Healthcare pathway analysis: Optimal matching and classification

Salamata Ba[†] and Adja Deguene Gueye[‡]

Project advisors: Nicolas Savy [§]*Romain Demeulemeester* [¶]

Abstract.

The concept of healthcare pathways is increasingly becoming a source of interest in improving the quality of care and optimizing the use of medical resources.
In the medical field, sequence analysis, which takes into account the chronology of care received by patients, helps us identify patterns and build typologies of healthcare pathways.
We execute this project on the McVicar and Anyadike-Danes (2002) dataset which describes the transition from school to work for 712 individuals from 1993 to 1999. The conclusions drawn in this study will be used as part of sequence analysis for thyroid cancer patients' pathways.
Here, we choose the Optimal Matching distance to measure dissimilarities between sequences. Then, we apply techniques such as hierarchical clustering and K-medoids to categorize the sequential patterns into a limited number of groups.
Special emphasis is put on studying the stability of the resulting clusters. Thanks to a few sampling techniques such as resampling without replacement, parametric bootstrapping, or jackknife, we measure two instability indexes found in the literature: variation of information and Rand index.
Results show that, when the resample size is chosen carefully, resampling allows to make a trade-off between the number of clusters and quality of clustering. Some individuals located in a particular cluster can cause instability. By examining these individuals closer, we see that they have similar trajectories and are quite distinguishable from the rest of the cluster. Although the Rand index is easily interpretable, we find that variation of information is a better indicator for comparing these techniques.
A continuation of this research project would be to predict the trajectory of a patient given previous medical history and evaluate the quality of prediction with respect to instability.

[†]Student in the Department of Applied Mathematics, National Institute of Applied Sciences (INSA), Toulouse, FR (sba@etud.insa-toulouse.fr,salamata68@gmail.com, https://www.linkedin.com/in/salamata-ba/).

[‡]Student in the Department of Applied Mathematics, National Institute of Applied Sciences (INSA), Toulouse, FR (adgueye@etud.insa-toulouse.fr, adjadeguene@gmail.com,https://www.linkedin.com/in/adja-deguene-gueye/).

[§]Lecturer-researcher at Toulouse Mathematics Institute (IMT) (nicolas.savy@math.univ-toulouse.fr).

[¶]PhD Student at Toulouse University Hospital Centre (CHU) (romain.demeulemeester04@gmail.com).

1. Introduction.

The ramp-up of healthcare pathways concept makes us rethink our health and medico-social system around the patient and their needs. The French national healthcare system aims to include issues like the aging of the population, the prevalence of chronic diseases, and equal access to care into its strategy. Therefore, understanding and analyzing healthcare pathways is becoming a major concern. This can help better optimize the care of patients with chronic illnesses or describe the modalities of therapeutic management for some patients.

In the context of thyroid cancer, data is provided by the National Health Data System (SNDS) and the Centre Hospitalier Universitaire de Toulouse. To analyze this data as pathways, we consider it to be sequences of events. A variety of events can be represented: hospitalizations, consultations, therapy, surgery, etc. The temporal properties of the pathways require taking into account the chronology of events, the time between two events, or the time when an event occurs.

Several methods of sequence analysis are implemented in this paper using the R software to obtain a typology of the different pathways. We also wish to characterize the stability of our clustering. However, the previous work performed by Romain Demeulemeester gave clusters that were quite stable. This is why we perform our research on an example dataset from McVicar and Anyadike-Danes (2002) called the MVAD dataset where we hope to observe a less stable clustering. It is included in TraMineR, an R package used to find characteristics of sequential data based on the course of its events. This dataset originates from a survey that aimed to evaluate the transition from school to work of 712 individuals, it summarizes their monthly activity on a period of 72 months, from July 1993 to June 1999. The possible states for each month were: ”employment”, ”joblessness, “higher education”, ”further education”, ”school” and ”training”.

In this research project, we first conduct an exploratory analysis of our sequential data. Then, we perform a panel of clustering methods based on the Optimal Matching distance to determine groups formed by the individuals and the predominant pathways of each group. Finally, we take some time to study the stability of the clusters we have just formed and find a global indicator of this stability. This is a very important aspect because it allows us to control these different clusters as well as whether or not the trajectories are representative enough. We use different techniques from the literature and compare them to evaluate their efficiency. We have also designed a Shiny App to visualize our results in an interactive way.

2. Notion of sequence.

2.1. Definition of a sequence.

In fields like biology or social sciences, it can be important to consider time spent in each state (for example, the number of consecutive nucleotides in the case of DNA sequences) and transitions between these states. The states are events that occur at given time points.

Therefore, analyzing data in the form of state sequences has gained more and more importance. A sequence describes longitudinal data where the position of each successive state receives a meaningful interpretation in terms of age, date, or more generally elapsed time or distance from the beginning of the sequence.

We represent a sequence x by listing the successive elements that form it: $x = (x_1, x_2, \ldots, x_l)$, with $x_j \in A$ the alphabet (list of states).

In social sciences for example, the analysis of state sequences makes it possible to study life trajectories and to answer some questions such as what are the standard trajectories, do they obey a certain social norm, what is their evolution over time, how are they related to gender, social origin and other cultural factors or why are some people more likely to follow a chaotic trajectory or remain stable in a state?

2.2. The MVAD dataset.

This dataset was used by McVicar and Anyadike-Danes (2002) to study school-to-work transition in Northern Ireland. It contains sequences representing the monthly follow-ups of employment status for 712 individuals over a period of six years starting in the month where they were first eligible to leave compulsory education (July 1993). The states are school (SC), further education (FE), higher education (HE), training (TR), employment (EM) ,and joblessness (JL).

2.3. Visualizing state sequences with the TraMineR package.

TraMineR offers several tools for visualizing and computing descriptive statistics of a set of sequences. For example, the index plot renders a set or subset of individual state sequences (using `seqiplot() or seqIplot()` functions) and the frequency plot represents them with respect to their frequencies (`seqfplot()`). A sequence is represented by horizontally stacked boxes that are colored according to the state of the individual at successive positions (Figure 1).

Figure 2 shows the state distributions of the whole set, the transversal entropy index, modal states and the mean time spent in each state.

The `seqdplot()` function produces a graphical view of the state distributions showing the

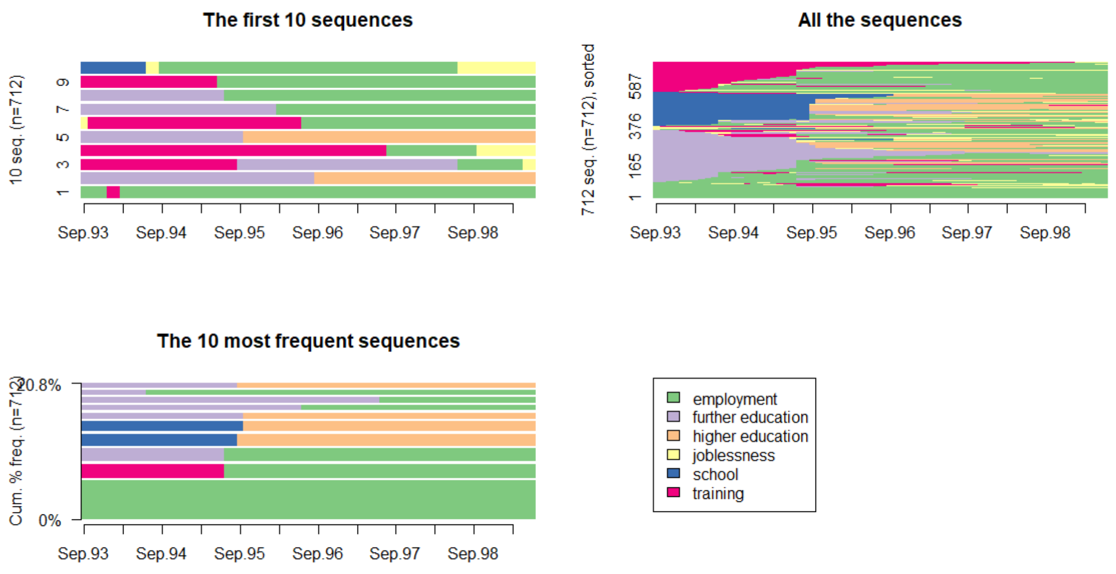


Figure 1: Index and frequency plots

states frequencies for each time unit.

