cfda: an R Package for Categorical Functional Data Analysis

Cristian Preda, Quentin Grimonprez, Vincent Vandewalle

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

cfda: an R Package for Categorical Functional Data Analysis

Cristian Preda
Université de Lille

Quentin Grimonprez
Inria Lille

Vincent Vandewalle
Université de Lille

Abstract

Categorical functional data represented by paths of a stochastic jump process with continuous time and finite set of states are considered. As an extension of the multiple correspondence analysis to an infinite set of variables, optimal encodings of states over time are approximated using an arbitrary finite basis of functions. That allows dimension reduction, optimal representation and visualisation of data in lower dimensional spaces. The methodology is implemented in the **cfda** R package and is illustrated using a real data set in the clustering framework.

Keywords: functional data, categorical data, stochastic process, multiple correspondence analysis.

1. Introduction

Most literature devoted to functional data considers data as sample paths of a real-valued stochastic process, $X = \{X_t, t \in \mathcal{T}\}$, $X_t \in \mathbb{R}^p$, $p \geq 1$ where $\mathcal{T}$ is some continuous set. Among a considerable record of papers on the subject, the monographs of Ramsay and Silverman (2005) and Ferraty and Vieu (2006) are still the main references presenting methodologies for visualisation, denoising, clustering and regression when dealing with functional data represented by real-valued functions. The **fda** (Ramsay, Wickham, Graves, and Hooker 2018) R (R Core Team 2020) package implements these methodologies and tools for dealing with such functional data.

In this paper we consider the case where the underlying stochastic model generating the data is a continuous-time stochastic process $X = \{X_t, t \in \mathcal{T}\}$ such that for all $t \in \mathcal{T}$, X_t is a categorical random variable rather than a real-valued one.

Let $(\Omega, \mathcal{A}, \mathsf{P})$ be a probability space, $\mathcal{S} = \{s_1, \ldots, s_K\}$, $K \geq 2$, be a set of K states and

$$X = \{X_t \ ; \ X_t : \Omega \longrightarrow \mathcal{S}, \quad t \in \mathcal{T}\} \tag{1}$$

be a family of categorical random variables indexed by $\mathcal{T}$. Thus, for some $\omega \in \Omega$, a path of X, $X(\omega)$, is a sequence of states $s_{i_j} = s_{i_j}(\omega)$ and time points $t_i = t_i(\omega)$ of transitions from one state to another one:

$$\{(t_0, s_{i_0}), (t_1, s_{i_1}), (t_2, s_{i_2}), \ldots\} \tag{2}$$

where $0 = t_0 < t_1 < t_2 < \ldots$ are the jump times in $\mathcal{T}$ and $s_{i_j} \in \mathcal{S}$ with $i_j \in \{1, \ldots, K\}$, $\forall j \geq 0$. This path is read as follows. At time $t_0 = 0$, ω is in some state s_{i_0} ; at time t_1, $t_1 > t_0$, ω moves randomly from s_{i_0} to the state s_{i_1}; then at time $t_2 > t_1$ it moves from the state s_{i_1} to state s_{i_2} and so on. If $\mathcal{T}$ is the interval of time $[0, T]$ for some $T > 0$, then the observation process stops when the time T is reached or some absorbing state is observed.

We call the sample paths of X given by sequences of type (2) *categorical functional data* generated by the process X.

Figure 1 presents the graphical representation of one observation of categorical functional random variable. The representation (a) is appropriate when no natural order relationship exists on the set of states $\mathcal{S}$, whereas the representation in (b) supposes that there exists some order relationship $(\prec)$ on $\mathcal{S}$: $s_1 \prec s_2 \prec \ldots \prec s_K$.

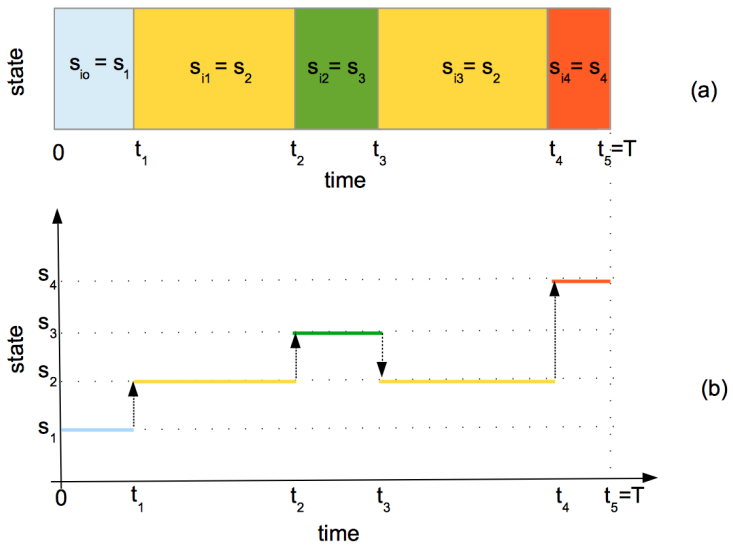


Figure 1: Categorical functional data: graphical representation.

