

# Extracting knowledge from life courses: clustering and visualization

Nicolas S. Müller<sup>1</sup>, Alexis Gabadinho<sup>2</sup>, Gilbert Ritschard<sup>1</sup>,  
Matthias Studer<sup>1</sup>

<sup>1</sup> Département d'économétrie, Université de Genève, {nicolas.muller,  
gilbert.ritschard, matthias.studer}@metri.unige.ch

<sup>2</sup> Laboratoire de démographie, Université de Genève, alexis.gabadinho@ses.unige.ch

**Abstract.** This article presents some of the facilities offered by our TraMiner R-package for clustering and visualizing sequence data. Firstly, we discuss our implementation of the optimal matching algorithm for evaluating the distance between two sequences and its use for generating a distance matrix for the whole sequence data set. Once such a matrix is obtained, we may use it as input for a cluster analysis, which can be done straightforwardly with any method available in the R statistical environment. Then we present three kinds of plots for visualizing the characteristics of the obtained clusters: an aggregated plot depicting the average sequential behavior of cluster members; a sequence index plot that shows the diversity inside clusters and an original frequency plot that highlights the frequencies of the  $n$  most frequent sequences. TraMiner was designed for analysing sequences representing life courses and our presentation is illustrated on such a real world data set. The material presented should, nevertheless, be of interest for other kind of sequential data such as DNA analysis or web logs.

## 1 Introduction

This paper discusses some of the tools made available in TraMiner, a package that we developed for the R statistical environment. TraMiner, which stands for life trajectory miner, is a toolbox for the analysis and visualization of sequential data. Though TraMiner is mainly intended for analyzing life courses, most of its features may be of interest for other kind of sequential data, such as DNA sequences or web logs for instance. The features discussed in this paper include the computation of the optimal matching (OM) distance between any two sequences, also known as the edit distance, and three kinds of graphical representation of sets of sequences. The OM distances can be used for clustering the sequences, and the graphical representations for characterizing the obtained clusters.

The methods and graphics are presented through a real world example. More specifically, we consider life courses of Swiss people born during the first half of the 20th century, using data from a retrospective survey carried out by the Swiss Household Panel. The life courses are made up of constituent events in familial life such as the leaving from the family home, the birth of the first child, the first

marriage or the first divorce. By using these events as our basis, it is possible to look at individual life courses in the form of a sequence of states, where each event that occurs in a person's life course corresponds to a change of state.

This article is divided in the following way. The first part presents the source data and the necessary transformations to build state sequences from events. The second part presents the optimal matching method and the operations cost problematic. The third part concerns the visualisation tools and their functioning. The fourth part reviews quickly the existing software for optimal matching analysis and presents our R module "TraMiner". We finally conclude on the possibilities that such methods bring to the social sciences.

## 2 Data

The data we extract from the answers to a survey is shown in a chart/table where each line represents an individual/person and each column a variable (Table 1).

**Table 1.** An example of data showing events

ind.	birth	leaving home	marriage	child	divorce
1	1974	1992	1994	1996	n/a

The move to a sequential representation is not without interest. The difficulty lies in representing a combination of events which either took place or did not, at a certain age, by a unique state. In a more formal manner, we define the state of a person at a given age as information on events that have already happened. From any given state, one can say which events have already taken place. One or more events which occur during the year  $t$  will cause the individual to go from the state he was in at  $t-1$  to a new state. The definition of states based on events is a problem specific to the type of data and the problematics of the research. A simple way to proceed would be to create a state for each combination of events. By so doing, the number of states would rise to  $2^n$  for  $n$  events, which renders the interpretation difficult whenever many events have to be taken into consideration. We have therefore chosen to group certain events in accordance with the research objectives.

For the purpose of this research, we retained four events which constitute familial life: the departure from the family home, the first marriage, the first divorce and the birth of the first child. Table 2 shows the encoding of the states which we have drawn up in relation to the four selected events. The number of events was reduced from 16 to 8, notably by eliminating impossible states (all those which contain a divorce without a previous marriage), or by combining two states (for example state 2 concerns married individuals who have not left the family home regardless of whether or not they have any children). By referring to

this list of states and to the example shown in Table 1, the result of the creation of a sequence of familial life is found in Table 3.

