CVNN(n_c, k) = Sep_{norm}(n_c, k) + Comp_{norm}(n_c, k), \tag{2.46}$$

  where the first term is defined by:

$$Sep_{norm}(n_c, k) = \frac{Sep(n_c, k)}{(max_{n_{c_{min}} \leq n_c \leq n_{c_{max}}} Sep(n_c, k))}. \tag{2.47}$$

  The second term of Eq. (2.46) is defined as:

$$Comp_{norm}(n_c) = \frac{Comp(n_c)}{(max_{n_{c_{min}} \leq n_c \leq n_{c_{max}}} Comp(n_c))}. \tag{2.48}$$

This index takes form from the summation of the inter-cluster separation and the intra-cluster compactness. Because the terms have different ranges, they need to be normalized like in Eq. (2.48) and Eq. (2.47). Note that a lower value for CVNN indicates a better clustering result.

- **PBM index** [57]

  The PBM index (acronym constituted of the initals of the names of its authors, Pakhira, Bandyopadhyay and Maulik) is calculated using the distances between the points and their cluster centers, and the distances between the cluster centers themselves.

  For a measure of the inter-cluster separation, let us denote by D_C the largest distance between two cluster centers:

$$D_C = max_{i,j \in n_c} d(c_i, c_j), \tag{2.49}$$

  where c_i and c_j are the centers of clusters i and j, respectively.

  On the other end, considering the within clusters compactness, let us denote by E_C the sum of the distances of the points of each cluster to their center, using:

$$E_C = \sum_{k=1}^{n_c} \sum_{i \in C_k} d(x_i, v_k), \tag{2.50}$$

  where x_i is a point in cluster C_k and v_k is the center of cluster k.

  Lets denote E_T as the sum of the distances of all the points to the center of the entire dataset that can be defined as:

$$E_T = \sum_{i=1}^{N} d(x_i, G_C), \tag{2.51}$$

  where where x_i is a point in the dataset and G_C is the center of the dataset, calculated using the average of all cluster centers.

  The PBM index is defined as:

$$PBM = \left(\frac{1}{n_c} \times \frac{E_T}{E_C} \times D_C \right)^2, \tag{2.52}$$

  where the first term is used to eliminate the chances that the second term becomes to small, and a lower value of n_c is preferred, meanwhile, for the second and third terms higher values are desired, which indicates better compactness and separation, respectively. Note that a higher value for PBM indicates a better clustering result.

- **Silhouette index** [58]

  Each cluster will be represented by a so-called *silhouette*, which is based on the comparison of its tightness and separation. This index shows which objects lie well within their cluster, and which ones are merely somewhere in between clusters. The maximum average silhouette value might be used to select an “appropriate“ number of clusters. The index can be represented as follows:

$$S = \frac{1}{n_c} \sum_{i=1}^{n_c} \left(\frac{1}{n_i} \sum_{x \in C_i} \frac{b(x) - a(x)}{max[b(x), a(x)]} \right), \tag{2.53}$$

where b and a functions are defined in the following way:

$$b(x) = min_{j, j \neq i} \left[\frac{1}{n_j} \sum_{y \in C_j} d(x, y) \right], \tag{2.54}$$

$$a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i, y \neq x} d(x, y), \tag{2.55}$$

where function $b(x)$ calculates the average distance of object x to all other objects of a cluster $C_j, j \neq i$ and selects the minimum of those values. Function $a(x)$ computes the average dissimilarity of object x to all other objects of its cluster C_i.

A good clustering will have objects of a cluster very distant from other objects in different clusters, then a high value of b is preferred. Moreover, a lower value of a indicates compact clusters.

- **Xie-Beni index** [59]

  The Xie-Beni improved index (XB) defines the intercluster separation as the minimum square distance between cluster centers, and the intracluster compactness as the maximum square distance between each data object and its cluster center. The optimal cluster number is reached when the minimum of XB is found.

  We consider the improved version from [60], which is defined as:

$$XB = \frac{max_{k=1,..,n_c} \frac{\sum_{j=1}^{n_k} (\|x_j - c_k\|)^2}{n_k}}{min_{i,l \neq i (\|c_i - c_l\|)^2}}, \tag{2.56}$$

  where x_j is the j element in cluster k, c_k is the center of cluster k, n_k is the number of elements in cluster k, while c_i and c_j are the centroids of clusters $i, j \in n_c$, respectively.

2.3 AliClu

Given the increasing availability of electronic medical records (EMRs) with longitudinal data, this data can be used for making clinical decisions so that physicians can choose personalized treatments for the patients. The work done sets out with the goal of finding an efficient way to cluster patients based on their temporal information from medical appointments. It is proposed the application of the Temporal Needlman and Wunsch algorithm [2], a modified version of the original Needleman and Wunsch [61], to align discrete sequences with the transition time information between symbols. This symbols may correspond to a patient's therapy, overall health status, or any other discrete state. The obtained TNW pairwise scores are then used to perform hierarchical clustering. To test the performance of the method, synthetic datasets were generated by continuous-time Markov chains. For datasets with 2 clusters and that were very well separated, the method successfully found the correct clusters with percentage of correct decisions above or equal to 80%. The tests with the Reuma.pt dataset were done in a non-automatic manner, i.e., the results of clustering were analysed individually. It was discovered 18 clusters in the biological sequences where successful separation based on the events and temporal information between them was achieved.

This is a very promising tool to analyse longitudinal patient data and unravel patterns that exist in clinical outcomes. Nevertheless, the validation of the resulting clusters, being a very important step in identifying the correct patterns, is not getting enough focus. Only external validation measures are considered and are not very extensive. Furthermore, AliClu results were only considered for non-automatic analysis, we look forward to having a more optimistic automatic approach. Therefore, we propose to add on top of AliClu, more CVIs. Hopefully, the new information that will be gathered can make the cluster decision process more automatic and trustworthy.

One last contribution we wish to give, concerns the visualization of the clusters. In previous work, the generation of the patients sequence was done manually, but being a repetitive process and also a fairly important one given the fact that this representation is what gives values to end users, we built a function that produces said clusters.

Chapter 3

Implementation

3.1 Clusterval

The *clusterval* package for python provides an implementation of all the indices described in the preceding sections and can be found in a github repository: https://github.com/Nuno09/clusterval.

The package evaluation of a clustering of data follows different steps, relative to which type of validation method it is using. For **external validation**:

1. Set the range of **k** to be tested;
2. Perform clustering on the data for **k** clusters;
3. Set the range of bootstrap samples to produce;
4. Generate a random sample from the original dataset;
5. Perform clustering of the sample, using the same **k**;
6. Calculate the values of each index;
7. Repeat step 5 until the limit of bootstrap samples has been reached;
8. Save the average values of the indices for the current **k**;
9. Repeat from step 4 with new **k**, until the max **k** value is reached;
10. Each index takes a vote on the number of clusters that gives the best index value. The majority answer is defined as the number of clusters for the dataset;

The **internal validation** steps:

1. Set the range of **k** to be tested;
2. Perform clustering on the data for **k** clusters;
3. Calculate the values of each index for the number of clusters **k**;
4. Repeat from step 2 with new **k**, until the max is reached;
5. Each index takes a vote on the number of clusters that gives the best index value. The majority answer is defined as the number of clusters for the dataset;

In the next subsection we can see the tables that describe the indices available in the package, with information on the name to use, the values range and the rule for the best value.

3.1.1 External Indices

Index	**Name in clusterval**	**Range**	**Rule**
Adjusted Rand	AR	[-1,1]	*max*
Jaccard	J	[0,1]	*max*
Fowlkes and Mallows	FM	[0,1]	*max*
F-Measure	F	(0,1]	*max*
Adjusted Wallace	AW	[-1,1]	*max*
Kulczynski	K	(0,1]	*max*
Phi	Phi	(-1,1]	*max*
Rogers-Tanimoto	RT	(0,1]	*max*
Sokal-Sneath	SS	(0,1]	*max*
Hubert	H	(-1,1]	*max*
Variation of Information	VI	[0,2log max(K, K')] [1]	*min*
Van Dongen	VD	[0,1)	*min*

Table 3.1: External clustering validation Indices.

[1]K = n_c from partiton C and K' = n_c from partition P.

3.1.2 Internal Indices

Index	Name in clusterval	Range	Rule
Xie-Beni index	XB	(0,+∞)	*min*
Dunn index	Dunn	(0,+∞)	*max*
Davies-Bouldin index	DB	(0,+∞)	*min*
SD index	SD	(0, +∞)	*min*
SDbw index	SDbw	(0, +∞)	*min*
CVNN index	CVNN	(0, +∞)	*min*
PBM index	PBM	(0,+∞)	*max*
Silhouette index	S	(0,+∞)	*max*

Table 3.2: Internal clustering validation indices.

3.2 Usage of *clusterval*

The *clusterval* package for Python can be installed by getting the stable version from the Python software repository, PyPi, with the following instruction:

```
pip install clusterval
```

Alternatively, the development version can be pulled from GitHub:

```
pip install git+https://github.com/Nuno09/clusterval.git
```

The package requires at least Python 3.8 and the tests were performed using Python 3.8.5 on a Linux machine running the Ubuntu distro.
Once the package is installed, it can be loaded in a Python environment like so:

```
from clusterval import Clusterval
```

Let's import some datasets from the *sklearn* package and load iris:

```
from sklearn.datasets import load_iris, make_blobs
data = load_iris()['data']
```

We are now able to use the full capabilities of the package.
The code below will create a Clusterval object that, for an input dataset, partitions the data in a default range of possible clusters (2-8 if not specified), using hierarchical agglomerative clustering following the ward criteria (if not specified). It calculates various CVIs across the resulting clustering, which will help make the decision on the correct number of clusters. The evaluate method is where the

calculations happen and can receive either a n-dimensional array or a distance matrix. **Note:** if given a distance matrix, K-means algorithm cannot be applied because it requires knowledge of the data points features. Moreover, some CVIs that need this information will not be calculated.

c = Clusterval() $\rightarrow$ creates object

c.evaluate(data) $\rightarrow$ evaluates the dataset, calculating the CVIs

In the following subsection we will go over all the possible commands and parameters for the package.

3.2.1 Available commands

The *clusterval* package produces *Clusterval* objects that can be used to perform clustering evaluation of any list-type dataset inputed.

The initialization of the instance with empty parameters means that some default attributes will be assigned, but they can also be specified upon creation. The possible attributes are:

Parameter	**Description**	**Default value**
min_k	Integer that sets the minimum number of clusters to test.	2
max_k	Integer that sets the maximum number of clusters to test.	8
algorithm	String that sets the clustering algorithm to use.	Hierarchical clustering with ward linkage.
bootstrap_samples	Integer that sets the number of bootstrap samples simulated	250.
index	String (or list of strings) containing the CVIs calculate.	All CVIs (tables3.1 and 3.2).

Table 3.3: Available *clusterval* parameters.

The range for the number of clusters to be tested is given by attributes **min_k** and **max_k**, which have default values of 2 and 8, respectively.

The clustering algorithm to be used is given by **algorithm** and the possible values are 'single', 'complete', 'centroid', 'average' and 'ward', for hierarchical clustering, and 'kmeans' for partitional clustering. Hierarchical clustering with ward linkage is the default.

The algorithm runs a bootstrapping simulation (**bootstrap_samples**) to produce the results, ideally it should be the highest possible value, keeping in mind that a higher value means a longer running time. By default it is set to 250 simulations.

Finally, we can set the indices that we want to consider in the evaluation. The options are listed in tables 3.1 and 3.2, and the value should be equal to the ones in the **'Name in clusterval'** column. Note that the desired indices should be inputed as strings or a list of strings, e.g, 'Dunn','S','AR' or, instead, ['Dunn', 'S', 'AR']. Moreover, it is also possible to choose all possible indices ('all'), the subset of external indices ('external') or the subset of internal indices ('internal'). The default value for the attribute **index** is 'all'.

After creating the *clusterval* object with the desired attributes, in order to make clustering evaluation we need to call the **evaluate** method. This method will perform clustering on the dataset that was passed, which can be a **n-dimensional array** or a **distance matrix**. Keeping in mind that when passing a distance matrix, K-means clustering is not possible and some indices that depend on feature knowledge will not be calculated. These indices are *XB, SD, SDbw, PBM and DB.*

From the resulting clustering it will calculate the validation indices. The method can be called like so:

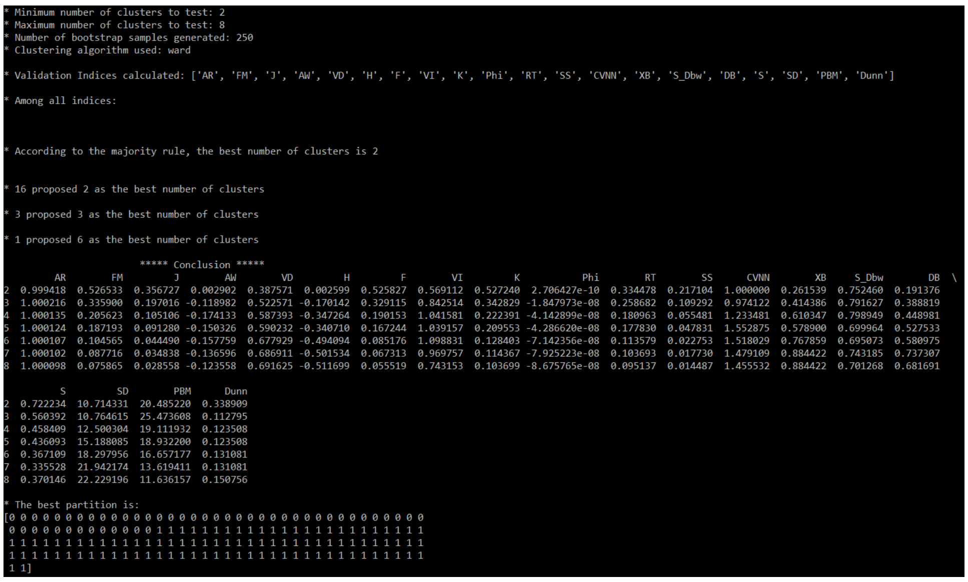
c.evaluate(data)

The method will output the predicted number of clusters, as well as information on the parameter values used.

To see more information about the evaluation there is the command:

print(c.long_info)

And the output will look similar to figure 3.1.



```
* Minimum number of clusters to test: 2
* Maximum number of clusters to test: 8
* Number of bootstrap samples generated: 250
* Clustering algorithm used: ward

* Validation Indices calculated: ['AR', 'FM', 'J', 'AW', 'VD', 'H', 'F', 'VI', 'K', 'Phi', 'RT', 'SS', 'CVNN', 'XB', 'S_Dbw', 'DB', 'S', 'SD', 'PBM', 'Dunn']

* Among all indices:


* According to the majority rule, the best number of clusters is 2


* 16 proposed 2 as the best number of clusters

* 3 proposed 3 as the best number of clusters

* 1 proposed 6 as the best number of clusters

                   ***** Conclusion *****
         AR        FM         J        AW        VD         H         F        VI         K           Phi        RT        SS      CVNN        XB     S_Dbw        DB  \
2  0.999418  0.526533  0.356727  0.002902  0.387571  0.002599  0.525827  0.569112  0.527240  2.706427e-10  0.334478  0.217104  1.000000  0.261539  0.752460  0.191376
3  1.000216  0.335900  0.197016 -0.118982  0.522571 -0.170142  0.329115  0.842514  0.342829 -1.847973e-08  0.258682  0.109292  0.974122  0.414386  0.791627  0.388819
4  1.000135  0.205623  0.105106 -0.174133  0.587393 -0.347264  0.190153  1.041581  0.222391 -4.142899e-08  0.180963  0.055481  1.233481  0.610347  0.798949  0.448981
5  1.000124  0.187193  0.091280 -0.150326  0.590232 -0.340710  0.167244  1.039157  0.209553 -4.286620e-08  0.177830  0.047831  1.552875  0.578900  0.699964  0.527533
6  1.000107  0.104565  0.044490 -0.157759  0.677929 -0.494094  0.085176  1.098831  0.128403 -7.142356e-08  0.113579  0.022753  1.518029  0.767859  0.695073  0.580975
7  1.000102  0.087716  0.034838 -0.136596  0.686911 -0.501534  0.067313  0.969757  0.114367 -7.925223e-08  0.103693  0.017730  1.479109  0.884422  0.743185  0.737307
8  1.000098  0.075865  0.028558 -0.123558  0.691625 -0.511699  0.055519  0.743153  0.103699 -8.675765e-08  0.095137  0.014487  1.455532  0.884422  0.701268  0.681691

          S         SD        PBM      Dunn
2  0.722234  10.714331  20.485220  0.338909
3  0.560392  10.764615  25.473608  0.112795
4  0.458409  12.500304  19.111932  0.123508
5  0.436093  15.188085  18.932200  0.123508
6  0.367109  18.297956  16.657177  0.131081
7  0.335528  21.942174  13.619411  0.131081
8  0.370146  22.229196  11.636157  0.150756

* The best partition is:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1]
```

Figure 3.1: Long information output for *clusterval*

From 3.1 we can obtain a lot of information on the clustering evaluation such as all values for the initialization attributes (**algorithm, min_k, max_k, bootstrap samples and indices to compute**), in addition to the **final selected number of clusters**, the **frequency that a certain number of clusters was selected by a CVI**, the **indices values for each cluster number**, and finally, the **best partition** for the selected number of clusters.

It's also possible to visualize the **dendrogram** produced by the hierarchical clustering [2]. To see the visualization, use the command:

c.plot()

[2]In linux systems, the installation of the library "python3-tk" might be needed.

There is also the possibility of using the package with other clustering algorithms other than the ones included. To do that we make use of the functions: **calculate_external** and **calculate_internal**. First import the functions:

from clusterval import calculate_external, calculate_internal, Clusterval

For external validation it is needed to input two partitions. The first generated by the clustering algorithm on the dataset and the second generated from applying clustering on a random subset of the dataset. Moreover, it is optional to provide the desired CVIs to calculate (see Table 3.1). Following is an example:

calculate_external(partition_C, partition_P, indices=['all'])

The function returns a dictionary mapping CVI to its value.

For internal validation we need to input the clustering structure obtained from the clustering algorithm, these should be in a Python dictionary format where keys are number of clusters and values are the partitions, in list format, e.g., taking $D = \{d_0, d_1, d_2, d_3, d_4\}$, for k=2, the clustering algorithm produces a partition like $[[d_0, d_1, d_2], [d_3, d_4]]$, hence, one entry of the dictionary will be $\{k : [[d_0, d_1, d_2], [d_3, d_4]]\}$. Moreover, it is needed to input the distance (pairwise) matrix for the dataset. If inputing already a distance matrix, please note that this needs to be in a dictionary format with pairs as tuples being the keys and the distance between those pairs, the values, e.g., $d_0, d_1 \in D$, then the dictionary for this pair will be $\{(d_0, d_1) : d(d_0, d_1)\}$.

The library provides a method to calculate the pairwise matrix from the dataset, already producing the mentioned dictionary, the user only needs to add the following in code or python environment:

c = Clusterval()

distance_dict = c._distance _dict(dataset)

With this information it will be possible to calculate the internal CVIs:

calculate_internal(clustering, data=dataset, distance_dict=distance_dict)

3.3 AliClu - clusters visual representation

As described in Section 2.3, AliClu takes a longitudinal dataset and analyses it's clustering structure. The tests done were with Reuma.pt as input, which consists of treatment information on rheumatoid arthritis patients. One example of a cluster formed by AliClu, with 2 elements, can be seen in Table 3.4.

id_patient	coded_sequence
3948652	0.H,187.F,785.Z
69331060	0.F,1109.Z

Table 3.4: Patient's sequences belonging to one cluster after AliClu analysed Reuma.pt

The sequences are interpreted in chain, with each symbol representing a treatment ("Z" meaning the end of treatments) and the numbers the time in days between one treatment and the next. Therefore, the first patient start with treatment "H", which stays for 187 days, after which starts treatment "F", that lasted 785 days and concluded the sequence for this patient. Patient with id=69331060, had only one treatment, "F", in a time interval of 1109 days.

This description follows a chain, that can be translated into a graph scheme, where the nodes are the treatments and the edges direct from one treatment to the next. The edges carry weight that is equal to the time in days from one node to the next.

Our solution is to create a graph for each patient inside each cluster and merge them, resulting in a "average" graph for the specific cluster. We chose to consider this notion of average graph because in theory, the objects in the same cluster share many similarities, therefore, the treatment will be similar. A function "print_nodes" that handles the graph generation was added in AliClu's code and can be seen in `https://github.com/Nuno09/AliClu/blob/master/Code/print_results.py`.

For the cluster represented in Table 3.4, the respective graph is shown in Figure 3.2.

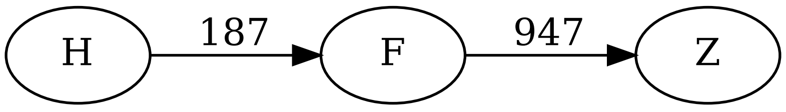


Figure 3.2: Graph representing the cluster in Table 3.4.

Note that, both sequences have a transition from state "F" to the final state "Z", and the average time being 947. Since the first patient start with a transition from "H" to "F", that information is added to the graph.

Chapter 4

Results

In this chapter we will give a demonstration of the library being used to evaluate the clustering results of various synthetic and real-world datasets.

We will first describe the experimental setup for obtaining the results, followed by the results achieved by using synthetic datasets. Next, we also compare the CVIs using 10 real datasets drawn from UCI repository [62]. Finally, we present the results obtained when using AliClu with *clusterval*, to evaluate the Reuma.pt dataset. In any case, it is important to note that results based on real datasets should be analyzed with caution since these datasets are usually intended to be used with supervised learning and, therefore, are not always well adapted to the clustering problem. On the other hand, the synthetic datasets avoid many problems found with real datasets. For instance, in synthetic datasets categories exist independent of human experience and their characteristics can be easily controlled by the experiment designer.