To the best of our knowledge, and quite surprisingly, there are no recent researches devoted to this type of functional data despite its ability to model real situations in different fields of applications: health and medicine (status of a patient over time), economy (status of the market), sociology (evolution of social status), and so on. As a start point of research on this topic we mention the works of Boumaza (1980), Deville (1982), Deville and Saporta (1983) and Saporta (1981). These works are devoted to the extension of factorial techniques (canonical and multiple correspondence analysis) towards functional data. Applications of these techniques are presented in Heijden, Teunissen, and van Orlé (1997) for analysing career

data and in Preda (1998) for studying spectral properties of the transition probability matrix of a stationary Markovian jump process with continuous time. In Cardot, Lecuelle, Schlich, and Visalli (2019) the authors are interested to cluster paths of semi-Markov processes using mixtures with application to sensory data.

In this article we present categorical functional data analysis as an extension of the multiple correspondence analysis towards functional data and its implementation in the **cfda** R package. The theoretical foundations of this work are given in Deville (1982) and are based on the concept of *optimal encoding* of the states of the process X with respect to maximum variance criterion among all encodings. In Section 2 we present the theoretical background of the optimal encoding methodology defining the *principal components* of the process X throughout the optimal encodings. The approximation of the optimal encodings of the states into a basis of functions and optimal representation of categorical functional data in lower dimensional spaces are detailed. The implementation of the optimal encodings is presented throughout the **cfda** R package in Section 3 where an application on real data set (care trajectories for patients diagnosed with severe infection) is performed in view of visualisation, descriptive statistics and clustering.

2. Categorical Functional Data Analysis

Introduced in Saporta (1981) and Deville (1982) under the name "analyse harmonique qualitative", multiple correspondence analysis is extended to categorical functional data. There are several ways to do this, we have chosen in this work to introduce it as a problem of finding the latent variables (principal components) that are the most related to the process $X = \{X_t, t \in \mathcal{T}\}$. Therefore, the principal components will allow to define optimal encoding of states $\mathcal{S} = \{s_1, \ldots, s_K\}$.

Without loss of generality, let suppose that $\mathcal{T} = [0, T]$, with $T > 0$. For $x, y \in \mathcal{S}$ and $\forall t \in [0, T]$, let denote by:

- $\mathbf{1}_t^x = \begin{cases} 1 & \text{if } X_t = x, \\ 0 & \text{otherwise,} \end{cases}$
- $p^x(t) = \mathsf{P}(X_t = x)$ and $p^{x,y}(t, s) = \mathsf{P}(X_t = x, X_s = y)$.

The general hypotheses considered in that framework are:

H_1: the process X is continuous in probability,

$$\lim_{h \to 0} \mathsf{P}(X_{t+h} \neq X_t) = 0$$

and

H_2: for each time $t \in [0, T]$ (except possibly a finite discrete set of time points), any state has a strictly positive probability to occur:

$$p^x(t) \neq 0, \forall x \in \mathcal{S}, \forall t \in [0, T].$$

2.1. The Principal Components

Let $L_2(\Omega)$ be the space of real random variables with finite second moment and, for some

$t \in [0,T]$, $L(X_t)$ be the linear space spanned by X_t. Then, the conditional expectation operator associated to X_t,

$$\mathsf{E}_t : L_2(\Omega) \to L(X_t),$$

$$z \in L_2(\Omega), \quad z \longmapsto \mathsf{E}_t(z) = \sum_{x \in \mathcal{S}} \mathsf{E}(z|X_t = x)\mathbf{1}_t^x,$$

is also the orthogonal projector on the space linearly spanned by the set of indicator random variables $\{\mathbf{1}_t^x, x \in \mathcal{S}\}$. Notice that E_t is self-adjoint, idempotent and of rank K.

For $z \in L_2(\Omega)$ and $t \in [0,T]$, the coefficient

$$\eta^2(z; X_t) = \frac{\mathsf{VAR}(\mathsf{E}_t(z))}{\mathsf{VAR}(z)}$$

is a measure of the correlation between z and the variable X_t. The empirical version of η^2 is known as Wilks' Lambda statistics, well known in multivariate ANOVA (Nath and Pavur 1985).

Let us recall that if $t_1, t_2, \ldots, t_p$ are p different time points in $[0,T]$, then the random variable z which maximizes

$$\sum_{i=1}^{p} \eta^2(z; X_{t_i}) \tag{3}$$

defines the first principal component of the multiple correspondence analysis of the set of p categorical variables $\{X_{t_1}, X_{t_2}, \ldots X_{t_p}\}$ (Escofier 1978). By an iterative procedure, the principal components of higher order are defined as maximizing (3) under orthogonality conditions with respect to the principal components of lower order.

In Saporta (1981) and Deville (1982) the authors extend the multiple correspondence analysis to the process $X = \{X_t, t \in [0,T]\}$ (seen as infinite set of categorical random variables). More specifically, the principal components are defined as the random variable z that maximizes the criterion

$$\int_0^T \eta^2(z; X_t)dt. \tag{4}$$

They show that under the hypotheses H_1 and H_2, the variable z which maximizes (4) is the variable associated to the largest eigenvalue of the following (stochastic) eigenvalue problem:

$$\int_0^T \mathsf{E}_t(z)dt = \lambda z. \tag{5}$$

The operator $Q = \displaystyle\int_0^T \mathsf{E}_t dt$ is positive, hermitian, and compact. Therefore, Q has a countable set of positive eigenvalues and eigenvectors, $\{(\lambda_i, z_i)\}_{i \geq 1}$ such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ and

$$\int_0^T \mathsf{E}_t(z_i)dt = \lambda_i z_i.$$

The variables $\{z_i\}_{i \geq 1}$ are called *principal components* of the process X. Notice that $z = 1$ (constant) is an eigenvector of Q associated to the largest eigenvalue $\lambda_{\max} = T$. It follows that the principal components $\{z_i\}_{i \geq 1}$ form a set of zero-mean uncorrelated random variables.