**Table 2.** List of states

	leaving home	marriage	children	divorce
0	no	no	no	no
1	yes	no	no	no
2	no	yes	yes/no	no
3	yes	yes	no	no
4	no	no	yes	no
5	yes	no	yes	no
6	yes	yes	yes	no
7	yes/no	yes	yes/no	yes

**Table 3.** An example of data as a sequence of states

individual	1974	...	1991	1992	1993	1994	1995	1996	1997	1998	...
1	0	...	0	1	1	3	3	6	6	6	...

The data used in this article comes from the retrospective biographical survey carried out by the Swiss Household Panel ([www.swisspanel.ch](http://www.swisspanel.ch)) in 2002. We have retained the individuals who were at least 30 years old at the time of the survey in order to have only complete sequences between the ages of 15 and 30. In this way, our sample is made up of 4318 individuals born between 1909 and 1972.

### 3 Optimal matching

The method of analysing sequences which we use in this work is that of optimal matching (OM). The algorithm used is inspired by the methods of sequence alignment and dynamic programming used in molecular biology, especially for the comparison of proteins, or sequences of ADN thought to be homologous [1; 2]. This method of working was devised in order to enable the rapid calculation of numerous sequences in order to find the correspondence between them. The first algorithms of OM based on the Levenshtein distance [3] appeared at the beginning of the 1970s and their first use in social science goes back to the article by Abbott and Forrest on their application to historical data [4]. We owe numerous methodological articles on the use of these methods in social sciences, especially in sociology, to Abbott [5; 6]. The interest in applying this method

to a life course is that one can then go on to a clustering using the distances calculated by OM.

### 3.1 Method

The OM method uses the Needleman-Wunsch algorithm to compute a distance between two sequences based on the Levenshtein distance [3; 2]. Take  $\Omega$ , the set of possible operations, and  $a[\omega]$  the result of the application of the operations  $\omega \in \Omega$  on the sequence  $a$ . We take into account 3 types of operations: the insertion of an element, the suppression of an element, or the substitution of one element by another. We attribute a cost  $c(\omega)$  to each  $\omega$  which corresponds to the cost of implementing the operation  $\omega \in \Omega$ . The OM distance between a sequence  $a$  and a sequence  $b$  can be formulated as follows:  $d(a, b) = \min\{c[\omega_1, \dots, \omega_k] \mid b = a[\omega_1, \dots, \omega_k], \omega \in \Omega, k \geq 0\}$ , with  $c[\omega_1, \dots, \omega_k] = \sum_{i=1}^k c[\omega_i]$ . In other words, for each pair of sequences, we look for the combination of operations with the lowest total cost that renders both sequences identical.

The Needleman-Wunsch algorithm for finding the minimal distance  $d(a, b)$  can be summarized in 4 points :

1. A matrix  $D$  of size  $m = (\text{length of } a + 1)$  and  $n = (\text{length of } b + 1)$  is created.
2. The value of cell  $d_{0,0}$  is set to 0 and the values of the first column and line are computed using the following recursive equation :

$$d_{i,0} = d_{i-1,0} + c[\omega(a_i, \phi)] \quad (1)$$

$$d_{0,j} = d_{0,j-1} + c[\omega(\phi, b_j)] \quad (2)$$

where  $c[\omega(a_i, \phi)] = c[\omega(\phi, b_j)]$  is the cost of an insertion/deletion operation.

3. The remaining cells are recursively filled with the following equation :

$$d_{i,j} = \min \begin{cases} d_{i,j-1} + c[\omega(\phi, b_j)] \\ d_{i-1,j-1} + c[\omega(a_i, b_j)] \\ d_{i-1,j} + c[\omega(a_i, \phi)] \end{cases} \quad (3)$$

where  $c[\omega(a_i, b_j)]$  is the cost of replacing the  $i$ th state of sequence  $a$  by the  $j$ th state of sequence  $b$ .

