WeightedCluster Library Manual

A practical guide to creating typologies of trajectories in the social sciences with R

Matthias Studer
Institute for Demographic and Life Course Studies
University of Geneva

Abstract

This manual has a twofold aim: to present the **WeightedCluster** library and offer a step-by-step guide to creating typologies of sequences for the social sciences. In particular, this library makes it possible to represent graphically the results of a hierarchical cluster analysis, to group identical sequences in order to analyse a larger number of sequences, to compute a set of measures of partition quality and also an optimized PAM (Partitioning Around Medoids) algorithm taking account of weightings. The library also offers procedures to facilitate the choice of a particular clustering solution and to choose the optimal number of groups.

In addition to the methods, we also discuss the building of typologies of sequences in the social sciences and the assumptions underlying this operation. In particular we clarify the place that should be given to the creation of typologies in the analysis of sequences. We thus show that these methods offer an important descriptive point of view on sequences by bringing to light recurrent patterns. However, they should not be used in a confirmatory analysis, since they can point to misleading conclusions.

Keywords: Analysis of sequences, trajectory, life course, optimal matching, distance, cluster, typology, weighting, cluster quality measure, R.

To cite this document or the **WeightedCluster** library, please use:

Studer, Matthias (2013). WeightedCluster Library Manual: A practical guide to creating typologies of trajectories in the social sciences with R. *LIVES Working Papers*, 24. DOI: http://dx.doi.org/10.12682/lives.2296-1658.2013.24.

1. Introduction

This manual has a twofold aim. It presents the functionalities offered by the **WeightedCluster** library for the construction and validation of weighted data clustering in R. At the same time, throughout this manual, we apply the methods presented to the analysis of sequences in the social sciences, so that it is also a step-by-step guide to building typologies of sequences. We also discuss some sociological implications and assumptions underlying these analyses.

In a general way, cluster analysis aims to construct a grouping of a set of objects in such a way that the groups obtained are as homogeneous as possible and as different from one another as possible. There are many different methods for performing this task, whose pertinence depends in particular on the objects analysed. The methods presented in this manual and available in the **WeightedCluster** library are based on a measure of dissimilarity between the objects, which makes it possible to *compare* two objects by quantifying their similarity.

We present two stages of cluster analysis based on dissimilarities: first the algorithms for grouping the objects and then the measure of the quality of the results obtained. This second stage is essential, since all the cluster analyses presented produce results, whether they are pertinent or not (Levine 2000). These measures also provide valuable help in choosing the best grouping from among the solutions resulting from different algorithms or in selecting the optimal number of groups. To do so, we use here the methods offered by the **WeightedCluster** library, such as a highly optimized PAM (Partitioning Around Medoids) algorithm or computation and visualization of the quality of a set of clustering solutions.

The particularity of the **WeightedCluster** library is that it takes account of the weighting of the observations in the two phases of the analysis previously described. There are at least two situations in which the use of weighting proves indispensable. Firstly, survey data are often weighted to correct representativity bias. In such cases, it is essential to use the weights in the clustering procedures so that the results are not biased. Secondly, weighting makes it possible to group the identical cases, which considerably reduces the memory and computing time used. Section A presents in detail the functionalities offered by the **WeightedCluster** library to automate this grouping. The **WeightedCluster** library offers functions to include weightings only for analyses for which, so far as we know, there are currently no other libraries which already do it. If this were to be the case, as for the hierarchical clustering procedures, we present the existing solutions.

As previously mentioned, this manual also aims to be a step-by-step guide to constructing of typologies of trajectories in R. As such, it is intended for a wide audience, and for this reason the most technical or most advanced parts are reserved for an appendix. [1] This approach will also lead us to discuss the sociological implications and assumptions underlying cluster analysis.

In this manual, we illustrate the methods presented with the aid of data from the study by McVicar and Anyadike-Danes (2002), available in **TraMineR** (Gabadinho *et al.* 2011), which will enable the reader to reproduce the set of analyses presented. These data describe the transition from school to work of young people in Northern Ireland, in the form of sequences of monthly statuses. The original aim of the study was to "identify young people 'at-risk' at age 16 and characterize their post-school career trajectories". This data set provides weights to correct representativity bias.

This manual is organized as follows. We start by presenting the theoretical issues raised by the construction of sequence typologies in the social sciences before turning to the cluster analysis methods as such. We then briefly summarize the different stages of cluster analysis before presenting several clustering algorithms available in R for weighted data. We then present several measures of the quality of a clustering and the main uses that can be made of them. Finally, we discuss the issues in the interpretation of cluster analysis and the risks

[1] However, a knowledge of the basic operations of sequence analysis is assumed. If this is not the case, an introduction to their practical implementation with **TraMineR** is available in Gabadinho *et al.* (2011).

that are entailed when one analyses the links between types of trajectories and explanatory factors, which is a common practice in the social sciences.

2. Typology of sequences in the social sciences

The creation of a typology is the method most used to analyse sequences (Hollister 2009; Aisenbrey and Fasang 2010; Abbott and Tsay 2000). This procedure aims to identify the recurrent patterns in the sequences or, in other words, the typical successions of states through which the trajectories run. The individual sequences are distinguished from one another by a multitude of small differences. The construction of a typology of sequences aims to efface these small differences so as to identify types of trajectories that are homogeneous and distinct from one another.

This analysis seeks to bring out recurrent patterns and/or "ideal-typical sequences" (Abbott and Hrycak 1990). These patterns can also be interpreted as interdependences between different moments in the trajectory. The search for such patterns is thus an important question in several issues in the social sciences (Abbott 1995). In particular it makes it possible to bring to light the legal, economic or social constraints that shape the construction of individual life courses. As Abbott and Hrycak (1990) note, while typical sequences can result from constraints, these typical sequences can also act on reality by serving as models for the actors, who anticipate their own future. These different possibilities of interpretation make the creation of typologies a powerful tool. In the study by McVicar and Anyadike-Danes (2002) for example, such an analysis should make it possible to identify the successions of states that lead to "at-risk" situations i.e. ones marked by a high rate of unemployment.

This grouping procedure is based on a *simplification* of the data. It thus offers a descriptive and exploratory point of view on the trajectories. By simplifying the information, one can identify the main characteristics and patterns of the sequences. However, there is a risk that this *simplification* is *unjustified* and does not correspond to the reality of the data. In other words, it is possible that the types identified are not clearly separated from one another or that they are not sufficiently homogeneous. This risk has often been neglected in the literature. As Levine (2000) in particular points out, all cluster analyses produce a result, whether it is pertinent or not. It is therefore always possible to interpret the results and make a theory out of them which often declares itself "without preliminary assumptions" or "flowing from the data", although this typology may also be the produce of a statistical artefact.

According to Shalizi (2009), the pertinence of a typology depends on three conditions. First, and this is the most important, the typology must not be dependent on the sampling, which means it must be generalizable to other observations. Secondly, a typology should extend to other properties. Finally, the typology obtained should also be grounded by a theory of the domain analysed. Shalizi (2009) gives two examples to illustrate his remarks. The classification of animal species, created on the basis of physical characteristics, also applies to other characteristics such as the song of a bird. By contrast, the classification of the stars into constellations made it possible to construct ungrounded theories.

For all the previously mentioned reasons, the **WeightedCluster** package provides several indexes to measure the quality of a partition obtained using an automatic grouping procedure. In our view, the use of such measures is an essential stage in validating the results and, in a more general way, of making the results of sequence analysis using clustering more credible.

Having presented the theoretical issues around cluster analysis, we turn to the practice, by presenting the different stages of cluster analysis.

3. Installation and loading

To use the **WeightedCluster** library, it has to be installed and loaded. It only has to be installed once, but it has to be reloaded with the `library` command each time R is started. These two stages are performed in the following way:

```
R> install.packages("WeightedCluster")
R> library(WeightedCluster)
```

4. Stages of cluster analysis

In a general way, cluster analysis takes place in four stages which we shall examine in more detail. First the *dissimilarities* are computed. These are then used to *group similar sequences* into types that are as homogeneous as possible and as different as possible from one another. Generally several different algorithms and different numbers of groups are tested. For each of the groupings obtained, the *quality of the clustering* is then computed. These measures make it possible to guide the choice of a particular solution and validate it. Finally, the results of the analysis are *interpreted* and, as appropriate, the association between the grouping obtained and other variables of interest is computed.

In this manual, we shall discuss the last three stages of the analysis. We offer here only an introduction to computation of the dissimilarities between sequences. In particular it should be noted that a dissimilarity measure is a quantification of the distance between two sequences or, more generally, between two objects. This quantification then makes it possible to *compare* the sequences. In cluster analysis for example, this information is necessary in order to group the most similar sequences. A matrix of dissimilarities containing the set of dissimilarities two by two, or in other words the quantification of all the possible comparisons, is generally used.

The choice of the measure of dissimilarity used to quantify the differences between sequences is an important question, but beyond the scope of this manual. Several articles address this issue (Aisenbrey and Fasang 2010; Hollister 2009; Lesnard 2010). Here, we use the optimal matching distance adopting the costs used in the original article by McVicar and Anyadike-Danes (2002).

In R, the distances between state sequences can be computed with the aid of the **TraMineR** library (Gabadinho *et al.* 2011). To compute these distances, one first has to create a state sequence object using the `seqdef` function. The distances are then computed with the `seqdist` function. Gabadinho *et al.* (2011) presents these different stages in detail.

Here, we start by loading the example data set with the `data` command. We then build the sequence object by specifying the columns containing the data (17 to 86) and the weighting of the observations. The `seqdist` function computes the matrix of distances between sequences.

```
R> data(mvad)
R> mvad.alphabet <- c("employment", "FE", "HE", "joblessness", "school",
         "training")
R> mvad.labels <- c("Employment", "Further Education", "Higher Education",
         "Joblessness", "School", "Training")
R> mvad.scodes <- c("EM", "FE", "HE", "JL", "SC", "TR")
R> mvadseq <- seqdef(mvad[, 17:86], alphabet = mvad.alphabet,
         states = mvad.scodes, labels = mvad.labels,
         weights = mvad$weight, xtstep = 6)
R> ## Defining the custom cost matrix
R> subm.custom <- matrix(
           c(0, 1, 1, 2, 1, 1,
             1, 0, 1, 2, 1, 2,
             1, 1, 0, 3, 1, 2,
             2, 2, 3, 0, 3, 1,
             1, 1, 1, 3, 0, 2,
             1, 2, 2, 1, 2, 0),
           nrow = 6, ncol = 6, byrow = TRUE)
R> ## Computing the OM dissimilarities
R> mvaddist <- seqdist(mvadseq, method = "OM", indel = 1.5, sm = subm.custom)
```

The remainder of this manual covers the three following stages. We shall first discuss the clustering methods available for the weighted data. We shall then present the measures of the quality of a clustering offered by the **WeightedCluster** library. These measures will enable us to choose a particular solution and measure its validity. Finally, we shall discuss the problems of interpreting the results of cluster analysis and computing the relationship between typologies and other variables of interest.