Moreover, we have that

$$\sum_{i\geq 1} \lambda_i = KT,$$

where K is the number of states. Thus, excluding the trivial eigenvalue $\lambda_{\max} = T$, the contribution of the i-th principal component z_i to (4) is

$$\mathrm{Ctr}(z_i) = \frac{\lambda_i}{(K-1)T}.$$

2.2. Optimal Encoding Functions

In order to solve (5), let denote by

$$\xi_t = \frac{1}{\lambda}\mathsf{E}_t(z), \quad \forall t \in [0,T]. \tag{6}$$

Under the hypotheses H_1 and H_2, for each $t \in [0,T]$ ξ_t is X_t-measurable i.e. $\mathsf{E}_t\xi_t = \xi_t$ and $\{\xi_t\}_{t\in[0,T]}$ is a L_2-continuous stochastic process.
From (5) it follows that

$$z = \int_0^T \xi_t dt. \tag{7}$$

Taking the conditional expectation with respect to X_t in (5), one obtains that the stochastic process $\{\xi_t\}_{t\in[0,T]}$ is eigenvector of the following (stochastic) eigenvalue problem posed in the space of L_2-continuous stochastic processes:

$$\int_0^T \mathcal{K}(t,s)\xi_s ds = \lambda\xi_t, \ \forall t \in [0,T], \tag{8}$$

with $\mathcal{K}(t,s) = \mathsf{E}_t\mathsf{E}_s$, for all $t,s \in [0,T]$. Recall that the spectral analysis of the kernel $\mathcal{K}(t,s)$ yields to the canonical analysis of X_t and X_s (Hotelling 1936).

It can be shown (Saporta (1981)) that the eigenvalue problems (5) and (8) are equivalent in that sense that they have the same set of eigenvalues $\{\lambda_i\}_{i\geq 1}$ and there is an one-to-one correspondence between the principal components z_i and the process $\xi_i = \{\xi_{i,t}, t \in [0,T]\}$, $\forall i \geq 1$. This correspondence is given by (6).
As in (5), the solution of (8) is unique up to a constant. To have unique eigenvectors, the usual constraint on $\{\xi_t\}_{t\in[0,T]}$ is that of total variance equals to one,

$$\int_0^T \mathsf{VAR}(\xi_t)dt = \int_0^T \mathsf{E}(\xi_t^2)dt = 1. \tag{9}$$

The relation (9) implies

$$\mathsf{VAR}(z) = \mathsf{E}(z^2) = \lambda. \tag{10}$$

It follows from (6) that ξ_t is X_t-measurable for all $t \in [0,T]$ and one can write

$$\xi_t = \sum_{x\in\mathcal{S}} a^x(t)\mathbf{1}_t^x, \tag{11}$$

where $\{a^x\}_{x\in\mathcal{S}}$ are deterministic functions on $[0,T]$ that we call *optimal encoding* functions. Introducing (11) in (8) one obtains the following eigenvalue equation,

$$\int_0^T \sum_{y\in\mathcal{S}} p^{x,y}(t,s)a^y(s)ds = \lambda a^x(t)p^x(t), \quad \forall t\in[0,T], \forall x\in\mathcal{S}, \tag{12}$$

where, $p^x(t) = \mathsf{P}(X_t = x)$ and $p^{x,y}(t,s) = \mathsf{P}(X_t = x, X_s = y)$.

The integral system (12) is a more "classic" one than (5) and (8). Under the hypothesis H_1 and H_2 it admits the sequence of eigenvalues $\{\lambda_i\}_{i\geq 1}$ associated to the optimal encoding eigen-functions $\{a_i^x, x\in\mathcal{S}\}_{i\geq 1}$.

Notice that the constraint conditions in (9) are expressed now in terms of optimal encoding functions,

$$\int_0^T \sum_{x\in\mathcal{S}} \left[a^x(t)\right]^2 p^x(t)dt = 1. \tag{13}$$

According to (7), for $i\geq 1$, the i-th principal component z_i is derived from the i-th optimal encoding functions $\{a_i^x\}$ as

$$z_i = \int_0^T \sum_{x\in\mathcal{S}} a_i^x(t)\mathbf{1}_t^x dt, \quad \forall i\geq 1. \tag{14}$$

2.3. Expansion Formulas and Dimension Reduction

As a summary of the previous section, the three equivalent eigen-problems stated in (5), (8) and (12) yield to the following elements of the analysis of X:

- the set of principal components $\{z_i\}_{i\geq 1}$ are zero-mean and uncorrelated:
  - $\mathsf{E}(z_i) = 0, \quad \forall i\geq 1.$
  - $\mathsf{COV}(z_i, z_j) = \begin{cases} \lambda_i & \text{if } i=j, \\ 0 & \text{otherwise.} \end{cases}$
- the set of eigen-processes $\{\xi_i = \{\xi_{i,t}, t\in[0,T]\}\}_{i\geq 1}$ which generates the principal components by (7),
$$z_i = \int_0^T \xi_{i,t}dt, \quad i\geq 1,$$
are zero-mean and of unit total variance.
- the optimal encoding functions, $\{a_i^x = \{a_i^x(t), t\in[0,T]\}\}_{x\in\mathcal{S},i\geq 1}$. They generate the eigen-processes ξ_i by (11),
$$\xi_{i,t} = \sum_{x\in\mathcal{S}} a_i^x(t)\mathbf{1}_t^x, \quad \forall t\in[0,T].$$
They satisfy the normalization condition (13).

Expansion Formulas. As an analogy to the Karhunen-Loève expansion for the scalar processes (Deville 1974), the following expansion formulas hold (Saporta 1981):

- for the process $X = \{X_t, t \in [0,T]\}$ throughout the indicators $\mathbf{1}^x = \{\mathbf{1}_t^x, t \in [0,T]\}$:

$$\mathbf{1}_t^x = \sum_{i \geq 1} z_i a_i^x(t) \frac{1}{p^x(t)}, \quad \forall x \in \mathcal{S}. \tag{15}$$

- for the bivariate joint probability function, $p^{x,y} = \{p^{x,y}(t,s), t, s \in [0,T]\}$: applying the Mercer theorem (Mercer 1909) to the integral equation (12), one has the following expansion formula:

$$p^{x,y}(t,s) = p^x(t)p^y(s) \sum_{i \geq 1} \lambda_i a_i^x(t) a_i^y(s), \ \forall t, s \in [0,T], \forall x, y \in \mathcal{S}. \tag{16}$$

  In particular, for $x = y$ and $s = t$ we obtain

$$p^x(t) = \left\{ \sum_{i \geq 1} \lambda_i \left[a_i^x(t)\right]^2 \right\}^{-1}, \ \forall t \in [0,T], \forall x \in \mathcal{S}. \tag{17}$$

Dimension Reduction. Using only the q first terms in the right-side part of (15) , $q \geq 1$, one obtains the best approximation of order q of X (viewed as a vector process $X = \{\mathbf{1}^x, x \in \mathcal{S}\}$) under the L_2 norm, among all the linear expansions of type

$$\mathbf{1}_t^x \approx \sum_{i=1}^{q} z_i a_i^x(t) \frac{1}{p^x(t)}, \quad \forall x \in \mathcal{S}.$$

Thus, the q first principal components,

$$\{z_1, \ldots, z_q\}, \quad q \geq 1,$$

allow for

- graphical representation of sample paths of X in $\mathbb{R}^q$ (especially for $q = 2$, one obtains a 2-D representation of categorical functional data),
- fit of clustering and regression models with X as explanatory variables.

2.4. Approximation of Optimal Encoding Functions: a Basis Expansion Approach

The eigenvalue equation in (12) provides the optimal encoding functions. For a two-state process, Saporta (1981) considers the birth-and-death process on $[0,1]$,

$$X_t = \begin{cases} 0, & \text{if } t < \theta, \\ 1, & \text{if } t \geq \theta, \end{cases} \tag{18}$$

where θ is a random variable uniformly distributed on $[0,1]$. The authors provide in this case explicit formulas for the eigenvalues $\{\lambda_i\}_{i \geq 1}$, the optimal encoding functions $\{a_i^x\}_{i \geq 1}$, $x \in \mathcal{S}$,

and the principal components $\{z_i\}_{i\geq 1}$. In Preda (1998), the author considers the case of stationary Markovian continuous time processes with reversible distribution. In this case, the system in (12) reduces to a system of linear second-order differential equations with constant coefficients.

In general, the solution of (12) is obtained by approximation. In his seminal work, Deville (1982) proposes to approximate the encoding functions $\{a_i^x\}_{i\geq 1}$, $x \in \mathcal{S}$, into a basis of functions of dimension m, $m \geq 1$. As in the classical framework of functional data (Ramsay and Silverman (2005)), the choice of m is a trade-off between complexity computation and precision of the approximation. In our simulation study (section 4) we show the influence of the choice of m on the approximation of optimal encodings.

Let $\{\phi_1, \ldots, \phi_m\}$, $\phi_i : [0, T] \to \mathbb{R}$, $i = 1, \ldots, m$, be a basis of functions (Fourier, B-splines, monomial, etc.) and for each $x \in \mathcal{S}$ consider the approximation:

$$a^x(t) \approx \alpha_{(x,1)}\phi_1(t) + \alpha_{(x,2)}\phi_2(t) + \ldots + \alpha_{(x,m)}\phi_m(t), \quad \forall t \in [0, T], \tag{19}$$

where $\alpha_x = \left(\alpha_{(x,1)}, \alpha_{(x,2)}, \ldots, \alpha_{(x,m)}\right)' \in \mathbb{R}^m$ is the column vector of the expansion coefficients of a^x into the basis $\{\phi_1, \ldots, \phi_m\}$.