4.1 Experimental setup

The synthetic datastes were created with the goal of covering all possible combinations of 5 factors: number of clusters (**K**), dimensionality of the data (**dim**), **noise**, cluster density (**dens**) and cluster overlapping (**overlap**).

Every dataset generated will consist of at least 200 samples, while the maximum number of samples is set at 500. Datasets will vary depending on the values of the 5 factors listed before. They can have either 2, 4 or 8 clusters. Dimensionality will take values 2, 4 and 8, while noise can be at 10% level. Clusters can also be of different sizes, we can define a ratio of 1 for symmetric clusters or a ratio of 4, where one cluster will be 4 times more dense than the rest. In the case of testing for overlap between clusters we will vary the standard deviation of the dataset, a value of 1.5 (good separation) or a value of 5.0 (overlap) are considered. Table 4.1 shows the possible values.

Parameter	Value
Nº samples	200-500
K	2,4,8
dim	2,4,8
noise	0, 0.1
dens	1, 4
overlap	1.5, 5.0

Table 4.1: Parameter values used for the synthetic datasets generation.

Moreover, we will use *clusterval* with 4 different clustering algorithms, 3 hierarchical ones (single, complete and ward) and 1 partitional (kmeans). Although the package also works with centroid and average linkages, those were not considered because readability and presentation of six algorithms would be difficult.

Considering all factors we get 72 possible combinations. From each we will create 5 datasets, which gives us 360 synthetic datasets. Multiplying by 4 possible clustering algorithms we end up with **1440 configurations** to be considered.

Figure 4.1 shows one example of 4 two-dimensional datasets we have used.

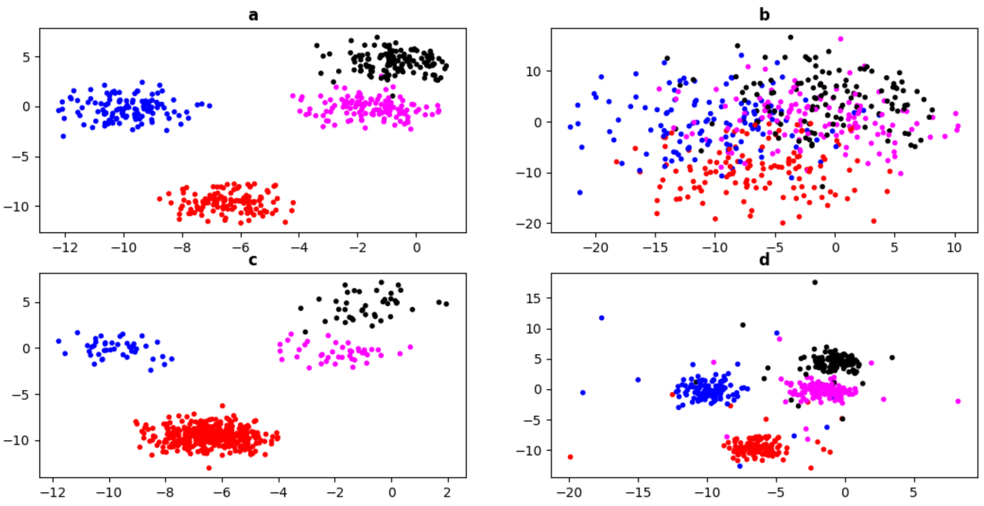


Figure 4.1: Two-dimensional plots of four synthetic datasets used in the experiment. **(a)** Shows a "normal" dataset with no cluster overlap, no density asymmetry and no noise. **(b)** Shows a similar dataset with high cluster overlap. **(c)** Shows a dataset with cluster density asymmetry. **(d)** Shows a dataset with noise.

The 10 real-world datasets compared are listed in Table 4.2. In this case, since we cannot change characteristics of the datasets, comparison is based on the algorithms used for clustering, hence we consider **40 configurations**.

Dataset	Nº instances	Nº attributes	Classes
Breast tissue	106	9	6
Ecoli	336	7	8
Glass	214	9	7
Haberman	306	3	2
Iris	150	4	3
Parkinsons	195	22	2
Vehicle	846	18	4
Vertebral column	310	6	3
Wine	178	13	3
Yeast	1484	8	10

Table 4.2: Real datasets from UCI repository.

4.2 Synthetic datasets

The overall results for the synthetic datasets are shown in Figure 4.2. The figure shows the percentage of correct results (from the 1440 tests) for the number of clusters from each validation metric, sorted from best to worst performing. From the figure, we can see *Xie-Beni* achieves the best result, 45% success rate. *PBM*, *Dunn* and *CVNN* also show a similar result, 44%, 42% and 41%, respectively. These four metrics are the only that achieve a higher result than 40%. It is noticeable to see that internal indicies perform better than most external indices, with the six best metrics being six out of eight internal validation indices. It is also interesting to see that all external CVIs have very similar results, with the exception of the *Adjusted Rand* which holds the lowest success rate from all CVIs (6%).

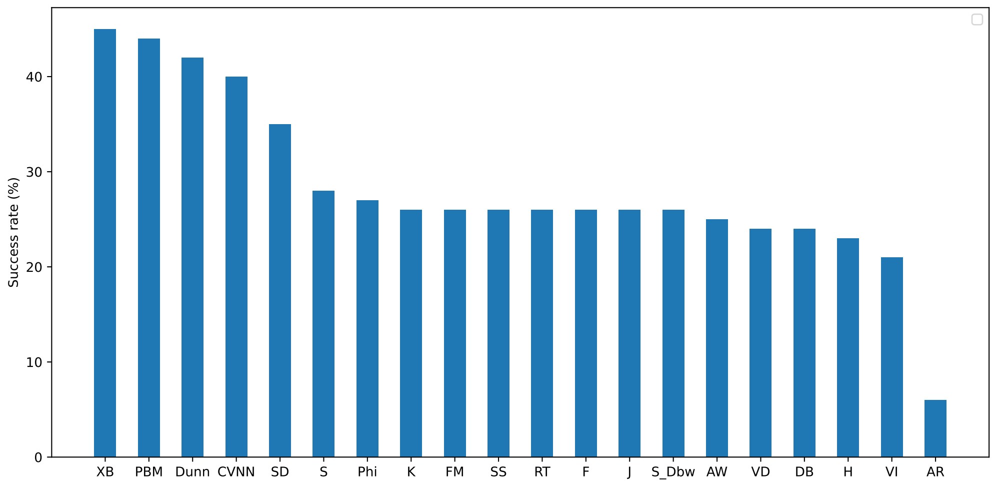


Figure 4.2: Overall results for the test with synthetic datasets.

Following we will show a similar figure for every parameter that was experimented. We will keep the figures ordered by the CVIs shown in Figure 4.2, so that we can see if a parameter influences the success rate of the CVIs, which would be signaled by an increasing graph.

Firstly, let's consider Figure 4.3 with respect to the number of clusters in the dataset. We can observe that all CVIs, except *AR*, do better when the number of clusters is two, which is expected. The average result for k=2 is 71,5%, which drops to 11,1% for k=4 and to 5.4% for k=8. Clearly, the CVIs have difficulties guessing the right number of clusters when this parameter increases and we can not choose one of the metrics as the best in this regard, although internal indices perform relatively better.

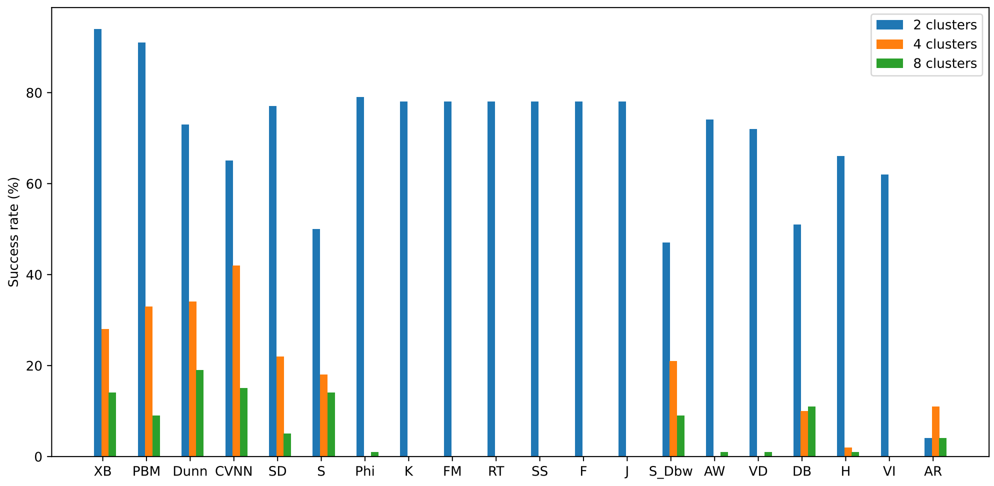


Figure 4.3: Results for the synthetic datasets, by cluster number.

With respect to dimensionality (Figure 4.4), we can notice an interesting trend. In general, the results get better with an increase in the number of features in the dataset. All indices, except of *SDbw* and *AR*, perform better with dim=8, which is unexpected since complexity is being added. Furthermore, we would like to highlight *PBM*, which seems to not be very sensitive to the number of features present, with 38% success rate for dim=2, 47% for dim=4 and 48% when the number of features is 8. On average, the success rate for dim=2 is 22.9%, for dim=4 it is 27.4% and for dim=8 it is 37.7%. We can also conclude that dimensionality does not affect significantly the overall trend.

Focusing on the results from the noise level experiment (Figure 4.5), we can say that the addition of noise to the dataset does not affect the CVIs performance severely where we see a drop on the average success rate of 2.7% when introducing noise (from 30.7% to 28%). In fact, some indices (*XB, CVNN, AR*) look to be more successful with a 10% noise addition. The overall trend is also kept.

With respect to the density of the clusters, Figure 4.6 shows that it does not affect with severity the results, with some exceptions (*PBM, SD*). Like with noise, a small drop happens when introducing cluster density, in this case of 4.5% (31.6% to 27.1%). The overall trend is also mostly kept.

Focusing on cluster overlap, Figure 4.7 tells us that when we compare well separated clusters with overlapped clusters the average results decrease by 13.3% (from 36% to 22.7%). Notable exception are the Van Dongen index (VD), which is barely affected by the change in the cluster structure, Variation

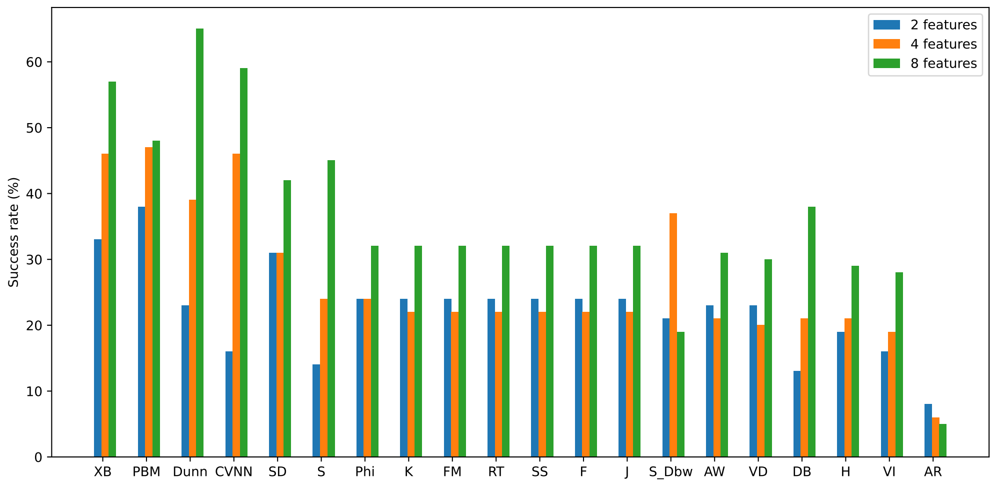


Figure 4.4: Results for the synthetic datasets, by dimensionality.

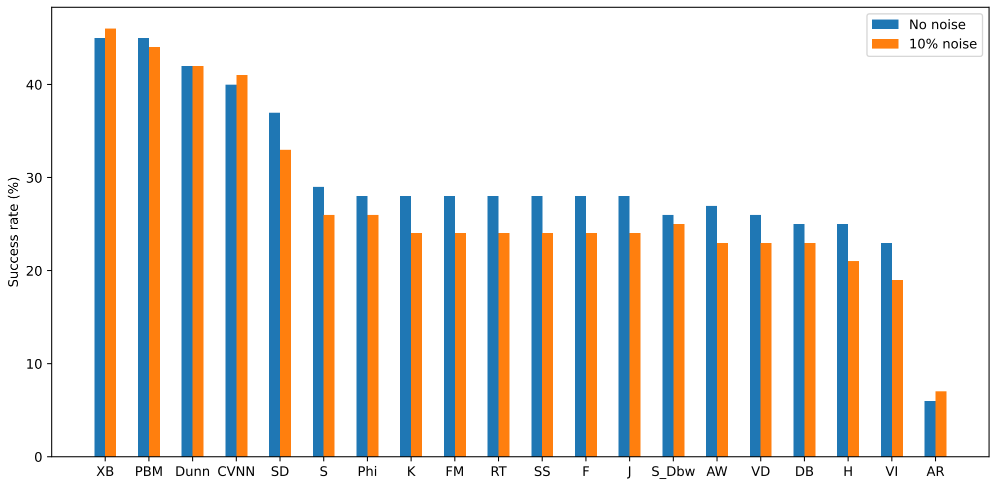


Figure 4.5: Results for the synthetic datasets, by noise level.

of Information (VI) and Adjusted Rand (AR) that performed slightly better on datasets with overlap. Internal indices continue to perform better than external ones and the trend is more or less unaffected.

Finally, Figure 4.8 shows how the clustering algorithm used in the experiment affects the performance of the indices used. Although no clear pattern can be found, it seems that the choice of algorithm does not have a big effect on the overall results since the decreasing of the graph is still maintained, with the exception of Davies-Bouldin (DB). The best average results are obtained when using the K-means algorithm (32.5%), followed by ward algorithm (31.5%), complete (26.8%) and single (26.5%). Moreover we can conclude that single algorithm produces weak results for Silhouette (S) and Davies-Bouldin (DB), when compared with the other algorithms.

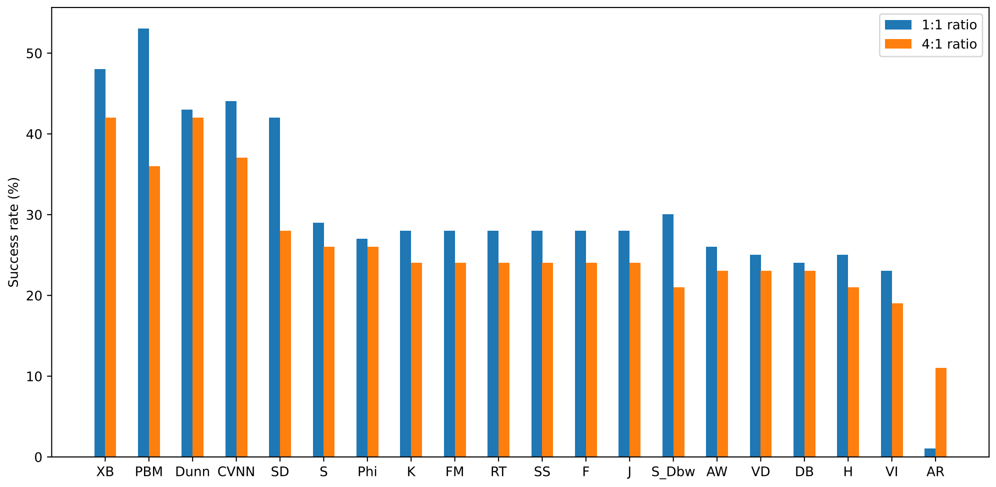


Figure 4.6: Results for the synthetic datasets, by cluster density.

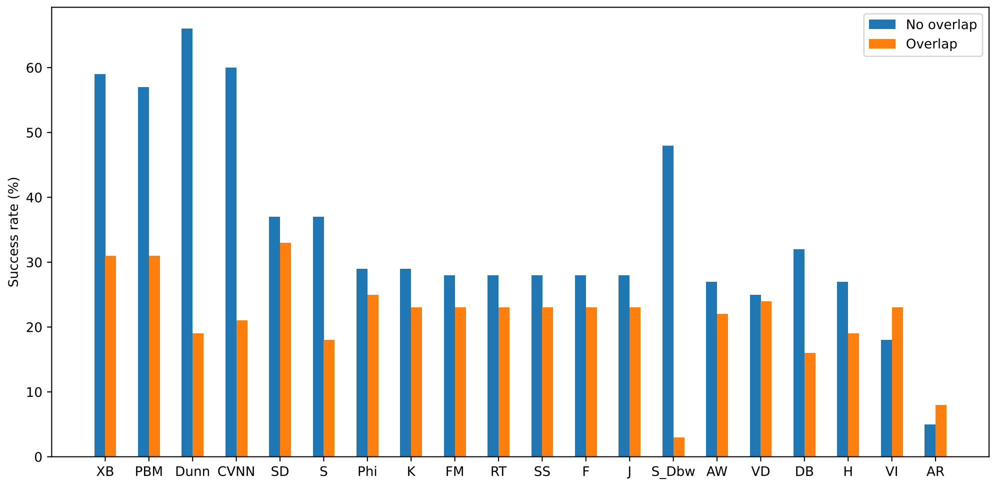


Figure 4.7: Results for the synthetic datasets, by cluster overlap.

4.3 Real-World datasets

In this section we show the results obtained for 10 real datasets (main characteristics in Table 4.2). We followed a similar style to the one from synthetic datasets, but since we have no control over the design, only one factor is analysed: the clustering algorithm used.

First, in Figure 4.9 we show the overall results for the real datasets. A quick comparison with the synthetic datasets (Figure 4.2) shows that there has been some changes in the rankings of the indices, the most glaring one being *PBM* going from second to last in order of success rate. Notably, the external indices in general maintain the same positions and results, while the internal validation metrics continue achieving the best results, except the already mentioned *PBM* nut also *CVNN* and *Dunn*. It is of note that the Silhouette performance remained consistent, with 28% for synthetics and 30% for real datasets.

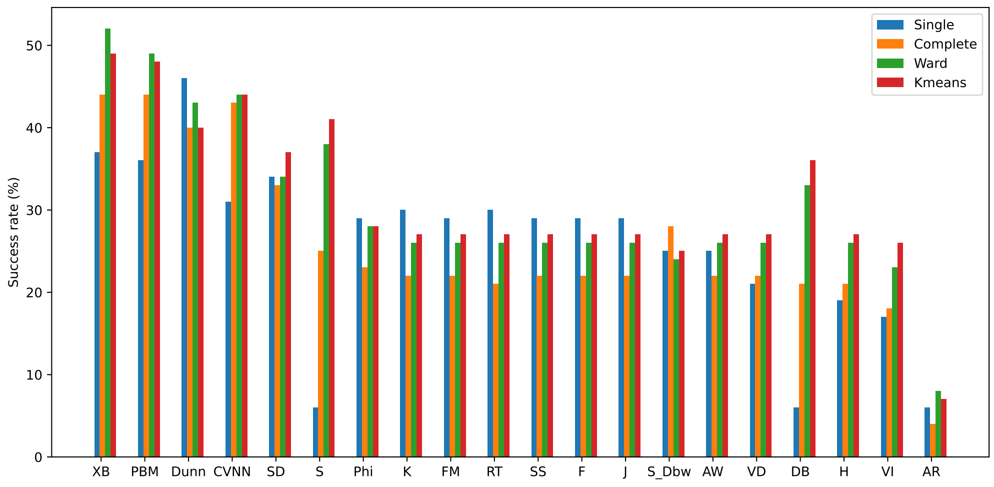


Figure 4.8: Results for the synthetic datasets, by clustering algorithm.

S jumped from rank 6th for synthetics to 1st for real world datasets. *DB* index had a similar behaviour and is 2nd for real datasets. While in synthetic results we see a 29.3% average success rate, for real datasets it drops to 20.1%.

Regarding the clustering algorithm used, Figure 4.10 shows that the results follow the overall trend. Noticeably, the algorithm chosen does not carry any influence when using external CVIs, with exception of Hubert statistic. *PBM* index could only guess correctly the number of clusters when using single linkage hierarchical clustering. Furthermore, ward is the algorithm that provides the best results on average with 22% success rate, followed by k-means (20.5%), complete (19.5%) and single (18.5%).

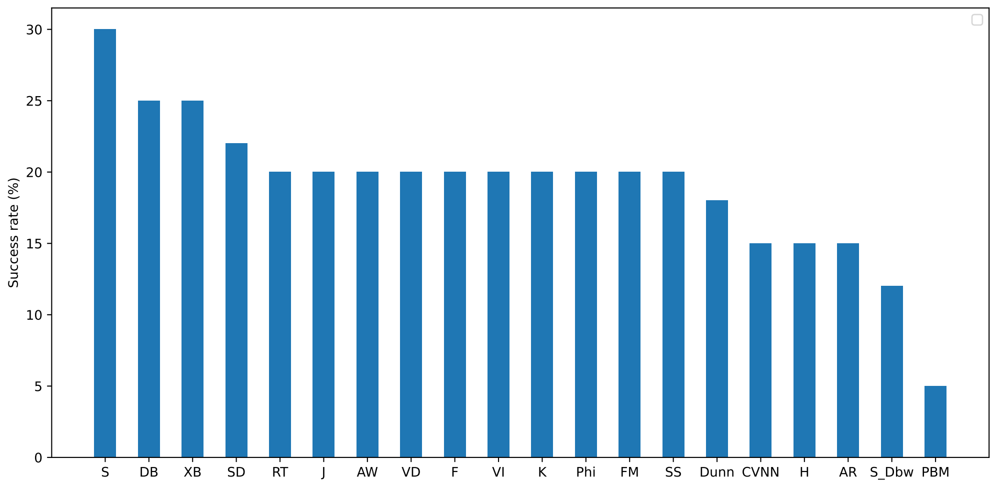


Figure 4.9: Overall results for the real-world datasets experiment.