5. Clustering

There are many different clustering algorithms. We present here some methods derived from two different logics: hierarchical clustering methods and those of partitioning into a predefined number of groups. We conclude by discussing the possible interactions between these types of algorithms.

5.1. Hierarchical clustering

We present here the procedures for hierarchical agglomerative clustering, which function as follows. One starts out from the observations, each of them being considered as a group. On each iteration, the two closest groups (or initially observations) are grouped, until all the observations form a single group. The agglomeration schedule, i.e. the succession of groupings performed, represents the clustering procedure in the form of a tree which is called a dendrogram. When the agglomeration schedule has been constructed, one selects the number of groups. The partition selected is obtained by "cutting" the grouping tree at the corresponding level.[2]

[2]There are also so-called descendant (or divisive) procedures which run in the same way, but in the opposite

In R, the agglomeration schedule (the succession of groupings performed) is created with the `hclust` function.[3] This function takes the following parameters: the distance matrix (as a `dist` object), the method (here "Ward", we shall return to this) and the weight vector `members`.

```
R> wardCluster <- hclust(as.dist(mvaddist), method = "ward",
         members = mvad$weight)
```

When the agglomeration schedule has been created, the final stages of this schedule can be visualized with the aid of the **WeightedCluster** library. This is done in two stages. One starts by constructing a tree of sequences from the agglomerative schema with the command `as.seqtree`. This function takes the following arguments: the clustering procedure (`wardCluster` in our case), the sequence object (`seqdata` argument), the distance matrix (`diss` argument) and maximum number of groupings to be represented (`ncluster`).

```
R> wardTree <- as.seqtree(wardCluster, seqdata = mvadseq, diss = mvaddist,
         ncluster = 6)
```

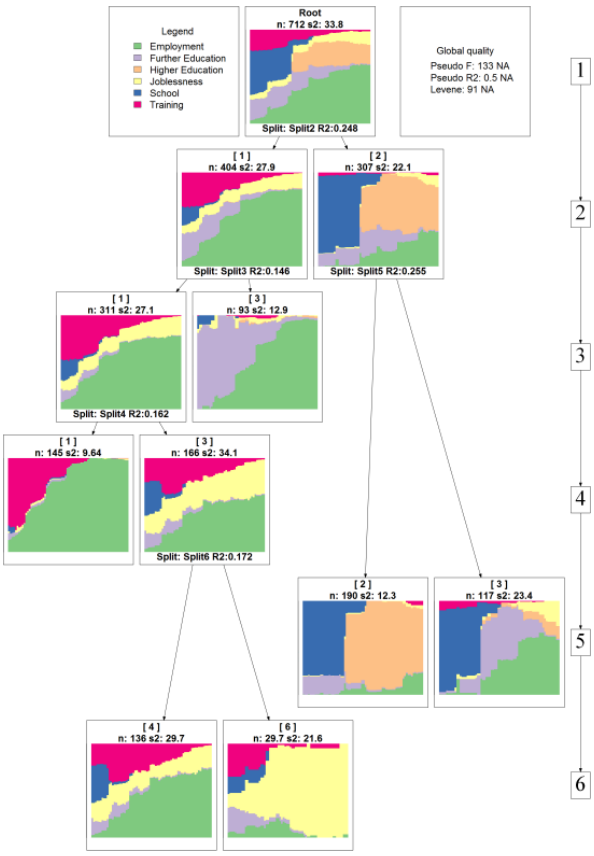
Once the tree is constructed, the `seqtreedisplay` function of the **TraMineR** library (Studer *et al.* 2011) makes it possible to represent it graphically. The free GraphViz software (Gansner and North 1999) must be installed and accessible for the function to work properly.[4] The option `showdepth=TRUE` displays the levels of the groupings on the right of the graph.

```
R> seqtreedisplay(wardTree, type = "d", border = NA, showdepth = TRUE)
```

direction. Instead of starting out from the observations that are regarded as groups, the procedure divides one of the groups at each step until each observation corresponds to a group. An algorithm is available in R with the `diana` function of the **cluster** library. Everything presented here is also compatible with the object returned by this function.

[3] Other possibilities exist.

[4] The program can be downloaded from `http://www.graphviz.org/`.



This graphic presents the result of this procedure and makes it possible to visualize the grouping logics of the sequences. The first distinction, considered to be the most important, separates the individuals who go into tertiary education from the others. The distinctions that are made when one moves from four to five groups enable us, however, to bring to light the fact that this group amalgamates two logics: those who go into "Higher Education" and those who go into "Further Education". This graphic provides an important aid for identifying the distinctions pertinent for the analysis. Thus, if one opts for a four-group solution, no further distinctions will be made between between the two logics ("Higher Education" and "Further Education"). By contrast, the distinction into five groups enables this to be identified.

This example illustrates the *simplification* of the data effected by the clustering procedures. Opting for only four groups, one amalgamates two logics which may (or may not) be pertinent for the analysis. It should be noted that in opting for only five groups, one amalgamates the distinctions that are made at a lower level. One is thus always performing a simplification.

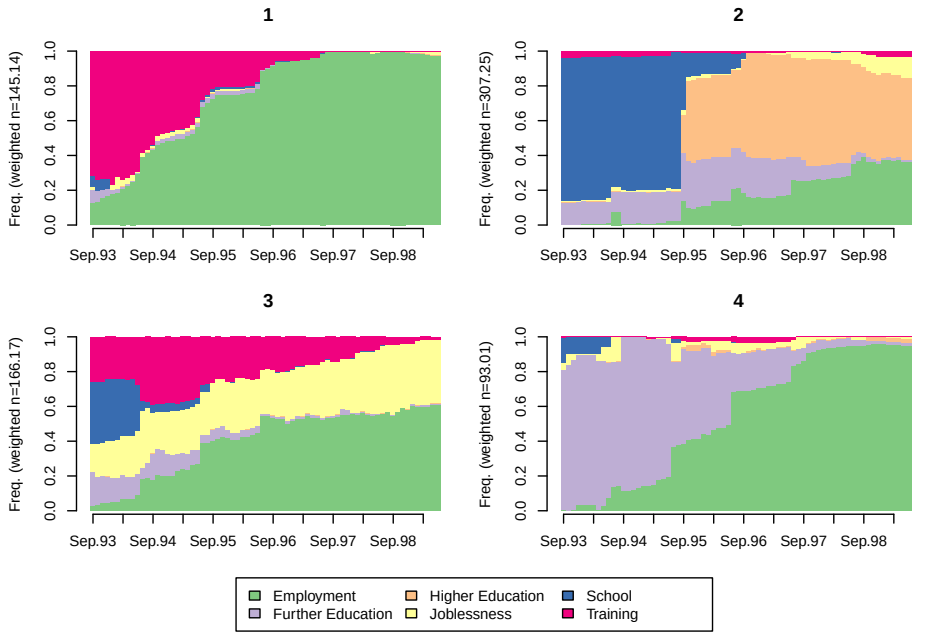
To obtain a grouping in a particular number of groups, one cuts the tree at a given level. This means that one keeps all the terminal nodes if the growing of the tree is stopped at the level presented on the right. In R, this information is recovered with the aid of the `cutree` function. For example, to use the grouping in four distinct classes:

```
R> clust4 <- cutree(wardCluster, k = 4)
```

This information can then be used in other analyses, for example, to represent the types obtained with the aid of a chronogram. Not surprisingly, the graphics obtained correspond

to the terminal nodes of the tree at level four.

```
R> seqdplot(mvadseq, group = clust4, border = NA)
```



As previously mentioned, the `hclust` function offers seven different hierarchical algorithms which are specified with the argument method. The **fastcluster** library provides an optimized version of this function (Müllner 2011). The hierarchical algorithms `diana` (divisive algorithm, see Kaufman and Rousseeuw 1990) and beta-flexible clustering with the `agnes` function are available in the **cluster** library (Maechler *et al.* 2005).[5] Table 1 lists these algorithms, giving the name of the function to be used, the allowance for weightings ("Indep" means that the algorithm is insensitive to weightings) and the interpretation of the clustering logic. A more detailed presentation of these algorithms is available in Müllner (2011) and Kaufman and Rousseeuw (1990).

In their article, Milligan and Cooper (1987) examine the results of different simulations in order to evaluate the performances of some of these algorithms. It should be noted that these simulations were carried out with numerical data and the Euclidian distance measure. The extension of these results to other types of distances is subject to debate. They thus report rather poor results for the "single", "complete", "centroid" and "median" methods. The "Ward" method generally performs fairly well except in the presence of outliers which bias the results. They report very variable results for the "average" method. Finally, the "beta-flexible" method with a value of beta close to -0.25 gives good results in the presence of various forms of error in the data (Milligan 1989). The best results are obtained with the "flexible UPGMA" algorithm, which is not currently available in R (Belbin *et al.* 1992).

Hierarchical procedures are open to several criticisms. First, and most importantly, the merging of two groups is done by maximizing a local criterion. These procedures optimize

[5] If these functions are used, it is necessary to specify the argument `diss=TRUE`, in order for the algorithm to use the distance matrix passed as an argument.

Table 1: Hierarchical clustering algorithms.

Name	Function	Weight	Interpretation and notes.
single	`hclust`	Indep	Merging of the groups with closest observations.
complete	`hclust`	Indep	Minimization of diameter of each new group (very sensitive to atypical data).
average (or UPGMA)	`hclust`	Yes	Average of distances.
McQuitty (or WPGMA)	`hclust`	Indep	Depends on previous mergings.
centroid	`hclust`	Yes	Minimization of distances between medoids.
median	`hclust`	Indep	Depends on previous mergings.
ward	`hclust`	Yes	Minimization of residual variance.
beta-flexible	`agnes`	No	For a value of β close to -0.25, set the argument `par.method=0.625`.

a local criterion, i.e. the loss of information due to a grouping is estimated locally. These local choices can lead to great differences at higher levels and it is not guaranteed that a local choice is the best one from a global point of view. In other words, it can often happen that a good choice at the local level leads to mediocre results at a higher level of grouping. Secondly, agglomerative procedures are non-deterministic, particularly when the distance measure takes only a few different values, giving rise to ties between which one has to decide; in particular this can be the case with optimal matching or the Hamming distance. Although a particular version of this algorithm generally produces the same dendrogram in each analysis[6], several versions of this same algorithm can lead to divergent results. Moreover, this occurrence can penalize the procedure, which has no criterion to make a choice (Fernández and Gómez 2008). We now present the PAM algorithm, which has the advantage of seeking to maximize a global criterion.