Plugging (19) in (12) and (13) one obtains the following classical eigen-problem:

$$G\alpha = \lambda F\alpha, \tag{20}$$

under the constraint

$$\alpha' F\alpha = 1, \tag{21}$$

where $\alpha \in \mathbb{R}^{m\times K}$ is the column vector obtained by the concatenation of the vectors $\{\alpha_x\}_{x\in\mathcal{S}}$ and G and F are square matrices of size $mK \times mK$ defined as follows:

- The matrix G is the covariance matrix of the random variables $\{V_{(x,i)}, x \in \mathcal{S}, i \in 1, \ldots m\}$, defined as

$$V_{(x,i)} = \int_0^T \phi_i(t)\mathbf{1}_t^x dt, \quad \forall x \in \mathcal{S}, \tag{22}$$

$$G = \left\{G_{(x,i),(y,j)} = \mathsf{COV}\left(V_{(x,i)}, V_{(y,j)}\right), \quad x, y \in \mathcal{S}, i, j = 1, \ldots, m\right\}, \tag{23}$$

$$G = \begin{array}{c} \\ \\ \\ (x,1) \\ \\ (x,i) \\ \\ (x,m) \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} \ldots & (y,1) & \ldots & (y,j) & \ldots & (y,m) & \ldots \end{array} \\ \begin{pmatrix} \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdot & \cdots & \mathsf{COV}\left(V_{(x,i)}, V_{(y,j)}\right) & \cdots & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \end{array}.$$

- The matrix F is defined by

$$F = \left\{ F_{(x,i),(y,j)} = \mathsf{E}\left(U_{(x,i),(y,j)}\right), \quad x, y \in \mathcal{S}, i, j = 1, \ldots, m \right\}, \tag{24}$$

where $U_{(x,i),(y,j)}$ is the random variable

$$U_{(x,i),(y,j)} = \int_0^T \phi_i(t)\phi_j(t)\mathbf{1}_t^x \mathbf{1}_t^y dt = \begin{cases} \displaystyle\int_0^T \phi_i(t)\phi_j(t)\mathbf{1}_t^x dt & \text{if } \ x = y, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

Thus, F is a block diagonal matrix, each block being a square matrix of size $m \times m$ corresponding to each x in $\mathcal{S}$, $\{U_{(x,i),(x,j)}, i, j = 1, \ldots, m\}$.

Example. *Let observe that if* $0 = t_0 < t_1 < \ldots < t_m = T$ *is a sequence of time points in* $[0, T]$ *and for* $i = 1, \ldots, m$*, one defines the basis (B-splines of order 1),*

$$\phi_i(t) = \begin{cases} 1 & \textit{if } \ t \in [t_{i-1}, t_i[, \\ \\ 0 & \textit{otherwise} \end{cases}$$

then the random variable $V_{(x,i)}$ *represents the time spent in the state* x *in the interval* $[t_{i-1}, t_i[$*. Since* $V_{(x,i)} = U_{(x,i),(x,i)}$*, then* F *is a diagonal matrix with elements* $\mathsf{E}(V_{(x,i)})$*,* $x \in \mathcal{S}$*,* $i = 1, \ldots, m$*.*

2.5. Estimation

Notice that the random variables $V_{(x,i)}$ and $U_{(x,i),(y,j)}$, $x, y \in \mathcal{S}$, $i, j = 1, \ldots, m$, are computed from $X = \{X_t, t \in [0, T]\}$ throughout the basis of functions $\{\phi_1, \ldots, \phi_m\}$.

Thus, if $\{X_1, \ldots, X_n\}$ is a sample of n paths of X corresponding to a random sample $(\omega_1, \ldots, \omega_n) \in \Omega^n$, then the corresponding samples $V_{(x,i)}(\omega)$ and $U_{(x,i),(x,j)}(\omega)$, $\omega \in \{\omega_1, \ldots, \omega_n\}$, provide two classical data sets, V and U, as:

- the V data set with n rows and Km columns for the V's random variables,

$$\mathrm{V} = \begin{array}{c|cccccccc} \omega & V_{(s_1,1)} & \cdots & V_{(s_1,m)} & \cdots & V_{(x,i)} & \cdots & V_{(s_K,m)} \\ \hline \omega_1 & V_{(s_1,1)}(\omega_1) & \cdots & V_{(s_1,m)}(\omega_1) & \cdots & V_{(x,i)}(\omega_1) & \cdots & V_{(s_K,m)}(\omega_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \omega_n & V_{(s_1,1)}(\omega_n) & \cdots & V_{(s_1,m)}(\omega_n) & \cdots & V_{(x,i)}(\omega_n) & \cdots & V_{(s_K,m)}(\omega_n) \end{array}$$

- and the U dataset with n rows and Km^2 columns for the U's random variables, respectively:

$$\mathrm{U} = \begin{array}{c|ccccccc} \omega & \cdots & U_{(x,i),(x,1)} & \cdots & U_{(x,i),(x,m)} & \cdots & U_{(s_K,m),(s_K,m)} \\ \hline \omega_1 & \cdots & U_{(x,i),(x,1)}(\omega_1) & \cdots & U_{(x,i),(x,m)}(\omega_1) & \cdots & U_{(s_K,m),(s_K,m)}(\omega_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \omega_n & \cdots & U_{(x,i),(x,1)}(\omega_n) & \cdots & U_{(x,i),(x,m)}(\omega_n) & \cdots & U_{(s_K,m),(s_K,m)}(\omega_n) \end{array}$$