The results look to be rather poor, but perhaps looking at the average values of each index will gives some more insight. For example, Table 4.3 shows the index values for each real datatset, with indication

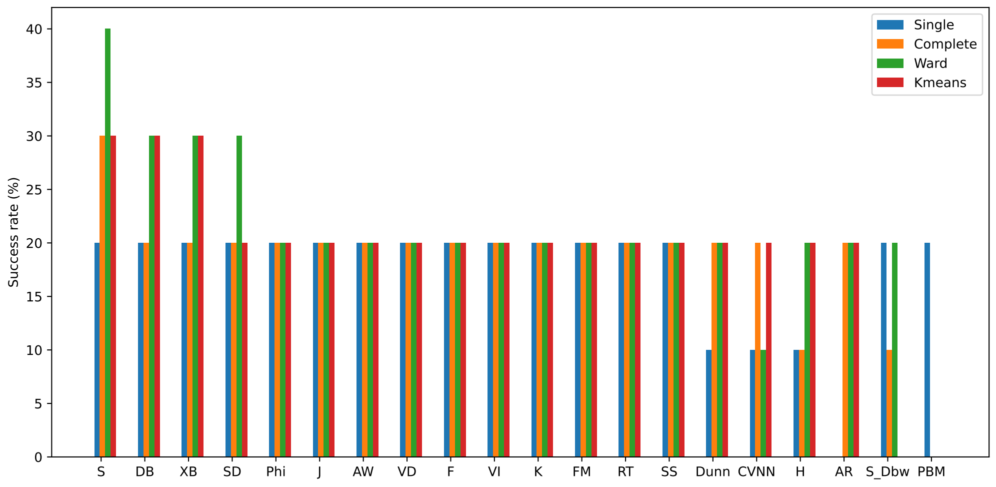


Figure 4.10: Results for the real-world datasets, by clustering algorithm.

of the number of clusters selected (NC), the index value for that NC and the value for the correct NC. In Table 4.3 we can see the results when using single-linkage hierarchical clustering. A pattern that can be seen in all datasets is that, with very few exceptions, the external indices chose NC=2, and upon comparing the values of the NC chosen and the value for the correct NC we see that both are either equal or very close together, but since the algorithm is choosing the first best value it will always choose 2. In the case of internal indices, the true value and the chosen value are also many times close. In the case of single-linkage the indices that guessed correctly were DB for Glass dataset, Haberman almost all (except CVNN, SDbw, DB and S, all with close values), H, Phi, CVNN and PBM guessed correctly for Iris and for Parkinsons all, except CVNN and PBM, chose the correct NC. Also stands out the high values given by PBM index for some of the datasets. In this cases the the clusters compactness is rather low and clusters are also very far from each other, resulting in high values for E_C and E_T for Eq. (2.52). Moreover, the maximum distance between clusters is also very high, contributing to the values observed.

Following we will present a similar table for each clustering algorithm results.

Table 4.4 shows the index results for complete-linkage. Considering all datasets, external indices choose NC=2 as the best result, with the exception of AR. Internal indices give similar results to single-linkage. For the datasets with NC=2 (Haberman and Parkinsons) most indices guessed correctly, much thanks to the bias shown by external indices. AR, CVNN and S guessed correctly for Iris. In the case of the Vehicles dataset, AR gives the same value for NC=3 and the true value (NC=4). SD chooses correctly the NC for the Vertebral dataset and AR for Wine dataset. For the Yeast dataset (NC=10), AR has the same value for NC=5 and the true value, and CVNN has very close values between NC=11 and NC=10.

Table 4.5 shows the results for Ward-linkage hierarchical clustering. We can see again that the majority of external CVIs will select NC=2 for all datasets, exceptions are AR, VD and VI. On the other hand, internal CVIs perform generally better and many times, when not choosing the correct number of clusters, the value given and the true value are very close. SDbw and S guess correctly for Glass, most

Table 4.3: Results for single-linkage hierarchical clustering.

	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	XB	SDbw	DB	S	SD	PBM	Dunn	
Breast tissue NC=6	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	5.0	2.0	2.0	2.0	2.0	4.0	3.0	2.0	NC
	1.0	1.0	1.0	1.0	0.0131	1.0	1.0	0.0484	1.0	$0.0346e^{-4}$	1.0	1.0	1.5919	0.0358	0.8558	0.0179	0.9753	0.002	15817895636.7681	3.3504	Value
	1.0	1.0	1.0	1.0	0.0495	1.0	1.0	0.196	1.0	$0.0076e^{-4}$	1.0	1.0	1.592	1.5207	1.0012	0.0667	0.8849	0.0037	4596788770.8825	0.1439	True value
Ecoli NC=8	2.0	2.0	2.0	2.0	2.0	6.0	2.0	2.0	2.0	2.0	2.0	2.0	13.0	2.0	13.0	12.0	12.0	2.0	2.0	2.0	NC
	1.0	0.9959	0.9919	0.9903	0.0067	0.7942	0.9959	0.0326	0.996	$0.0519e^{-6}$	0.9843	0.9839	1.9654	0.5005	0.9855	0.213	0.8381	26.1205	0.1575	0.392	Value
	1.0	0.9741	0.9491	0.947	0.0319	0.7564	0.9739	0.1389	0.9743	$0.0087e^{-6}$	0.9101	0.9031	1.9725	1.0669	0.9868	0.2585	0.7782	40.2984	0.0213	0.2315	True value
Glass NC=7	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	8.0	7.0	8.0	2.0	2.0	2.0	NC
	1.0	1.0	1.0	1.0	0.0049	1.0	1.0	0.0235	1.0	$0.0364e^{-5}$	1.0	1.0	1.0	0.2769	0.9533	0.064	0.9133	1.6232	13.881	0.4934	Value
	1.0	0.9862	0.9727	0.9736	0.0287	0.8655	0.9862	0.1314	0.9863	$0.0061e^{-5}$	0.9514	0.9468	1.9347	0.4086	0.9562	0.064	0.9123	2.4593	3.7762	0.2356	True value
Haberman NC=2	2.0	2.0	2.0	2.0	2.0	12.0	2.0	2.0	2.0	2.0	2.0	2.0	11.0	2.0	7.0	6.0	7.0	2.0	2.0	2.0	NC
	1.0	0.9956	0.9912	0.9955	0.0047	0.8226	0.9956	0.0257	0.9956	$0.081e^{-6}$	0.9827	0.9825	1.9338	0.2333	0.9771	0.06	0.8884	2.188	626.6558	0.2734	Value
	1.0	0.9956	0.9912	0.9955	0.0047	0.7172	0.9956	0.0257	0.9956	$0.081e^{-6}$	0.9827	0.9825	2.0	0.2333	0.9939	0.1166	0.8328	2.188	626.6558	0.2734	True value
Iris NC=3	5.0	2.0	2.0	2.0	2.0	3.0	2.0	8.0	2.0	3.0	2.0	2.0	3.0	2.0	8.0	2.0	2.0	2.0	3.0	2.0	NC
	1.0017	0.5267	0.3569	0.0029	0.3873	0.0026	0.526	0.4385	0.5275	$0.0027e^{-7}$	0.3345	0.2172	0.9676	0.2617	0.7422	0.1918	0.7219	10.3191	23.095	0.3389	Value
	0.9992	0.5177	0.349	0.0028	0.3924	0.0026	0.5174	0.5808	0.5181	$0.0027e^{-7}$	0.3345	0.2114	0.9676	0.3997	0.7941	0.287	0.6711	11.1153	23.095	0.1691	True value
Parkinsons NC=2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	9.0	2.0	2.0	2.0	2.0	2.0	2.0	4.0	NC
	1.0	1.0	1.0	1.0	0.011	1.0	1.0	0.0419	1.0	$0.0432e^{-5}$	1.0	1.0	1.7793	0.2262	0.989	0.1131	0.8293	0.1467	37883.3426	0.3069	Value
	1.0	1.0	1.0	1.0	0.011	1.0	1.0	0.0419	1.0	$0.0432e^{-5}$	1.0	1.0	2.0	0.2262	0.989	0.1131	0.8293	0.1467	37883.3426	0.2391	True value
Vehicle NC=4	2.0	2.0	2.0	2.0	2.0	6.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	2.0	10.0	18.0	10.0	2.0	2.0	7.0	NC
	1.0	0.9952	0.9905	0.9914	0.0052	0.7842	0.9952	0.0273	0.9952	$0.0143e^{-7}$	0.9813	0.9811	0.9998	0.7688	1.6267	0.2925	0.7308	0.2336	11779.3024	0.0879	Value
	1.0	0.9887	0.9775	0.9737	0.0122	0.689	0.9886	0.0641	0.9887	$0.0057e^{-7}$	0.9571	0.9559	1.9998	2.1053	3.5002	0.6072	0.3905	0.288	4069.1276	0.0711	True value
Vertebral NC=3	2.0	2.0	2.0	2.0	2.0	3.0	2.0	2.0	2.0	2.0	2.0	2.0	12.0	2.0	11.0	2.0	2.0	2.0	2.0	2.0	NC
	1.0	0.9956	0.9913	0.9876	0.0044	0.7085	0.9956	0.0243	0.9957	$0.0796e^{-6}$	0.9829	0.9828	1.9261	0.0988	0.9706	0.0494	0.9304	0.4059	44659.3307	1.7066	Value
	1.0	0.9912	0.9825	0.9816	0.009	0.7085	0.9912	0.0493	0.9912	$0.0394e^{-6}$	0.9662	0.9655	1.9915	0.3237	0.9796	0.1079	0.8436	0.4178	20325.9782	0.2437	True value
Wine NC=3	7.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	NC
	1.0029	0.9921	0.9843	0.9859	0.0141	0.7665	0.9921	0.0568	0.9921	$0.0341e^{-5}$	0.9704	0.9691	1.4262	0.2724	0.9905	0.1362	0.7712	0.0856	231323.8092	0.105	Value
	1.0	0.9496	0.9027	0.8968	0.0503	0.6167	0.9488	0.1749	0.9504	$0.0082e^{-5}$	0.8393	0.8229	1.9265	1.3147	2.6186	0.1666	0.7389	0.1141	132724.6337	0.0684	True value
Yeast NC=10	2.0	2.0	2.0	2.0	2.0	26.0	2.0	2.0	2.0	2.0	2.0	2.0	6.0	2.0	27.0	27.0	27.0	2.0	3.0	2.0	NC
	1.0	0.9873	0.9747	0.9749	0.0102	0.6525	0.9872	0.0487	0.9874	$0.000674e^{-7}$	0.9515	0.9507	0.9684	0.5838	0.9749	0.1614	0.814	29.9106	0.0969	0.3645	Value
	1.0	0.9718	0.9445	0.9444	0.025	0.6404	0.9714	0.1286	0.9721	$0.000274e^{-7}$	0.8992	0.8948	1.9672	0.8344	1.0943	0.3759	0.6631	39.4823	0.0163	0.2576	True value

Table 4.4: Results for complete-linkage hierarchical clustering.

	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	XB	SDbw	DB	S	SD	PBM	Dunn	
Breast tissue NC=6	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	7.0	2.0	2.0	2.0	2.0	3.0	4.0	2.0	NC
	1.0	1.0	1.0	1.0	0.0126	1.0	1.0	0.0471	1.0	$0.0353e^{-4}$	1.0	1.0	1.2659	0.0358	0.8558	0.0179	0.9753	0.001	30737824621.0059	3.3504	Value
	0.9999	0.7781	0.6365	0.5671	0.2276	0.5807	0.777	0.4297	0.7792	$0.00251e^{-4}$	0.6569	0.4683	1.2668	0.5861	3.9167	0.2535	0.6474	0.0017	21169659897.027	0.0852	True value
Ecoli NC=8	3.0	2.0	2.0	2.0	2.0	2.0	2.0	13.0	2.0	2.0	2.0	2.0	3.0	2.0	9.0	2.0	2.0	2.0	2.0	2.0	NC
	1.0001	0.5558	0.3839	0.0554	0.3541	0.0488	0.5547	0.2602	0.5569	$0.00197e^{-7}$	0.3556	0.2376	0.8279	0.5448	1.2501	0.4846	0.4309	23.8457	0.1525	0.1131	Value
	1.0	0.2491	0.1272	-0.0878	0.5652	-0.1818	0.2253	0.8937	0.2757	-0.00088^{-7}	0.2433	0.068	1.4669	0.9528	1.2607	0.7665	0.3208	33.3466	0.0481	0.0867	True value
Glass NC=7	2.0	2.0	2.0	2.0	2.0	4.0	2.0	2.0	2.0	2.0	2.0	2.0	6.0	2.0	2.0	2.0	10.0	2.0	3.0	8.0	NC
	1.0	0.9519	0.907	0.8952	0.049	0.7935	0.9512	0.1642	0.9527	$0.0378e^{-6}$	0.8447	0.8298	0.6287	0.4125	1.0681	0.3683	0.5204	3.0753	10.6288	0.2335	Value
	1.0	0.8786	0.7832	0.7557	0.1701	0.7324	0.8783	0.4103	0.8789	$0.0185e^{-6}$	0.767	0.6441	1.4854	0.825	1.3606	0.6064	0.4182	4.2837	7.081	0.2163	True value
Haberman NC=2	6.0	2.0	2.0	2.0	2.0	3.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	2.0	2.0	2.0	2.0	4.0	4.0	2.0	NC
	1.0001	0.9054	0.8256	0.8028	0.11	0.5794	0.904	0.2534	0.9069	$0.0606e^{-7}$	0.7574	0.7046	0.7901	0.503	1.0583	0.4831	0.414	1.2729	273.096	0.0674	Value
	1.0	0.9054	0.8256	0.8028	0.11	0.5414	0.904	0.2534	0.9069	$0.0606e^{-7}$	0.7574	0.7046	1.05	0.503	1.0583	0.4831	0.414	1.2773	156.1184	0.0674	True value
Iris NC=3	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	2.0	7.0	2.0	3.0	4.0	4.0	8.0	NC
	1.0002	0.5857	0.4139	0.1537	0.3479	0.1464	0.5853	0.5486	0.5861	$0.0152e^{-6}$	0.4019	0.2611	0.9211	0.4077	0.6644	0.3283	0.5534	11.0139	22.7123	0.1529	Value
	1.0002	0.2868	0.1638	-0.2079	0.5363	-0.2936	0.2814	0.8831	0.2924	$-0.0317e^{-6}$	0.2108	0.0893	0.9211	0.4769	0.8796	0.3764	0.5534	11.0604	18.2166	0.1033	True value
Parkinsons NC=2	5.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	6.0	2.0	7.0	2.0	2.0	2.0	3.0	2.0	NC
	1.0003	0.9396	0.884	0.877	0.0569	0.6304	0.9384	0.186	0.9407	$0.0457e^{-6}$	0.8111	0.7922	0.4601	0.2573	2.1823	0.2369	0.6623	0.1088	43628.8918	0.3069	Value
	1.0	0.9396	0.884	0.877	0.0569	0.6304	0.9384	0.186	0.9407	$0.0457e^{-6}$	0.8111	0.7922	1.0	0.2573	5.9713	0.2369	0.6623	0.1088	39996.9435	0.3069	True value
Vehicles NC=4	3.0	2.0	2.0	2.0	2.0	2.0	2.0	20.0	2.0	2.0	2.0	2.0	13.0	2.0	12.0	2.0	2.0	2.0	5.0	20.0	NC
	1.0	0.664	0.4954	0.2806	0.2707	0.2384	0.6621	-0.1637	0.6659	$0.000247e^{-7}$	0.4531	0.3298	0.5297	0.2417	5.5044	0.2172	0.6634	0.2151	194017.3308	0.136	Value
	1.0	0.3048	0.1769	-0.2064	0.4928	-0.2757	0.3003	0.9488	0.3093	$-0.000283e^{-7}$	0.2181	0.0971	0.8529	0.5687	5.7079	0.3816	0.4944	0.2232	182207.1983	0.0417	True value
Vertebral NC=3	5.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	12.0	2.0	2.0	2.0	2.0	3.0	2.0	2.0	NC
	1.0001	0.9952	0.9904	0.9939	0.0261	0.7628	0.9952	0.0648	0.9952	$0.0652e^{-6}$	0.9829	0.9811	1.471	0.0988	0.9835	0.0494	0.9304	0.354	44659.3307	1.7066	Value
	1.0	0.7893	0.6498	0.5506	0.191	0.433	0.7868	0.4152	0.7919	$0.00283e^{-6}$	0.5891	0.4831	1.8569	0.4455	2.5764	0.3222	0.6195	0.354	28377.3439	0.0843	True value
Wine NC=3	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	7.0	2.0	2.0	2.0	2.0	2.0	8.0	9.0	NC
	1.0003	0.6102	0.4362	0.1426	0.2982	0.1138	0.6068	0.5134	0.6137	$0.06183e^{-7}$	0.388	0.2795	0.3738	0.2333	0.8684	0.2273	0.6588	0.0846	885517.8082	0.0656	Value
	1.0003	0.3336	0.199	-0.2721	0.4784	-0.2935	0.331	0.8177	0.3363	$-0.1554e^{-7}$	0.2127	0.1108	0.636	0.4692	1.1407	0.3665	0.5381	0.0916	428035.2159	0.0227	True value
Yeast NC=10	5.0	2.0	2.0	2.0	2.0	2.0	2.0	27.0	2.0	2.0	2.0	2.0	11.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	NC
	1.0	0.9971	0.9942	0.9961	0.0211	0.7616	0.9971	-0.5733	0.9971	$0.00191e^{-7}$	0.9893	0.9884	0.8363	0.2952	1.1252	0.2095	0.7043	37.7372	0.1924	0.3454	Value
	1.0	0.1628	0.0844	-0.3224	0.5921	-0.5375	0.1553	1.1909	0.1709	$-0.000062e^{-7}$	0.1219	0.0441	0.8486	1.5559	1.3928	0.9663	0.3276	62.3918	0.0278	0.0539	True value

CVIs are correct about Haberman and Parkinsons, AR, CVNN and PBM output the correct NC for Iris, regarding Vertebral column dataset, XB, DB, S, SD, PBM and Dunn give the correct result, for Wine dataset AR and SDbw are correct, finally for Yeast dataset, AR and CVNN are the two CVIs that come closer to NC=10, with the first having equal value (NC=4) and true value, and CVNN choosing NC=9 but having very close value to the true value.

Finally, Table 4.6 shows the results for K-means clustering algorithm. Again we can observe that the majority of external indices chooses NC=2, and internal CVIs come closer to true values. For datasets Haberman and Parkinsons, the overall results continue to be good, due to the fact that external CVIs show a bias for NC=2. AR, CVNN, SD, PBM and Dunn guess the correct number for Iris. Regarding the Vertebral column dataset, XB, DB, S and PBM output the correct number of clusters, for Wine dataset only AR, and, finally, for Yeast dataset, CVNN is able to guess the correct number of clusters.

4.4 Reuma dataset

In this section, we repeat the experiment I that is described in the paper about AliClu [1], which consists on performing alignment with sequences created with the patient treatment history found on Reuma.pt dataset. The goal of this experiment is to try to find similarities between patients based on different variables and, ultimately, be able to stratify them according to their similarity regarding the disease and treatment evolution. The difference in our replication of the experiment is that we have more clustering validation indices (CVIs) that could help us decide on the best clustering structure and an automatic function that generates the chain of treatments for each cluster. Like the previous work, the most relevant results will be presented.

Reuma.pt [63] is a Portuguese rheumatology database developed by the Portuguese Society of Rheumatology (SPR) that follows rheumatics patients nationwide, with the goal of monitoring disease progression and assuring treatment efficacy and safety. It includes mostly patients with rheumatoid arthritis (RA), ankylosing spondylitis (AS), psoriatic arthritis (PsA) and juvenile idiopathic arthritis (JIA) being treated with biological therapies or receiving synthetic disease modifying anti-rheumatic drugs (DMARDs). Data has been gathered since 2008 in 21 Rheumatology Departments assigned to the Portuguese National Health Service, 2 Military Hospitals (Lisboa and Porto), 1 public-private institution and 6 private centers. Patient data is being collected in a regular basis, more or less every three/six months.

The dataset provided for this study comes from two centres and only contains patients diagnosed with RA. In total there is information about 426 patients and 9305 appointments. For each patient several variables are measured and calculated during time such as age, disease duration, gender, medical scores, patient questionnaires, active therapies and others. More detailed information on the dataset can be found in [1], or even more in [63].

The algorithms in AliClu were used to create the sequences and the parameters used for the clustering processed were bootstrap_samples=250, min_k=2, max_k=20, all CVIs will be considered and we will vary the algorithm used. If any parameteres are changed it will be indicated.

From previous work we know that negative gap values do not produce good results, therefore we focus

Table 4.5: Results for ward-linkage hierarchical clustering.