The transversal entropy plot: (`seqHtplot()` function) displays the evolution over positions of the cross-sectional entropies. If we denote by p_i the proportion of cases in state i at the considered position, the entropy is:

$$h(p_1, \ldots, p_a) = -\sum_{1}^{a} p_i \log(p_i)$$

The entropy is 0 when all cases are in the same state and is maximal when we have the same proportion of cases in each state. It can be seen as a measure of the diversity of states observed at the considered position. (see [1] for more details).

The `seqmsplot()` function displays modal states which are the most frequent states at each position. The height of the bar at each position is proportional to the frequency of the displayed state.

Finally, one can also easily visualize the average time spent in each state with function `seqmtplot()`.

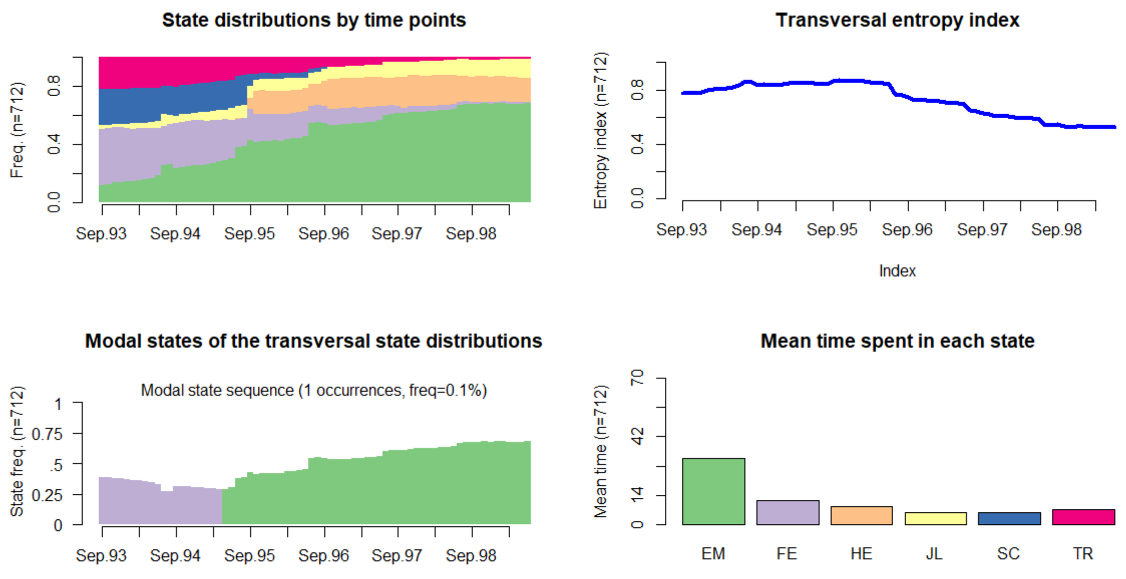


Figure 2: Some descriptive statistics

3. Clustering of sequences.

3.1. Measuring sequence dissimilarities: Optimal Matching.

The Optimal Matching algorithm (0M) is a very common approach used to assess the dissimilarity of time-ordered sequences. It is inspired by the sequence alignment and dynamic programming methods used in molecular biology, in particular for the comparison of proteins or DNA sequences assumed to be homologous.

The algorithm defines the distance between two sequences as the minimal cost of transforming one sequence into the other. Indeed, to transform a sequence, two types of operations are considered: the insertion or deletion of a state in the sequence (indel) or the substitution of one state by another one. Then, costs are assigned to these operations and the algorithm minimizes the total cost of the transformation by calculating the cheapest set of elementary operations.

Let us consider two sequences $S_1 = (s_1, s_2, \ldots, s_{T_1})$ et $S_2 = (s'_1, s'_2, \ldots, s'_{T_2})$.
If we assume that $A = a_1, a_2, \ldots, a_n$ is the sequence of operators such that the application of all the operators of this sequence A to the first sequence S_1 gives the second sequence S_2 and $c(A) = \sum_{i=1}^{n} c(a_i)$ the total cost of the transformation, then the optimal matching dissimilarity is:

$$d(S_1, S_2) = \min_A \{c(A) \text{ such that } S_2 = A(S_1) = a_1 \circ a_2 \circ \ldots a_n(S_1)\}$$

As we can expect, assigned costs affect the results of the Optimal Matching algorithm. The choice of a cost system is arbitrary, it depends on the importance given to the temporal distance which separates the same events and to the distance between events that occur at the same time units. The operations of insertion and deletion generate a time lag. On the contrary, substitutions keep time order. It is possible to favor substitutions or even prohibit time shifts by setting a high indel cost relatively to substitution costs.

TraMineR offers a `seqdist()` function which calculates the Optimal Matching dissimilarity. The indel cost is usually set as constant and independent of the position and state. The costs of substitutions are assumed to be constant when we use the option method= "CONSTANT" or determined from the estimated transition rates as $2 - p(s_i|s_j) - p(s_j|s_i)$ where $p(s_i|s_j)$ is the probability of observing state s_i at time $t+1$ given that state s_j has been observed at time t when we set option method = "TRATE". This option sets high substitution costs when changes between s_i and s_j are seldom and low costs if they are frequent. More details are provided in the TraMineR documentation [1].

3.2. Clustering techniques.

3.2.1. Hierarchical Clustering.

In our case, hierarchical clustering aims to identify groups of healthcare pathways with similar patterns and build typologies for these sequences. This clustering method uses the dissimilarity matrix computed by the Optimal Matching algorithm to decide which sequences should be combined or not. This algorithm tries to create the hierarchy of clusters in the form of a tree. This tree-shaped structure is known as the dendrogram.
Hierarchical clustering works as follows:

1. Consider every data point as an individual cluster
2. Merge the clusters which are highly similar (the two nearest clusters using the dissimilarity matrix) into the same cluster.
3. Update the dissimilarity matrix between clusters
4. Repeat steps 2 and 3 until only a single cluster remains.

Dissimilarity between individuals is given by the optimal matching algorithm and we find dissimilarity between clusters using Ward's method: distance between clusters equals to the weighted distance between their centroids:

$$d^2(C_1, C_2) = \frac{n_1 n_2}{n_1 + n_2} d^2(g_1, g_2)$$

where g_i is the center of gravity of the cluster C_i. This criterion allows to minimize the total within-cluster variance or, equivalently, maximize the between-cluster variance. This means that we try to make elements of the same cluster as close as possible and elements belonging to different clusters as far apart as possible.

3.2.2. K-medoids algorithm.

The K-medoids clustering algorithm, also called Partitioning Around Medoids algorithm, attempts to minimize the distance between sequences labeled to be in a cluster and a sequence designated as the center of that cluster (the medoid). Medoids are representative objects of a cluster. The sum of dissimilarities of the medoid to all the objects in the cluster is minimal. The algorithm works as follows:

1. A set of medoids is chosen at random.
2. The distances to the other sequences are computed using the dissimilarity matrix.
3. Data are clustered according to the medoid they are most similar to.
4. The medoids set is optimized via an iterative process.

3.2.3. Discussion on the selection of the clustering algorithm.

Hierarchical clustering:

For this clustering technique, it is not necessary to specify the number of clusters in advance (we explore all possibilities). However, that only postpones this decision. We can also evaluate the different partitions found (one for each possible value of the number of clusters) with a measure such as the silhouette coefficient, an evaluation of how close each sequence in one cluster is to sequences in the neighboring clusters. Moreover, it is easy to decide the number of clusters by merely looking at the dendrogram.

On the other hand, its algorithmic complexity is heavy. At each iteration, to decide which clusters to join, we will need the pairwise distances between all pairs of points in the dataset. For this reason, hierarchical clustering is more suitable for small datasets.

K-medoids algorithm

Although we have to choose the number of clusters in advance, this clustering method is simple to understand and easy to implement. K-Medoids algorithm is fast and converges in a fixed number of steps. Furthermore, it is less sensitive to outliers than hierarchical clustering.

The main disavantage is that it may results in different clusters for different runs on the same dataset because initially, we pick k medoids randomly from the full dataset and assigns them to each cluster one by one so that it becomes initial medoid of that cluster.

Remark: Why not use the K-means algorithm ?
Medoids are similar in concept to centroids, but they are always restricted to be members of the data set while K-means centroids are fictitious. Medoids are especially useful when the centroid is not representative of the dataset which is the case for 3D images and gene expression or when we want to find a representative clustering using a distance other than squared euclidean distance (recall that K-means is designed for Euclidean distance).