Therefore, the matrices G and F are estimated from the sample $\{X_1, \ldots, X_n\}$ by the matrices $\widehat{G}$ and $\widehat{F}$, the covariance matrix estimator of the random variables V's and the mean estimator of the random variables U's. For each i and j in $\{1, \ldots, m\}$ and x and y in $\mathcal{S}$ one has:

$$\widehat{G}_{(x,i),(y,j)} = \widehat{\mathsf{COV}}\left(V_{(x,i)}, V_{(y,j)}\right) = \frac{1}{n-1}\left(\sum_{h=1}^{n} V_{(x,i)}(\omega_h)V_{(y,j)}(\omega_h) - n\bar{V}_{(x,i)}\bar{V}_{(y,j)}\right)$$

and

$$\widehat{F}_{(x,i),(y,j)} = \begin{cases} \bar{U}_{(x,i),(y,j)} = \dfrac{1}{n}\displaystyle\sum_{h=1}^{n} U_{(x,i),(y,j)}(\omega_h) & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

An estimate of i-th eigen vector of (20), $\alpha^{(i)}$, $i \geq 1$, is the i-th eigenvector $\widehat{\alpha}_i$ of the eigen-equation (26),

$$\widehat{G}\widehat{\alpha} = \widehat{\lambda}\widehat{F}\widehat{\alpha}, \tag{26}$$

under the constraint

$$\widehat{\alpha}'\widehat{F}\widehat{\alpha} = 1. \tag{27}$$

Then, for each $x \in \mathcal{S}$, the i-th encoding eigen-function a_i^x is estimated by

$$\widehat{a}_i^x = \sum_{j=1}^{m} \widehat{\alpha}_{i,(x,j)}\phi_j, \quad i \geq 1. \tag{28}$$

The estimates for the encoding functions allow to compute the principal components z_i for each unit ω in the sample $(\omega_1, \ldots, \omega_n)$, as

$$\widehat{z}_i(\omega) = \int_0^T \sum_{x\in\mathcal{S}} \widehat{a}_i^x(t)\mathbf{1}_t^x(\omega)dt = \sum_{x\in\mathcal{S}}\sum_{j=1}^{m} \widehat{\alpha}_{i,(x,j)}V_{(x,j)}(\omega), \quad i \geq 1. \tag{29}$$

Notice that the variance of $\widehat{z}_i$ equals the i-th eigenvalue $\widehat{\lambda}_i$ of (26),

$$\widehat{\mathsf{VAR}(\widehat{z}_i)} = \widehat{\lambda}_i, \quad i \geq 1. \tag{30}$$

Confidence bounds. Bootstrapping from the V and U datasets, throughout (26) one obtains an estimate of the covariance matrix of $\widehat{\alpha}_i$ denoted with $\widehat{\Sigma}_i$. Therefore, for each $t \in [0, T]$, we have

$$\widehat{\mathsf{VAR}(a_i^x(t))} = \phi(t)'\widehat{\Sigma}_{(i,x)}\phi(t),$$

where $\phi(t)$ is the column vector $\phi(t) = (\phi_1(t), \ldots, \phi_m(t))'$ and $\widehat{\Sigma}_{(i,x)}$ is the covariance matrix of $\widehat{\alpha}_{i,x} = \left(\widehat{\alpha}_{i,(x,1)}, \ldots, \widehat{\alpha}_{i,(x,m)}\right)$. Notice that $\widehat{\Sigma}_{(i,x)}$ is a submatrix of $\widehat{\Sigma}_i$.
Then, for a confidence level $1-u$, $u \in [0, 1]$, a confidence interval for $a_i^x(t)$ is obtained as

$$\mathtt{CI}^{1-u}(a_i^x(t)) = \widehat{a_i^x(t)} \pm \zeta_{1-\frac{u}{2}}\sqrt{\widehat{\mathsf{VAR}(a_i^x(t))}},$$

where $\zeta_{1-\frac{u}{2}}$ is the quantile of order $1-\frac{u}{2}$ of the standard normal distribution.

3. The cfda Package through Examples

The **cfda** R package provides functions to analyze categorical functional data allowing to compute basic statistics such as transition tables or visualisation, and compute the optimal encodings. It uses **ggplot2** (Wickham 2016) package to display graphics and the **snow** (Tierney, Rossini, Li, and Sevcikova 2018) and **foreach** (Microsoft and Weston 2019) packages for code parallelization.

Other packages for analyzing sequences of categorical data exist, but not in a functional way. The **TraMineR** (Gabadinho, Studer, Müller, Bürgin, Fonta, and Ritschard 2019) R package provides functions to perform descriptive analysis, and distance functions between sequences to perform clustering analysis. The **WeightedCluster** (Studer 2013) package relies on **TraMineR** and implements a clustering method and some cluster quality statistics. The **msm** (Jackson 2011) package estimates a continuous-time (hidden) Markov multi-state model. The R packages **ClickClust** (Melnykov 2016) and **clickstream** (Scholz 2016) model these sequences as discrete Markov chains to cluster them with mixture models or k-means.