	AR	**FM**	**J**	**AW**	**VD**	**H**	**F**	**VI**	**K**	**Phi**	**RT**	**SS**	**CVNN**	**XB**	**SDbw**	**DB**	**S**	**SD**	**PBM**	**Dunn**	
Breast tissue NC=6	7.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	7.0	2.0	2.0	2.0	2.0	3.0	7.0	2.0	NC
	1.0004	1.0	1.0	1.0	0.0143	1.0	1.0	0.0519	1.0	$0.0330e^{-4}$	1.0	1.0	1.1184	0.0358	0.8558	0.0179	0.9753	0.0018	42484339254.6111	3.3504	Value
	0.999	0.5617	0.3863	0.2177	0.335	0.2685	0.5563	0.5576	0.5672	$0.0012e^{-4}$	0.4717	0.2401	1.1793	0.4037	3.8481	0.2445	0.6768	0.002	36464035150.7726	0.1727	True value
Ecoli NC=8	3.0	2.0	2.0	2.0	2.0	2.0	2.0	13.0	2.0	2.0	2.0	2.0	7.0	2.0	13.0	2.0	2.0	2.0	2.0	13.0	NC
	1.0	0.5643	0.3918	0.0649	0.3408	0.0562	0.563	0.2873	0.5655	$0.00227e^{-7}$	0.3591	0.2437	0.7881	0.5652	1.3238	0.4891	0.4288	21.4332	0.1456	0.1071	Value
	1.0	0.1073	0.0444	-0.1368	0.696	-0.4526	0.085	1.1473	0.1355	$-0.000262e^{-7}$	0.1224	0.0227	1.1144	1.6949	1.5632	0.9828	0.231	37.2734	0.0408	0.0817	True value
Glass NC=7	9.0	2.0	2.0	2.0	3.0	4.0	2.0	2.0	2.0	2.0	2.0	2.0	5.0	3.0	7.0	2.0	7.0	3.0	6.0	4.0	NC
	1.0001	0.8967	0.8121	0.7869	0.1502	0.7369	0.8963	0.3412	0.8972	$0.0193e^{-6}$	0.7717	0.6837	0.7031	0.6681	1.0301	0.5887	0.4597	7.5725	10.005	0.1335	Value
	0.9999	0.5926	0.4184	0.2493	0.3071	0.2916	0.5896	0.6234	0.5956	$0.00743e^{-6}$	0.4805	0.2648	0.9702	1.1535	1.0301	0.6527	0.4597	8.772	9.0722	0.0884	True value
Haberman NC=2	3.0	2.0	2.0	2.0	2.0	2.0	2.0	12.0	2.0	2.0	2.0	2.0	4.0	2.0	9.0	2.0	2.0	2.0	6.0	6.0	NC
	1.0001	0.5356	0.3646	0.0087	0.3878	0.0075	0.5339	0.4239	0.5373	$0.000448e^{-7}$	0.3367	0.2232	0.7231	0.5988	1.1163	0.4777	0.3976	1.7637	227.2767	0.0464	Value
	0.9998	0.5356	0.3646	0.0087	0.3878	0.0075	0.5339	0.5653	0.5373	$0.000448e^{-7}$	0.3367	0.2232	1.0294	0.5988	1.3791	0.4777	0.3976	1.7637	126.2456	0.0221	True value
Iris NC=3	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	2.0	6.0	2.0	2.0	2.0	3.0	2.0	NC
	1.0002	0.5264	0.3566	0.0029	0.3876	0.0026	0.5257	0.5686	0.5271	$0.00267e^{-7}$	0.3345	0.217	0.9743	0.2617	0.697	0.1918	0.7219	10.7094	25.3669	0.3389	Value
	1.0002	0.3352	0.1964	-0.1186	0.524	-0.17	0.3283	0.8428	0.3422	$-0.185e^{-7}$	0.2587	0.1089	0.9743	0.4144	0.7918	0.3892	0.5602	10.7598	25.3669	0.1128	True value
Parkinsons NC=2	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	2.0	4.0	2.0	2.0	2.0	4.0	2.0	NC
	1.0001	0.9463	0.8966	0.8911	0.0563	0.6704	0.9455	0.182	0.9472	$0.0481e^{-6}$	0.8316	0.8126	0.4577	0.2325	1.7744	0.2215	0.694	0.108	50462.6458	0.2643	Value
	1.0	0.9463	0.8966	0.8911	0.0563	0.6704	0.9455	0.182	0.9472	$0.0481e^{-6}$	0.8316	0.8126	1.006	0.2325	5.941	0.2215	0.694	0.108	42827.9141	0.2643	True value
Vehicles NC=4	3.0	2.0	2.0	2.0	2.0	2.0	2.0	20.0	2.0	2.0	2.0	2.0	5.0	2.0	13.0	2.0	2.0	3.0	5.0	8.0	NC
	1.0	0.5648	0.3928	0.0834	0.3499	0.0749	0.5639	-0.2892	0.5657	$0.0000751e^{-7}$	0.3678	0.2445	0.8188	0.2907	5.6209	0.2411	0.6309	0.1892	218169.7858	0.0805	Value
	1.0	0.2476	0.1371	-0.245	0.5189	-0.364	0.2407	1.0488	0.2548	$-0.000383e^{-7}$	0.1832	0.0737	0.8677	0.6086	5.6501	0.4356	0.4425	0.1964	146478.8882	0.0451	True value
Vertebral NC=3	4.0	2.0	2.0	2.0	2.0	2.0	2.0	12.0	2.0	2.0	2.0	2.0	2.0	3.0	4.0	3.0	3.0	3.0	3.0	3.0	NC
	1.0	0.6758	0.5091	0.3156	0.2784	0.2758	0.6741	0.4332	0.6776	$0.0158e^{-7}$	0.472	0.3422	1.037	0.4409	1.7297	0.3151	0.6314	0.4025	34218.4527	0.0979	Value
	1.0	0.6501	0.4813	0.2906	0.2972	0.2712	0.648	0.5461	0.6523	$0.0156e^{-7}$	0.4698	0.3187	1.9889	0.4409	2.563	0.3151	0.6314	0.4025	34218.4527	0.0979	True value
Wine NC=3	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	2.0	3.0	2.0	2.0	4.0	7.0	2.0	NC
	1.0002	0.5705	0.3979	0.0801	0.3354	0.0693	0.569	0.5433	0.572	$0.0618e^{-7}$	0.3654	0.2485	0.2957	0.2373	0.9933	0.2293	0.6609	0.1148	1031483.5847	0.0716	Value
	1.0002	0.2962	0.1713	-0.2393	0.527	-0.3074	0.2918	0.874	0.3006	$-0.1554e^{-7}$	0.2063	0.0939	0.6533	0.5387	0.9933	0.3487	0.5605	0.127	456485.2784	0.0224	True value
Yeast NC=10	4.0	2.0	2.0	2.0	2.0	2.0	2.0	27.0	2.0	2.0	2.0	2.0	9.0	3.0	6.0	6.0	6.0	3.0	6.0	23.0	NC
	1.0	0.5467	0.3739	0.0035	0.3828	0.0029	0.5439	-1.1173	0.5495	$0.0000309e^{-9}$	0.3346	0.2302	0.9405	0.9286	1.5359	0.903	0.2703	60.964	0.0372	0.0513	Value
	1.0	0.0762	0.0297	-0.1406	0.7389	-0.5424	0.0577	1.5557	0.1006	$-0.00888e^{-9}$	0.0906	0.0151	1.1853	2.0998	1.6862	1.1317	0.2088	99.9504	0.0194	0.0303	True value

Table 4.6: Results for K-means clustering.

	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	XB	SDbw	DB	S	SD	PBM	Dunn	
Breast tissue NC=6	7.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	7.0	2.0	2.0	2.0	2.0	3.0	7.0	2.0	NC
	1.0005	1.0	1.0	1.0	0.0124	1.0	1.0	0.0462	1.0	$0.0358e^{-4}$	1.0	1.0	1.1233	0.0358	0.8558	0.0179	0.9753	0.0068	40953254951.0304	3.3504	Value
	0.9992	0.5749	0.3997	0.2262	0.3231	0.2739	0.5704	0.6707	0.5795	$0.0012e^{-4}$	0.4717	0.2503	1.1793	0.4037	3.8481	0.2808	0.6768	0.0083	36464035150.7726	0.1727	True value
Ecoli NC=8	3.0	2.0	2.0	2.0	2.0	2.0	2.0	13.0	2.0	2.0	2.0	2.0	3.0	2.0	13.0	2.0	2.0	2.0	2.0	3.0	NC
	1.0001	0.5537	0.382	0.051	0.3542	0.0451	0.5528	0.5572	0.5546	$0.00182e^{-7}$	0.3539	0.2361	0.9191	0.545	1.3621	0.4881	0.4274	28.6348	0.1515	0.0874	Value
	1.0	0.0867	0.0339	-0.1301	0.7323	-0.4919	0.0657	1.2508	0.1146	$-0.0306e^{-7}$	0.1049	0.0173	1.1129	1.209	1.5175	0.909	0.2687	49.3509	0.0481	0.0631	True value
Glass NC=7	5.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	5.0	4.0	9.0	4.0	4.0	3.0	4.0	4.0	NC
	1.0011	0.8963	0.8114	0.7856	0.1529	0.7292	0.8959	0.3407	0.8967	$0.0192e^{-6}$	0.7717	0.6826	0.8174	0.6294	0.9303	0.4522	0.5882	7.8025	12.6043	0.1633	Value
	1.0004	0.4942	0.3237	0.1051	0.3532	0.1349	0.4885	0.7273	0.5001	$0.00349e^{-6}$	0.3985	0.1934	1.085	1.2382	1.2943	0.6952	0.4034	9.7968	9.1859	0.06	True value
Haberman NC=2	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	5.0	2.0	6.0	2.0	2.0	2.0	7.0	12.0	NC
	1.0001	0.5658	0.3945	0.1285	0.3828	0.1275	0.5658	0.5676	0.5658	$0.00749e^{-7}$	0.3925	0.2457	0.7846	0.5116	0.9972	0.484	0.3967	1.0628	212.8548	0.0476	Value
	0.9999	0.5658	0.3945	0.1285	0.3828	0.1275	0.5658	0.5676	0.5658	$0.00749e^{-7}$	0.3925	0.2457	1.0667	0.5116	1.1687	0.484	0.3967	1.0628	144.9489	0.0262	True value
Iris NC=3	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	2.0	5.0	2.0	2.0	3.0	3.0	3.0	NC
	1.0002	0.5176	0.3487	-0.0065	0.404	-0.006	0.5171	0.5752	0.5181	$-0.00622e^{-7}$	0.3306	0.2112	0.9122	0.2555	0.6892	0.2024	0.7009	8.4861	25.0692	0.0988	Value
	1.0002	0.3236	0.1875	-0.1225	0.5377	-0.1812	0.3157	0.8504	0.3316	$-0.1983e^{-7}$	0.2537	0.1034	0.9122	0.4107	0.7912	0.3919	0.5553	8.4861	25.0692	0.0988	True value
Parkinsons NC=2	4.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	5.0	2.0	5.0	2.0	2.0	2.0	8.0	2.0	NC
	1.0001	0.9351	0.8767	0.8612	0.0748	0.6537	0.9342	0.2101	0.9361	$0.043e^{-6}$	0.8111	0.7809	0.8284	0.2573	2.441	0.2369	0.6623	0.1751	49577.66	0.3069	Value
	1.0	0.9351	0.8767	0.8612	0.0748	0.6537	0.9342	0.2101	0.9361	$0.043e^{-6}$	0.8111	0.7809	1.0	0.2573	5.9713	0.2369	0.6623	0.1751	39996.9435	0.3069	True value
Vehicle NC=4	3.0	2.0	2.0	2.0	2.0	2.0	2.0	20.0	2.0	2.0	2.0	2.0	5.0	2.0	19.0	2.0	2.0	3.0	5.0	19.0	NC
	1.0	0.5905	0.4183	0.1382	0.3251	0.1248	0.5899	-0.3032	0.5912	$0.000125e^{-7}$	0.3916	0.2645	0.7662	0.2482	5.5044	0.2239	0.6597	0.1959	230520.7614	0.0918	Value
	1.0	0.2343	0.1284	-0.2531	0.536	-0.3867	0.2272	1.0603	0.2418	$-0.000409e^{-7}$	0.1748	0.0687	0.8642	0.6249	5.6484	0.4562	0.4524	0.2092	145428.5273	0.0387	True value
Vertebral NC=3	4.0	2.0	2.0	2.0	2.0	2.0	2.0	12.0	2.0	2.0	2.0	2.0	2.0	3.0	2.0	3.0	3.0	4.0	3.0	5.0	NC
	1.0	0.5228	0.3533	-0.0017	0.4059	-0.0016	0.5221	0.5452	0.5235	$-0.0000891e^{-7}$	0.3326	0.2146	1.015	0.5342	0.9588	0.3185	0.6144	1.5009	45095.1951	0.0973	Value
	0.9998	0.4899	0.3242	-0.0119	0.4326	-0.012	0.4888	0.6295	0.491	$-0.000675e^{-7}$	0.328	0.1938	1.9607	0.5342	2.5848	0.3185	0.6144	1.5113	45095.1951	0.0688	True value
Wine NC=3	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	4.0	2.0	7.0	2.0	2.0	6.0	7.0	6.0	NC
	1.0001	0.5453	0.3739	0.0315	0.3614	0.0277	0.5442	0.5593	0.5463	$0.0145e^{-7}$	0.3459	0.23	0.5243	0.2687	0.475	0.2394	0.6406	0.0696	1140154.9197	0.0603	Value
	1.0001	0.2661	0.1485	-0.2009	0.5795	-0.3085	0.2585	0.9133	0.2738	$-0.1699e^{-7}$	0.2035	0.0802	0.828	0.5254	2.405	0.3474	0.568	0.0799	473716.2925	0.0163	True value
Yeast NC=10	3.0	2.0	2.0	2.0	2.0	2.0	2.0	27.0	2.0	2.0	2.0	2.0	10.0	3.0	27.0	2.0	6.0	3.0	6.0	27.0	NC
	1.0	0.5944	0.4219	0.1374	0.3184	0.1214	0.5934	-0.9362	0.5954	$0.0001281e^{-7}$	0.3902	0.2674	1.0642	0.6985	1.5093	0.8008	0.2699	60.8302	0.0394	0.0419	Value
	1.0	0.0538	0.0196	-0.1302	0.7760	-0.6036	0.0384	1.6648	0.0756	$-0.00011e^{-7}$	0.0688	0.0099	1.0642	1.3503	1.6670	1.1290	0.2011	100.5881	0.0184	0.0239	True value

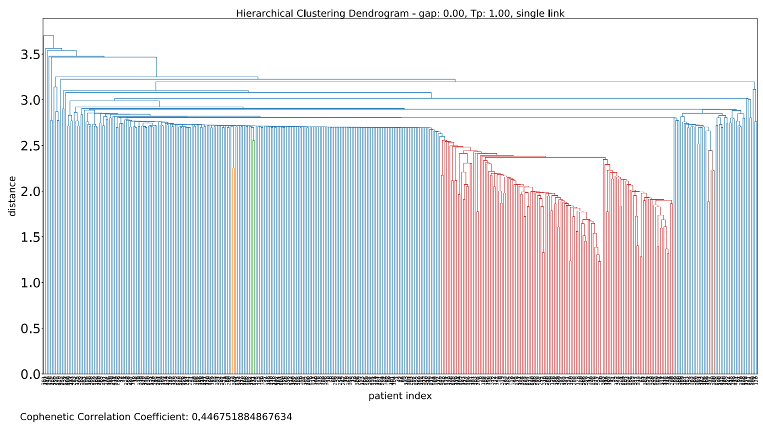


Figure 4.11: Dendrogram of single-linkage hierarchical clustering with gap = 0.0 and Tp = 1.00.

only on gap values that are positive.

We divided our experiments in two groups: **1) automatic** generation of results, i.e., generation of clusters, CVIs analysis and final result, and **2) semi-automatic** generation of results, i.e., making ourselves the decision on the number of clusters and final partition, based on our analysis of a range of clustering results and its associated CVIs.

The tests consisted on performing hierarchical clustering using the five variations mentioned earlier in Subsection 2.1.1. There was no test for partitional clustering K-means because the dataset used is not an n-dimensional one, but just a sequence of events for each patient (e.g. Table 4.7). Therefore, there is no possibility of calculating cluster centroids, which is what K-means is based in.

patient_id	**Temporal sequences**
33496	0.B,156.C,314.D,1696.Z
33499	0.B,148.Z

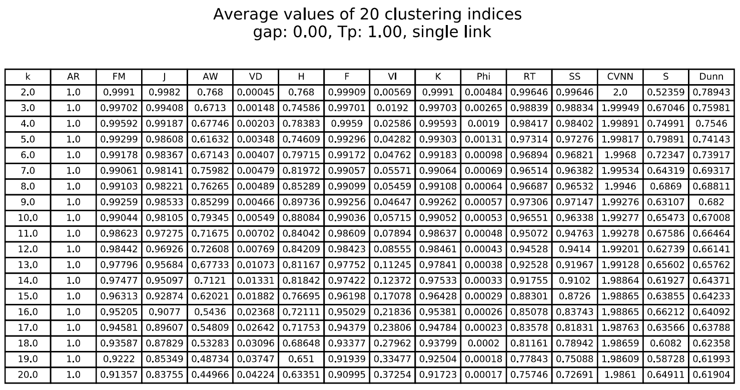
Table 4.7: Prefix-encoded temporal sequences for patients 33496 and 33499.

4.4.1 Automatic results

Regarding the automatic results, for the Average and Single linkages, clusterval chose two as the number of clusters for tests tp=0.25 and tp=1.00. While for tp=2.00, single link chooses again k=2, but average link ouputs k=7. For this two clustering algorithms, when the results is a two cluster structure, it is generated one very small partition (1-8 elements) and a very big one with all other elements. An example of the type of cluster structure from these results is Figure 4.11, that as said before does not show any well defined form.

The average values from the CVIs can be seen in Table 4.12. The values for different k are very similar, when rounding to five decimal points, which makes it hard to decide on the best number of clusters. Although, we can see that most indices have the best value for k=2, hence why clusterval chose that as the cluster number.

In order to help understand the results chosen we can also look at the average standard deviation


Average values of 20 clustering indices
gap: 0.00, Tp: 1.00, single link

k	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	S	Dunn
2.0	1.0	0.9991	0.9982	0.768	0.00045	0.768	0.99909	0.00569	0.9991	0.00484	0.99646	0.99646	2.0	0.52359	0.78943
3.0	1.0	0.99702	0.99408	0.6713	0.00148	0.74586	0.99701	0.0192	0.99703	0.00265	0.98839	0.98834	1.99949	0.67046	0.75981
4.0	1.0	0.99592	0.99187	0.67746	0.00203	0.78383	0.9959	0.02586	0.99593	0.0019	0.98417	0.98402	1.99891	0.74991	0.7546
5.0	1.0	0.99299	0.98608	0.61632	0.00348	0.74609	0.99296	0.04282	0.99303	0.00131	0.97314	0.97276	1.99817	0.79891	0.74143
6.0	1.0	0.99178	0.98367	0.67143	0.00407	0.79715	0.99172	0.04762	0.99183	0.00098	0.96894	0.96821	1.9968	0.72347	0.73917
7.0	1.0	0.99061	0.98141	0.75982	0.00479	0.81972	0.99057	0.05571	0.99064	0.00069	0.96514	0.96382	1.99534	0.64319	0.69317
8.0	1.0	0.99103	0.98221	0.76265	0.00489	0.85289	0.99099	0.05459	0.99108	0.00064	0.96687	0.96532	1.9946	0.6869	0.68811
9.0	1.0	0.99259	0.98533	0.85299	0.00466	0.89736	0.99256	0.04647	0.99262	0.00057	0.97306	0.97147	1.99276	0.63107	0.682
10.0	1.0	0.99044	0.98105	0.79345	0.00549	0.88084	0.99036	0.05715	0.99052	0.00053	0.96551	0.96338	1.99277	0.65473	0.67008
11.0	1.0	0.98623	0.97275	0.71675	0.00702	0.84042	0.98609	0.07894	0.98637	0.00048	0.95072	0.94763	1.99278	0.67586	0.66464
12.0	1.0	0.98442	0.96926	0.72608	0.00769	0.84209	0.98423	0.08555	0.98461	0.00043	0.94528	0.9414	1.99201	0.62739	0.66141
13.0	1.0	0.97796	0.95684	0.67733	0.01073	0.81167	0.97752	0.11245	0.97841	0.00038	0.92528	0.91967	1.99128	0.65602	0.65762
14.0	1.0	0.97477	0.95097	0.7121	0.01331	0.81842	0.97422	0.12372	0.97533	0.00033	0.91755	0.9102	1.98864	0.61927	0.64371
15.0	1.0	0.96313	0.92874	0.62021	0.01882	0.76695	0.96198	0.17078	0.96428	0.00029	0.88301	0.8726	1.98865	0.63855	0.64233
16.0	1.0	0.95205	0.9077	0.5436	0.02368	0.72111	0.95029	0.21836	0.95381	0.00026	0.85078	0.83743	1.98865	0.66212	0.64092
17.0	1.0	0.94581	0.89607	0.54809	0.02642	0.71753	0.94379	0.23806	0.94784	0.00023	0.83578	0.81831	1.98763	0.63566	0.63788
18.0	1.0	0.93587	0.87829	0.53283	0.03096	0.68648	0.93377	0.27962	0.93799	0.0002	0.81161	0.78942	1.98659	0.6082	0.62358
19.0	1.0	0.9222	0.85349	0.48734	0.03747	0.651	0.91939	0.33477	0.92504	0.00018	0.77843	0.75088	1.98609	0.58728	0.61993
20.0	1.0	0.91357	0.83755	0.44966	0.04224	0.63351	0.90995	0.37254	0.91723	0.00017	0.75746	0.72691	1.9861	0.64911	0.61904

Figure 4.12: Average clustering indices values for dendrogram of Figure 4.11

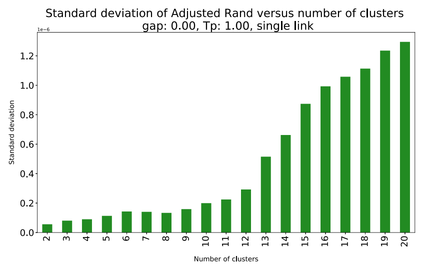


Figure 4.13: Standard deviation of AR versus number of clusters for the dendrogram of Figure 4.11

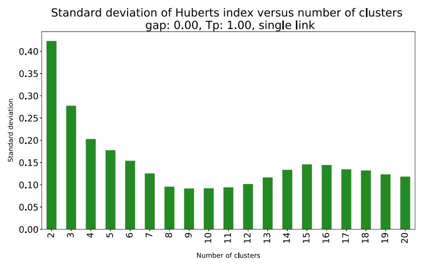


Figure 4.14: Standard deviation of H versus number of clusters for the dendrogram of Figure 4.11

of the CVIs, for example for Adjusted Rand and Huberts indices, represented in Figures 4.13 and 4.14, respectively. We can observe that the two indices offer opposite preferences for the number of clusters, when considering the standard deviation of the values. Adjusted Rand is in line with the output of clusterval (k=2), however, Huberts index suggest that higher values for the number k represent better configurations, and this also is accurate with the average values for H in Table 4.12.