3.3. Application to MVAD dataset.

In this section, we show the clustering results we get with the MVAD dataset.

Hierarchical Clustering

Figure 3 shows the Dendrogram obtained by applying a hierarchical clustering.

Once the dendrogram is created, it is necessary to cut the tree to determine the optimum number of clusters. This can be done based on the inertia decrease graph displayed in Figure 4.

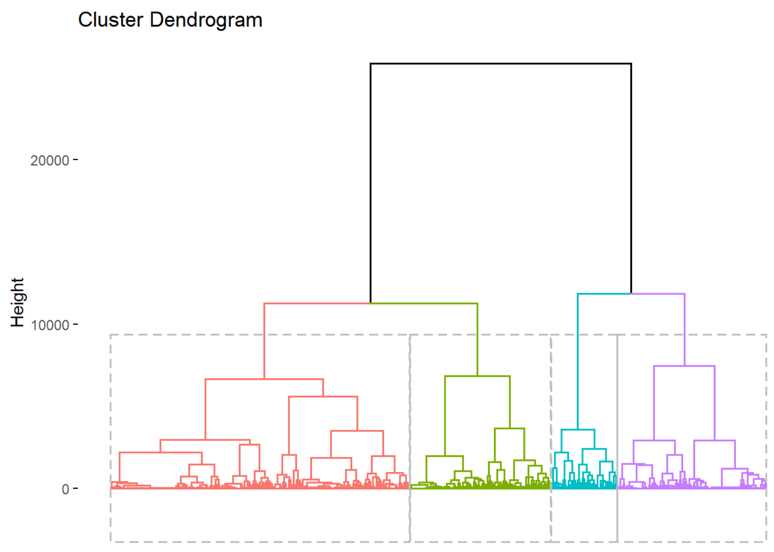


Figure 3: Dendrogram: Hierarchical clustering of MVAD dataset

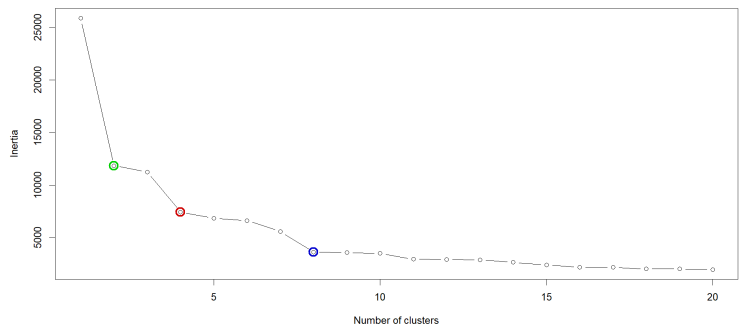


Figure 4: Decrease of Inertia

We first partition our dataset into four clusters. Using the functions of TraMineR, we observe the cluster patterns by plotting their transversal state distributions:

We can see in Figure 5 that the first cluster groups youngsters who make an early transition to employment just after the end of compulsory schooling. The second cluster is formed by young people who continue their studies. The third group corresponds to slow transition to employment with first an important spell of further education. The last is made up of people who go through long periods of unemployment or training after compulsory schooling.

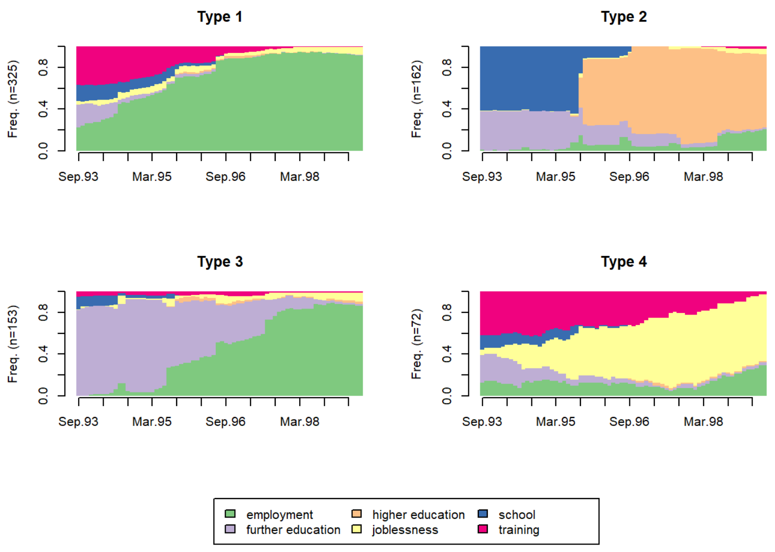


Figure 5: Transversal state distribution for each cluster

KMedoids algorithm

As for hierarchical clustering, we choose to partition our dataset into 4 clusters. We find that the medoids are: 32, 60, 117 and 510. We then observe these medoids in order to get a better understanding of each cluster:

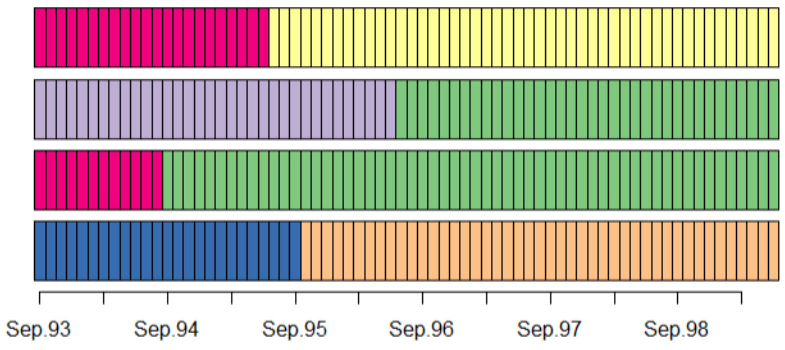


Figure 6: Sequence representation of the 4 medoids.

We find very similar trajectories to the ones we had using hierarchical clustering. This can be further characterized by looking at a cross table which compares these two clusterings:

HC	M32	M60	M117	M510
1	6	298	17	4
2	128	0	33	1
3	0	0	153	0
4	1	4	8	59

4. Study of the clustering stability.

4.1. Background.

Clustering stability is defined as the ability of a clustering algorithm to render similar results when applied to several datasets from the same underlying model (von Luxburg, 2010 [4]). Studying stability can help us choose an optimal number of clusters.

How can we measure clustering stability/instability?

Let $d(\mathcal{C}, \mathcal{C}')$ be the distance between two clusterings $\mathcal{C}$ and $\mathcal{C}'$.
For a fixed probability distribution $\mathcal{P}$, a fixed number of clusters K and a fixed sample size n, the clustering instability is the expected distance between two clusterings $\mathcal{C}_{\mathcal{K}}(\mathcal{S}_{\mathrm{n}}), \mathcal{C}_{\mathcal{K}}(\mathcal{S}'_{\mathrm{n}})$ on different datasets $\mathcal{S}_{\mathrm{n}}, \mathcal{S}'_{\mathrm{n}}$ of size n. We define:

$$\mathrm{Instab}(K, n) = \mathbb{E}[d(C_K(S_{\mathrm{n}}), C_K(S'_{\mathrm{n}}))] \tag{4.1}$$

How can we choose the number of clusters while preserving stability?

A general method for measuring clustering stability is described as follows:

- For $k = 2, \ldots, k_{max}$:
    - Let us generate perturbed versions S_b(with $b = 1, ..., b_{max}$) of our original data set, for example by subsampling or adding noise.
    - We cluster each dataset S_b using the clustering algorithm into k clusters to obtains clustering C_b.
    - Then, we compute pairwise distances $d(C_b, C'_b)$ between these different clusterings.
    - Instability is the mean distance between these clusterings:

      $$\widehat{\mathrm{Instab}}(K, n) = \frac{1}{b_{max}^2} \sum_{b,b'=1}^{b_{max}} d(C_b, C'_b) \tag{4.2}$$

    - Finally, we choose the parameter k that gives the lowest instability:

      $$K := \underset{k}{\mathrm{argmin}}\, \widehat{\mathrm{Instab}}(K, n)$$

4.2. Different distances used to measure stability.

Clusterings defined on the same dataset can be easily compared using common clustering distances such as Rand index, Jaccard coefficient, or variation of information distance. There are many other criteria used to measure the distance between clusterings (Hamming distance, minimal matching distance, etc).