4. The minimal distance is then found in the cell  $d_{m,n}$ .

This algorithm is detailed in several other publications ([2; 7; 1]).

**Example** We will now present a visual example of how the distance between a sequence **a** (ECD) and sequence **b** (ABCD) is computed by the algorithm. Firstly, the matrix  $D$  of size 3x4 is initialized with its cell  $d_{0,0} = 0$ . To simplify the example, we attribute a cost of 1 to all operations  $\omega \in \Omega$ . The first line and the first column are recursively filled according to the equations (2) and (1). The resulting matrix is on the left panel of Table 4. Each cell is then recursively

defined by equation (3). For example, cell  $d_{1,1}$  takes its value from the minimum of these three possibilities :  $d_{0,0} + c[\omega(E, A)]$ ,  $d_{0,1} + c[\omega(E, \phi)]$  or  $d_{1,0} + c[\omega(\phi, A)]$ . The  $d_{1,1}$  value is then 1, corresponding to a substitution of A by E ( $\omega(E, A)$ ). After filling the entire matrix, we find that the minimal sum of costs to transform sequence **a** into sequence **b** is 2, corresponding to the substitution of A by E and then the deletion of B ( $\omega(B, \phi)$ ).

**Table 4.** On the left : initial matrix, on the right : completed matrix

	A	B	C	D
	0	1	2	3
E	1			
C	2			
D	3			

	A	B	C	D
	0	1	2	3
E	1	<b>1</b>	2	3
C	2	2	<b>2</b>	3
D	3	3	3	<b>2</b>

### 3.2 Cost definition

As previously seen, a cost  $c$  can be ascribed to the operations  $w \in \Omega$ . The costs of substitution, in which we were particularly interested, can be represented in the form of a symmetrical matrix which defines a value for each pair of states. In the context of a use in social science, it is extremely difficult to base the attribution of these values on a theoretical model. Such practice has given rise to a debate [8]. It is indeed difficult to determine the cost of transforming one state into another, yet it is interesting and sometimes of fundamental importance to be able to differentiate between these costs. In order to do that, two available methods have been tried out on our data. The cost of transforming a state  $i$  into a state  $j$  is calculated in terms of the rate of longitudinal transition:  $c[w(i, j)] = c[w(j, i)] = 2 - p(i_t | j_{t-1}) - p(j_t | i_{t-1})$ . The basic cost is fixed at 2 and the greater the transition probability  $p(i_t | j_{t-1})$  of going from state  $i$  to state  $j$ , and vice versa, the more the cost decreases. Thus, the substitutions corresponding to the most frequently observed transitions will be less costly than those which never occur. Another method offered by the software T-COFFEE/SALTT [9], involves iteratively calculating an optimal substitution costs matrix (Gauthier et al. [10]).

In this paper, a value of 1 was attributed to the cost of insertion and deletion in the solution based on the rate of transition. The reason for this choice was to favour the operations of insertion or deletion in the case where a sequence has the same states than another one but one or two years later or sooner. This allows to keep a small distance between this kind of sequences.

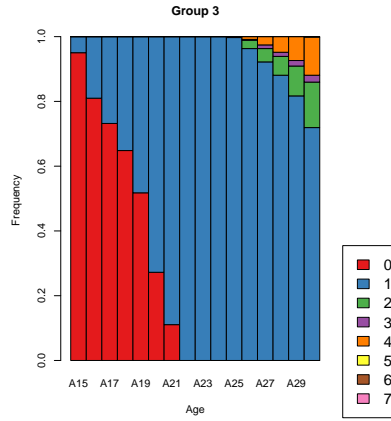
### 3.3 Classification

We are now able to produce a matrix for distances measuring the differences between individuals' life courses. It can be used in a agglomerative hierarchical clustering using the Ward criterion. We chose a 4 clusters solution for two

reasons: in the first place, the dendrogram resulting from the hierarchical clustering shows a clear split at the 4 clusters level; in the second place, the 4 clusters receive good interpretation both visually and through logistic regression models.