Regarding the Tp values, we notice that when set to 1.0 and 2.0 the result is fewer clusters, from 2 up to 7, which results in clusters hard to read due to how big the graph gets, also, we noticed many repeating events in different clusters which makes them not very distinct. The information given from a small number of clusters brings virtually no value to be applied in the real world.

For the automatic tests, the most interesting results were given by complete-linkage and ward-linkage hierarchical clustering. We can see in Figure 4.15 the resulting dendrogram for complete-linkage, which already shows a bit more structure than the one from Figure 4.11.

The algorithm selected k=20 as the cluster number and we can see the average values of the indices for each k in Table 4.15. External CVIs have better values for high number of clusters, in the range of 15 to 20 and can be confirmed that k=20 is the maximum for most, hence the output of the algorithm.

Regarding internal CVIs, CVNN and S have better values for k=7, with k=5 and k=6 also close. Dunn is in line with the results for external CVIs. The standard deviations versus k values shown in Figure 4.17 supports the average results for CVNN and S, with k=7 being relatively low compared to most other cluster numbers. Due to this opposing results, it is not obvious which k is the best. Analysing the resulting clusters we could see that each seems to be distinct, i.e., containing one event or chain, for

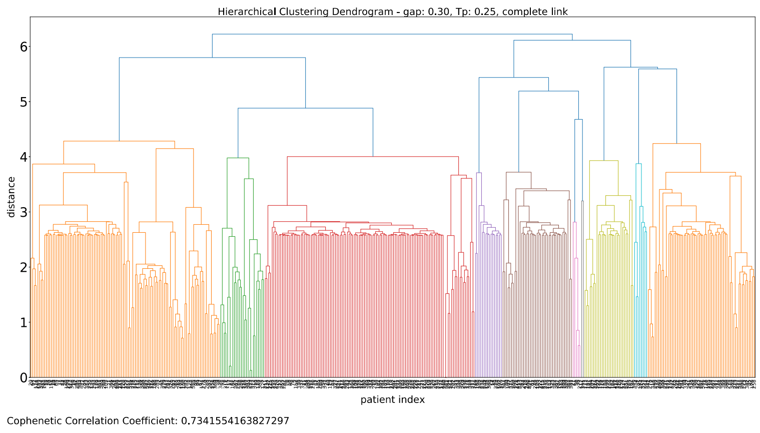
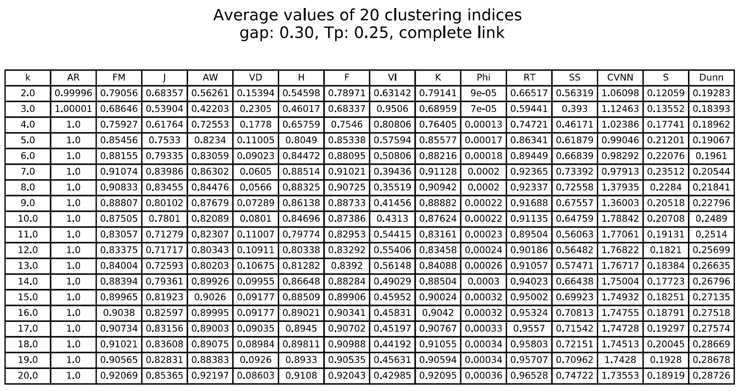

Hierarchical Clustering Dendrogram - gap: 0.30, Tp: 0.25, complete link
distance
0
1
2
3
4
5
6
patient index
Cophenetic Correlation Coefficient: 0.7341554163827297

Figure 4.15: Dendrogram of complete-linkage hierarchical clustering with gap = 0.3 and Tp = 0.25


Average values of 20 clustering indices
gap: 0.30, Tp: 0.25, complete link

k	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	S	Dunn
2.0	0.99996	0.79056	0.68357	0.56261	0.15394	0.54598	0.78971	0.63142	0.79141	9e-05	0.66517	0.56319	1.06098	0.12059	0.19283
3.0	1.00001	0.68646	0.53904	0.42203	0.2305	0.46017	0.68337	0.9506	0.68959	7e-05	0.59441	0.393	1.12463	0.13552	0.18393
4.0	1.0	0.75927	0.61764	0.72553	0.1778	0.65759	0.7546	0.80806	0.76405	0.00013	0.74721	0.46171	1.02386	0.17741	0.18962
5.0	1.0	0.85456	0.7533	0.8234	0.11005	0.8049	0.85338	0.57594	0.85577	0.00017	0.86341	0.61879	0.99046	0.21201	0.19067
6.0	1.0	0.88155	0.79335	0.83059	0.09023	0.84472	0.88095	0.50806	0.88216	0.00018	0.89449	0.66839	0.98292	0.22076	0.1961
7.0	1.0	0.91074	0.83986	0.86302	0.0605	0.88514	0.91021	0.39436	0.91128	0.0002	0.92365	0.73392	0.97913	0.23512	0.20544
8.0	1.0	0.90833	0.83455	0.84476	0.0566	0.88325	0.90725	0.35519	0.90942	0.0002	0.92337	0.72558	1.37935	0.2284	0.21841
9.0	1.0	0.88807	0.80102	0.87679	0.07289	0.86138	0.88733	0.41456	0.88882	0.00022	0.91688	0.67557	1.36003	0.20518	0.22796
10.0	1.0	0.87505	0.7801	0.82089	0.0801	0.84696	0.87386	0.4313	0.87624	0.00022	0.91135	0.64759	1.78842	0.20708	0.2489
11.0	1.0	0.83057	0.71279	0.82307	0.11007	0.79774	0.82953	0.54415	0.83161	0.00023	0.89504	0.56063	1.77061	0.19131	0.2514
12.0	1.0	0.83375	0.71717	0.80343	0.10911	0.80338	0.83292	0.55406	0.83458	0.00024	0.90186	0.56482	1.76822	0.1821	0.25699
13.0	1.0	0.84004	0.72593	0.80203	0.10675	0.81282	0.8392	0.56148	0.84088	0.00026	0.91057	0.57471	1.76717	0.18384	0.26635
14.0	1.0	0.88394	0.79361	0.89926	0.09955	0.86648	0.88284	0.49029	0.88504	0.0003	0.94023	0.66438	1.75004	0.17723	0.26796
15.0	1.0	0.89965	0.81923	0.9026	0.09177	0.88509	0.89906	0.45952	0.90024	0.00032	0.95002	0.69923	1.74932	0.18251	0.27135
16.0	1.0	0.9038	0.82597	0.89995	0.09177	0.89021	0.90341	0.45831	0.9042	0.00032	0.95324	0.70813	1.74755	0.18791	0.27518
17.0	1.0	0.90734	0.83156	0.89003	0.09035	0.8945	0.90702	0.45197	0.90767	0.00033	0.9557	0.71542	1.74728	0.19297	0.27574
18.0	1.0	0.91021	0.83608	0.89075	0.08984	0.89811	0.90988	0.44192	0.91055	0.00034	0.95803	0.72151	1.74513	0.20045	0.28669
19.0	1.0	0.90565	0.82831	0.88383	0.0926	0.8933	0.90535	0.45631	0.90594	0.00034	0.95707	0.70962	1.7428	0.1928	0.28678
20.0	1.0	0.92069	0.85365	0.92197	0.08603	0.9108	0.92043	0.42985	0.92095	0.00036	0.96528	0.74722	1.73553	0.18919	0.28726

Figure 4.16: Average clustering indices values for dendrogram of Figure 4.15

example, cluster 1 which separates the event "H->F->Z" from all others, or cluster 20 with all "A->H->Z" events. Although, some of the clusters still look to be relatively long and with many connections, as is the case for cluster 14 with 21 patients and 5 events related. The clusters for the selected k (20) can be seen in Appendix A.1.

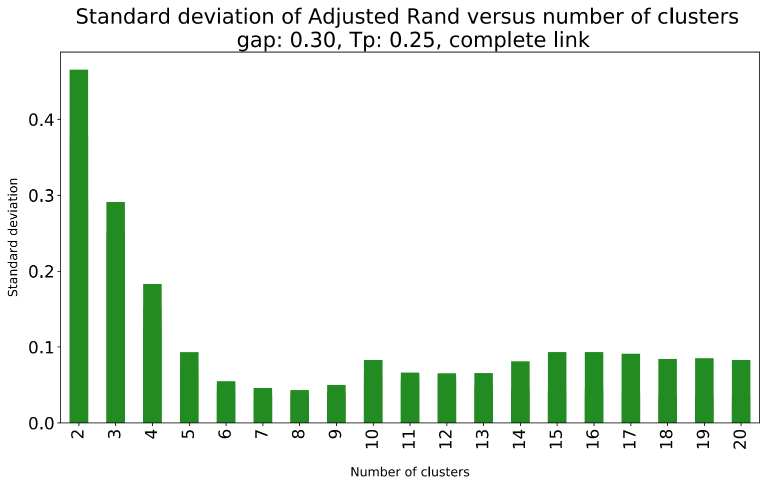


Figure 4.17: Standard deviation of AR versus number of clusters for the dendrogram of Figure 4.15

On the other hand, for Ward-linkage hierarchical clustering the algorithm chooses the dendrogram show in Figure 4.18. It seems to represent a more well defined cluster than the complete link and the algorithm chose k=20 as the best number of clusters.

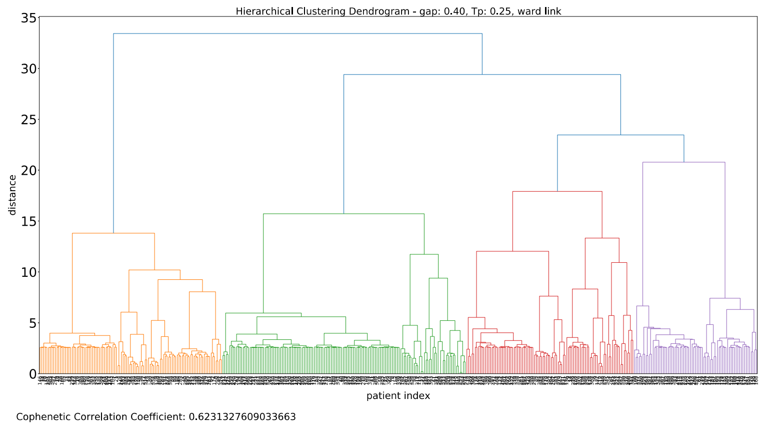
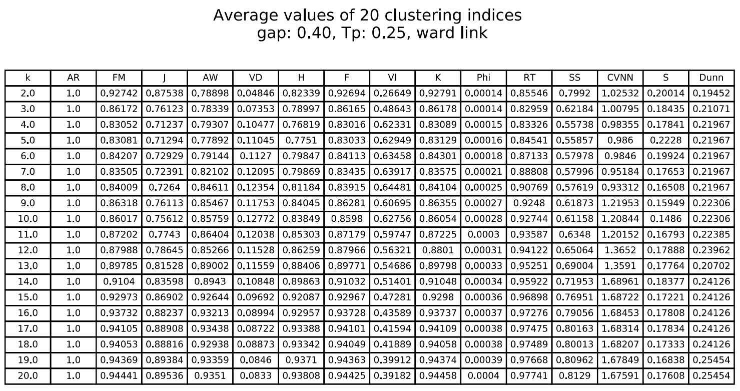


Figure 4.18: Dendrogram of ward-linkage hierarchical clustering with gap = 0.4 and Tp = 0.25

The average values for the indices are shown in Table 4.19 and the vast majority of indices have better values for high values of k (more than 17), although values are generally not very far from each other, similar to the results seen for complete linkage.



Average values of 20 clustering indices
gap: 0.40, Tp: 0.25, ward link

k	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	S	Dunn
2.0	1.0	0.92742	0.87538	0.78898	0.04846	0.82339	0.92694	0.26649	0.92791	0.00014	0.85546	0.7992	1.02532	0.20014	0.19452
3.0	1.0	0.86172	0.76123	0.78339	0.07353	0.78997	0.86165	0.48643	0.86178	0.00014	0.82959	0.62184	1.00795	0.18435	0.21071
4.0	1.0	0.83052	0.71237	0.79307	0.10477	0.76819	0.83016	0.62331	0.83089	0.00015	0.83326	0.55738	0.98355	0.17841	0.21967
5.0	1.0	0.83081	0.71294	0.77892	0.11045	0.7751	0.83033	0.62949	0.83129	0.00016	0.84541	0.55857	0.986	0.2228	0.21967
6.0	1.0	0.84207	0.72929	0.79144	0.1127	0.79847	0.84113	0.63458	0.84301	0.00018	0.87133	0.57978	0.9846	0.19924	0.21967
7.0	1.0	0.83505	0.72391	0.82102	0.12095	0.79869	0.83435	0.63917	0.83575	0.00021	0.88808	0.57996	0.95184	0.17653	0.21967
8.0	1.0	0.84009	0.7264	0.84611	0.12354	0.81184	0.83915	0.64481	0.84104	0.00025	0.90769	0.57619	0.93312	0.16508	0.21967
9.0	1.0	0.86318	0.76113	0.85467	0.11753	0.84045	0.86281	0.60695	0.86355	0.00027	0.9248	0.61873	1.21953	0.15949	0.22306
10.0	1.0	0.86017	0.75612	0.85759	0.12772	0.83849	0.8598	0.62756	0.86054	0.00028	0.92744	0.61158	1.20844	0.1486	0.22306
11.0	1.0	0.87202	0.7743	0.86404	0.12038	0.85303	0.87179	0.59747	0.87225	0.0003	0.93587	0.6348	1.20152	0.16793	0.22385
12.0	1.0	0.87988	0.78645	0.85266	0.11528	0.86259	0.87966	0.56321	0.8801	0.00031	0.94122	0.65064	1.3652	0.17888	0.23962
13.0	1.0	0.89785	0.81528	0.89002	0.11559	0.88406	0.89771	0.54686	0.89798	0.00033	0.95251	0.69004	1.3591	0.17764	0.20702
14.0	1.0	0.9104	0.83598	0.8943	0.10848	0.89863	0.91032	0.51401	0.91048	0.00034	0.95922	0.71953	1.68961	0.18377	0.24126
15.0	1.0	0.92973	0.86902	0.92644	0.09692	0.92087	0.92967	0.47281	0.9298	0.00036	0.96898	0.76951	1.68722	0.17221	0.24126
16.0	1.0	0.93732	0.88237	0.93213	0.08994	0.92957	0.93728	0.43589	0.93737	0.00037	0.97276	0.79056	1.68453	0.17808	0.24126
17.0	1.0	0.94105	0.88908	0.93438	0.08722	0.93388	0.94101	0.41594	0.94109	0.00038	0.97475	0.80163	1.68314	0.17834	0.24126
18.0	1.0	0.94053	0.88816	0.92938	0.08873	0.93342	0.94049	0.41889	0.94058	0.00038	0.97489	0.80013	1.68207	0.17333	0.24126
19.0	1.0	0.94369	0.89384	0.93359	0.0846	0.9371	0.94363	0.39912	0.94374	0.00039	0.97668	0.80962	1.67849	0.16838	0.25454
20.0	1.0	0.94441	0.89536	0.9351	0.0833	0.93808	0.94425	0.39182	0.94458	0.0004	0.97741	0.8129	1.67591	0.17608	0.25454

Figure 4.19: Average clustering indices values for dendrogram of Figure 4.18

Moreover, the standard deviation shown in Figure 4.20 is lower for values of k between 10 and 12, although other values for the number of clusters also also have low standard deviations. Again, the algorithm does give a valid answer, in line with the values here presented, although other values of k also represent a good structure. The number of clusters chosen was 20 and analysing each cluster individually we again notice a good separation of the events, with most being well contained, others big due to long sequences, like cluster 5 and 14. Moreover, in comparison with the complete-linkage results, clusters have more elements/patients and cluster 20 is the same for both with sequence “A->H->F“.

The clusters for the chosen k (20), can be seen in Appendix A.2.

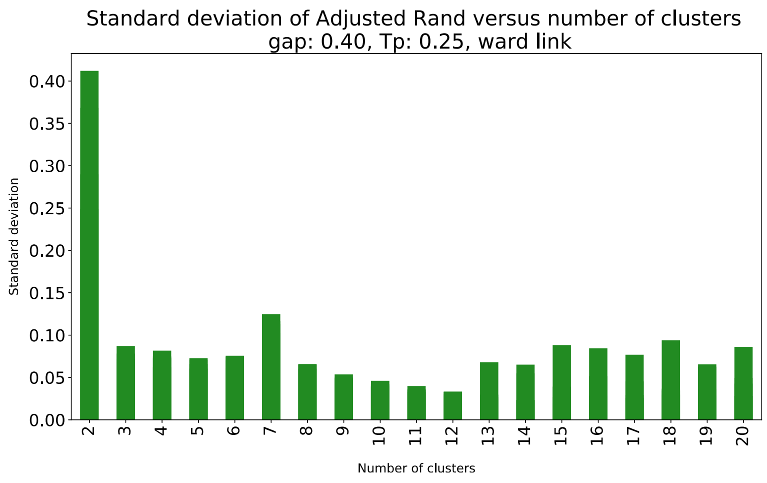


Figure 4.20: Standard deviation of AR versus number of clusters for the dendrogram of Figure 4.18

4.4.2 Semi-automatic results

The second part of the experiment consisted in running the same previous tests but instead of letting the algorithm choose the number of clusters we will do it ourselves.

Most clustering algorithms tend to form few and large clusters that don't seem to carry much information. Much like the automatic results, complete-linkage and ward-linkage hierarchical clustering generated the most interesting cluster structures. Firstly, we can see in Figure 4.21 the dendrogram chosen for complete link. Looking at the average index values in Table 4.22 we can see that almost all indices have better values for high k number (range of 16 to 20), with exceptions of internal CVIs, CVNN and S (range 9 to 12). The standard deviation for two of the indices was checked in order to give more insight, but looking at Figures 4.23 and 4.24 the values are similar for the cluster number in the said ranges. In the end, k=20 was chosen because it was creating more simple sequence chains that seem to hold more information in the problem context. In comparison with automatic results, complete-linkage differs slightly on the cluster sequences, both in time and states, with the semi-automatic one having subjectively harder to read chains due to being longer. Moreover, cluster 20 is the same for both.

In the Appendix A.1, a representation of the clusters is shown.

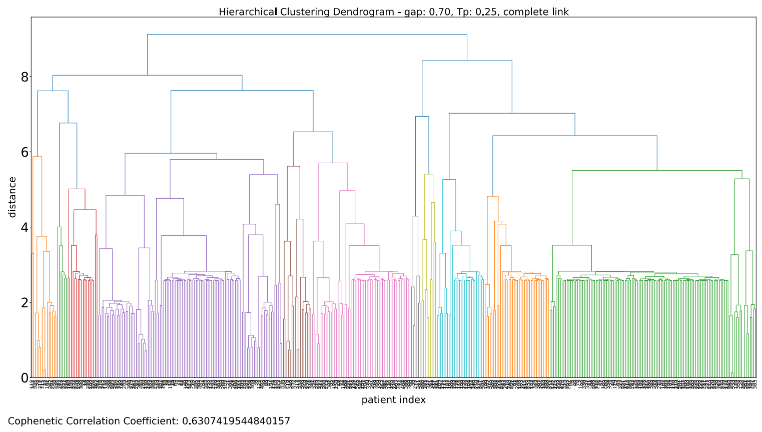
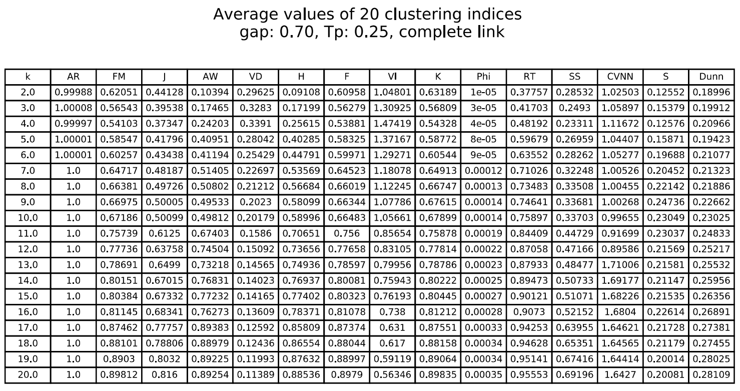


Figure 4.21: Dendrogram of complete-linkage hierarchical clustering with gap = 0.7 and Tp = 0.25