The idea is to count the pairs of points on which two clusterings C and C' agree or disagree on their classification. We set:

N_{11} the number of points pairs that are in the same cluster under both C and C'
N_{00} the number of points pairs that are in different clusters under both C and C'
N_{01} the number of points pairs that are in the same cluster under C' but not under C
N_{10} the number of points pairs that are in the same cluster under C but not under C'

$N_{11} + N_{00}$ is the number of agreements between C and C' while $N_{01} + N_{10}$ is the number of disagreements between C and C'. These counts can be obtained from the contingency table as in Subsection 3.3.
Furthermore, $N_{11} + N_{00} + N_{01} + N_{10} = \binom{n}{2} = n(n-1)/2$ where n is the number of points in the whole dataset.

Rand index [5]

The Rand index represents the frequency of occurrence of agreements over the total pairs, or the probability that C and C' will agree on a randomly chosen pair.

$$R(C, C') = \frac{N_{11} + N_{00}}{n(n-1)/2}$$

The value of the Rand index is between 0 and 1, it is 0 if the two clusterings do not agree on any pair of points and equal to 1 if, on the contrary, the two clusterings are the same.

Jaccard index [5]

Also called the Jaccard similarity coefficient, this index is an improved version of the Rand index. Like the rand index, the two clusterings become more similar as it gets closer to 1.
It is given as:

$$J(C, C') = \frac{N_{11}}{N_{11} + N_{01} + N_{10}}$$

Variation of information [5]

Let P(k) be the probability of the outcome being in cluster C_k of clustering $\mathcal{C}$. If we assume that each point has an equal probability of being picked, we have $P(k) = \frac{n_k}{n}$.

We can define entropy H of a clustering C as:

$$H(C) = -\sum_{k=1}^{K} P(k)\, \log\, P(k)$$

In order to introduce variation of information, we first define mutual information between two clusterings $\mathcal{C}$ and $\mathcal{C}'$.
Denote by $P(k,k')$ the probability that a point belong to cluster C_k of clustering $\mathcal{C}$ and cluster $C'_{k'}$ of clustering $\mathcal{C}'$. Mutual information between $\mathcal{C}$ and $\mathcal{C}'$ is given by:

$$I(C,C') = -\sum_{k=1}^{K}\sum_{k'=1}^{K'} P(k,k')\, \log\, \frac{P(k,k')}{P(k)P'(k')} \tag{4.3}$$

It is the information one clustering has about the other.
Hence, variation of information, which measures the distance between two clusterings, is as follows:

$$VI(C,C') = H(C) + H(C') - 2I(C,C') = H(C|C') + H(C'|C) \tag{4.4}$$

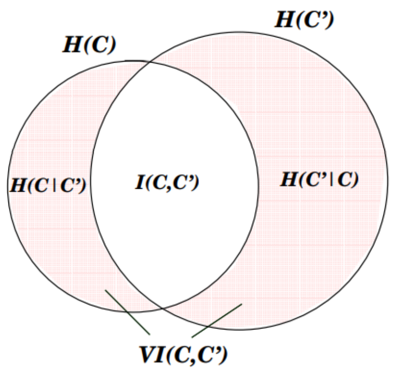


Figure 7: Representation of the variation of information (Meila, 2003 [5])

4.3. Three approaches to studying stability.

We return to our MVAD dataset. Here, we will study instability based on the procedure described in section 4.1 for different versions of our dataset.

4.3.1. Resampling without replacement.

A resample is obtained by selecting at random a subset of size s of the data points. We choose a resampling without replacement method to avoid repetitions of sequences leading to wrong conclusions. For different values of k (number of clusters), we independently generate m resamples of fixed size s. Then, we apply to each of these subsets the same clustering procedure that was used on the full dataset, using the same parameters. Finally, the extent to which the clustering assignments obtained from the resamples agrees with that of the full sample is measured using the Rand index or the variation of the information.

This method can help find a good compromise between the number of clusters k and the stability of the clustering: the optimal number of clusters maximizes the Rand index and minimizes information variation. Note that the size of the subsamples can have an effect on the results. We can generalize this method by pairwise comparisons of our resamples.
The results on the MVAD dataset are shown in Section 5.

4.3.2. Identification and study of unstable individuals by jackknife.

The jackknife is one of the earliest yet simplest techniques for sampling. It is a method that requires less computational power compared to others.
Suppose we have a dataset $S = (x_1, x_2, ..., x_n)$. For $i = 1, 2, ..., n$, the ith jackknife sample is the dataset for which the ith observation of S has been removed. We have:

$$S_{(i)} = (x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n), \forall i = 1, .., n.$$

In our study, we use the jackknife method to find individuals that alter stability. To this end, we construct all jackknife samples in which we have removed the ith individual of our dataset, for i=1,...,712. We then compute the distance matrix for each sample: we take the distance matrix of the original clustering and we remove its component (i, i). Using this distance matrix, we perform K-medoids clustering. Finally, we measure instability between each resulting clustering and the original clustering of all individuals except individual i.

We then search for individuals for which variation of information is significantly greater than 0 or Rand index smaller than 1. If we consider 4 clusters, there are 54 individuals that meet this criteria.

By taking a closer look, we notice that all unstable individuals except one are in the same cluster (cluster of medoid 60). We also plot the sequences that influence stability versus the rest of the cluster in order to spot some differences:

We can see that unstable individuals are the ones that undergo training (exclusively) for a shorter period than the rest of the cluster before being employed. To study further characteristics of these individuals, we can perform logistic regression on the cluster of medoid 60, where the event is true when the individual has an unstable sequence.

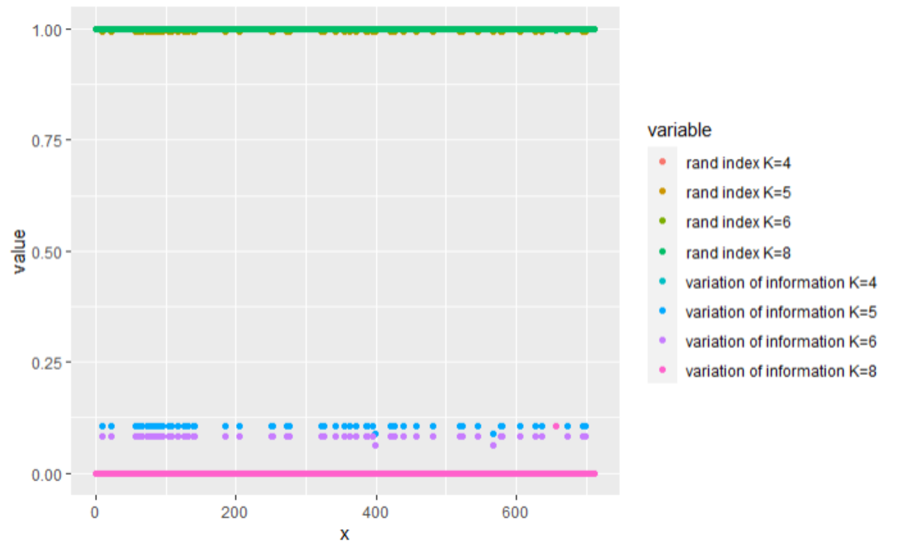


Figure 8: Rand index and Variation of information of the xth jackknife for different K

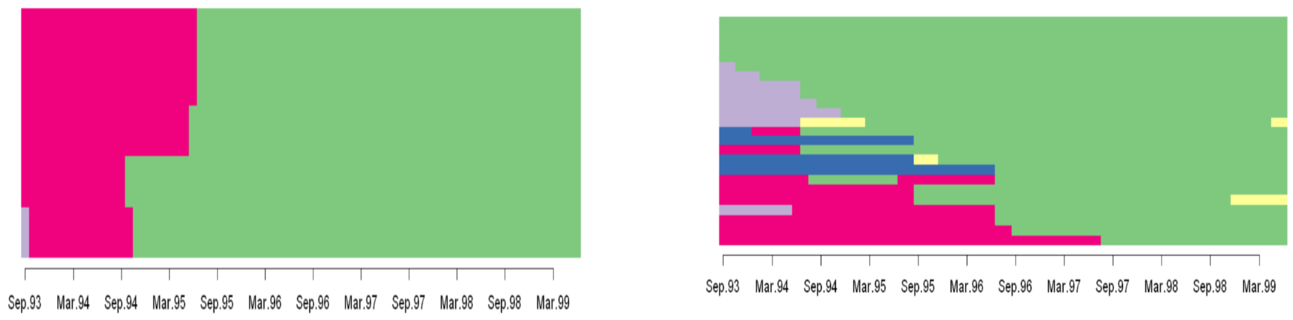


Figure 9: Plot of the unstable sequences vs. Plot of the rest of the cluster

We have deleted non-significant variables then computed the result of this logistic regression in the following table:

(Intercept)	Belfastyes	N.Easternyes	Southernyes	S.Easternyes	Grammaryes	funempyes	livbothyes
0.1008383	2.3209769	2.1781206	0.9916048	2.4917641	0.9633586	1.3416169	1.3097214

We can see that the characteristics of these individuals are very different when their sequences are unstable and when they are not. For example, the probabilities of coming from

Belfast or an Eastern school for a person with an unstable sequence are 2 times higher than for the rest of the cluster. The chances for their father to be unemployed (funemp) are also 34% higher. In addition, they are 31% more likely to be living with both parents.