3.1. Data

Real Dataset

The **cfda** package is illustrated with the `care` dataset. It contains 2929 care trajectories for patients diagnosed with a severe infection. Each month from the diagnosis of the infection, the follow-up of each patient is filled in using one of the following 4 states: `"D"`, the patient has not a medical follow-up, `"C"`, the patient has a medical follow-up but no treatment, `"T"`, the patient has a medical follow-up with a treatment but the infection is not suppressed and `"S"`, the patient has a medical follow-up with a treatment and the infection is suppressed.

The dataset can be loaded running:

```
R> data(care)
R> head(care, 10)
```

```
id time state
 3    0     D
 3    5     D
 9    0     D
 9    1     D
13    0     D
13    7     D
15    0     D
15    4     T
15    7     C
15    8     D
```

The dataset is in a specific format required by every function in this package. Data must be provided as a data.frame with 3 columns named `id`, `time` and `state`. the `id` column contains the identifiers of the different individuals, the `time` column the different time of records of state changes and the `state` column containing the state that occurs at the corresponding

time. For example, in the dataset above, four individuals (3, 9, 13 and 15) are visible. The individual 15 has an initial state (`D`) at time $t = 0$, it stays in this state until $t = 4$ months, at which he moves to a new state (`T`), and so on.

Note that within each individual, the time values must be ordered. Concerning `id` and `state`, the used format is quite versatile: character, factor or integer can be used.

Generate a Dataset

There are two functions for generating datasets: `generate_Markov` and `generate_2State`. The `generate_Markov` function generates individuals following a Markov process where different time values are defined by an exponential law. The function is defined as below:

```
generate_Markov(n, K, P, lambda, pi0, Tmax, labels)
```

with `n` the number of individuals, `K` the number of state, `P` the transition matrix of size `K*K` between states containing the probabilities of changing from one state to another, `lambda` a vector of length `K` containing the parameters of the exponential law associated with each state, `pi0` a vector of length `K` containing the probabilities of being in the different states at time $t = 0$, `Tmax` the maximal time of each individual, `labels` a vector containing the state names.

The `generate_2State` function generates individuals starting at time $t = 0$ with state 0 and with with a unique change into the state 1 at a time $t \sim \mathcal{U}(0, 1)$. The only argument of `generate_2State` is `n` the number of individuals.

Visualize a Dataset

The `summary_cfd` function gives an overview of the dataset by printing information such as the number of individuals, the time range or the number of states, etc. All the printed information are returned in a list.

```
R> summary_cfd(care)
```

```
Number of rows: 10017
Number of individuals: 2929
Time Range: 0 - 50
Same time start value for all ids: TRUE
Same time end value for all ids: FALSE
Number of states: 4
States:
   D, T, C, S
Number of individuals visiting each state:
   D    C    T    S
2905 1154 1014 1063
```

One can note that all individuals have the same start time value but not the same end time value. This does not meet the constraints for performing a functional data analysis. A subsample of the dataset must be extracted before performing the encoding computation.

A sample of the individuals from the `care` dataset is plotted using the `plotData` function (see Figure 2). Each line corresponds to an individual of the dataset, the successive changes of states are represented by different colors.