## 4 Visualization

Life state sequences are difficult to visualize for several reasons. Firstly, states are categorical values that cannot be easily plotted over time like time series. Furthermore, it is not possible to calculate an "average" sequence for representing each cluster. Plotting individual life sequences like time series, i.e. with time on the X axis and the arbitrarily sorted states on the Y axis, would become unreadable when a lot of sequences are drawn. That is why we need specific plots for visualizing life state sequences.



**Fig. 1.** Relative frequencies of individuals in each state at each age (cluster 3)

In this paper, we consider three kinds of plots. The first one can be used to get a quick overview of the clusters, such as shown for cluster 3 on Fig. 1. For each possible age, from 15 to 30, we find the proportion of individuals being in each different states (from 0 to 7). These plots give only a very general overview of the clusters as it shows only aggregated results. That is why we need additional types of graphics.

The *frequencies plots* represent each state of the life sequences of the  $n$  most frequent sequences. Fig. 2 shows the 10 most frequent sequences in each of the 4 clusters, as they were defined by the hierarchical clustering. Each color corresponds to a state, as they are described in Table 2. The height of each sequence is proportional to its relative frequency.

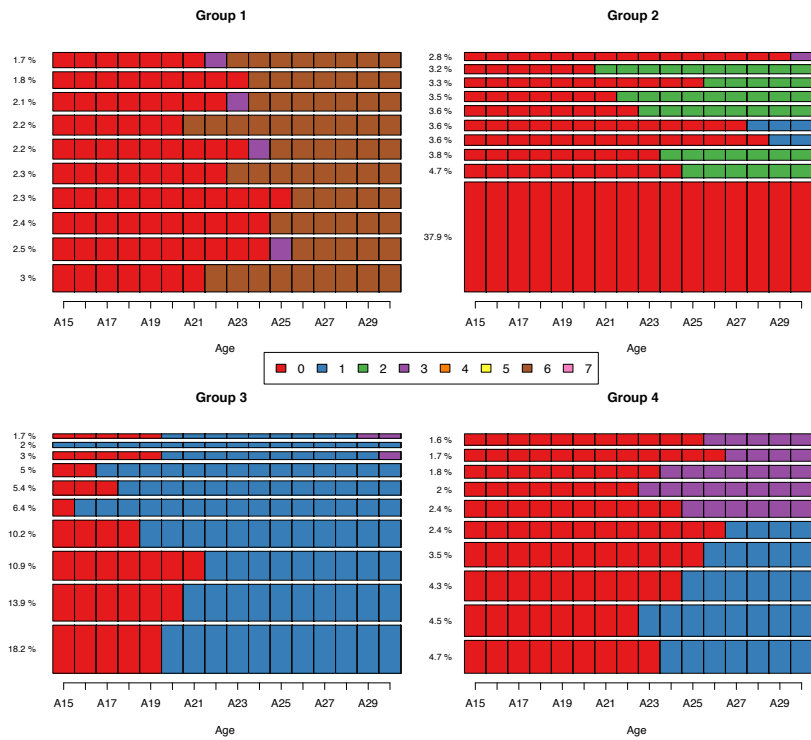
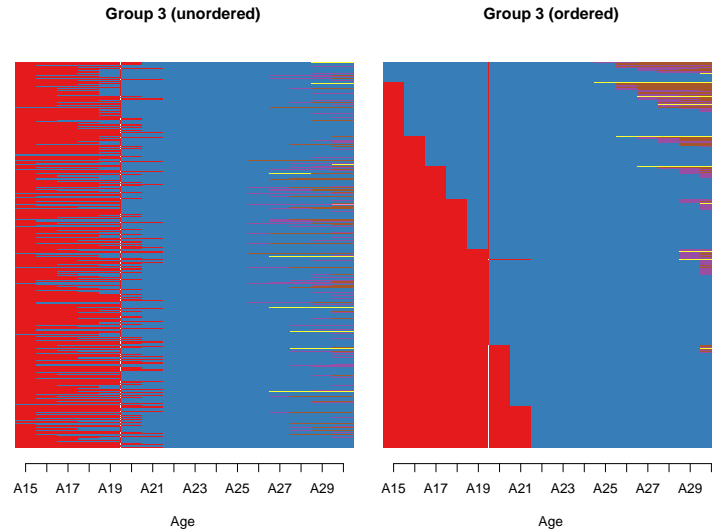