4.3.3. Parametric bootstrap sampling.

Here, we implement a recent method based on the parametric bootstrap and introduced by Matthias Studer in 2021 [7]. The idea is to compare ”the cluster quality of an observed typology with the quality obtained by clustering similar but non-clustered data”(Studer, 2021), which he calls the null model. As a null model, three models are proposed: “randomized sequencing” which measures the added value of the typology compared with the situation in which all paths are likely, “randomized duration” which measures the captured structure of the data, and a combination of both models. The parametric bootstrap procedure works by repeating n times the following operations:

1. generate similar but non-clustered data using a “null” model
2. cluster the generated data
3. compute the value of the clustering quality index (CQI) of interest

We then obtain the n CQI values of the clustering of our null model that we can compare with the n values of our own clustering. We want the cluster quality to be much higher than the one obtained with non-clustered data. We choose the average silhouette width (ASW) as our CQI. This index links for each sequence its distance to the center of its cluster (homogeneity) to its distance to the closest other cluster (separation) in order to measure clustering quality.

In Figure 10, we can see the results obtained using k-medoids clustering for k ranging from 2 to 7. It can be noted that using hierarchical clustering gives the same results.

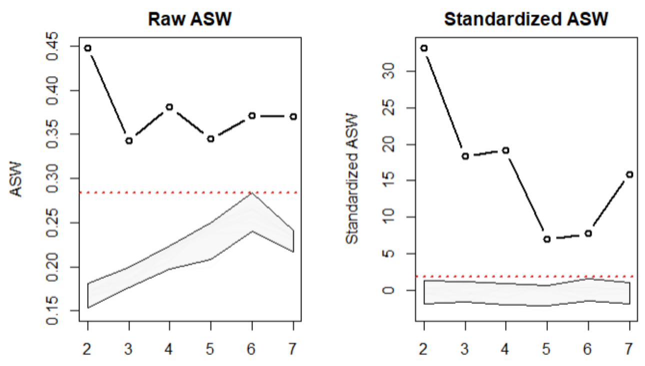


Figure 10: Observed and bootstrapped values of the ASW for a varying number of clusters using the combined (sequencing and duration) randomization null model

This method is not conclusive because ASW is higher than the null method for all numbers of clusters.

4.4. Does the time step have an influence on stability?.

We would like to know if the way the dataset is constructed, i.e the time step between two state observations, has an influence on stability. To answer this question, we construct a data set by grouping states by trimester: for each trimester, we take the most recurrent state to represent that trimester. This leaves us with 25 states for each individual.

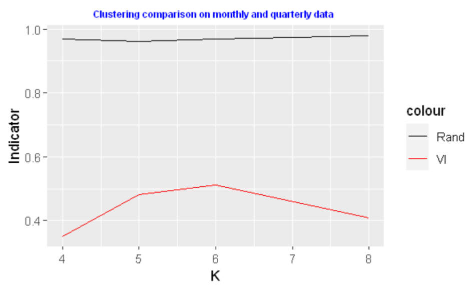


Figure 11: Influence of time step on clustering

In the figure Figure 11, we compare the clustering obtained on the full monthly dataset with the one obtained on the full quarterly dataset. We observe that the Rand index is almost constant whatever the number of clusters k, which suggests that the time step has no influence on clustering. However, there is still a small variation in the information between the two clusterings. Hence, we study the behavior of these clusterings when resampling is used.

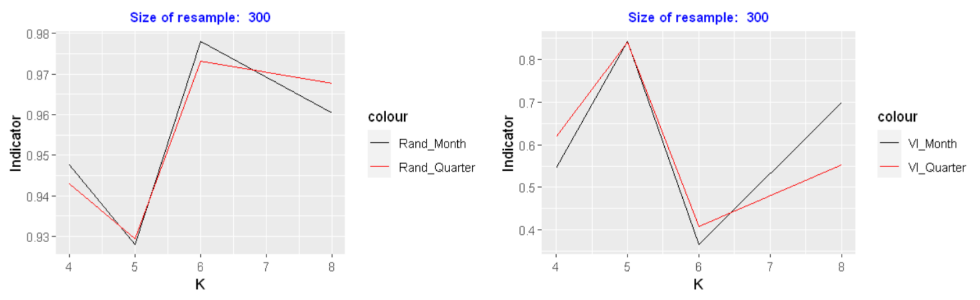


Figure 12: Stability of clusterings depending on the time step

As we can see in Figure 12, on the MVAD data set, the stability does not depend very much on the time step.

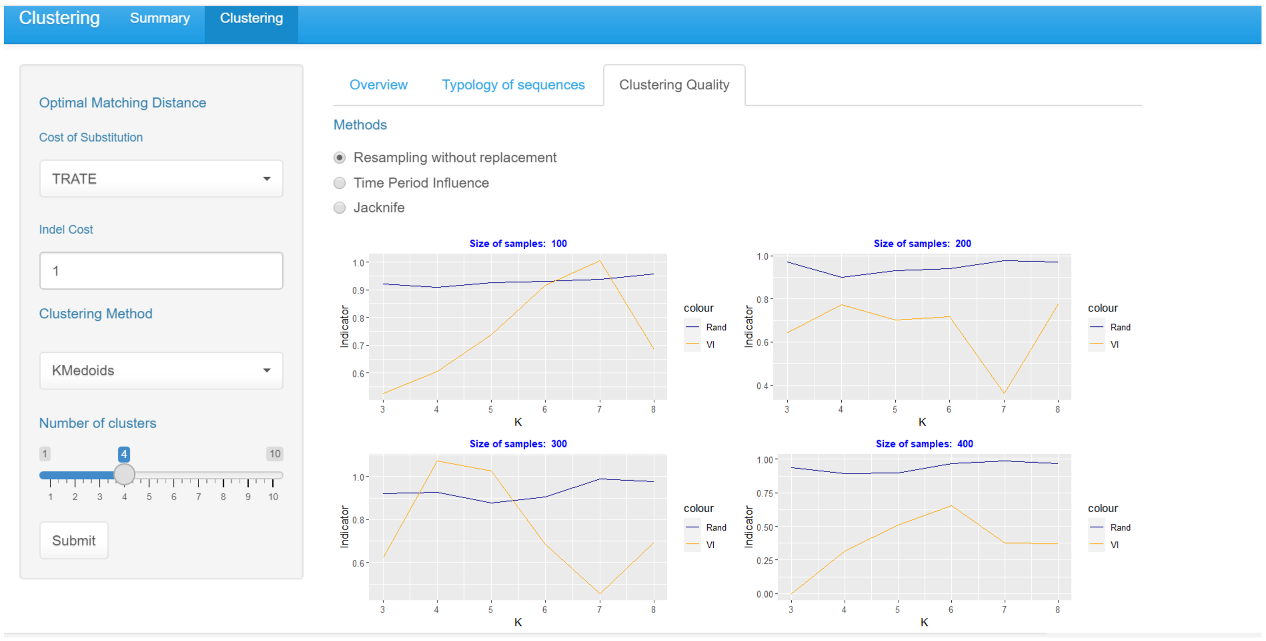
5. Shiny App.

We have designed a Shiny App to visualize our results in an interactive way. Several parameters can be changed directly in the application, such as the costs for calculations of the optimal matching distance (substitution and indel costs), the clustering algorithm to use or the number of clusters that we want to consider.

We thus find in the application visualizations of the typology of sequences obtained according to the chosen parameters, as well as the results of the clustering quality study with the three approaches described in Subsection 4.3.

Results of resampling without replacement:

We apply the procedure described in Subsection 4.3.1 to study whether our clustering results are stable against resampling for a given number k of clusters. We perform our resampling scheme with 5 resamples and various resampling sizes to see its influence on the results. The full dataset consists of 712 individuals. Here, we choose resamples of size 100, 200, 300 and 400.