5.2. Partitioning Around Medoids

The PAM (for "Partitioning Around Medoids") algorithm follows a different logic from hierarchical algorithms (Kaufman and Rousseeuw 1990). It aims to obtain the best partitioning of a data set into a predefined number k of groups. Compared to the other algorithms presented, this algorithm has the advantage of maximizing a global criterion and not only a local criterion.

The aim of the algorithm is to identify the k best representatives of groups, called medoids. More precisely, a medoid is defined as the observation of a group having the smallest weighted sum of distances from the other observations de this group. This algorithm thus seeks to minimize the weighted sum of distances from the medoid.

In brief, the functioning of this algorithm can be described in two phases. In a first stage, one initializes the algorithm, seeking the observations that most diminish the weighted sum of the distances from the existing medoids, having chosen the medoid from the whole data set at the start. When the initial solution has been constructed, the second phase of the

[6]The algorithms generally make a choice which depends on the order of the observations, which makes it replicable so long as the order remains unchanged.

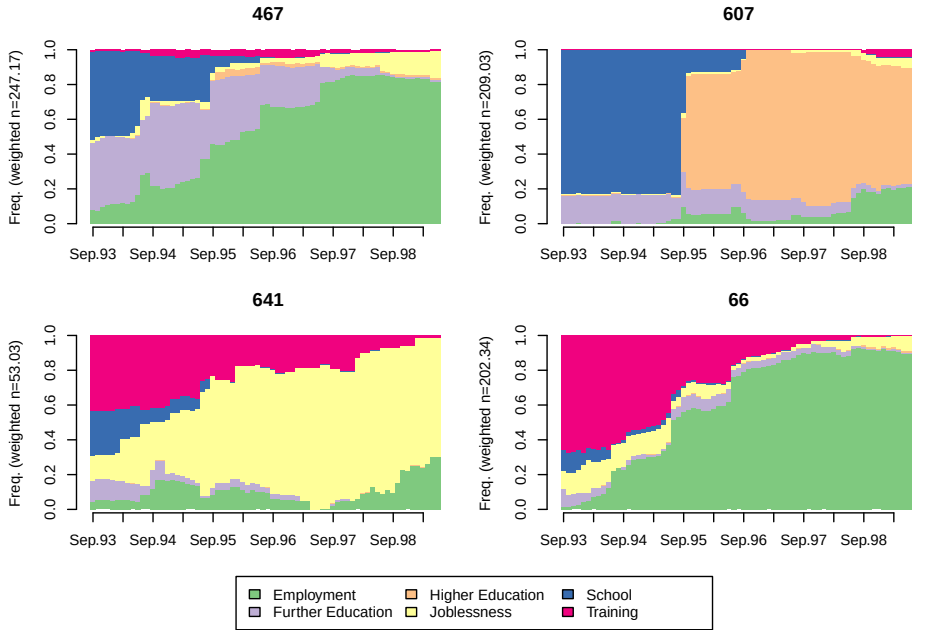
algorithm, called "swapping", starts. For each observation, one computes the potential gain that would be realized if one of the existing medoids were replaced this observation. The gain is computed at the global level and in terms of the weighted distances from the closest medoids. The medoid is then replaced by the observation that leads to the greatest possible gain. These operations are repeated until it is no longer possible to improve the current solution.

The algorithm is available in the library R **cluster** (Maechler *et al.* 2005; Struyf *et al.* 1997), but does not make it possible to use weighted data. The `wcKMedoids` function in the **WeightedCluster** library, partly based on the code available in the **cluster** library, takes weightings into account and also implements the optimizations suggested by Reynolds *et al.* (2006), which make it faster. This function also consumes only half as much memory, which makes it adequate for analysing very large data sets. Annex B presents the gains in computing time in more detail.

```
R> pamclust4 <- wcKMedoids(mvaddist, k = 4, weights = mvad$weight)
```

The `clustering` element contains the cluster membership of each observation. This information can then be used, for example to represent the sequences with the aid of a chronogram.

```
R> seqdplot(mvadseq, group = pamclust4$clustering, border = NA)
```



The number assigned to each group corresponds to the index of the medoid of this group. The medoids can thus be recovered by using the command `unique(pamclust4$clustering)`. The following command uses this possibility to display the medoid sequences of each group:

```
R> print(mvadseq[unique(pamclust4$clustering), ], format = "SPS")
```

```
    Sequence
[1] (TR,22)-(EM,48)
[2] (SC,25)-(HE,45)
[3] (SC,10)-(FE,12)-(EM,48)
[4] (TR,22)-(JL,48)
```

The PAM algorithm has several disadvantages. First, it is necessary to specify in advance the number k of groups. When testing several sizes k of partition, one cannot be sure that the types obtained interlock as in the case of hierarchical procedures, and the computing time can be correspondingly long. Secondly, the PAM algorithm always creates "spherical" groups[7] centred on their medoid, and this structure does not necessarily correspond to the reality of the data.[8] But most importantly, the algorithm is dependent on the initial choice of medoids, which is not always optimal.

5.3. Combining the algorithms

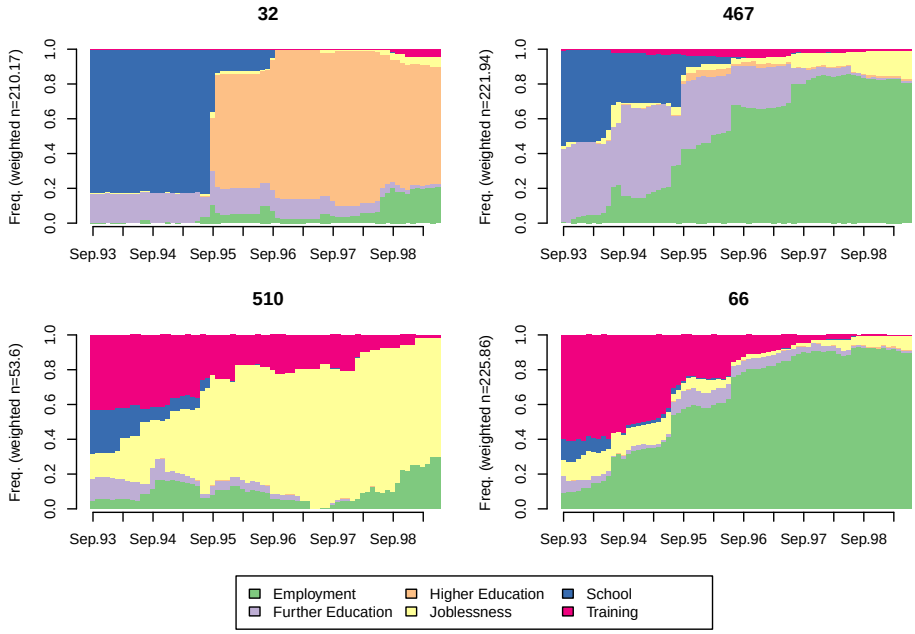
The PAM algorithm has the advantage of optimizing a global criterion, but the algorithm is dependent on the initial choice of medoids, which is not always optimal. To overcome this limitation, we can try to initialize the PAM algorithm using the result of the hierarchical clustering procedure. To do this, the clustering obtained by the hierarchical method (the argument `initialclust=wardCluster`) is specified as the starting point of the `wcKMedoids` function (or the `wcKMedRange` function, see section 6.3).

```
R> pamwardclust4 <- wcKMedoids(mvaddist, k = 4, weights = mvad$weight,
         initialclust = wardCluster)
```

This leads here to a slightly different solution, but with better quality.

```
R> seqdplot(mvadseq, group = pamwardclust4$clustering, border = NA)
```

[7] Which is also the case for the "Ward" criterion in hierarchical procedures.
[8] For instance, the algorithm might fail to recognize the *true* structure if the data are organized in squares.



We have presented several types of cluster analyses, which have led us to different solutions. How does one choose among these solutions? This question is all the more important since we could also have selected different numbers of groups for each procedure, which would have led us to a large number of possibilities. The measures of quality of a partition that we now present assist in this choice by offering a basis for comparing these different solutions.

6. Measuring the quality of a partition

Measures of the quality of a partition have two objectives. First, some of them give an idea of the statistical quality of the partition. Secondly, these measures assist in the choice of the best partition from a statistical standpoint. They are a valuable aid in selecting the number of groups or the best algorithm.

6.1. Presentation of measures

The **WeightedCluster** library offers several measures of partition quality, which are listed in Table 2. The choice of these measures is mainly inspired by Hennig and Liao (2010), together with the "C-index" which figured among the best indexes according to Milligan and Cooper (1985). The presentation we make here is centred on the concepts. The interested reader can however refer to Annex C, which presents the mathematical details of these measures and the adjustments made to take account of the weighting of the observations. Alongside the names of the quality measures, Table 2 presents their main characteristics with the aid of the following information:

- Abrv: abbreviation used in the WeightedCluster library.
- Range: interval of the possible values.
- Min/Max: Does a good partition minimize or maximize this measure?

- Interpretation of the value.

Table 2: Measures of the quality of a partition.

Name	Abrv.	Range	Min/Max	Interpretation
Point Biserial Correlation	PBC	$[-1;1]$	Max	Measure of the capacity of the clustering to reproduce the distances.
Hubert's Gamma	HG	$[-1;1]$	Max	Measure of the capacity of the clustering to reproduce the distances (order of magnitude).
Hubert's Somers' D	HGSD	$[-1;1]$	Max	Measure of the capacity of the clustering to reproduce the distances (order of magnitude) taking into account ties in distances.
Hubert's C	HC	$[0;1]$	**Min**	Gap between the partition obtained and the best partition theoretically possible with this number of groups and these distances.
Average Silhouette Width	ASW	$[-1;1]$	Max	Coherence of assignments. High coherence indicates high between-group distances and strong within-group homogeneity.
Average Silhouette Width (weighted)	ASWw	$[-1;1]$	Max	As previous, for floating point weights.
Calinski-Harabasz index	CH	$[0;+\infty[$	Max	Pseudo F computed from the distances.
Calinski-Harabasz index	CHsq	$[0;+\infty[$	Max	As previous, but using *squared* distances.
Pseudo R^2	R2	$[0;1]$	Max	Share of the discrepancy explained by the clustering solution (only to compare partitions with identical number of groups).
Pseudo R^2	R2sq	$[0;1]$	Max	As previous, but using *squared* distances.