Average values of 20 clustering indices
gap: 0.70, Tp: 0.25, complete link

k	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	S	Dunn
2.0	0.99988	0.62051	0.44128	0.10394	0.29625	0.09108	0.60958	1.04801	0.63189	1e-05	0.37757	0.28532	1.02503	0.12552	0.18996
3.0	1.00008	0.56543	0.39538	0.17465	0.3283	0.17199	0.56279	1.30925	0.56809	3e-05	0.41703	0.2493	1.05897	0.15379	0.19912
4.0	0.99997	0.54103	0.37347	0.24203	0.3391	0.25615	0.53881	1.47419	0.54328	4e-05	0.48192	0.23311	1.11672	0.12576	0.20966
5.0	1.00001	0.58547	0.41796	0.40951	0.28042	0.40285	0.58325	1.37167	0.58772	8e-05	0.59679	0.26959	1.04407	0.15871	0.19423
6.0	1.00001	0.60257	0.43438	0.41194	0.25429	0.44791	0.59971	1.29271	0.60544	9e-05	0.63552	0.28262	1.05277	0.19688	0.21077
7.0	1.0	0.64717	0.48187	0.51405	0.22697	0.53569	0.64523	1.18078	0.64913	0.00012	0.71026	0.32248	1.00526	0.20452	0.21323
8.0	1.0	0.66381	0.49726	0.50802	0.21212	0.56684	0.66019	1.12245	0.66747	0.00013	0.73483	0.33508	1.00455	0.22142	0.21886
9.0	1.0	0.66975	0.50005	0.49533	0.2023	0.58099	0.66344	1.07786	0.67615	0.00014	0.74641	0.33681	1.00268	0.24736	0.22662
10.0	1.0	0.67186	0.50099	0.49812	0.20179	0.58996	0.66483	1.05661	0.67899	0.00014	0.75897	0.33703	0.99655	0.23049	0.23025
11.0	1.0	0.75739	0.6125	0.67403	0.1586	0.70651	0.756	0.85654	0.75878	0.00019	0.84409	0.44729	0.91699	0.23037	0.24833
12.0	1.0	0.77736	0.63758	0.74504	0.15092	0.73656	0.77658	0.83105	0.77814	0.00022	0.87058	0.47166	0.89586	0.21569	0.25217
13.0	1.0	0.78691	0.6499	0.73218	0.14565	0.74936	0.78597	0.79956	0.78786	0.00023	0.87933	0.48477	1.71006	0.21581	0.25532
14.0	1.0	0.80151	0.67015	0.76831	0.14023	0.76937	0.80081	0.75943	0.80222	0.00025	0.89473	0.50733	1.69177	0.21147	0.25956
15.0	1.0	0.80384	0.67332	0.77232	0.14165	0.77402	0.80323	0.76193	0.80445	0.00027	0.90121	0.51071	1.68226	0.21535	0.26356
16.0	1.0	0.81145	0.68341	0.76273	0.13609	0.78371	0.81078	0.738	0.81212	0.00028	0.9073	0.52152	1.6804	0.22614	0.26891
17.0	1.0	0.87462	0.77757	0.89383	0.12592	0.85809	0.87374	0.631	0.87551	0.00033	0.94253	0.63955	1.64621	0.21728	0.27381
18.0	1.0	0.88101	0.78806	0.88979	0.12436	0.86554	0.88044	0.617	0.88158	0.00034	0.94628	0.65351	1.64565	0.21179	0.27455
19.0	1.0	0.8903	0.8032	0.89225	0.11993	0.87632	0.88997	0.59119	0.89064	0.00034	0.95141	0.67416	1.64414	0.20014	0.28025
20.0	1.0	0.89812	0.816	0.89254	0.11389	0.88536	0.8979	0.56346	0.89835	0.00035	0.95553	0.69196	1.6427	0.20081	0.28109

Figure 4.22: Average clustering indices values for dendrogram of Figure 4.21

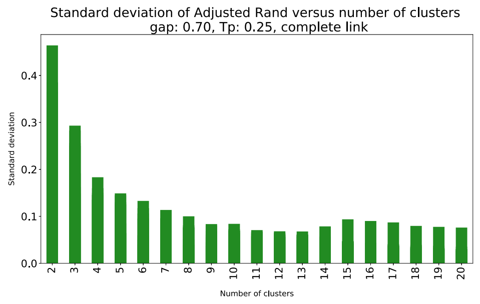


Figure 4.23: Standard deviation of AR versus number of clusters for the dendrogram of Figure 4.21

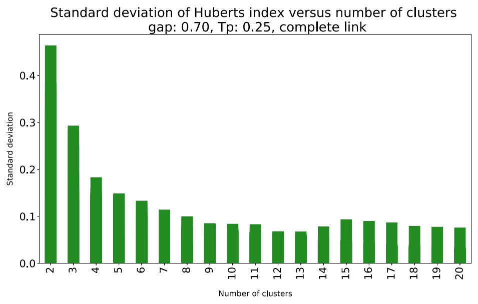


Figure 4.24: Standard deviation of H versus number of clusters for the dendrogram of Figure 4.21

Finally, from ward link we can see the dendrogram in Figure 4.25. By analysing the index values from Table 4.26 we see that k in the range 16 to 20 has the best index values for every CVI (ignoring k=2, which is not desirable for being to general) and the standard deviations of two of the indices versus the number of clusters, represented in Figures 4.27 and 4.28, shows that there is not a value in that range that stands out, making the decision of the final number of clusters hard. Nevertheless, following the indices results, k=16 has the best results for the vast majority of the considered CVIs, therefore it was chosen as final k. In comparison with the automatic results, ward-linkage has bigger graphs, which is expected due to having bigger and less clusters. Furthermore, cluster 20 is the same for both approaches.

In the Appendix B.2, a representation of the clusters is shown.

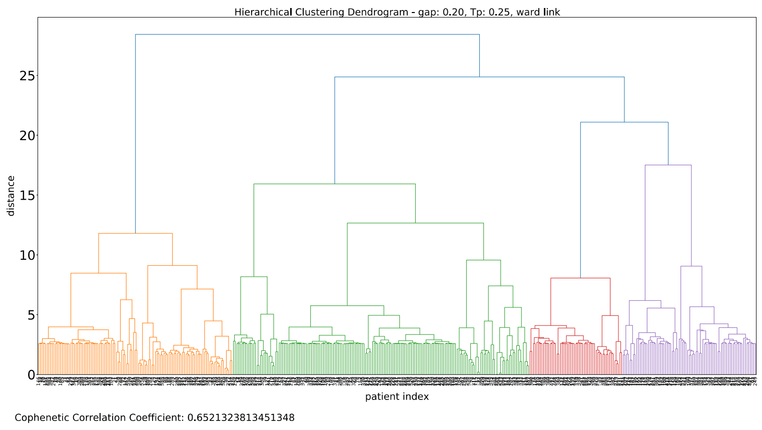
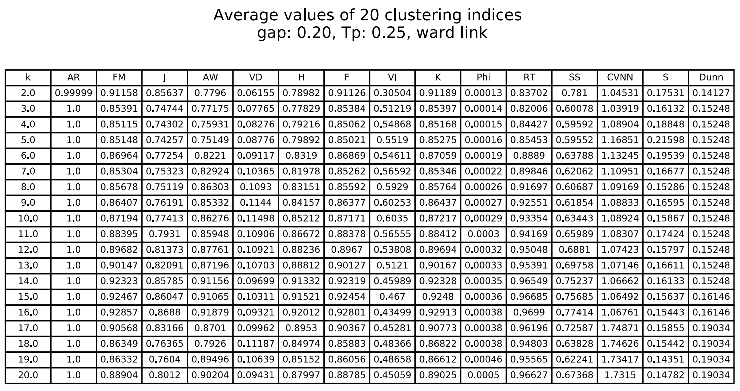


Figure 4.25: Dendrogram of ward-linkage hierarchical clustering with gap = 0.2 and Tp = 0.25


Average values of 20 clustering indices
gap: 0.20, Tp: 0.25, ward link

k	AR	FM	J	AW	VD	H	F	VI	K	Phi	RT	SS	CVNN	S	Dunn
2.0	0.99999	0.91158	0.85637	0.7796	0.06155	0.78982	0.91126	0.30504	0.91189	0.00013	0.83702	0.781	1.04531	0.17531	0.14127
3.0	1.0	0.85391	0.74744	0.77175	0.07765	0.77829	0.85384	0.51219	0.85397	0.00014	0.82006	0.60078	1.03919	0.16132	0.15248
4.0	1.0	0.85115	0.74302	0.75931	0.08276	0.79216	0.85062	0.54868	0.85168	0.00015	0.84427	0.59592	1.08904	0.18848	0.15248
5.0	1.0	0.85148	0.74257	0.75149	0.08776	0.79892	0.85021	0.5519	0.85275	0.00016	0.85453	0.59552	1.16851	0.21598	0.15248
6.0	1.0	0.86964	0.77254	0.8221	0.09117	0.8319	0.86869	0.54611	0.87059	0.00019	0.8889	0.63788	1.13245	0.19539	0.15248
7.0	1.0	0.85304	0.75323	0.82924	0.10365	0.81978	0.85262	0.56592	0.85346	0.00022	0.89846	0.62062	1.10951	0.16677	0.15248
8.0	1.0	0.85678	0.75119	0.86303	0.1093	0.83151	0.85592	0.5929	0.85764	0.00026	0.91697	0.60687	1.09169	0.15286	0.15248
9.0	1.0	0.86407	0.76191	0.85332	0.1144	0.84157	0.86377	0.60253	0.86437	0.00027	0.92551	0.61854	1.08833	0.16595	0.15248
10.0	1.0	0.87194	0.77413	0.86276	0.11498	0.85212	0.87171	0.6035	0.87217	0.00029	0.93354	0.63443	1.08924	0.15867	0.15248
11.0	1.0	0.88395	0.7931	0.85948	0.10906	0.86672	0.88378	0.56555	0.88412	0.0003	0.94169	0.65989	1.08307	0.17424	0.15248
12.0	1.0	0.89682	0.81373	0.87761	0.10921	0.88236	0.8967	0.53808	0.89694	0.00032	0.95048	0.6881	1.07423	0.15797	0.15248
13.0	1.0	0.90147	0.82091	0.87196	0.10703	0.88812	0.90127	0.5121	0.90167	0.00033	0.95391	0.69758	1.07146	0.16611	0.15248
14.0	1.0	0.92323	0.85785	0.91156	0.09699	0.91332	0.92319	0.45989	0.92328	0.00035	0.96549	0.75237	1.06662	0.16133	0.15248
15.0	1.0	0.92467	0.86047	0.91065	0.10311	0.91521	0.92454	0.467	0.9248	0.00036	0.96685	0.75685	1.06492	0.15637	0.16146
16.0	1.0	0.92857	0.8688	0.91879	0.09321	0.92012	0.92801	0.43499	0.92913	0.00038	0.9699	0.77414	1.06761	0.15443	0.16146
17.0	1.0	0.90568	0.83166	0.8701	0.09962	0.8953	0.90367	0.45281	0.90773	0.00038	0.96196	0.72587	1.74871	0.15855	0.19034
18.0	1.0	0.86349	0.76365	0.7926	0.11187	0.84974	0.85883	0.48366	0.86822	0.00038	0.94803	0.63828	1.74626	0.15442	0.19034
19.0	1.0	0.86332	0.7604	0.89496	0.10639	0.85152	0.86056	0.48658	0.86612	0.00046	0.95565	0.62241	1.73417	0.14351	0.19034
20.0	1.0	0.88904	0.8012	0.90204	0.09431	0.87997	0.88785	0.45059	0.89025	0.0005	0.96627	0.67368	1.7315	0.14782	0.19034

Figure 4.26: Average clustering indices values for dendrogram of Figure 4.25

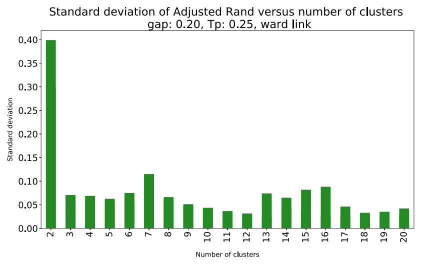


Figure 4.27: Standard deviation of AR versus number of clusters for the dendrogram of Figure 4.25

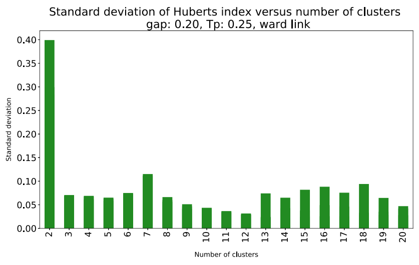


Figure 4.28: Standard deviation of H versus number of clusters for the dendrogram of Figure 4.25

Chapter 5

Conclusions

5.1 Conclusion Remarks

We proposed an increase in the number and variety of clustering validation indices to be used by AliClu, with the goal of giving more robustness and automation to the method. To achieve this, we built a tool, named *clusterval*, that serves the purpose of clustering validation. With 20 metrics included, 12 with an external validation approach and 8 with an internal one, *clusterval* has the potential to be useful when doing clustering of data, where the partitions produced should be in some way evaluated in order the unravel the clustering structure that best represents the hidden patterns.

In order to test the performance of *clusterval*, synthetic datasets were generated. Data points were randomly assigned to clusters with some variation in parameters for each partition, namely the number of clusters, dimensionality, density, overlap and noise. In total, 1440 different datasets were generated. The results shown in Section 4.2 tell us how each validation metric performs. For the overall of the datasets generated, XB, PBM, Dunn and CVNN acheived the best results, with 45%, 44%, 42% and 41%, respectively, and the average success rate of all CVIs was 29.3%. When looking at the influence of the number of clusters in the dataset, the results were quite good for k=2, with 71.5%, but a big drop was observed for k=4 and k=8, with 11.1% and 5.4%, respectively. Interestingly, when the datasets have a higher number of features (dimensionality), the results were better for dim=8, with 48% average success. Noise level and density of custers do not seem to have a big influence on the results with only a decrease of 2.7%. and 4.5%, respectively. When introducing overlap we see a decrease of 13.3% on the average success rate. Moreover, we applied 4 different clustering algorithms, 3 of them from the hierarchical family (Single, Complete, Ward) and one from the partitional family (k-means). There is some algorithms that seem to affect more some of the CVIs (e.g. Silhouette and Davies-Bouldin perfom very poorly with Single-linkage hierarchicalclustering), but overall the choice of algorithm does not produce unexpected results.

The methods were further tested with 10 real-world datasets and in more detail Reuma.pt dataset. The real-world datasets were chosen with the goal of having a good variety of configurations, some small and others large, some datasets having many features and other few, same for the number of clusters.

Even though, the datasets are not specific for the clustering problem, but rather for classification, we tested the tool for each, with different clustering algorithms. The results reported in Section 4.3, in comparison with the synthetic datasets, have poorer results on overall, with some indices performing better for the real-world datasets, but also others performing worse. The average success rate considering all datasets and all clustering algorithms used was 20.1%, which is very poor. Further analysing the individual scores for each CVI we can identify that the values chosen are very close to the true values for the number of clusters, sometimes even equal. Hence it shows that a completely automatic approach to cluster validation can lead to misleading results.

In the experiment with Reuma.pt, *clusterval* was implemented with the AliClu tool [1] and two tests were conducted: 1) automatic generation of the results, and 2) semi-automatic generation of the results, where the user takes the final decision. The dataset used was the same as in [1] for experiment I, and we tested for values of gap = [0.0, 0.1], Tp = 0.25, 1.00, 2.00 and 5 hierarchical clustering algorithms.

For the automatic tests, single, average and centroid linkage produce very unstructured partitions for every gap and Tp, with the resulting clustering being very few clusters with extremely long sequences. Complete and ward linkage algorithms lead to the most interesting results, the number of clusters selected by both was k=20, and the resulting clusters were easily interpreted, events are contained and seem to carry useful information for the problem context. The clusters generated can be seen in Appendix A.

For the semi-automatic tests, similarly to automatic tests, single, average and centroid linkage cannot achieve easily interpreted results with the hierarchical dendrograms having no structure. For complete linkage, the final analysis of the CVIs and dendrogram lead us to choose k=20 (like automatic) and for ward linkage, k=16. The clusters generated can be seen in Appendix B.

We noticed that the bigger cluster, for the clustering structures chosen, is always the same, with 150 patients and sequence "A->H->Z". The difference between the other clusters being the distribution of the remaining patients.

5.2 Future Work

In conclusion, the addition of more clustering validation indices thanks to the *clusterval* tool does provide useful results, with well separated and contained clusters, for the Reuma.pt dataset. And more importantly in a more automated fashion. The development of the sequence graphs generation also eases the appliance of the methods and provides end users with a readily working model.

Although, the synthetic and real-world datasets tests suggest that a fully automatic approach is not able to produce results that shows its effectiveness, hence a more iterative process is recommended. Despite this, there is room for improvement. For instance, making the algorithm not choose directly the best scores for the CVIs, but rather take into account the problem context or have the voting system for the CVIs assign different weights to those metrics.

Bibliography

[1] K. Rama, H. Canhão, A. Carvalho, and S. Vinga. Aliclu - temporal sequence alignment for clustering longitudinal clinical data. *BMC Medical Informatics and Decision Making*, 19, 12 2019. doi: 10.1186/s12911-019-1013-7.

[2] H. Syed and A. Das. Temporal needleman-wunsch. 10 2015. doi: 10.1109/DSAA.2015.7344785.

[3] A. Jain and R. Dubes. *Algorithms for Clustering Data*, volume 32. 01 1988. doi: 10.2307/1268876.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3): 264–323, Sept. 1999. ISSN 0360-0300. doi: 10.1145/331499.331504. URL https://doi.org/10.1145/331499.331504.

[5] M.-W. O. Dictionary. Cluster analysis. *<http://www.merriam-webster-online.com>*, 2021.

[6] C. Fraley and A. E. Raftery. How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. *The Computer Journal*, 41(8):578–588, 01 1998. ISSN 0010-4620. doi: 10.1093/comjnl/41.8.578. URL https://doi.org/10.1093/comjnl/41.8.578.

[7] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2009.09.011. URL http://www.sciencedirect.com/science/article/pii/S0167865509002323. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[8] R. R. Sokal. The principles and practice of numerical taxonomy. *TAXON*, 12(5):190–199, 1963. doi: https://doi.org/10.2307/1217562. URL https://onlinelibrary.wiley.com/doi/abs/10.2307/1217562.

[9] M. Anderberg. Cluster analysis for applications. *Academic Press*, 1973. doi: https://dx.doi.org/10.1016/c2013-0-06161-0.

[10] A. Saxena, M. Prasad, A. Gupta, N. Bharill, o. Patel, A. Tiwari, M. Er, W. Ding, and C.-T. Lin. A review of clustering techniques and developments. *Neurocomputing*, 267, 07 2017. doi: 10.1016/j.neucom.2017.06.053.

[11] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(65)90241-X. URL http://www.sciencedirect.com/science/article/pii/S001999586590241X.

[12] F. B. Baker and L. J. Hubert. A graph-theoretic approach to goodness-of-fit in complete-link hierarchical clustering. *Journal of the American Statistical Association*, 71(356):870–878, 1976. ISSN 01621459. URL http://www.jstor.org/stable/2286853.

[13] R. R. Sokal. Distance as a measure of taxonomic similarity. *Systematic Zoology*, 10(2):70–79, 1961. ISSN 00397989. URL http://www.jstor.org/stable/2411724.

[14] Jianchang Mao and A. K. Jain. A self-organizing network for hyperellipsoidal clustering (hec). In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 5, pages 2967–2972 vol.5, 1994. doi: 10.1109/ICNN.1994.374705.

[15] R. R. Sokal. Numerical taxonomy. *Scientific American*, 215(6):106, 1966. ISSN 00368733.

[16] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 62(317): 86–101, 1967. ISSN 01621459. URL http://www.jstor.org/stable/2282912.

[17] J. Ward. Hierarchical groupings to optimize an objective function. *J. Am. Stat. Assoc.*, pages 234–244, 01 1963.

[18] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. *SIGMOD Rec.*, 27(2):73–84, June 1998. ISSN 0163-5808. doi: 10.1145/276305.276312. URL https://doi.org/10.1145/276305.276312.

[19] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996. ISSN 0163-5808. doi: 10.1145/235968.233324. URL https://doi.org/10.1145/235968.233324.

[20] S. Guha, R. Rastogi, and K. Shim. Rock: a robust clustering algorithm for categorical attributes. In *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, pages 512–521, 1999. doi: 10.1109/ICDE.1999.754967.

[21] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200512992.

[22] G. Ball and D. Hall. *Isodata, a Novel Method of Data Analysis and Pattern Classification*. Architectural design. Stanford Research Institute, 1965. URL https://books.google.pt/books?id=Ti3BGwAACAAJ.

[23] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. 09 2009. ISBN 9780470317488.

[24] V. Estivill-Castro and J. Yang. Fast and robust general purpose clustering algorithms. In R. Mizoguchi and J. Slaney, editors, *PRICAI 2000 Topics in Artificial Intelligence*, pages 208–218, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-44533-3.

[25] J. Bezdek. *Pattern Recognition With Fuzzy Objective Function Algorithms.* 01 1981. ISBN 978-1-4757-0452-5. doi: 10.1007/978-1-4757-0450-1.

[26] R. R. Yager and D. P. Filev. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1279–1284, 1994. doi: 10.1109/21.299710.

[27] I. Gath and A. B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–780, 1989. doi: 10.1109/34.192473.

[28] R. J. Hathaway, J. C. Bezdek, and Yingkang Hu. Generalized fuzzy c-means clustering strategies using l/sub p/ norm distances. *IEEE Transactions on Fuzzy Systems*, 8(5):576–582, 2000. doi: 10.1109/91.873580.

[29] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. doi: 10.1109/TNN.2005.845141.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL `http://www.jstor.org/stable/2984875`.

[31] C. Aggarwal and C. Reddy. *DATA CLUSTERING Algorithms and Applications.* 08 2013.

[32] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, page 515–524, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134924. doi: 10.1145/584792.584877. URL `https://doi.org/10.1145/584792.584877`.

[33] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Cluster validity methods: Part i. *SIGMOD Rec.*, 31 (2):40–45, June 2002. ISSN 0163-5808. doi: 10.1145/565117.565124. URL `https://doi.org/10.1145/565117.565124`.

[34] R. Sokal and C. Michener. A statistical method of evaluating systematic relationships. *The University of Kansas Science Bulletin*, 38:1409–1438, 01 1958.

[35] R. Sokal and F. Rohlf. Sokal rr, rohlf fj. the comparison of dendrograms by objective methods. taxon 11: 33-40. *Taxon*, 11:33–40, 02 1962. doi: 10.2307/1217208.

[36] G. H. Ball. Data analysis in the social sciences: What about the details? In *Proceedings of the November 30–December 1, 1965, Fall Joint Computer Conference, Part I*, AFIPS '65 (Fall, part I), page 533–559, New York, NY, USA, 1965. Association for Computing Machinery. ISBN 9781450378857. doi: 10.1145/1463891.1463950. URL `https://doi.org/10.1145/1463891.1463950`.

[37] On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62 (320):1159–1178, 1967. ISSN 01621459. URL `http://www.jstor.org/stable/2283767`.

[38] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. doi: 10.1080/01621459.1971.10482356. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

[39] J. C. Dunn†. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1): 95–104, 1974. doi: 10.1080/01969727408546059. URL https://doi.org/10.1080/01969727408546059.

[40] D. Davies and D. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1:224 – 227, 05 1979. doi: 10.1109/TPAMI.1979.4766909.

[41] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

[42] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:25, 12 1985. doi: 10.1007/BF01908075.

[43] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983. ISSN 01621459. URL http://www.jstor.org/stable/2288117.