Optimal Matching Distance

Cost of Substitution: TRATE

Indel Cost: 1

Clustering Method: KMedoids

Number of clusters: 4

Methods

● Resampling without replacement

○ Time Period Influence

○ Jacknife

Figure 13: Results of resampling without replacement

First, we can see in Figure 13 that depending on the chosen distance (Rand or variation of information), we do not always obtain the same optimal number of clusters.

Recall that the optimal number of clusters k is defined as the one which minimizes instability. However, it is important to note that Instab(k, n) trivially scales with k, regardless of the underlying data structure. It can then be necessary to have previously defined a range of values for k or to have used normalization protocols to correct the trivial scaling of the instability (See [2]).

If we simultaneously observe Rand index and variation of information, we would conclude that 7 is the optimal number of clusters for resampling sizes of 200, 300 and 400 because Rand index is high and variation in information is low at these values in the figure. However, if we choose a small resampling size (100), the results are not similar, we would prefer to choose 3 clusters.

The resampling size should be chosen carefully. If it is too small, we can destroy the structure that we want to discover and if it is too large we can observe trivial stability. We recommend trying one-third of the data as the resample size.

This resampling technique can be a good way to choose the right number of clusters for a given study depending on the quality of the Rand index or the variation in information that one is ready to lose.

Relative sensitivity of instability indicators

Figure 13 also shows that the variation of information is a much more sensitive distance than the Rand index which does not vary much. We can also observe these results in Figure 8 where we notice that variation of information is easier to visualize than Rand index. However, Rand is easier to explain.

6. Conclusion.

State Sequence Analysis (SSA) is an effective way to extract a great deal of information from data relating to changes over time at an individual level, such as MVAD's social data .

Thanks to optimal matching distance, we can build typologies of sequences by applying adapted clustering algorithms like hierarchical clustering or K-medoids algorithm.

Researchers are increasingly focusing on studying the stability of clusters formed by these algorithms. There are various methods and indicators for validation of results obtained by sequence clustering and for determining the optimal number of clusters. In this paper, we have used methods based on resampling without replacement, parametric bootstrapping and Jackknife.

Resampling allows to make a trade-off between the number of clusters and quality of clustering. However, the resampling size should be chosen so as not to change the structure of the data too much.

Parametric boostrap, which is a recent method for determining the optimal number of clusters, has not shown its relevancy in this study.

The jackknife method has allowed us to determine individuals that decrease stability and find which of their other characteristics influenced this behaviour.

Moreover, we have noticed that variation of information is more sensitive to stability than the Rand index. It detects more changes between two compared clusterings.

This study will hopefully help set the basis to constructing stable clusters for patients in healthcare.

To continue this project, we could try defining a method for choosing the costs used in the Optimal Matching algorithm with respect to cluster stability, predict the class of a new sequence based on the closest medoid (for example) and/or find covariates on which could depend clustering results.