The first three measures, namely "Point Biserial Correlation" (Milligan and Cooper 1985; Hennig and Liao 2010), "Hubert's Gamma" and "Hubert's Somers' D" (Hubert and Arabie 1985), follow the same logic. They measure the capacity of a partition of the data to reproduce the distance matrix. Whereas the first measures the capacity to reproduce the exact value of the distances, the other two are based on the concordances. This implies that, according to the latter two indexes, a partition is valid if the distances between the groups are greater than those within the groups. Technically, this capacity is measured by computing the association between the distance matrix and a second measure of distance which takes the value 0 for observations that are in the same group and 1 otherwise. Pearson's correlation ("Point Biserial Correlation"), Goodman and Kruskal's Gamma ("Hubert's Gamma") or Somers' D ("Hubert's Somers' D") is used.

The "Hubert's C" index compares the partition obtained with the best partition that could have been obtained with this number of groups and this distance matrix. In contrast to the other indexes, a small value indicates a good partition of the data.

The Calinski-Harabasz indexes (Calinski and Harabasz 1974) are based on the statistic F of the analysis of variance. This measure gave very good results in the simulations of Milligan and Cooper (1985). However, its extension to non Euclidean distance (such as optimal matching) is subject to debate. One may question its pertinence if the measure of distance is Euclidian (or squared Euclidean), in which case this measure amounts to using the statistic F on the coordinates that can be associated with the observations, for example with a principal

coordinate analysis (Studer *et al.* 2011). For this case the **WeightedCluster** library provides this statistic using the squared distances when the distance is Euclidian or the distance itself when the measure is already a squared Euclidian distance, such as the Hamming distance.

R-squared calculates the share of discrepancy explained by a partition (Studer *et al.* 2011). This measure is only pertinent to compare partitions containing the same number of groups, since it does not penalize complexity.

Finally, the measure "Average Silhouette Width" proposed by Kaufman and Rousseeuw (1990) is particularly interesting. It is based on the coherence of the assignment of an observation to a given group, comparing the average weighted distance of an observation from the other members of its group and its average weighted distance from the closest group. A value is calculated for each observation, but more attention is paid to the average silhouette. If this is weak, it means that the groups are not clearly separated or that the homogeneity of the groups is low. Interestingly, Kaufman and Rousseeuw (1990) put forward orders of magnitude for interpreting this measure, which are reproduced in Table 3.

Table 3: Orders of magnitude for interpreting the *ASW* measure

ASW	Interpretation proposed
$0.71 - 1.00$	Strong structure identified.
$0.51 - 0.70$	Reasonable structure identified.
$0.26 - 0.50$	Structure is weak and could be artificial. Try other algorithms.
≤ 0.25	No structure.

The original formulation of Kaufman and Rousseeuw (1990) supposes that one weighting unit is equivalent to one observation. This is the case if the weighting results from aggregation (see Annex A) or if the data are not weighted. When the weights aim at correcting representativity bias (as it is the case here), we propose a variant of this measure called "ASWw". The details are given in Annex C.1. Generally, the measures gives very similar results, "ASWw" tends to be a little higher.

These measures are calculated with the `wcClusterQuality` function. The value returned by the function is a list comprising two elements. The `stats` element contains the values of the measures of quality.

```
R> clustqual4 <- wcClusterQuality(mvaddist, clust4, weights = mvad$weight)
R> clustqual4$stats

   PBC     HG   HGSD    ASW   ASWw     CH     R2   CHsq   R2sq     HC
  0.43   0.57   0.57   0.23   0.23 140.43   0.37 282.38   0.54   0.19
```

According to the measure `ASWw=0.23`, the solution in four groups obtained using the Ward criterion could be a statistical artefact, since it is less than 0.25.

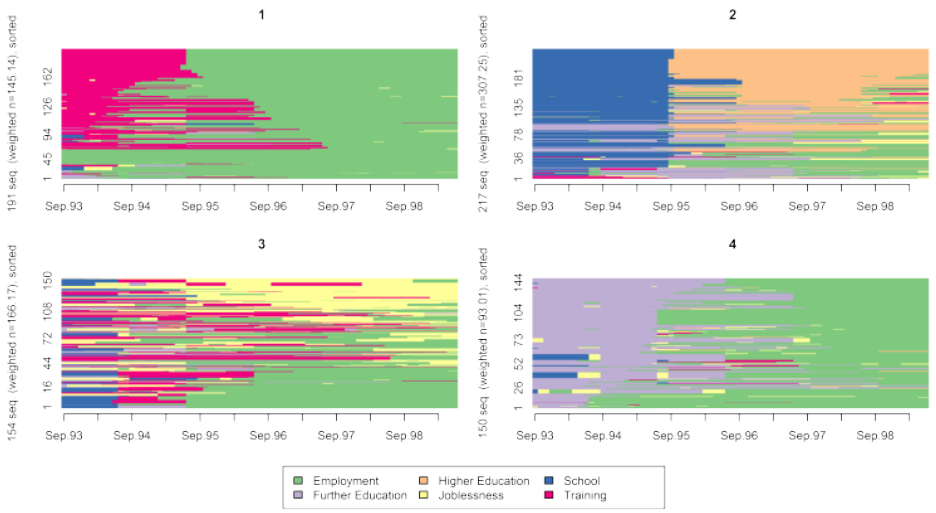
The `ASW` element of the object `clustqual4` contains the two variant of the average silhouette of each group taken separately. According to this measure, group 3 is particularly ill-defined since its average silhouette is negative.

```
R> clustqual4$ASW

     ASW   ASWw
1  0.577  0.580
2  0.230  0.232
3 -0.186 -0.182
4  0.434  0.440
```

6.2. Use the silhouette to represent the clusters

The silhouette can be computed separately for each sequence, which makes it possible to identify the sequences most characteristic of a grouping (sequences with a silhouette width close to one). The `wcSilhouetteObs` function computes these values. In the example below, we use the silhouettes to order the sequences in index plots.

```
R> sil <- wcSilhouetteObs(mvaddist, clust4, weights = mvad$weight,
         measure = "ASWw")
R> seqIplot(mvadseq, group = clust4, sortv = sil)
```



The most characteristic sequences of each cluster are represented at the top of each graphic. According to the definition of the silhouette, the sequences we call "characteristic" are those that are close to the centre of their group but distant from the closest group. In group 1, for example, the characteristic sequence is to enter employment after two years of apprenticeship. By contrast, the sequences at the bottom of each graphic are poorly represented and/or poorly assigned. Thus, in group 3, for example, the sequences "school – apprenticeship – employment" are closer to another group (most likely no. 1) than their own group.

6.3. Choice of a partition

The measures of the quality of a partition facilitate the choice of the best partition among a set of possibilities. They can thus be used to identify the algorithm that gives the best

results. The `wcKmedoids` function used for PAM directly computes these values, which are stored in the elements `stats` and `ASW` as previously. The following code displays the quality measures for the partition identified with the aid of PAM. This partition thus seems better than that obtained with Ward.

```
R> pamclust4$stats

  PBC    HG  HGSD   ASW  ASWw    CH    R2  CHsq  R2sq    HC
  0.6   0.8   0.8   0.4   0.4 198.2   0.5 534.6   0.7   0.1
```

These measures also make it possible to compare partitions with different numbers of groups. Only pseudo R^2 should not be used for this purpose, since it does not penalize for complexity. Computing the quality of all these different possibilities soon becomes laborious. The as.clustrange function in the **WeightedCluster** library automatically computes these values for a set of numbers of groups derived from one hierarchical clustering procedure (`wardCluster` in our example). This function requires the following arguments: the matrix of dissimilarities (`diss`), the weightings (optional, `weights` argument) and the maximum number of cluster that one wants to retain (`ncluster`). In the following example, we estimate the clustering quality for groupings in $2, 3, \ldots,$ ncluster $= 20$ groups.

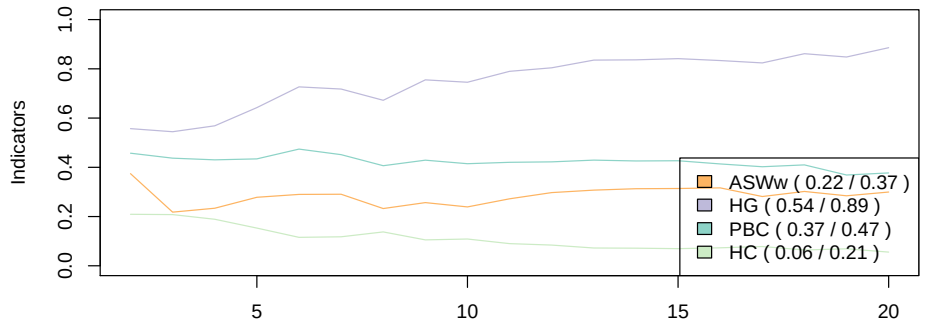
```
R> wardRange <- as.clustrange(wardCluster, diss = mvaddist,
         weights = mvad$weight, ncluster = 20)
R> summary(wardRange, max.rank = 2)

     1. N groups 1.  stat 2. N groups 2.  stat
PBC            6   0.4740           2   0.4573
HG            20   0.8858          18   0.8614
HGSD          20   0.8842          18   0.8597
ASW            2   0.3725          16   0.3028
ASWw           2   0.3742          16   0.3165
CH             2 234.1011           3 164.0037
R2            20   0.6954          19   0.6805
CHsq           2 469.5188           3 326.5015
R2sq          20   0.8624          19   0.8432
HC            20   0.0556          18   0.0641
```

The `summary` function presents the best number of groups according to each quality measure and the value of these statistics. The `max.rank` argument specifies the number of partitions to be displayed. According to the "Point Biserial Correlation" (PBC) measure, partitioning in six groups is the best partition, whereas for the "ASW" index a solution in two groups seems preferable. The maximum value identified for this latter index indicates that we may be dealing with statistical artefacts (see Table 3). It should be noted that pseudo-R^2 and its version based on the high squared distances will always give a maximum for the highest number de groups, since these statistics cannot diminish.