Fig. 2. 10 most frequent sequences in each cluster

Alternatively, we can visualize each sequence by a stacked horizontal line. We obtain this way a so-called indexplot ([11]). This is similar to the previous representation, except that all the sequences are drawn. In order to get a better view of what kind of sequences are in a cluster, it is useful to sort them before plotting them. In this case, we have computed an optimal matching distance between a random sequence in the cluster and all the other ones, following an idea from Brzinsky-Fay et al. [12]. The example from Fig. 3 shows clearly the benefit of sorting the sequences for getting a usable representation of the cluster. As we can see, the first kind of plot (Fig. 1) is not an accurate overview of the clusters content. That is why using graphs based on individual data is important to better understand the classification results and to observe the diversity inside each cluster.

## 5 Software

We have implemented the methods for computing distances and visualizing the sequences in a package named "TraMiner" for the R statistical software environ-



**Fig. 3.** Sorted and unsorted indexplots

ment. This section reviews some of the other softwares available for sequence analysis. It then introduces the TraMiner package and shortly comments about the complexity of the different methods.

### 5.1 Review of existing softwares

The major problem for the social scientist who wants to do sequence analysis is the lack of implementation of these methods in standard statistical software. So far, among the most widely used statistical packages, only Stata and SAS provide optimal matching analysis through third-party modules [12; 13]. The software TDA [7] is also able to compute optimal matching distances but is no longer under development. The same is true for OPTIMIZE developed by Abbott. The packages T-COFFEE/SALTT [9] and CHESA [14] compute, among other things, optimal matching distances, but provide no plotting tools.

In the academic field, R is becoming an important tool for several reasons : it is open-source software, its programming language is high-level and derived from S (a widespread statistical programming language) and most of the statistical methods are accessible through modules. That is why we decided to create a module in R for sequence analysis. Indeed, the plotting sub-system is efficient and flexible enough to allow us to create custom graphics like the ones presented in this paper. It also permits to directly access and interact with many other already implemented methods, such as hierarchical clustering, logistic regression or multi-dimensional scaling. The purpose of TraMiner is to allow the user to do everything related to sequence analysis within the R software environment,

starting from sequence data creation to visualizing the results of a clustering. This paper is centered on the optimal matching method, however TraMiner offers other distances or measures such as those proposed by Elzinga ([15; 16]).

## 5.2 Complexity

The main algorithm, as we have said before, is the optimal matching method to compute distances between sequences. As the distances are symmetrical (the distance between A and B equals the one between B and A), the number of distances computed is  $\frac{n*(n-1)}{2}$ . To compute the distance between two sequences, the algorithm builds a matrix which size is defined by the length of the sequences. The solution is found by filling all the cells of this matrix with the recursive equation (3). The complexity of this part of the algorithm is  $O(\ell_1 * \ell_2)$ , where  $\ell_1$  is the length of the first sequence and  $\ell_2$  the length of the second one. Assuming that all the sequences are of equal length, as this is the case in this paper, then the total complexity is  $O(n^2 * \ell^2)$ . The plotting methods require only the drawing of the sequences and thus are of complexity  $O(n)$ .

In order to speed up the computation of distances, the Needleman-Wunsch algorithm is written in C instead of the R programming language; as a result, a R shared library is created and the method is called from R. The C version is on our test machine about 15 times faster than the one written in R. In order to reduce the number of distances to be computed, we implemented a process that consider only one instance of multiple instances of a same sequence. Our life sequences data set contains for instance only 841 different sequences among 4318. The number of distances computed was thus reduced from  $\frac{4318*4317}{2} = 9320403$  to  $\frac{841*840}{2} = 353220$ . The computation took less than 15 seconds on our test machine (Intel Core2Duo@2GHZ).