Appendix A. R code.

```
###### Loading data and packages

# Package names
packages <- c("TraMineR", "TraMineRextras", "seqhandbook", "cluster",
              "WeightedCluster", "factoextra", 'fossil','gridExtra',
              'mcclust', 'ramify')
install.packages(packages)

library(TraMineR)
library(TraMineRextras)
library(factoextra)
library(seqhandbook)
library(cluster)
library(WeightedCluster)
library(fossil)
library(gridExtra)
library(mcclust)
library(ramify)


#loading data
data(mvad)
print(seqstatl(mvad[, 17:86])) # alphabet: list of states


#####  Descriptive analysis of MVAD dataset

# Creating a state sequence object
mvad.alphabet <- c("employment", "FE", "HE", "joblessness", "school", "training")

mvad.labels <- c("employment", "further education", "higher education",
                 "joblessness", "school", "training")

mvad.scodes <- c("EM", "FE", "HE", "JL", "SC", "TR")
mvad.seq <- seqdef(mvad, 17:86, alphabet = mvad.alphabet,
                   states = mvad.scodes, labels = mvad.labels, xtstep = 6)




##### Visualizing sequences

par(mfrow = c(2, 2))
#Plot the first 10 sequences in the mvad.seq sequence object.
```

```
seqiplot(mvad.seq, with.legend = FALSE, border = NA)
#Plot all the sequences in the data set, sorted by states from start
seqIplot(mvad.seq, sortv = "from.start", with.legend = FALSE)
#Plot the 10 most frequent sequences.
seqfplot(mvad.seq, with.legend = FALSE, border = NA)
#Plot the legend for the state colors
seqlegend(mvad.seq)


##### Visualizing some descriptive analysis
-------------------------------------------------------------------------
par(mfrow = c(2, 2))
#plot the state distributions by time points
seqdplot(mvad.seq, with.legend = FALSE, border = NA)
#plot the transversal entropy index
seqHtplot(mvad.seq)
#Plot the sequence of modal states of the transversal state distributions
seqmsplot(mvad.seq, with.legend = FALSE, border = NA)
#Plot the mean time spent in each state of the alphabet.
seqmtplot(mvad.seq, with.legend = FALSE)


##### Compute Optimal Matching distance

--------------------------------------------------------------
#Compute the optimal matching distances using substitution costs
#based on transition rates observed in the data and a 1 indel cost.
dist.om1 <- seqdist(mvad.seq, method = "OM", indel = 1, sm = "TRATE")


##### Hierarchical clustering

--------------------------------------------------------------
#build a Ward hierarchical clustering of the sequences from
#the optimal matching distances

clusterward1 <- hclust(dist(dist.om1), method = "ward.D2")

#plot the dendrogram
plot(as.dendrogram(clusterward1), leaflab = "none")

--------------------------------------------------------------
#Choosing the number of clusters: decrease of inertia

options(repr.plot.width=5, repr.plot.height= 5)
# represent the jumps of inertia of the dedrogram according
#to the number of selected classes.

inertie <- sort(clusterward1$height, decreasing = TRUE)
```

```
plot(inertie[1:20], type = "b", xlab = "Number of clusters",
     ylab = "Inertia")
points(c(2, 4, 8), inertie[c(2, 4, 8)],
       col = c("green3", "red3", "blue3"), cex = 2, lwd=3)
```

```
# We choose a number of 4 clusters:
fviz_dend(clusterward1, k = 4, show_labels = FALSE, rect = TRUE)
```

```
# Typologies of sequences

cl1.4 <- cutree(clusterward1, k = 4)
cl1.4fac <- factor(cl1.4, labels = paste("Type", 1:4))

#Plot all the sequences within each cluster.
seqIplot(mvad.seq, group = cl1.4fac, sortv = "from.start")

#Plot the state distribution within each cluster.
seqdplot(mvad.seq, group = cl1.4fac, border = NA)

# sort the sequences by Multidimensional scaling to make the mats more readable
ordre <- cmdscale(as.dist(dist.om1), k = 1)
seqIplot(mvad.seq, group = cl1.4fac, sortv = ordre,
          space = 0, border = NA, yaxis = FALSE)

#Plot the representative sequences of each cluster.
seqrplot(mvad.seq, diss = dist.om1, group = cl1.4fac,
    border = NA, cex.legend = 1.5, cex.plot = 5)

#heatmap
seq_heatmap(mvad.seq, clusterward1, labcol=17:86)

# size of classes
table(cl1.4)



##### KMedoid Clustering
```

```
# choice of k=4 clusters
MedoidClust <- wcKMedoids(dist.om1, k=4, initialclust=cl1.4)

# Contingency table: comparison with hierarchical clustering
table(cl1.4, MedoidClust$clustering)

# Typologies
MedoidClust.4 <- as.numeric(as.factor(MedoidClust$clustering))
```

```
seqplot.rf(mvad.seq, diss=dist.om1,
           group=MedoidClust$clustering,which.plot="medoids")

# Medoids
medoid_index= sort(unique(MedoidClust$clustering))
medoid_index
seqplot(mvad.seq[medoid_index,], idxs=as.vector(medoid_index))

# Visualing clusters

#another way to apply KMedoids algorithm
Pam <- pam(x = dist.om1, k=4, diss=TRUE) #

clusplot(dist.om1,Pam$clustering,diss=TRUE,
         labels=2,color=TRUE,
         col.txt=Pam$clustering, shade=T )

# Quality of clustering: silhouette coef
fviz_silhouette(Pam)
```

```
##### Stability Study


###### Resampling without replacement
```

```
ResamplingWR <- function(NbSamples=3, K=2:5, SizeResampling=300,
                         index=0, ClusterAlgo, distance){
  #ClusterAlgo: "KMedoids" or "Hierarchical Clustering"
  #index=0 for Rand and 1 for Variation of Information

  dfR <- matrix(nrow = length(K), ncol = NbSamples)
  dfVI <- matrix(nrow = length(K), ncol = NbSamples)

  il=1 #row incrementer in df
  mean.index.rand <- rep(0,length(K))
  mean.index.vi <- rep(0,length(K))

  for (k in K){

    #clustering on full dataset
    if (ClusterAlgo == "KMedoids"){
      clusterOrigin <- pam(distance,k,diss=T)$clustering
    }
    if (ClusterAlgo == "Hierarchical Clustering"){
      clusterOrigin <- cutree(hclust(dist(distance), method = "ward.D2"), k)
    }

    #clustering on resamples
    for (b in 1:NbSamples){

      #build resamples
      isample <- sample(1:nrow(mvad.seq), size=SizeResampling)
      dist.sample <- distance[isample,isample]

      #clustering
      if (ClusterAlgo == "KMedoids"){
        clusters.sample <- pam(x = dist.sample, k, diss=T)$clustering
      }
      if (ClusterAlgo== "Hierarchical Clustering"){
        clusters.sample <- cutree(hclust(dist(dist.sample), method = "ward.D2"), k)
      }

      # computing indexes between clustering on full dataset
      # and clustering on resamples
      dfR[il,b] <- rand.index(clusterOrigin[isample],clusters.sample)
      dfVI[il,b] <- vi.dist(clusterOrigin[isample],
                            clusters.sample, parts = FALSE)

    }

    #compute the mean of indexes for the b resamples
    mean.index.rand[il] <- mean(dfR[il,])
    mean.index.vi[il]<- mean(dfVI[il,])
    il= il + 1

  }


  dfR <-  data.frame(dfR,mean.index.rand )
  dfVI <-  data.frame(dfVI,mean.index.vi )

  rownames(dfR) <- paste("K=",K)
  rownames(dfVI) <- paste("K=",K)
  colnames(dfR) <- c(paste("Subsample",1:NbSamples, sep=""),"meanRand")
  colnames(dfVI) <- c(paste("Subsample",1:NbSamples, sep=""),"meanVI")

  if (index==0){return (dfR)} #Rand
  if (index==1) {return (dfVI)} #VI

}
```

```
PlotResamplingWR <- function(K=1:7, SizeResampling, NbSamples=5,
                                          ClusterAlgo ,  distance){

   plot_df <- data.frame(K, Rand= ResamplingWR(NbSamples,K, SizeResampling,
                                 index=0,ClusterAlgo ,  distance)$meanRand,

                                 VI= ResamplingWR(NbSamples,K, SizeResampling,
                                 index=1,ClusterAlgo ,  distance)$meanVI  )

   fig <- ggplot(plot_df ,  aes(x = K)) +

     geom_line(  aes(x = K,y=Rand,color="Rand"))+
     geom_line(aes(x=K,  y=VI,  color= "VI"))+  ylab("Indicator")+
     labs(title=paste("Size  of  samples:  ",  SizeResampling))+
     scale_color_manual(values= c("Rand"= "darkblue","VI"= "orange"))+
     theme(plot.title  =  element_text(hjust=0.5,color="blue",size=10,face="bold"))

   return  (fig)

}
```

```
fig1 <- PlotResamplingWR(K=1:7,  SizeResampling  =  100,  NbSamples=5,
                                        ClusterAlgo ="KMedoids"  ,
                                        distance  =  dist.om1)

fig2 <- PlotResamplingWR(K=1:7,  SizeResampling  =  200,  NbSamples=5,
                                        ClusterAlgo  =  "KMedoids"  ,
                                        distance  =  dist.om1)

fig3 <- PlotResamplingWR(K=1:7,  SizeResampling  =  300,  NbSamples=5,
                                        ClusterAlgo ="KMedoids",
                                        distance  =  dist.om1)

fig4 <- PlotResamplingWR(K=1:7,  SizeResampling  =  400,  NbSamples=5,
                                        ClusterAlgo  =  "KMedoids",
                                        distance=  dist.om1)

grid.arrange(fig1 ,  fig2 ,  fig3 ,  fig4 ,  ncol=2 )
```

```
##### Bootstraping

pamRange <- wcKMedRange(dist.om1,  2:7)
bcq2 <- seqnullcqi(mvad.seq ,  pamRange,  R=10,  model=c("combined"),
                        seqdist.args=list(method="OM",  sm='TRATE'),
                        kmedoid  =  TRUE)

## Different  kind  of  plots
```

```
plot(bcq2, stat="ASW", type="line")
plot(bcq2, stat="ASW", type="density")
plot(bcq2, stat="ASW", type="boxplot")
```

```
##### JACKNIFE: Identification and study of unstable individuals

jackseq=list()