The presentation offered by `summary` is useful for comparing the results of two procedures (see below). However, it only presents two or three solutions and is often useful to observe

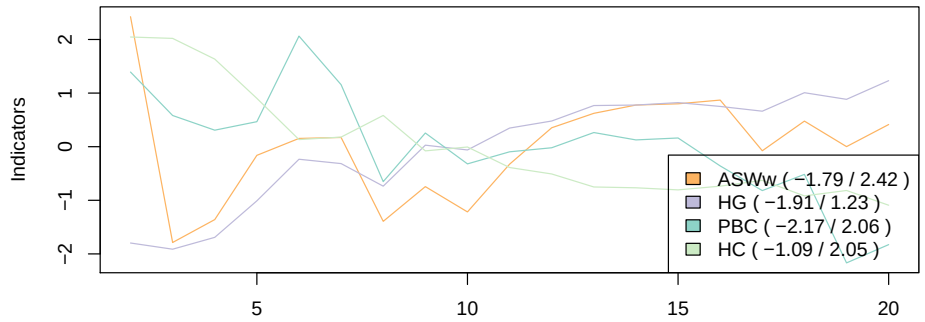
the changes in these measures so as to identify breaking points and the partitions that offer the best compromise among several measures. The `plot` function associated with the object returned by `as.clustrange` represents this change graphically. If required, the `stat` argument specifies the list of measures to be displayed (`"all"` displays all of them).

```
R> plot(wardRange, stat = c("ASWw", "HG", "PBC", "HC"))
```



The solution in six groups is here a local maximum for the measures “HC”, “PBC” and “HG”, which makes it a good one if one wants to keep a limited number of groups. The graphic is sometimes somewhat difficult to read, because the average values of each measure differ. To palliate this problem, the argument `norm="zscore"` standardizes the values, which makes it easier to identify the maxima and minima.[9] The command below facilitates the identification of the partitions that give good results. The solutions in six and 17 groups appear to be good in the present case.

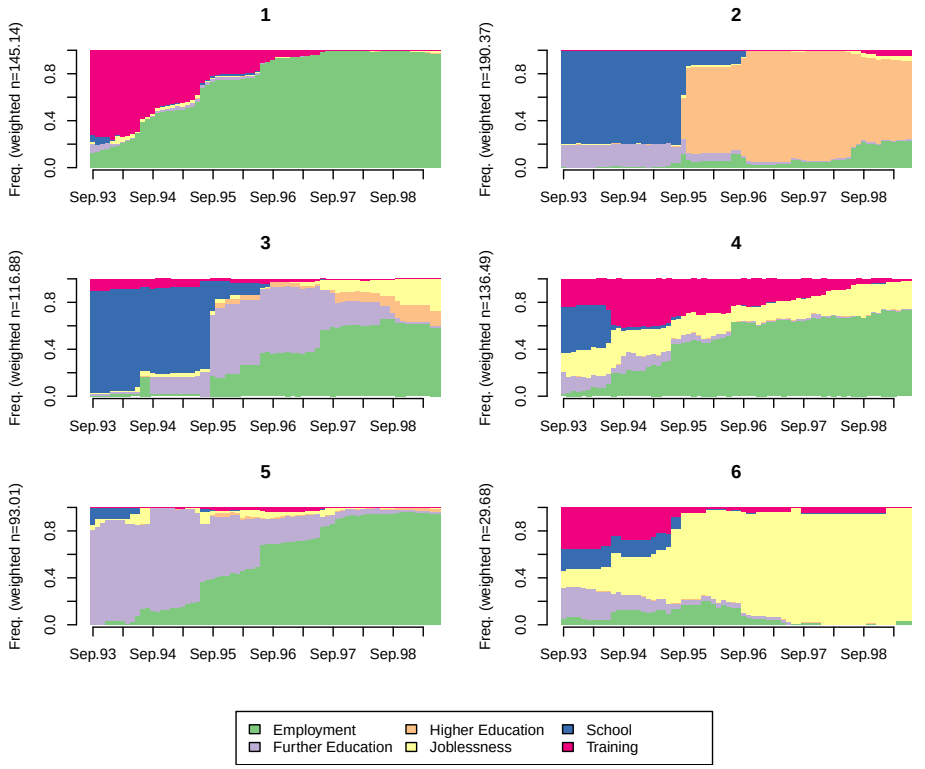
```
R> plot(wardRange, stat = c("ASWw", "HG", "PBC", "HC"), norm = "zscore")
```



The object returned by the `as.clustrange` function also contains a `data.frame` called `clustering` containing all the partitions tested. One can thus use `as.clustrange` directly rather than `cutree`. The solution in six groups can be displayed in the following way:

[9] For a more robust standardization based on the median, `norm="zscoremed"` can be used.

```
R> seqdplot(mvadseq, group = wardRange$clustering$cluster6, border = NA)
```



The `as.clustrange` function accepts several types of arguments, including the set of clustering procedures available in the **cluster** library, even if the latter do not accept weighting. However, it is not possible to use it directly with non-hierarchical algorithms such as PAM. To do this, we use the `wcKMedRange` function, which automatically computes the partitions for a series of values of k (number of groups). The object returned is the same as that previously presented for the hierarchical clustering procedures and the same presentations can be used as previously. This function takes as a parameter a distance matrix, `kvals`, a vector containing the number of groups of the different partitions to be created, and a vector of `weights`. The supplementary arguments are passed to `wcKMedoids`.

```
R> pamRange <- wcKMedRange(mvaddist, kvals = 2:20, weights = mvad$weight)
R> summary(pamRange, max.rank = 2)

     1. N groups 1.  stat 2. N groups 2.  stat
PBC            4    0.565           5   0.5323
HG            20    0.910          19   0.8936
HGSD          20    0.908          19   0.8920
ASW            2    0.389          20   0.3716
ASWw          20    0.390           2   0.3900
CH             2  252.008           3 205.7979
R2            20    0.712          19   0.7024
```

```
CHsq           4  534.585           2 483.1366
R2sq          20    0.887          19   0.8808
HC            20    0.045          19   0.0528
```

The presentation offered by the `summary` function is particularly useful for comparing the solutions of different clustering procedures. It can thus be noted that the PAM results generally have a higher quality. Alongside the solution in two groups, which may somewhat oversimplify, the solution in four groups seems appropriate here. It should be pointed out, however, that these partitions could still be statistical artefacts since the "ASW" is less than 0.5.

The various tools that we have presented have enabled us to construct a typology of sequences. To do so, we have tested various algorithms and numbers of groups before opting for a partition in four groups created using the PAM method. In a general way, we suggest that readers test a larger number of algorithms than have been tested here. Lesnard (2006) suggests for example using the "average" method or the "flexible" method (see Table 1). The tools presented make it possible to make these comparisons.

6.4. Naming the clusters

Once the typology has been created, it is customary to name the types obtained so as to make them easier to interpret. The factor function makes it possible to create a categorial variable. To do so, one specifies the variable of the types (here `pamclust4$clustering`), the values that this variable takes with the `levels` argument (these values can be found in the previous graphics), and the labels that one wants to assign to each of these types with the `labels` argument. The `labels` must be specified in the same order as the levels argument so that first item of `levels` (here 66) corresponds to the first item of `labels` (here "Training – Employment").

```
R> mvad$pam4 <- factor(pamclust4$clustering, levels = c(66, 467,
         607, 641), labels = c("Train-Empl", "School-Empl", "High Ed",
         "Unempl"))
```

It is also possible to automatically label the clusters using the medoid sequence of each cluster. The `seqclustname` function can be used for this purpose. One needs to specify the sequence object (`seqdata` argument), the clustering (`group` argument), the distance matrix (`diss` argument). If `weighted=TRUE` (default), then the weights of the `seqdata` object are used to find the medoids. Finally, on can set `perc=TRUE` to add the percentage of observation in each groups to the labels.

```
R> mvad$pam4.auto <- seqclustname(mvadseq, pamclust4$clustering,
         mvaddist)
R> table(mvad$pam4.auto, mvad$pam4)


                    Train-Empl School-Empl High Ed Unempl
  TR/22-EM/48              189           0       0      0
  SC/10-FE/12-EM/48          0         307       0      0
```

```
SC/25-HE/45                0            0       160        0
TR/22-JL/48                0            0         0       56
```

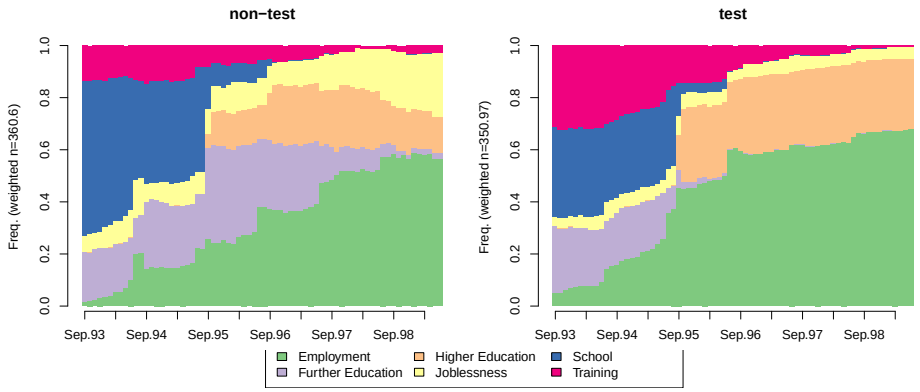
Once the clusters have been named, the typology is constructed. Before concluding, we return in more detail to the question of the simplification induced by cluster analysis and the bias this can introduce into the analysis. We illustrate this question by presenting the computation of the links between typology and explanatory factors and showing how this can bias the results.

7. Linking trajectory types and explanatory factors

Many research questions in the social sciences connect trajectories (or sequences) with explanatory factors. For example, one may want to know whether the occupational trajectories of men differ significantly from those of women. Similarly, Widmer *et al.* (2003) seek to bring to light the appearance of new trajectories in the construction of family life. To do so, one generally correlates a typology of family sequences with the cohort of individuals with the aid of a chi-square test or logistic regression (Abbott and Tsay 2000). In this section, we present this technique and also the dangers it implies for the analysis.

Suppose one seeks to measure the links between the variable `test` that we have created for our purposes[10] and our trajectories. This variable takes two modalities, "test" and "non-test", and groups the sequences in the following way:

```
R> seqdplot(mvadseq, group = mvad$test, border = NA)
```



The "non-test" individuals seem in particular to have a strong probability of undergoing a period of unemployment. Is this difference significant? One can run a chi-square test between the variable `test` and `pam4` (which covers our types) in the following way for weighted data. One starts by constructing a crossed table with the `xtabs` function.[11] This function takes as a parameter a formula in which the left side of the equation refers to the weightings and the right side lists the factors that form the crossed table. The `data` argument specifies where variables that appear in the formula are to be found. One then runs a chi-square test on this table.