[44] E. Achtert, S. Goldhofer, H. Kriegel, E. Schubert, and A. Zimek. Evaluation of clusterings – metrics and visual support. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1285–1288, 2012.

[45] A. Severiano, F. R. Pinto, M. Ramirez, and J. A. Carriço. Adjusted wallace coefficient as a measure of congruence between typing methods. *Journal of Clinical Microbiology*, 49(11):3997–4000, 2011. ISSN 0095-1137. doi: 10.1128/JCM.00624-11. URL https://jcm.asm.org/content/49/11/3997.

[46] D. L. Wallace. A method for comparing two hierarchical clusterings: Comment. *Journal of the American Statistical Association*, 78(383):569–576, 1983. ISSN 01621459. URL http://www.jstor.org/stable/2288118.

[47] S. Kulczyński. *Die Pflanzenassoziationen der Pieninen.* Mémoires de l'Académie polonaise des sciences et des lettres: Classe des sciences mathématiques et naturelles. Imprimerie de l'Université, 1928. URL https://books.google.pl/books?id=cmSiHAAACAAJ.

[48] B. Desgraupes. Clustering indices. 11 2017.

[49] M. Meila. Comparing clusterings – an information based distance. *Journal of Multivariate Analysis*, 98:873–895, 05 2007. doi: 10.1016/j.jmva.2006.11.013.

[50] T. Cover and J. Thomas. *Elements of Information Theory*, volume 36, pages i – xxiii. 10 2001. ISBN 9780471200611. doi: 10.1002/0471200611.fmatter_indsub.

[51] S. V. Dongen. Performance criteria for graph clustering and markov cluster experiments. Technical report, NATIONAL RESEARCH INSTITUTE FOR MATHEMATICS AND COMPUTER SCIENCE IN THE, 2000.

[52] S. Wagner and D. Wagner. Comparing clusterings - an overview. *Technical Report 2006-04*, 01 2007.

[53] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: Part ii. *SIGMOD Rec.*, 31(3):19–27, Sept. 2002. ISSN 0163-5808. doi: 10.1145/601858.601862. URL https://doi.org/10.1145/601858.601862.

[54] M. Ramze Rezaee, B. Lelieveldt, and J. Reiber. A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, 19(3):237 – 246, 1998. ISSN 0167-8655. doi: https://doi.org/10.1016/S0167-8655(97)00168-2. URL http://www.sciencedirect.com/science/article/pii/S0167865597001682.

[55] M. Halkidi and M. Vazirgiannis. Clustering validity assessment: finding the optimal partitioning of a data set. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 187–194, 2001. doi: 10.1109/ICDM.2001.989517.

[56] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, and S. Wu. Understanding and enhancement of internal clustering validation measures. *IEEE Transactions on Cybernetics*, 43(3):982–994, 2013. doi: 10.1109/TSMCB.2012.2220543.

[57] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik. Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3):487–501, 2004. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2003.06.005. URL https://www.sciencedirect.com/science/article/pii/S0031320303002838.

[58] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

[59] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991. doi: 10.1109/34.85677.

[60] M. Kim and R. Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15):2353 – 2363, 2005. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2005.04.007. URL http://www.sciencedirect.com/science/article/pii/S016786550500125X.

[61] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970. ISSN 0022-2836. doi: https://doi.org/10.1016/0022-2836(70)90057-4. URL http://www.sciencedirect.com/science/article/pii/0022283670900574.

[62] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

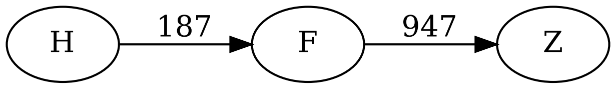
[63] A. F. H. Canhão and F. M. et al. Reuma.pt - the rheumatic diseases portuguese register. *Acta Reumatologica Portuguesa*, 36(1):45–56, Jan. 2011.

Appendix A

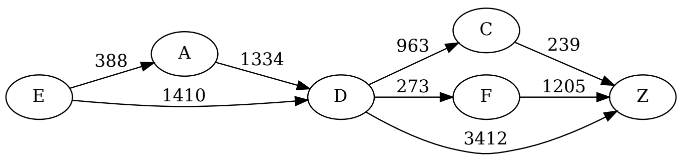
Reuma.pt clusters (automatic run)