## 6 Conclusion

We presented in this paper some of the salient tools offered by our TraMiner R-package for extracting useful knowledge from sequential data. More specifically, we discussed OM, one among the methods implemented for evaluating the closeness between sequences describing life courses. We presented also three kinds of complementary graphics for visualizing sequence data. The first of them gives some kind of average view of a set of sequences, the second one depicts the frequencies of the more frequent sequences and the third one reveals the within group discrepancy. If the first and third kind of plots have already been proposed in the literature, the frequency plots of sequences is an original contribution. Such visualization tools prove to be of great help for the experts. They highly facilitate the drawing of significant interpretations regarding the structuring of events and their timing. TraMiner is still under development and should in the near future include many other features, among which descriptive indicators of individual sequences as well as of set of sequences and frequent episodes mining tools. Since TraMiner runs in R it is also quite straightforward to use

for instance the computed distances with procedures such as multidimensional scaling offered by other R packages. By offering the social scientist, and indeed any interested end user, the possibility to run all her/his analyses of sequences and produce all her/his graphics within a same free and open-source software, TraMiner should help popularize sequential analysis in the social sciences.

## Bibliography

- [1] Deonier, R., Tavaré, S., Waterman, M.: Computational Genome Analysis: an Introduction. Springer (2005)
- [2] Needleman, S.B., Wunsch, C.: General method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **Vol. 48** (1970) pp. 443–453
- [3] Kruskal, J.: An overview of sequence comparison. In: Time warps, string edits, and macromolecules. The theory and practice of sequence comparison. Don Mills, Ontario:Adison-Wesley (1983) pp. 1–44
- [4] Abbott, A., Forrest, J.: Optimal matching methods for historical sequences. *Journal of Interdisciplinary History* **Vol. 16** (1986) 471–494
- [5] Abbott, A., Hrycak, A.: Measuring resemblance in sequence data: An optimal matching analysis of musician's careers. *American Journal of Sociology* **Vol. 96**(1) (1990) 144–185
- [6] Abbott, A., Tsay, A.: Sequence analysis and optimal matching methods in sociology, Review and prospect. *Sociological Methods and Research* **Vol. 29**(1) (2000) 3–33 (With discussion, pp 34-76).
- [7] Rohwer, G., Pötter, U.: TDA user's manual. Software, Ruhr-Universität Bochum, Fakultät für Sozialwissenschaften, Bochum (2002)
- [8] Wu, L.: Some comments on "sequence analysis and optimal matching methods in sociology : Review and prospect". *Sociological Methods and Research* **Vol. 29**(1) (2000) pp. 41–64
- [9] Notredame, C., Bucher, P., Gauthier, J.A., Widmer, E.: T-COFFEE/SALTT: User guide and reference manual. disponible sur <http://www.tcoffee.org/saltd> (2005)
- [10] Gauthier, J.A., Widmer, E.D., Bucher, P., Notredame, C.: How much does it cost? Optimization of costs in sequence analysis of social science data. *Sociological Methods and Research* (2008) (forthcoming).
- [11] Scherer, S.: Early career patterns : A comparison of Great Britain and West Germany. *European Sociological Review* **Vol. 17** (2) (2001) pp. 119–144
- [12] Brzinsky-Fay, C., Kohler, U., Luniak, M.: Sequence analysis with Stata. *The Stata Journal* **Vol. 6, number 4** (2006) pp. 435–460
- [13] Lesnard, L.: Describing social rhythms with optimal matching (2007)
- [14] Elzinga, C.H.: CHESA 2.1 User manual. User guide, Dept of Social Science Research methods, Vrije Universiteit, Amsterdam (2007)
- [15] Elzinga, C.H.: Sequence similarity : A nonaligning technique. *Sociological Methods & Research* **Vol. 32, No. 1** (2003) pp. 3–29
- [16] Elzinga, C.H.: Combinatorial representations of token sequences. *Journal of Classification* **Vol. 22** (2005) pp. 87–118