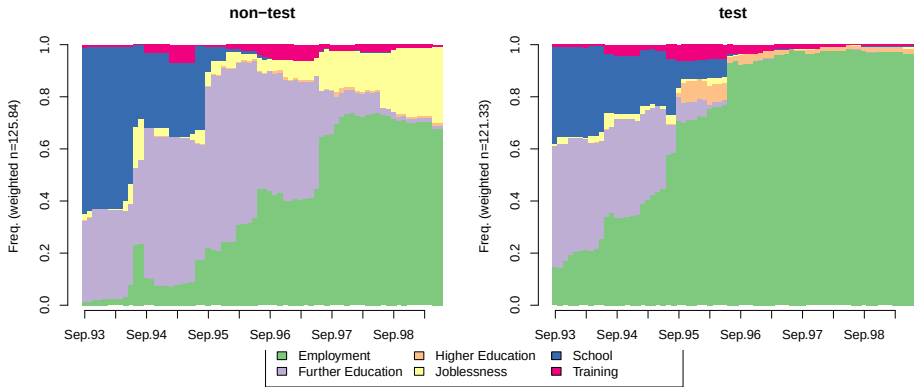
[10] The detail of the creation of this variable is given in Annex D.

[11] For unweighted data, the `table` function can be used.

```
R> tb <- xtabs(weight ~ test + pam4, data = mvad)
R> chisq.test(tb)


Pearson's Chi-squared test

data:  tb
X-squared = 0.0172, df = 3, p-value = 0.9994
```

The result is non-significant, which means that according to this procedure the trajectories do not differ in terms of the variable `test`. What is the problem? Is this really the case? No, the problem arises from the simplification effected by cluster analysis. By using the typology in the chi-square test, one implicitly makes the assumption that this typology is sufficient to describe the complexity of the trajectories, which is not case here. In fact the `test` variable explains the variation of the trajectories *within the types*. As an example, the following graphic shows the differences in trajectories according to the `test` variable for the individuals classified in the type "School-Employment". The "non-test" individuals thus seem to have a greater risk of undergoing a period of "Unemployment". Hence the variability within this type does not seem to be negligible.

```
R> SchoolEmpl <- mvad$pam4 == "School-Empl"
R> seqdplot(mvadseq[SchoolEmpl, ], group = mvad$test[SchoolEmpl],
          border = NA)
```



Let us return to the underlying assumptions already mentioned and attempt to explain it. In using the typology, one assigns its type to each sequence. One thus ignores the gap between the trajectory really followed by an individual and its type. One possible justification for this operation would be to say that the types obtained correspond to the models that have actually generated the trajectories. The gaps between the trajectories followed and these models (i.e. the types) could thus be assimilated to a kind of random error term containing no pertinent information. This therefore amounts to make the assumption that the trajectories are generated by well established models that are clearly distinct from one another. Moreover, we are implicitly making the assumption that we have actually succeeded in finding the true models with the aid of cluster analysis.

In parallel, a second assumption is also being made, namely that the types obtained are equally different from one another.[12] But this is not the case. The "Apprenticeship-Employment" and "School-Employment" types are closer, because they have the same trajectory endpoint. By contrast, the "Higher Education" and "unemployed" types are particularly distant from each other.

These two assumptions are strong ones. One postulates the existence of models that have actually generated the trajectories, which is debatable from a sociological standpoint. In accordance with the life course paradigm, one can thus suppose that the individuals are subject to various influences and constraints which, each in their own way, contribute to the construction of the trajectory (or part of it). This is a long way from the search for clearly defined models of trajectory.

Once again, these assumptions may prove pertinent if the groups obtained are very homogeneous and very different from one another — a situation in which the average silhouette should be relatively high ($ASW > 0.7$, for example). But this is not the case here, since the average silhouette is less than 0.5.

The solution to this problem consists in using discrepancy analysis (Studer *et al.* 2011). This type of analysis makes it possible to measure the strength of the link by providing a pseudo-R^2, i.e. the share of the variation of the sequences explained by a variable, and also the significance of the association. One is thus freed from the assumption of trajectory models by directly computing the link, without preliminary clustering. Studer *et al.* (2011) gives an introduction to its implementation in R as well as a general presentation of the method. Only a brief overview is therefore offered here to support the argument.

Briefly, a bivariate test of the association between the sequences and the variable `test` can be computed with the `dissassoc` function available in the **TraMineR** library. The results that interest us here can be read on the line Pseudo R^2. It can thus be observed that the `test` variable explains 3.6% of the variability of the trajectories and the p-value is highly significant.

```
R> set.seed(1)
R> dsa <- dissassoc(mvaddist, mvad$test, weights = mvad$weight,
         weight.permutation = "diss", R = 5000)
R> print(dsa$stat)

                t0 p.value
Pseudo F   25.8285  0.0002
Pseudo Fbf 25.9594  0.0002
Pseudo R2   0.0351  0.0002
Bartlett    2.4336  0.7984
Levene     20.0631  0.0004
```

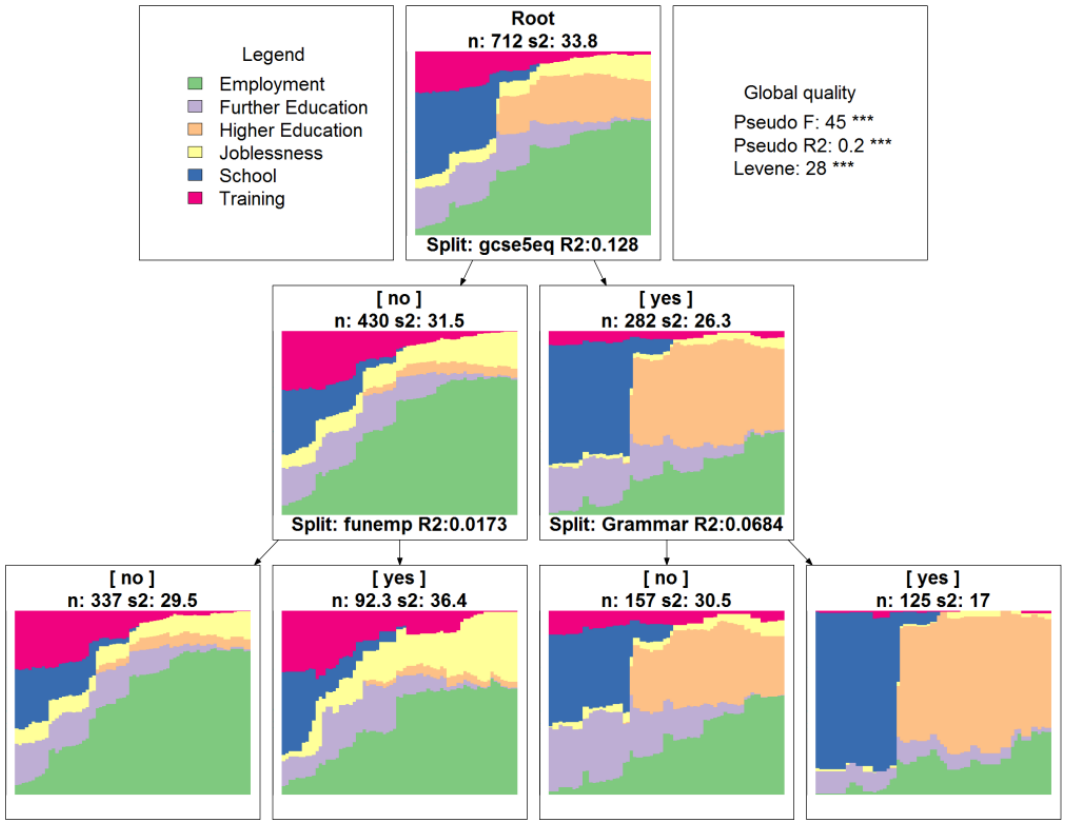
We may include several covariates by building a sequence regression trees with the `seqtree` and `seqtreedisplay`[13] functions (Studer *et al.* 2011).[14]

[12] Kohonen maps make it possible to visualize these distances between types of trajectories (Rousset and Giret 2007).

[13] The free GraphViz software (Gansner and North 1999) must be installed and accessible for the function to work properly. It can be downloaded from `http://www.graphviz.org/`.

[14] The ANOVA approach can also be extended to the multifactor case using the `dissmfacw` function.

```
R> tree <- seqtree(mvadseq ~ gcse5eq + Grammar + funemp, data = mvad,
         diss = mvaddist, weight.permutation = "diss")
R> seqtreedisplay(tree, type = "d", border = NA)
```



This analysis highlights the main effect as well as the interactions between covariates and the sequences. Here, the `gcse5eq` covariate (having good results at the end of compulsory schooling) has the most important effect. For those with good results, the most explanatory covariate is having been in a `Grammar` school. On the contrary, for those with poor results, the most important covariate is having an unemployed father.

Discrepancy analysis leads to a change of paradigm and frees us from the concept of clearly defined models of trajectories. Rather than relying on the search for models, we consider that the trajectories are set in a context which influences the construction of the trajectory in its own way. In other words, we seek to understand to what extent interindividual variability is explained by a context, while accounting for the diversity of the paths followed. From a conceptual standpoint, the assumptions underlying discrepancy analysis are very close to the principles put forward by the life course paradigm. Elder (1999) thus stresses the need to analyse the life course in times and places (the context) while preserving the interindividual variability and the agency of the actors.

In our view, building sequence typologies is a powerful method which has the advantage offering a descriptive point of view on the sequences while reducing the complexity of the analysis. However, its use in combination with inferential methods should be conducted with caution, since it can lead to misleading conclusions, as shown with the `test` variable.

8. Conclusion

This aim of this manual has been twofold: to present the **WeightedCluster** library and offer a step-by-step guide to building typologies of sequences for the social sciences. This library makes it possible, in particular, to represent graphically the results of a hierarchical cluster analysis, to group identical sequences in order to analyse a larger number of sequences,[15] to compute a set of measures of the quality of a partition, and also an optimized version of the PAM algorithm taking account of weightings. The library also offers procedures for facilitating the choice of a particular clustering solution and defining the number of groups.

In addition to the methods, we have also discussed the construction of sequence typologies in the social sciences. We argue that these methods offer an important descriptive viewpoint on sequences. Cluster analysis makes it possible to bring out recurrent patterns and/or "ideal-typical sequences" (Abbott and Hrycak 1990). The search for patterns is an important issue in several problem areas of the social sciences (Abbott 1995). In particular it makes it possible to bring to light the legal, economic or social constraints that frame the construction of individual courses. As Abbott and Hrycak (1990) note, while typical sequences may result from constraints that are found recurrently, these typical sequences can also act on reality by serving as models for the actors, who anticipate their own future. These different possibilities of interpretation make the creation of typologies a powerful tool.

All cluster analyses produce results, whatever their pertinence (Levine 2000). It is therefore necessary to discuss their quality so as to specify the scope of the results and not make unwarranted generalizations. In our view, this stage is too often absent from cluster analyses. In our case, this quality was low, as often happens in sequence analysis. With a higher quality (i.e. $ASW > 0.7$ for example), one can be more affirmative, since the partition probably reflects a strong structure identified in the data.