dist_jk<-list()

nrowseq=nrow(mvad.seq)
ijack<-matrix(nrow=nrowseq,ncol=nrowseq-1)
for (i in 1:nrowseq) {
    jackseq[[i]]=mvad.seq[setdiff(1:nrowseq,i),]
    dist_jk[[i]]=dist.om1[setdiff(1:nrowseq,i),setdiff(1:nrowseq,i)]
}
```

```
K=c(4,5,6,8)
NbSamples=nrowseq
dfR <- matrix(nrow = length(K), ncol = NbSamples)
dfVI <- matrix(nrow = length(K), ncol = NbSamples)

mean.index.vi <- rep(0,length(K))
mean.index.rand <- rep(0,length(K))
il=1

for (k in K){
    clusterO<- pam(dist.om1,k,diss=T)$clustering
    for (b in 1:NbSamples){

      clusterOrigin<-clusterO[setdiff(1:712,b)]
      clusters.jk <- pam(x = dist_jk[[b]], k, diss=T)$clustering

      dfR[il,b] <- rand.index(clusterOrigin,clusters.jk)

      dfVI[il,b] <- vi.dist(clusterOrigin, clusters.jk, parts = FALSE)
    }
    il=il+1

}
```

```
mean.cluster.vi<-rep(0,length(K))
mean.cluster.rand<-rep(0,length(K))
for(i in 1:length(K)){
mean.cluster.vi[i]<-mean(dfVI[i,])
mean.cluster.rand[i]<-mean(dfR[i,])/100}
#we divide by 100 to have the same scale
#and to be able to choose the best cluster


df <- data.frame(x=K, mean.cluster.rand, mean.cluster.vi)

ggplot(df, aes(x, y = value, color = variable)) +
    geom_point(aes(y = mean.cluster.rand, col = "rand index")) +
    geom_point(aes(y = mean.cluster.vi, col = "variation of information"))

mean.index.vi <- rep(0,712)
mean.index.rand <- rep(0,712)

for(i in 1:712){
  mean.index.vi[i]<-mean(dfVI[,i])
  mean.index.rand[i]<-mean(dfR[,i])}
```

```
n1=1
n2=712

df <- data.frame(x=n1:n2, mean.index.rand[n1:n2], mean.index.vi[n1:n2])

ggplot(df, aes(x, y = value, color = variable)) +
    geom_point(aes(y = mean.index.rand[n1:n2], col = "rand index")) +
    geom_point(aes(y = mean.index.vi[n1:n2], col = "variation of information"))


df <- data.frame(x=n1:n2, dfR[1,n1:n2], dfVI[1,n1:n2],dfR[2,n1:n2],
                 dfVI[2,n1:n2],dfR[3,n1:n2], dfVI[3,n1:n2],dfR[4,n1:n2],
                 dfVI[4,n1:n2])

ggplot(df, aes(x, y = value, color = variable)) +
    geom_point(aes(y = dfR[1,n1:n2], col = "rand index K=4")) +
    geom_point(aes(y = dfVI[1,n1:n2], col = "variation of information K=4"))+
    geom_point(aes(y = dfR[2,n1:n2], col = "rand index K=5")) +
    geom_point(aes(y = dfVI[2,n1:n2], col = "variation of information K=5"))+
                 geom_point(aes(y = dfR[3,n1:n2], col = "rand index K=6")) +
    geom_point(aes(y = dfVI[3,n1:n2], col = "variation of information K=6")) +
                 geom_point(aes(y = dfR[4,n1:n2], col = "rand index K=8")) +
    geom_point(aes(y = dfVI[4,n1:n2], col = "variation of information K=8"))

#For K=4

#number of individuals for wchich Rand index varies
lenindvar<-length(which(mean.index.rand<1))
print(paste("length?", lenindvar, sep=" " ))
```

```
#individuals for wchich Rand index varies
indvar<-which(mean.index.rand<1)
print(paste("Who are they? ="),indvar, sep=" ")

print(paste("What are their clusters? ="),
pam(x = dist.om1, 4, diss=T)$clustering[indvar], sep=" ")
```

```
clustint<-which(MedoidClust.4==2)
par(mfrow=c(1,2))

#individuals from the medoid 60 cluster that does not vary the information
seqplot.rf(mvad.seq[setdiff(clustint,indvar),],
           diss=dist.om1[setdiff(clustint,indvar),setdiff(clustint,indvar)],
           which.plot="medoids")

#individual from the medoid 60 cluster that vary the information
seqplot.rf(mvad.seq[indvar,],diss=dist.om1[indvar,indvar],which.plot="medoids")
```

```
# Logistic Regression
gvi<-ifelse(clustint %in% indvar,FALSE,TRUE)

glm.gvi <- glm(gvi ~ male+catholic+Belfast+ N.Eastern+
                 Southern+S.Eastern+Grammar+ funemp+gcse5eq+fmpr+livboth,
                 data = mvad[clustint,],
                 family = "binomial")
summary(glm.gvi)

#exp(coefficients(glm.gvi))
#exp(confint(glm.gvi))
```

```
##### Time step influence ?


# Regrouping the data by quarter
# by choosing the most frequent state in the quarter

newcol<-as.integer(linspace(1,70,23.33))[1:23]
lin=nrow(mvad.seq);col=ncol(mvad.seq)

indiv=matrix(0,ncol=23,nrow=lin)
for(i in 1:lin){
  maxi=matrix(0,nrow=6,ncol=23)
  for(k in 1:length(newcol)){
    for(j in newcol[k]:newcol[k]+2){
      if(mvad.seq[i,j]=='EM'){maxi[1,k]=maxi[1,k]+1}
      else if(mvad.seq[i,j]=='FE'){maxi[2,k]=maxi[2,k]+1}
      else if(mvad.seq[i,j]=='HE'){maxi[3,k]=maxi[3,k]+1}
      else if(mvad.seq[i,j]=='JL'){maxi[4,k]=maxi[4,k]+1}
      else if(mvad.seq[i,j]=='SC'){maxi[5,k]=maxi[5,k]+1}
      else if(mvad.seq[i,j]=='TR'){maxi[6,k]=maxi[6,k]+1}
    }
    indiv[i,k]=mvad.scodes[as.numeric(which.max(maxi[,k]))]
  }

}
newmvad<-cbind(indiv,mvad.scodes[mvad.seq[,col]])
mvadnew<-as.data.frame(newmvad)

colnames(mvadnew)=c("Autumn93","Winter93","Spring94","Summer94",
                    "Autumn94","Winter94","Spring95","Summer95",
                    "Autumn95","Winter95","Spring96","Summer96",
                    "Autumn96","Winter96","Spring97","Summer97",
                    "Autumn97","Winter97","Spring98","Summer98",
                    "Autumn98","Winter98","Spring99","Summer99")
```

```
mvad.tri2.seq <- seqdef(mvadnew)
dist.om.tri2 <- seqdist(mvad.tri2.seq, method = "OM", indel = 1, sm = "TRATE")
```

```
TimeInfluence <- function(K=3:6){
  dfR <- matrix(nrow = length(K), ncol = 1)
  dfVI <- matrix(nrow = length(K), ncol = 1)
  il=1 #row incrementor in df

  for (k in K){
    clusterMonth <- pam(dist.om1,k,diss=TRUE)$clustering #monthly data
    clusterQuarter <- pam(dist.om.tri2,k,diss=TRUE)$clustering #quarterly data

    dfR[il,1] <- rand.index(clusterMonth,clusterQuarter)
    dfVI[il,1] <- vi.dist(clusterMonth,clusterQuarter,parts = FALSE)

    il=il+1
  }

  return (list(dfR= dfR,dfVI= dfVI))
}
```

```
df1 <- data.frame(K, TimeInfluence() )

ggplot(df1, aes(x = K)) +
  geom_line( aes(x = K,y=dfR,color="Rand"))+
  geom_line(aes(x=K, y=dfVI, color= "VI"))+ ylab("Indicator")+
  labs(title="Clustering comparison on monthly and quarterly data ")+
  scale_color_manual(values= c("Rand"= "black","VI"= "red"))+
  theme(plot.title = element_text(hjust=0.5,color="blue",size=7,face="bold"))
```

```
TimeInfluenceResampling <- function(K=3:6, SizeResampling=300){

   #monthly data
   df1R <- matrix(nrow = length(K), ncol = 1)
   df1VI <- matrix(nrow = length(K), ncol = 1)

   #quarterly data
   df2R <- matrix(nrow = length(K), ncol = 1)
   df2VI <- matrix(nrow = length(K), ncol = 1)

   il=1 #row incrementor in df

   for (k in K){
     clusterOrigin1 <- pam(dist.om1,k,diss=TRUE)$clustering #monthly
     clusterOrigin2 <- pam(dist.om.tri2,k,diss=TRUE)$clustering #quarterly

     #same sample for the two kind of data
     isample <- sample(1:nrow(mvad.seq), size=SizeResampling)
     dist1.sample <- dist.om1[isample,isample]
     dist2.sample <- dist.om.tri2[isample,isample]

     clusters.sample1 <- pam(x = dist1.sample, k, diss=TRUE)$clustering
     clusters.sample2 <- pam(x = dist2.sample, k, diss=TRUE)$clustering

     df1R[il,1] <- rand.index(clusterOrigin1[isample],clusters.sample1)
     df1VI[il,1] <- vi.dist(clusterOrigin1[isample],
                            clusters.sample1, parts = FALSE)

     df2R[il,1] <- rand.index(clusterOrigin2[isample],clusters.sample2)
     df2VI[il,1] <- vi.dist(clusterOrigin2[isample],
                            clusters.sample2, parts = FALSE)

     il=il+1

   }

   return (list(df1R= df1R,df1VI= df1VI,df2R= df2R,df2VI= df2VI))
}
```

```
SizeResampling= 300
df <- data.frame(K, TimeInfluenceResampling(K, SizeResampling) )

fig1 <- ggplot(df, aes(x = K)) +
    geom_line( aes(x = K,y=df1R,color="Rand_Month"))+
    geom_line(aes(x=K, y=df2R, color= "Rand_Quarter"))+ ylab("Indicator")+
    labs(title=paste("Size of resample: ", SizeResampling))+
    scale_color_manual(values= c("Rand_Month"= "black","Rand_Quarter"= "red"))+
    theme(plot.title = element_text(hjust=0.5,color="blue",size=10,face="bold"))

fig2 <- ggplot(df, aes(x = K)) +
    geom_line( aes(x = K,y=df1VI,color="VI_Month"))+
    geom_line(aes(x=K, y=df2VI, color= "VI_Quarter"))+ ylab("Indicator")+
    labs(title=paste("Size of resample: ", SizeResampling))+
    scale_color_manual(values= c("VI_Month"= "black","VI_Quarter"= "red"))+
    theme(plot.title = element_text(hjust=0.5,color="blue",size=10,face="bold"))

grid.arrange(fig1, fig2, ncol=2 )
```

References.

[1] Nicolas S. Muller Alexis Gabadinho Gilbert Ritschard and Matthias Studer. *Analyzing and Visualizing State Sequences in R with TraMineR*. Journal of Statistical Software. 2011. URL: http://www.jstatsoft.org/v40/i04.

[2] Eytan Domany Erel Levine. "Resampling Method For Unsupervised Estimation Of Cluster Validity". In: (Feb. 2008).

[3] Brendan Halpin. "Optimal Matching Analysis and Life-Course Data: The Importance of Duration". In: (Avril 2010).

[4] Ulrike von Luxburg. "Clustering stability: an overview". In: (July 2010).

[5] Marina Meilă. "Comparing Clusterings - an information based distance". In: *Magnetic Resonance in Medicine* (Apr. 2005).

[6] Matthias Studer. *Clustering of Weighted Data*. 2021. URL: https://cran.rstudio.com/web/packages/WeightedCluster/WeightedCluster.pdf.

[7] Matthias Studer. "Validating Sequence Analysis Typologies Using Parametric Bootstrap". In: (June 2021).