A.1 Complete link, g=0.3, tp=0.25, 20 clusters

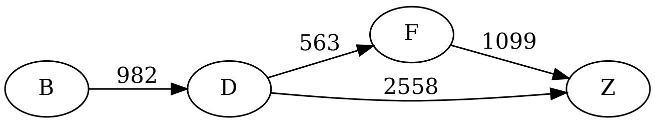
A.1.1 Cluster 1 - 2 patients



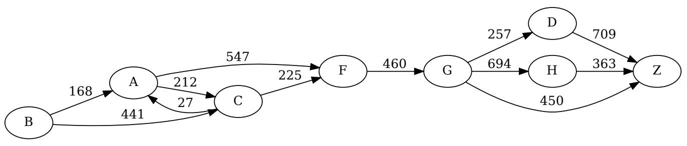
A.1.2 Cluster 2 - 3 patients



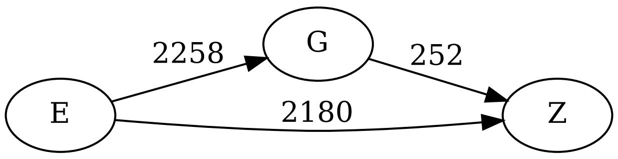
A.1.3 Cluster 3 - 4 patients



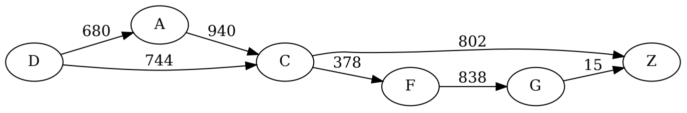
A.1.4 Cluster 4 - 5 patients



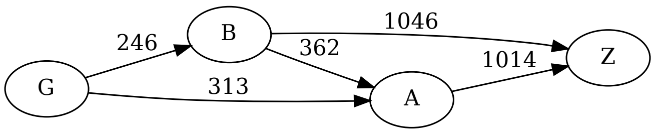
A.1.5 Cluster 5 - 5 patients



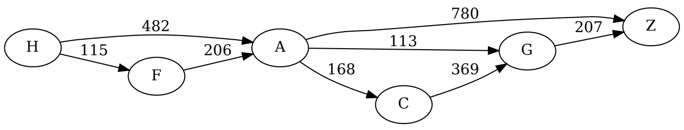
A.1.6 Cluster 6 - 6 patients



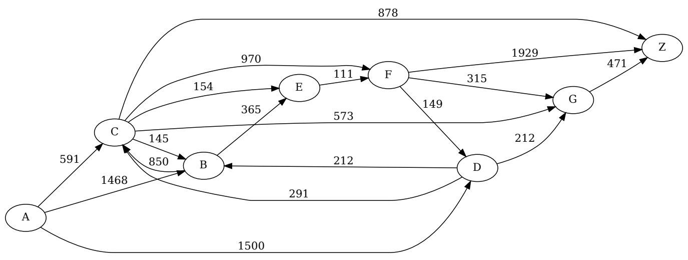
A.1.7 Cluster 7 - 7 patients



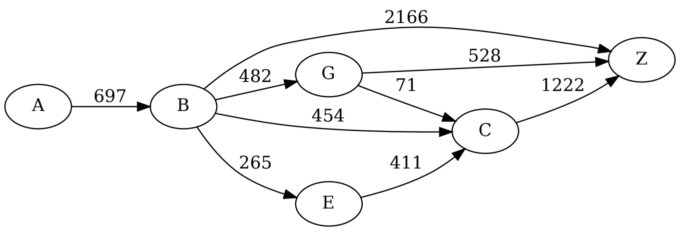
A.1.8 Cluster 8 - 7 patients



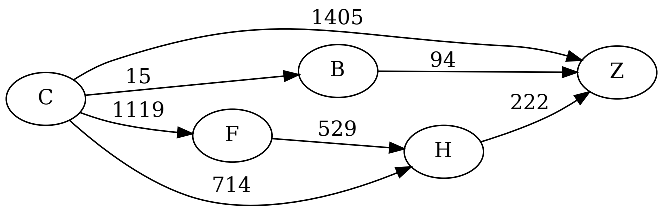
A.1.9 Cluster 9 - 13 patients



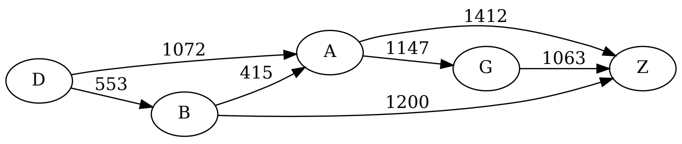
A.1.10 Cluster 10 - 13 patients



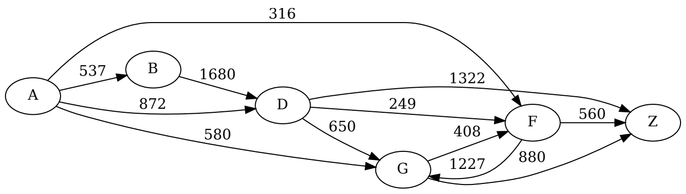
A.1.11 Cluster 11 - 16 patients



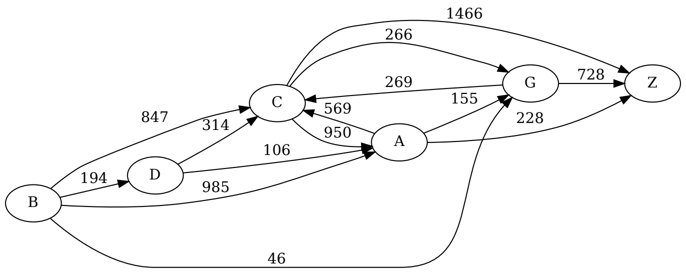
A.1.12 Cluster 12 - 16 patients



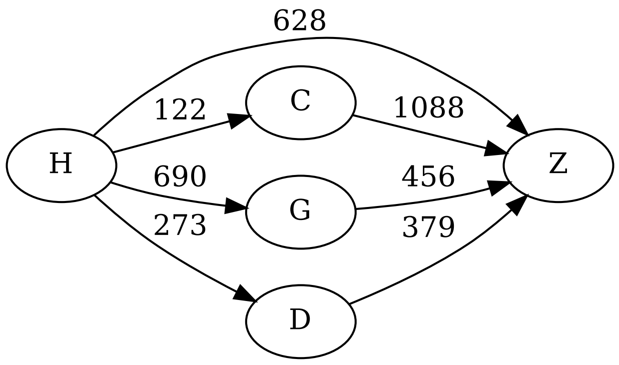
A.1.13 Cluster 13 - 18 patients



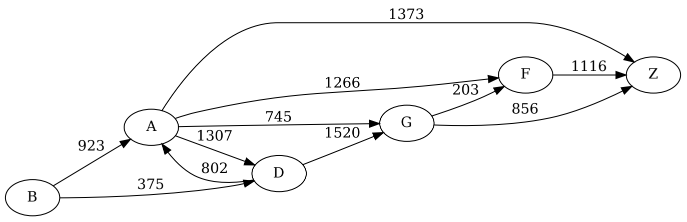
A.1.14 Cluster 14 - 21 patients



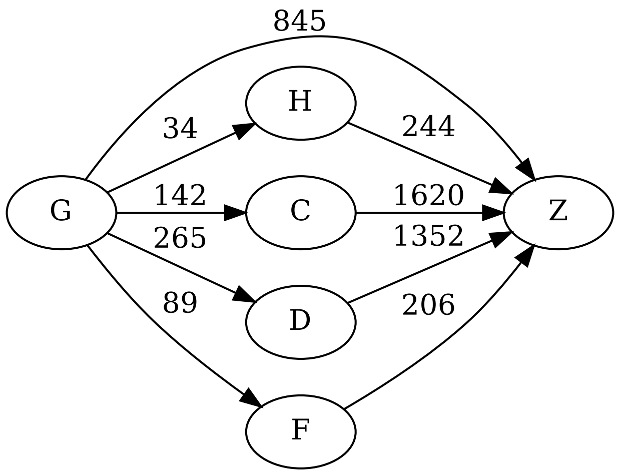
A.1.15 Cluster 15 - 22 patients



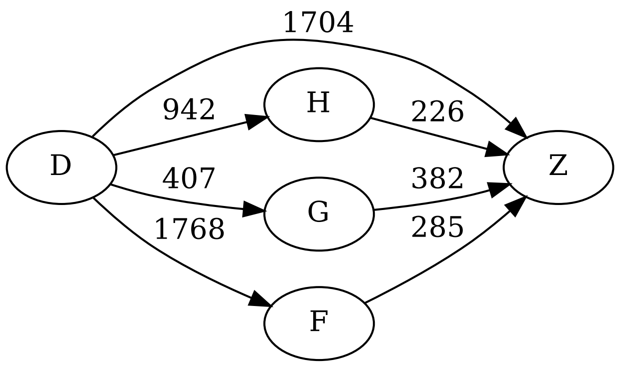
A.1.16 Cluster 16 - 31 patients



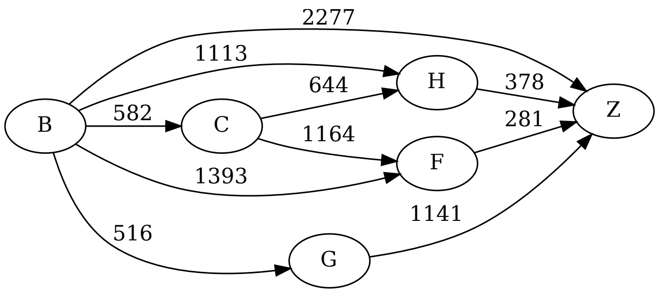
A.1.17 Cluster 17 - 34 patients



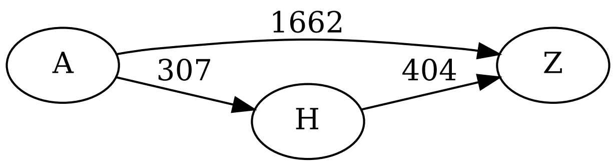
A.1.18 Cluster 18 - 41 patients



A.1.19 Cluster 19 - 55 patients

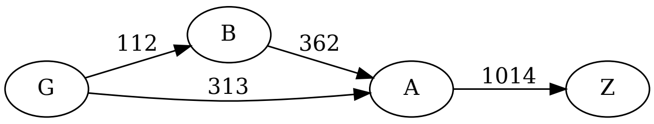


A.1.20 Cluster 20 - 105 patients

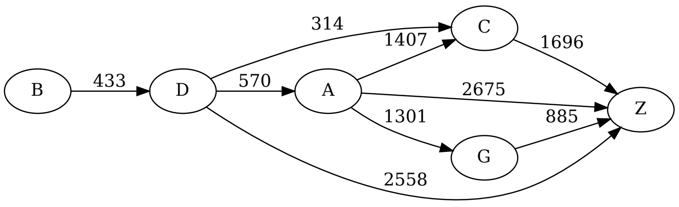


A.2 Ward link, g=0.4, tp=0.25, 20 clusters

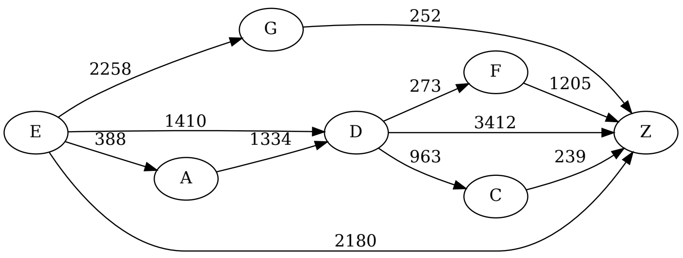
A.2.1 Cluster 1 - 5 patients



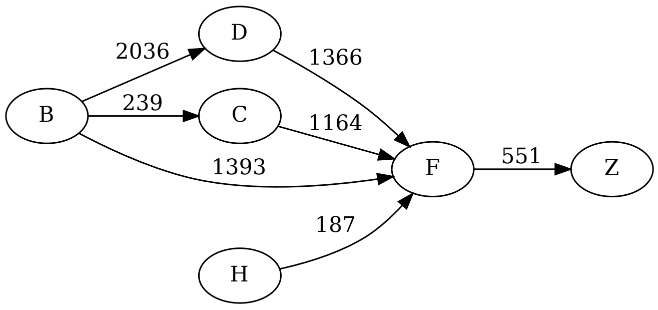
A.2.2 Cluster 2 - 7 patients



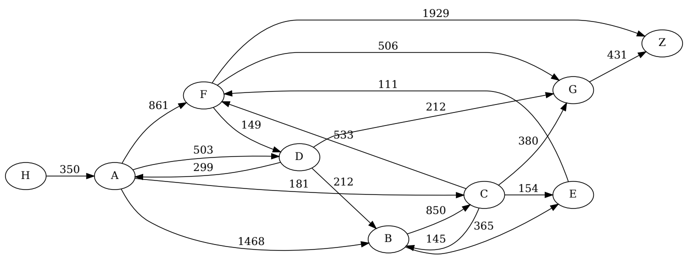
A.2.3 Cluster 3 - 8 patients



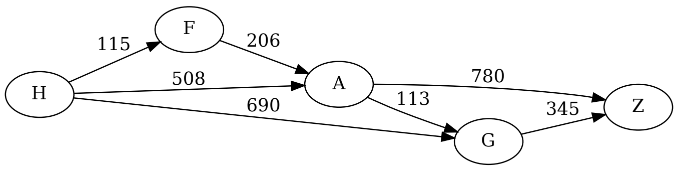
A.2.4 Cluster 4 - 8 patients



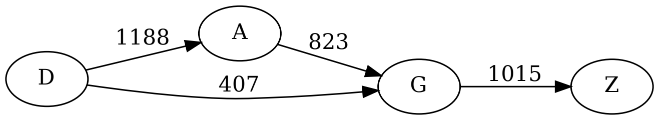
A.2.5 Cluster 5 - 8 patients



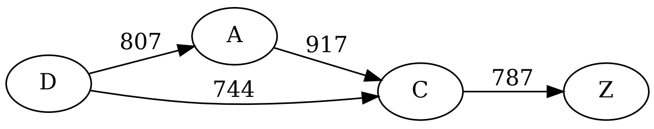
A.2.6 Cluster 6 - 8 patients



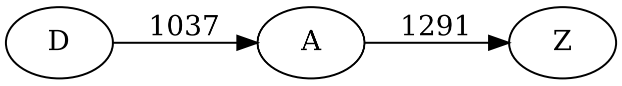
A.2.7 Cluster 7 - 9 patients



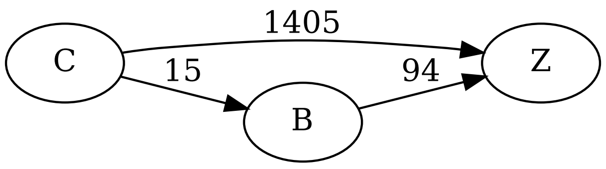
A.2.8 Cluster 8 - 10 patients



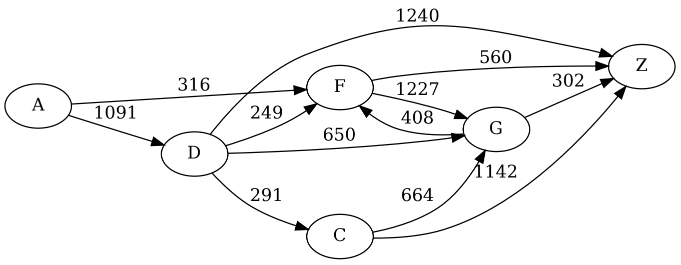
A.2.9 Cluster 9 - 10 patients



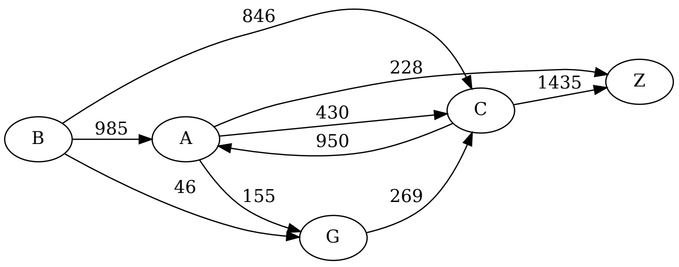
A.2.10 Cluster 10 - 14 patients



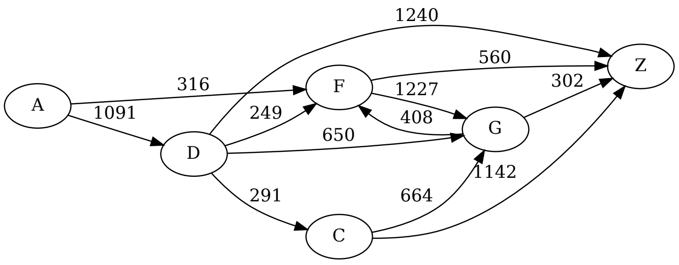
A.2.11 Cluster 11 - 14 patients



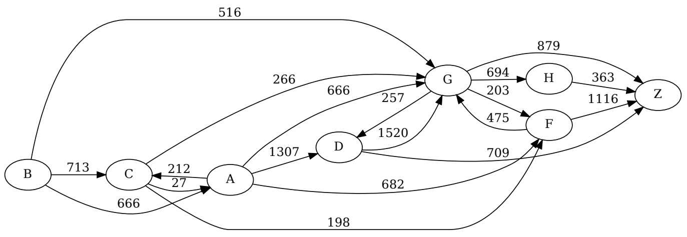
A.2.12 Cluster 12 - 17 patients



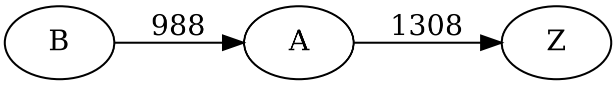
A.2.13 Cluster 13 - 17 patients



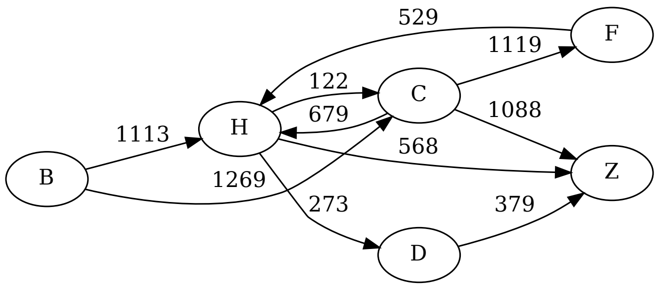
A.2.14 Cluster 14 - 18 patients



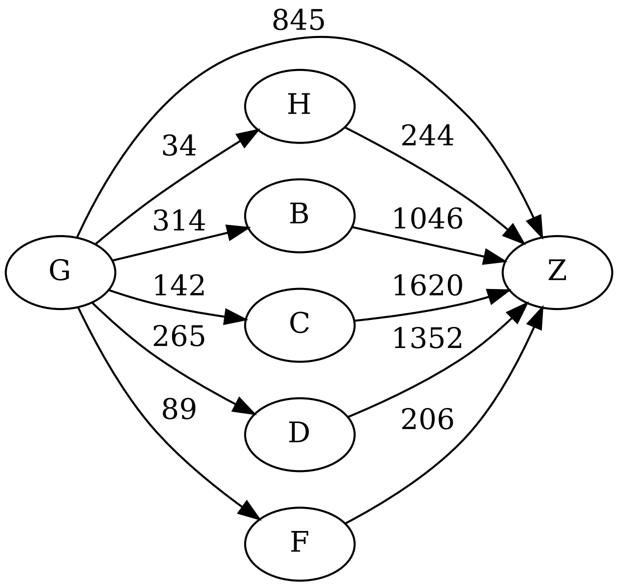
A.2.15 Cluster 15 - 20 patients



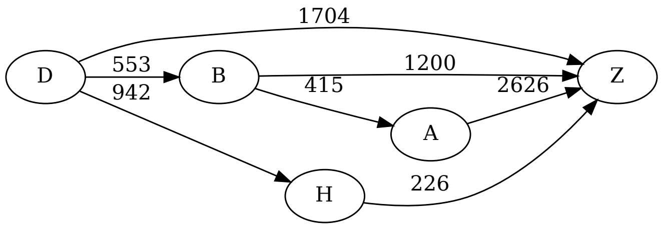
A.2.16 Cluster 16 - 24 patients



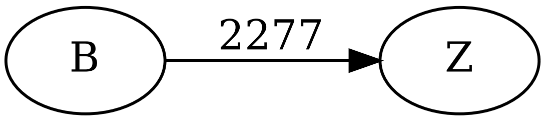
A.2.17 Cluster 17 - 36 patients



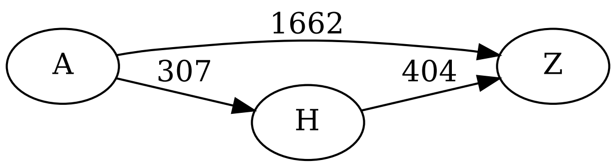
A.2.18 Cluster 18 - 40 patients



A.2.19 Cluster 19 - 46 patients



A.2.20 Cluster 20 - 105 patients

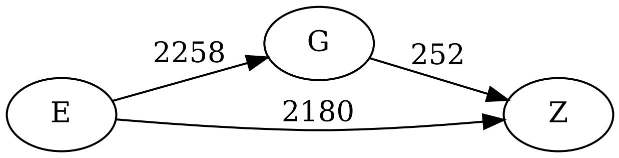


Appendix B

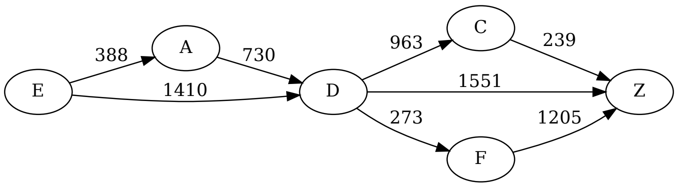
Reuma.pt clusters (semi-automatic run)

B.1 Ward link, g=0.2, tp=0.25, 16 clusters

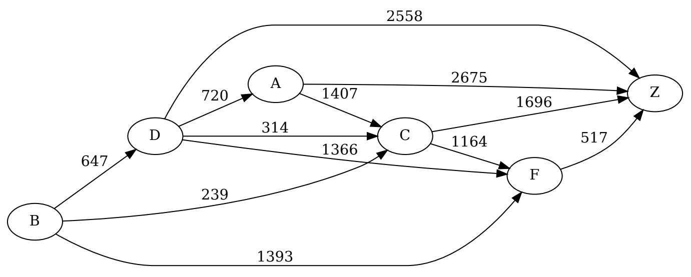
B.1.1 Cluster 1 - 5 patients



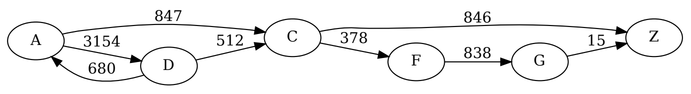
B.1.2 Cluster 2 - 9 patients



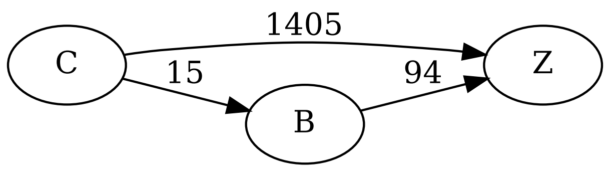
B.1.3 Cluster 3 - 13 patients



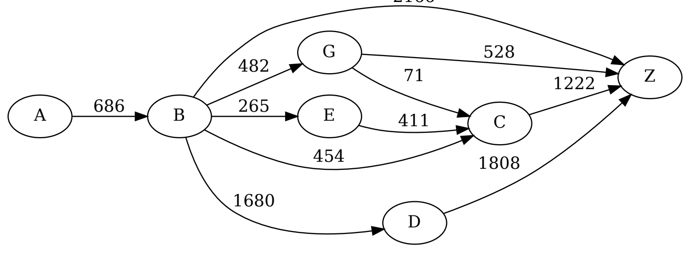
B.1.4 Cluster 4 - 13 patients



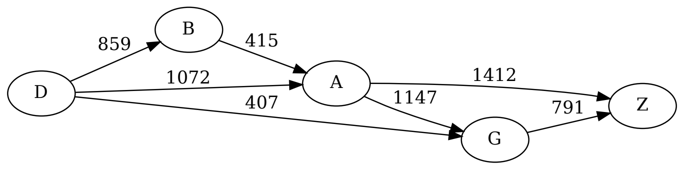
B.1.5 Cluster 5 - 14 patients



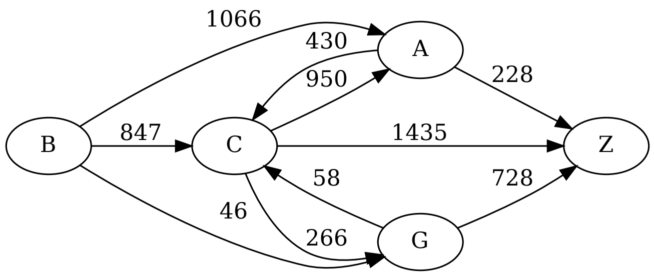
B.1.6 Cluster 6 - 14 patients



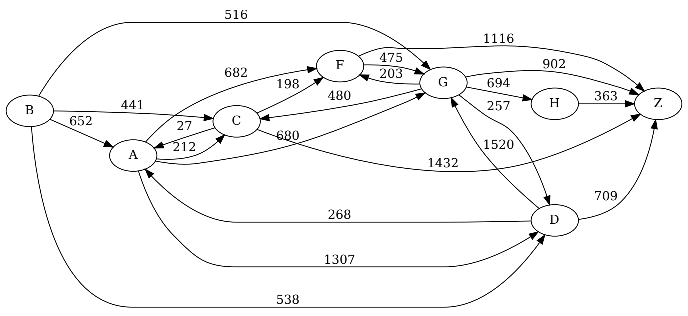
B.1.7 Cluster 7 - 16 patients



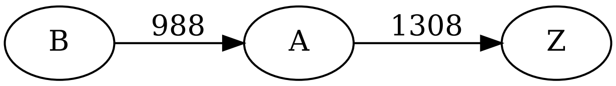
B.1.8 Cluster 8 - 18 patients



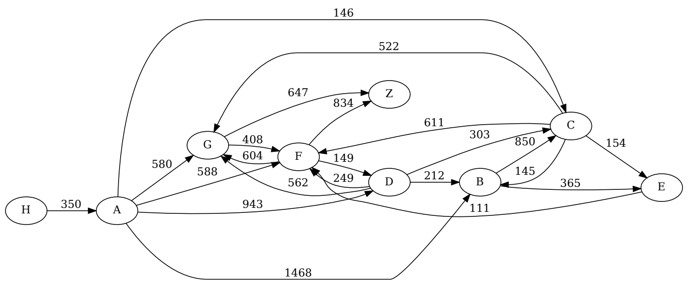
B.1.9 Cluster 9 - 18 patients



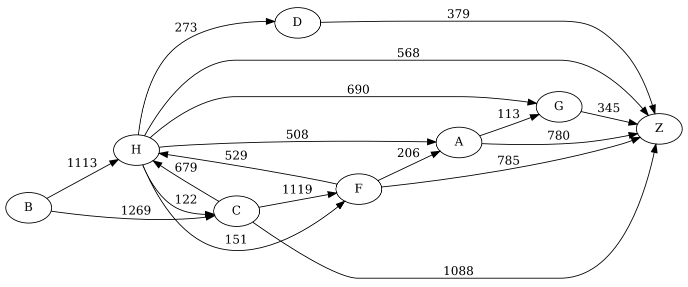
B.1.10 Cluster 10 - 20 patients



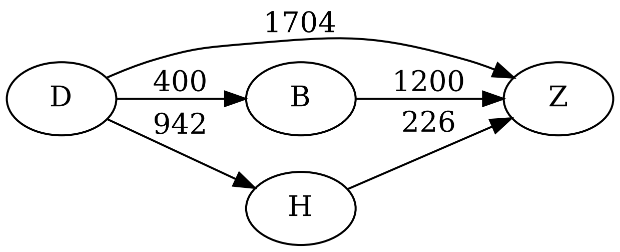
B.1.11 Cluster 11 - 20 patients



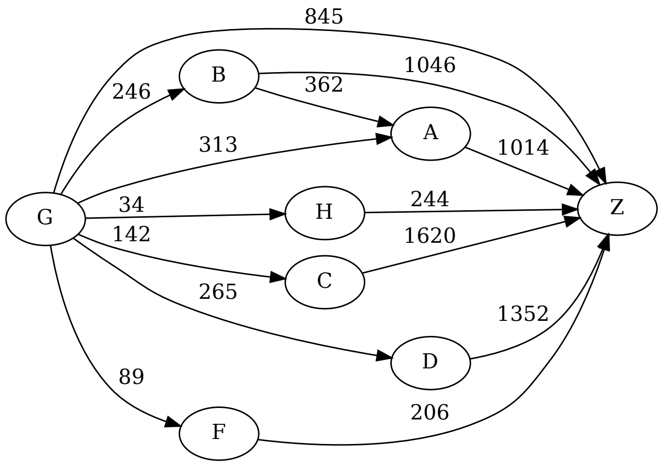
B.1.12 Cluster 12 - 33 patients



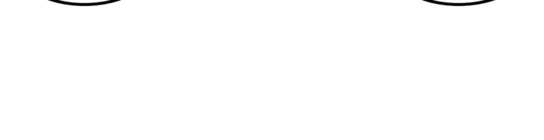
B.1.13 Cluster 13 - 39 patients



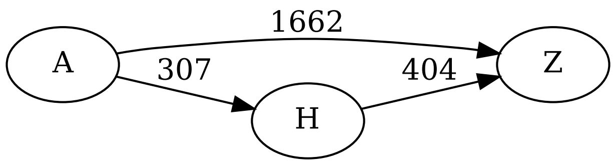
B.1.14 Cluster 14 - 41 patients



B.1.15 Cluster 15 - 46 patients

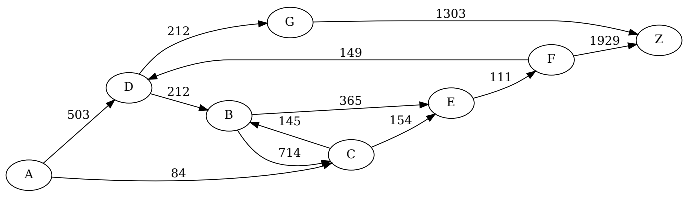


B.1.16 Cluster 16 - 105 patients

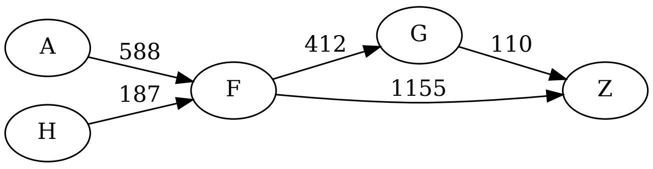


B.2 Complete link, g=0.7, tp=0.25, 20 clusters

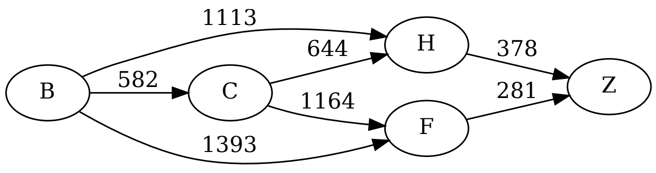
B.2.1 Cluster 1 - 2 patients



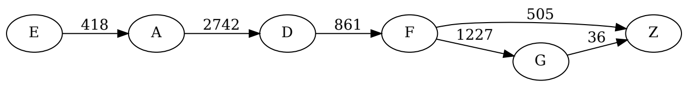
B.2.2 Cluster 2 - 4 patients



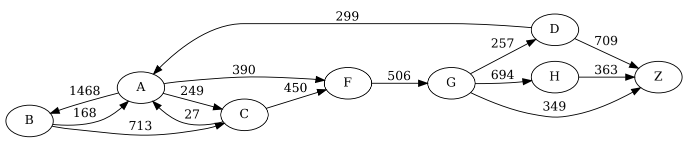
B.2.3 Cluster 3 - 5 patients



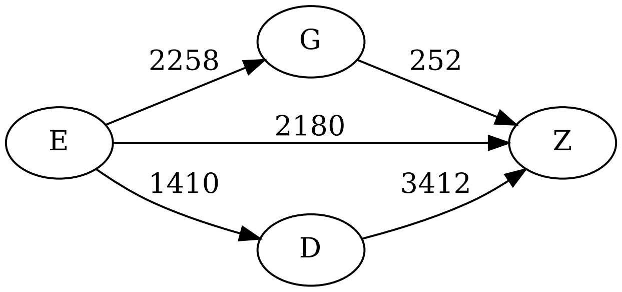
B.2.4 Cluster 4 - 5 patients



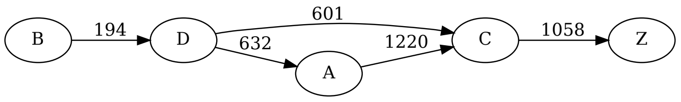
B.2.5 Cluster 5 - 6 patients



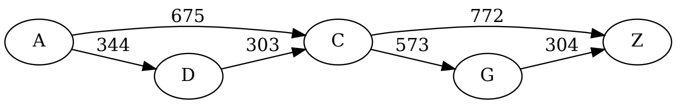
B.2.6 Cluster 6 - 6 patients



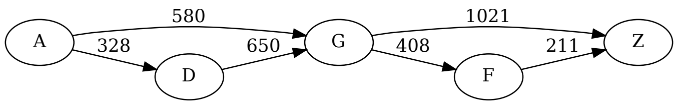
B.2.7 Cluster 7 - 7 patients



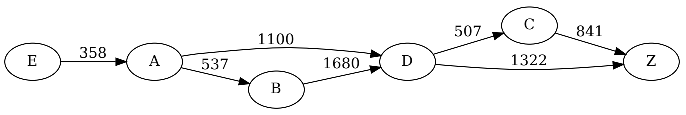
B.2.8 Cluster 8 - 8 patients



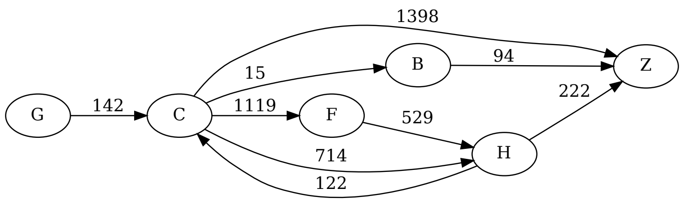
B.2.9 Cluster 9 - 9 patients



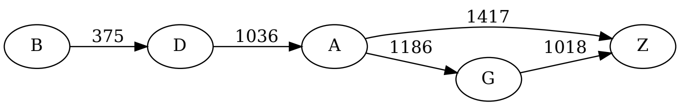
B.2.10 Cluster 10 - 10 patients



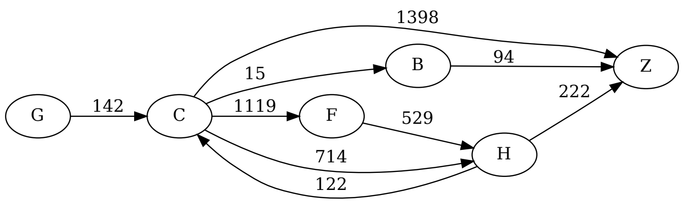
B.2.11 Cluster 11 - 13 patients



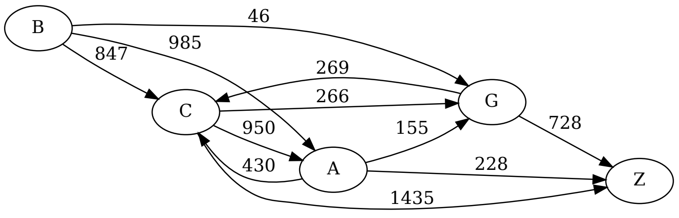
B.2.12 Cluster 12 - 15 patients



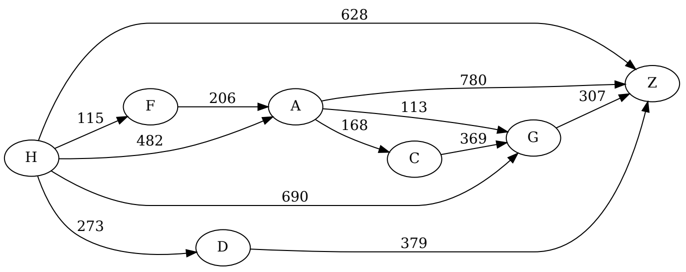
B.2.13 Cluster 13 - 18 patients



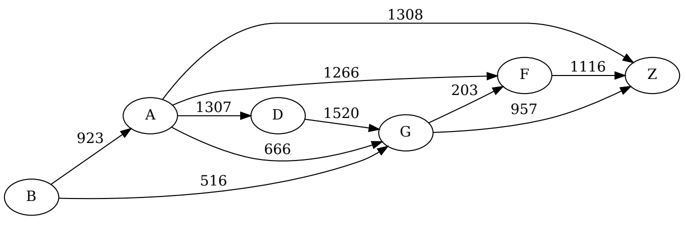
B.2.14 Cluster 14 - 19 patients



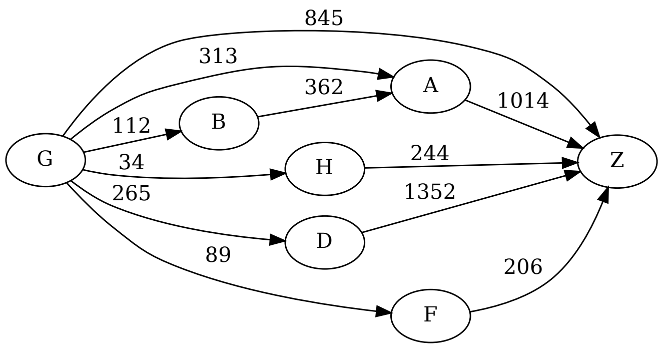
B.2.15 Cluster 15 - 28 patients



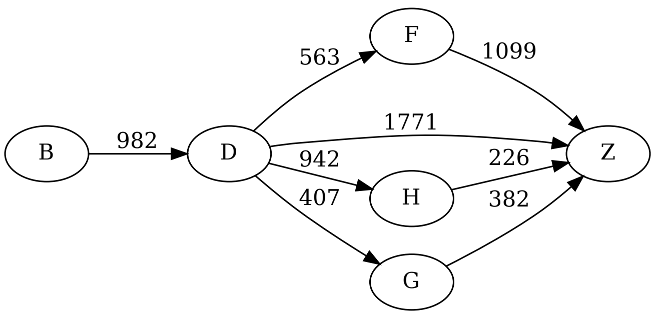
B.2.16 Cluster 16 - 33 patients



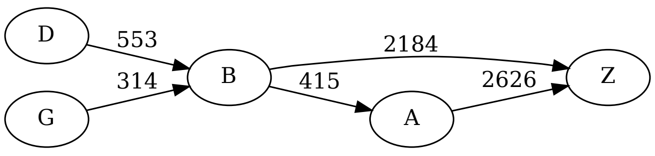
B.2.17 Cluster 17 - 38 patients



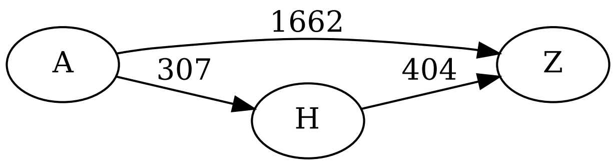
B.2.18 Cluster 18 - 43 patients



B.2.19 Cluster 19 - 51 patients



B.2.20 Cluster 20 - 105 patients


$$\begin{tikzcd}
A \arrow[rr, bend left, "1662"] \arrow[dr, "307"] & & Z \\
& H \arrow[ur, "404"] &
\end{tikzcd}$$