This is a powerful but also risky tool. In giving a unique name to each group, one tends to remove from the analysis the diversity of the situations found within each group. By way of an example, we noted that the length of the periods of unemployment varies significantly within the group that we named "Unemployed" and that this state is also found in other groups. This simplification makes sense if the aim is to bring out the recurrent patterns for descriptive purposes.

However, the typologies should not be used in an explanatory procedure. That would amount to postulating the existence of clearly defined models that actually generated the trajectories and have been identified through cluster analysis. Not only can this procedure result in misleading conclusions if these assumptions are not verified, but the assumptions are debatable from a sociological viewpoint. In accordance with the life course paradigm, one can suppose that the individuals are subject to various influences and constraints which, each in their own way, contribute to the construction of the trajectory (or part of it). We are thus a long way from the search for clearly defined trajectory models.

References

Abbott A (1995). "Sequence Analysis: New Methods for Old Ideas." *Annual Review of Sociology*, **21**, 93–113.

[15] The detail of this procedure is set out in Annex A.

Abbott A, Hrycak A (1990). "Measuring Resemblance in Sequence Data: An Optimal Matching Analaysis of Musicians' Careers." *American Journal of Sociolgy*, **96**(1), 144–185.

Abbott A, Tsay A (2000). "Sequence Analysis and Optimal Matching Methods in Sociology, Review and Prospect." *Sociological Methods and Research*, **29**(1), 3–33. (With discussion, pp 34-76).

Aisenbrey S, Fasang AE (2010). "New Life for Old Ideas: The "Second Wave" of Sequence Analysis Bringing the "Course" Back Into the Life Course." *Sociological Methods and Research*, **38**(3), 430–462.

Belbin L, Faith DP, Milligan GW (1992). "A Comparison of Two Approaches to beta-Flexible Clustering." *Multivariate Behavioral Research*, **3**(3), 417–433.

Calinski RB, Harabasz J (1974). "A Dendrite Method for Cluster Analysis." *Communications in Statistics*, **3**, 1–27.

Elder GH (1999). *Children of the Great Depression*. Westview Press, Boulder.

Fernández A, Gómez S (2008). "Solving Non-Uniqueness in Agglomerative Hierarchical Clustering Using Multidendrograms." *Journal of Classification*, **25**, 43–65. ISSN 0176-4268. 10.1007/s00357-008-9004-x, URL http://dx.doi.org/10.1007/s00357-008-9004-x.

Gabadinho A, Ritschard G, Müller NS, Studer M (2011). "Analyzing and visualizing state sequences in R with TraMineR." *Journal of Statistical Software*, **40**(4), 1–37. URL http://www.jstatsoft.org/v40/i04/.

Gansner ER, North SC (1999). "An Open Graph Visualization System and Its Applications to software engineering." *Software - Practice and Experience*, **30**, 1203–1233.

Hennig C, Liao TF (2010). "Comparing latent class and dissimilarity based clustering for mixed type variables with application to social stratification." *Technical report*, Department of Statistical Science, UCL, Department of Sociology, University of Illinois.

Hollister M (2009). "Is Optimal Matching Suboptimal?" *Sociological Methods Research*, **38**(2), 235–264. doi:10.1177/0049124109346164.

Hubert L, Arabie P (1985). "Comparing Partitions." *Journal of Classification*, **2**, 193–218.

Hubert L, Levin J (1976). "A general statistical framework for assesing categorical clustering in free recall." *Psychological Bulletin*, **83**, 1072–1080.

Kaufman L, Rousseeuw PJ (1990). *Finding groups in data. an introduction to cluster analysis*. Wiley, New York.

Lesnard L (2006). "Optimal Matching and Social Sciences." *Manuscript*, Observatoire Sociologique du Changement (Sciences Po and CNRS), Paris.

Lesnard L (2010). "Setting Cost in Optimal Matching to Uncover Contemporaneous Socio-Temporal Patterns." *Sociological Methods Research*, **38**(3), 389–419. doi:10.1177/0049124110362526. URL http://smr.sagepub.com/cgi/content/abstract/38/3/389.

Levine J (2000). "But What Have You Done For Us Lately." *Sociological Methods & Research*, **29 (1)**, pp. 35–40.

Maechler M, Rousseeuw P, Struyf A, Hubert M (2005). "Cluster Analysis Basics and Extensions." Rousseeuw et al provided the S original which has been ported to R by Kurt Hornik and has since been enhanced by Martin Maechler: speed improvements, silhouette() functionality, bug fixes, etc. See the 'Changelog' file (in the package source).

McVicar D, Anyadike-Danes M (2002). "Predicting successful and unsuccessful transitions from school to work using sequence methods." *Journal of the Royal Statistical Society A*, **165**(2), 317–334.

Milligan G (1989). "A study of the beta-flexible clustering method." *Multivariate Behavioral Research*, **24**(2), 163–176.

Milligan G, Cooper M (1985). "An examination of procedures for determining the number of clusters in a data set." *Psychometrika*, **50**(2), 159–179. URL http://ideas.repec.org/a/spr/psycho/v50y1985i2p159-179.html.

Milligan GW, Cooper MC (1987). "Methodology Review: Clustering Methods." *Applied Psychological Measurement*, **11**(4), 329–354. doi:10.1177/014662168701100401.

Müllner D (2011). *fastcluster: Fast hierarchical clustering routines for R and Python*. Version 1.1.3, URL http://math.stanford.edu/~muellner.

Oksanen J, Blanchet FG, Kindt R, Legendre P, Minchin PR, O'Hara RB, Simpson GL, Solymos P, Stevens MHH, Wagner H (2012). *vegan: Community Ecology Package*. R package version 2.0-3, URL http://CRAN.R-project.org/package=vegan.

Reynolds A, Richards G, de la Iglesia B, Rayward-Smith V (2006). "Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms." *Journal of Mathematical Modelling and Algorithms*, **5**, 475–504. ISSN 1570-1166. 10.1007/s10852-005-9022-1, URL http://dx.doi.org/10.1007/s10852-005-9022-1.

Rousset P, Giret JF (2007). "Classifying qualitative time series with SOM: the typology of career paths in France." In *Proceedings of the 9th international work conference on Artificial neural networks*, IWANN'07, pp. 757–764. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-73006-4. URL http://dl.acm.org/citation.cfm?id=1768409.1768513.

Shalizi C (2009). "Distances between Clustering, Hierarchical Clustering." *Lectures notes*, Carnegie Mellon University.

Struyf A, Hubert M, Rousseeuw P (1997). "Clustering in an Object-Oriented Environment." *Journal of Statistical Software*, **1**(4), 1–30. ISSN 1548-7660. URL http://www.jstatsoft.org/v01/i04.

Studer M, Ritschard G, Gabadinho A, Müller NS (2011). "Discrepancy Analysis of State Sequences." *Sociological Methods and Research*, **40**(3), 471–510. doi:10.1177/0049124111415372.

Widmer ED, Levy R, Pollien A, Hammer R, Gauthier JA (2003). "Entre standardisation, individualisation et sexuation: une analyse des trajectoires personnelles en Suisse." *Revue suisse de sociologie*, **29**(1), 35–67.

A. Aggregating identical sequences

The algorithms that we have presented all take into account the weighting of observations. This makes it possible to group identical observations by giving them a higher weighting. One can then perform cluster analysis on these grouped data, which considerably reduces the computing time and the memory used.[16] The analysis is completed by "disaggregating" so as to reintegrate the typology into the initial data.

These different operations are performed easily with the `wcAggregateCases` function in the **WeightedCluster** library. This function identifies the identical cases in order to group them. In a second stage, the object returned also makes it possible to perform the reverse operation, i.e. to disaggregate the data.

Let us return to the example that we have used from the start of this manual. The following code makes it possible to identify the identical sequences. The `wcAggregateCases` function takes two parameters: `a data.frame` (or a matrix) which contains the cases to be aggregated, and an optional vector of `weights`.

```
R> ac <- wcAggregateCases(mvad[, 17:86], weights = mvad$weight)
R> ac


Number of disaggregated cases:  712
Number of aggregated cases:  490
Average aggregated cases:  1.45
Average (weighted) aggregation:  1.45
```

The object returned by the `wcAggregateCases` function (here `ac`) gives some basic information on the grouping. It can be seen that initial data set contains 712 observations, but only 490 different sequences. The object returned contains three particularly interesting elements.

- `aggIndex`: index of unique objects.
- `aggWeights`: Number of times (weighted if necessary) that each unique object of `aggIndex` appears in the data.
- `disaggIndex`: index of initial objects in the list of unique objects. This information makes it possible to disaggregate the data. An example of its use will be given later.

With this information, we can create an object `uniqueSeq` of the unique sequences weighted by the number of times they appear in the data. In the following code, `ac$aggIndex` makes it possible to select the unique sequences and the vector `ac$aggWeights` contains the weighting of each of these sequences.

[16] The quantity of memory required increases quadratically.

```
R> uniqueSeq <- seqdef(mvad[ac$aggIndex, 17:86], alphabet = mvad.alphabet,
         states = mvad.scodes, labels = mvad.labels, weights = ac$aggWeights)
```

We can then compute different clustering solutions. Here, we compute the distance matrix before using this information for the `wcKMedoids` function. As before, we use here the vector `ac$aggWeights` to apply the weightings.

```
R> mvaddist2 <- seqdist(uniqueSeq, method = "OM", indel = 1.5, sm = subm.custom)
R> pamclust4ac <- wcKMedoids(mvaddist2, k = 4, weights = ac$aggWeights)
```

Now that the clustering has been done, the information contained in `ac$disaggIndex` makes it possible to move back. For example, the typology in the original (non-aggregated) `data.frame` can be added with the aid of the following code. The vector `pamclust4ac$clustering` contains the membership of each unique sequence in the clusters. Using the index `ac$disaggIndex`, we return to the original data set, i.e. we obtain the membership of each (non-unique) sequence in the clusters.

```
R> mvad$acpam4 <- pamclust4ac$clustering[ac$disaggIndex]
```

The following table gives the distribution of the original cases between the typology we obtained from the disaggregated cases (variable `pam4`) and the one obtained from the aggregated data (variable `acpam4`). It can be seen that the two solutions contain the same cases; only the labels differ.

```
R> table(mvad$pam4, mvad$acpam4)


              28  87 414 444
  Train-Empl   0   0 189   0
  School-Empl  0 307   0   0
  High Ed    160   0   0   0
  Unempl       0   0   0  56
```

The aggregation of identical cases is a very useful functionality for large data sets. It is often not possible to compute the set of distances because of insufficient memory. In such cases, the use of `wcAggregateCases` could solve the problem.

B. Notes on performances

The algorithms for partitioning around medoids available in the **WeightedCluster** library are highly optimized. Internally, the library offers several variants of the PAM algorithm. The choice among these variants depends on the type of the distance object passed to the function and on the `method` argument.

The `diss` argument may be a distance matrix or a `dist` object. In the first case, each distance is registered twice, which can rapidly give rise to problems of available memory, but

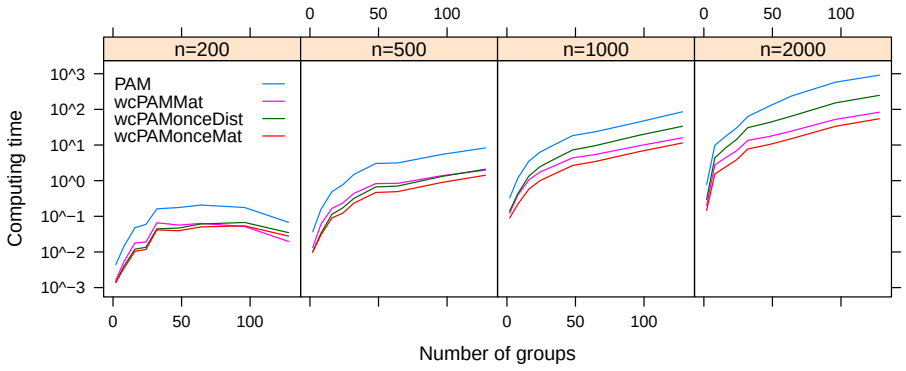
the algorithm is generally faster (see the performance comparisons below). If the argument is of the `dist` type, only the lower triangle of the distance matrix is stored in memory, but this gain comes at the expense of the speed of the algorithm.

In contrast to the PAM algorithm available in the **cluster** library, the distance matrix is not copied internally, making it possible to perform a clustering of a considerably larger number objects before reaching the memory limits of the machine.

The `method` argument specifies the algorithm used. Two versions are available: the original version of the PAM algorithm and "PAMonce". The "PAMonce" algorithm implements the optimizations proposed by Reynolds *et al.* (2006), which consist in evaluating the cost of the suppression of a medoid only once (rather than n times in the original version). We have also included in this algorithm a second optimization which consists in not evaluating the gain from replacing a medoid by a given object if the distance between them is zero. This optimization is consistent with the mathematical definition of a measure of distance, whereby two objects are identical if, and only if, their distance is zero. This optimization only makes sense insofar as the objects to be grouped contain duplicates and in particular only so long as the measure of dissimilarity used respects this condition. It should be noted that measures of dissimilarity generally do respect it.

To measure the impact of these optimizations on performance, we ran several simulations. They grouped in $k \in (2, 8, 16, 24, 32, 48, 64, 96, 128)$ groups a set of $n \in (200, 500, 1000, 2000)$ observations whose x and y coordinates were randomly generated with a uniform distribution. Figure 1 shows the changes in computing time (on a logarithmic scale) according to the values of n and k. Figure 2 shows the changes in total relative time, i.e. divided by the time taken by the fastest algorithm for this solution, with the same parameters.

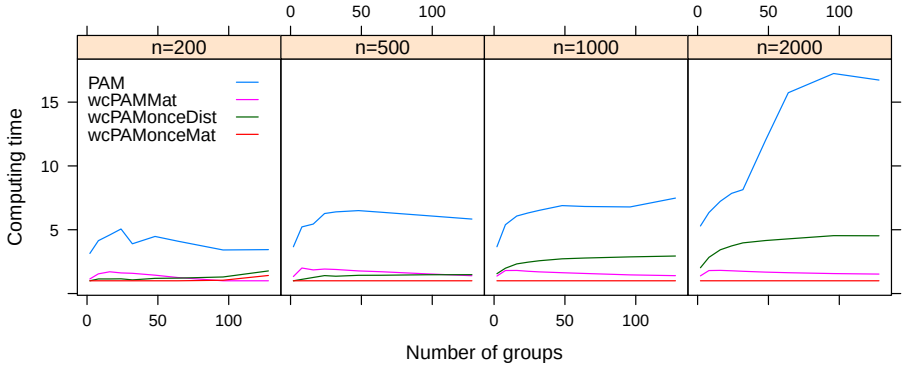
Figure 1: Changes in computing time according to n and k



The solutions proposed by **WeightedCluster** are faster and the differences increase as k and n increase. The use of a distance matrix rather than the `dist` object makes it possible to reduce the computing time. The gain in memory is thus indeed achieved at the expense of computing time. The same is true for the optimizations of "PAMonce", which brings a significant gain. Relative to the cluster library, the gains are particularly great (around a factor of 15) when n is greater than 1000 and k is large.

It should also be noted that if the data contain identical cases, the gains are potentially still greater, since these cases can grouped by using `wcAggregateCases`. This operation

Figure 2: Changes in relative time according to n and k



considerably reduces the memory requirement and the computing time.

C. Details of quality measures

C.1. Average Silhouette Width (ASW)

Originally proposed by Kaufman and Rousseeuw (1990), this index is based on a notion of coherence of the assignment of an observation to a given group, which is measured by comparing the average weighted distance, labelled a_i, of an observation i with the other members of its group and the average weighted distance from the closest group, labelled b_i.

Let k be the group of observation i, W_k the sum of the weightings of the observations belonging to group k, w_i the weight of observation i and ℓ one of the other groups; the silhouette of an observation is computed as follows:

$$a_i = \frac{1}{W_k - 1} \sum_{j \in k} w_j d_{ij} \tag{1}$$

$$b_i = \min_{\ell} \frac{1}{W_\ell} \sum_{j \in \ell} w_j d_{ij} \tag{2}$$

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{3}$$

Equation 1 supposes that the weighting unit is equivalent to one observation, because the weighted sum of distances is divided by $W_k - 1$. This assumption is not problematic when the weights are computed from an aggregation of identical cases (see Annex A) or when the data are not weighted. However, when some of the w_i or some of the W_k are lower than one, the a_i are undefined and the value of the silhouette cannot be interpreted. Those situations are quite frequent, when the weights aim at correcting the sample representativeness (as it was the case in our example). To address this issue, we propose to use aw_i instead of a_i when

computing the silhouette. This aw_i value is computed as follows.

$$aw_i = \frac{1}{W_k} \sum_{j \in k} w_j d_{ij} \tag{4}$$

There are two interpretations for the aw_i value. First, it is the distance between the observation and its own group, when all the observations of the group are used to define the group. In the original formulation, the observation is removed from the group before computing the a_i value. Second, the aw_i value can be interpreted as the a_i value when the weighting unit is as small as possible, i.e. when it tends to zero.

For both variant, the index finally used corresponds to the weighted average of the silhouettes s_i. This value is returned in the `stats` element. The weighted average of the silhouettes of each group is given in the `ASW` element and measures separately the coherence of each group.

C.2. C index (HC)

Developed by Hubert and Levin (1976), this index compares the partition obtained with the best partition that could be obtained with this number of groups and this distance matrix. The index ranges between 0 and 1, with a small value indicating a good partition of the data. More formally, it is defined as follows. Let S be the sum of the within-group distances weighted by the product of the weights of each observation, W the sum of the weights of the within-group distances, S_{min} the weighted sum of the W smallest distances, and S_{max} the weighted sum of the W greatest distances:

$$C_{index} = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{5}$$

C.3. "Hubert's Gamma" (HG, HGSD) and "Point Biserial Correlation" (PBC)

These indexes measure the capacity of a partition to reproduce the distance matrix by computing the association between the original distance d and a second matrix d_{bin} taking the value 0 for observations classified in the same partition and 1 otherwise. To use the weightings, we use W^{mat}, the matrix of the product of the weightings $W^{mat}_{ij} = w_i \cdot w_j$.

Hubert and Arabie (1985) suggest measuring this association by Goodman and Kruskal's Gamma (HG) or Somers' D (HGSD) to take account of the ties in the distance matrix. **WeightedCluster** uses the following formulae based on the fact that d_{bin} takes only two different values. Let T be the weighted cross table between the values of d in rows (ℓ rows) and of d_{bin} in two columns (0 or 1) computed by using the weightings W^mat; the number of concordant pairs C, the number of discordant pairs D and the number of ties on d E are defined as follows:

$$C = \sum_{i=1}^{\ell} \sum_{i'=1}^{i-1} T_{i'0} \qquad D = \sum_{i=1}^{\ell} \sum_{i'=1}^{i-1} T_{i'1} \qquad E = \sum_{i=1}^{\ell} T_{i0} + T_{i1}$$

$$HG = \frac{C - D}{C + D} \qquad HGSD = \frac{C - D}{C + D + E}$$

Hennig and Liao (2010) suggest using Pearson's correlation instead, a solution also known as "Point Biserial Correlation" (PBC equation 6) (Milligan and Cooper 1985). Let s_d and s_{d_bin} be the standard deviation weighted by W^{mat} of d and d_{bin} respectively, $s_{d,d_{bin}}$ the covariance weighted by W^mat between d and d_{bin}; this correlation is computed as follows:

$$PBC = \frac{s_{d,d_{bin}}}{s_{d_{bin}} \cdot s_{d_{bin}}} \tag{6}$$

C.4. CH index (CH, CHsq, R2, R2sq)

Calinski and Harabasz (1974) suggest using the statistic F of the analysis of variance by computing it based on the squared Euclidean distances. On the same bases, one can compute a pseudo R^2, i.e. the share of variability explained by a partition. Studer *et al.* (2011) suggest an extension of these measures to the weighted data.

D. Construction of the test variable

The `test` variable is constructed to explain the variability within the clusters. To do so, we use an MDS for weighted data (Oksanen *et al.* 2012).

```
R> library(vegan)
R> worsq <- wcmdscale(mvaddist, w = mvad$weight, k = 2)
```

Cluster analysis mainly explains the differences on the first axis. So we create the variable test according to the second axis. In order for the chi-square test to be close to zero, the proportions of "test" and "non-test" must be equivalent in each group. To satisfy these two constraints, we use the median of the second axis of the MDS in each group as a separation point.

```
R> library(isotone)
R> mvad$test <- rep(-1, nrow(mvad))
R> for (clust in unique(pamclust4$clustering)) {
         cond <- pamclust4$clustering == clust
         values <- worsq[cond, 2]
         mvad$test[cond] <- values > weighted.median(values,
             w = mvad$weight[cond])
     }
R> mvad$test <- factor(mvad$test, levels = 0:1, labels = c("non-test",
         "test"))
```

Affiliation:

Matthias Studer
Institute for Demographic and Life Course Studies

University of Geneva
CH-1211 Geneva 4, Switzerland
E-mail: matthias.studer@unige.ch
URL: http://mephisto.unige.ch/weightedcluster/