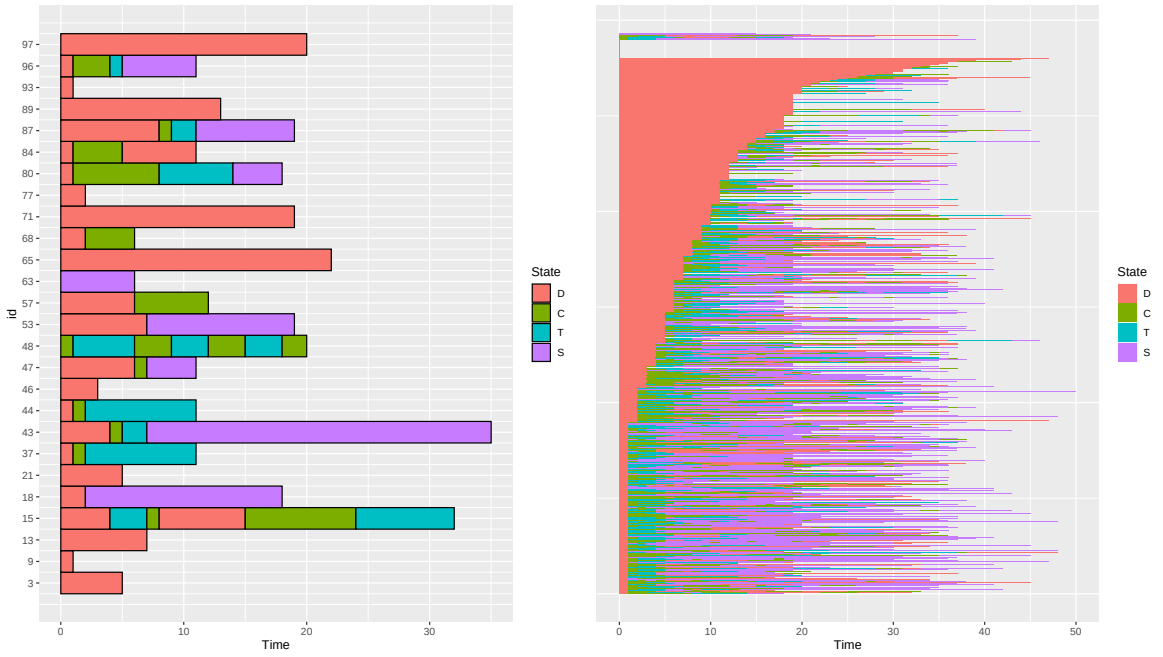


Figure 2: A sample (left) and all individuals (right) from the `care` dataset plotted using the `plotData` function.

```
plotData(data, col, addId, addBorder, sort)
```

The `plotData` function takes in argument a formatted data.frame (`data`) and additional aesthetic parameters:

`group` a vector of length the number of individuals of `data` containing a variable describing the group of each individual. Individuals from different groups are displayed on different subplots. Individuals whose group is `NA` are ignored.

`addId` a boolean to add the individuals id on the y-axis.

`addBorder` a boolean to add the black border around each individual.

`sort` a boolean to sort individuals according to the duration in their first state.

`col` allows users to customize state colors by providing a vector of the same length as the number of state. `col` is a character (named) vector containing defined color names from R (e.g. `c("red", "blue", "darkgreen")`) or RGB colors (e.g. `c("#E41A1C", "#377EB8", "#4DAF4A")`).

`nCol` only when `group` is used, the number of columns used to display the different groups.

```
R> plotData(care[care$id <= 100, ])
R> plotData(care, addBorder = FALSE, addId = FALSE, sort = TRUE)
```

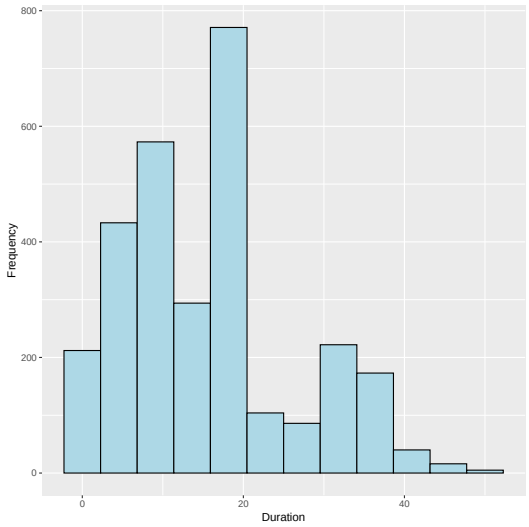


Figure 3: Distribution of the duration of trajectories.

Extract a Dataset Meeting the Constraints

To compute the encodings, each individual must have the same start and end time. This is not the case in the `care` dataset. So, we select patients with a follow-up of at least 18 months and works from $t = 0$ to $t = 18$ months.

First, we compute the duration of each individual using the `compute_duration` function. It computes the last time value minus the first time value for every individual. It returns a named vector, with the id as names, containing the duration. The results can be plotted using the `hist` function with the output of `compute_duration` as argument; it returns a `ggplot` object that can be modified.

```
R> duration <- compute_duration(care)
R> head(duration)

 3  9 13 15 18 21
 5  1  7 32 18  5

R> hist(duration)
```

The resulting plot is displayed in Figure 3. Most trajectories last less than 40 months. To restrict individuals to a maximal time value of 18 months, we use the `cut_data` function that has two parameters: `data` and `Tmax`, the maximal time value. After applying this function, all individuals have `Tmax` as ending time.

```
R> idToKeep <- names(duration[duration >= 18])
R> care2 <- cut_data(care[care$id %in% idToKeep, ], 18)
R> head(care2)

   id time state
1  15    0     D
```

```
2  15    4     T
3  15    7     C
4  15    8     D
5  15   15     C
6  15   18     C
7  18    0     D
8  18    2     S
9  18   18     S
10 43    0     D
```

3.2. Basic Statistics for Categorical Functional Data

Time Spent in each State

An interesting statistic is the time spent in each state per individual that can be computed with `compute_time_spent` function. It returns a matrix with `n` rows (number of individuals) and `K` columns (number of states) with the computed time. A `plot` function is provided to plot the distribution of each state. Figure 4 displays the graphic for the `care` dataset. We note that people tends to stay longer without medical follow-up (`D`) than in the other states.

```
R> timeSpent <- compute_time_spent(care2)
R> head(timeSpent)
```

```
    D C  T  S
15 11 4  3  0
18  2 0  0 16
43  4 1  2 11
48  0 7 11  0
53  7 0  0 11
65 18 0  0  0
```

```
R> plot(timeSpent)
```

Number of Jumps

The `compute_number_jumps` function counts the number of changes in each individual's state. It has two arguments: `data`, the dataset in the right format and `countDuplicated`, a binary value indicating if jumps in the same state must be ignored (`FALSE`) or not, the default is `FALSE`. An `hist` function is provided to plot the distribution of the number of jumps. For the `care` dataset, the number of jumps varies between 0 and 8 (cf. Figure 5), with most patients with less than 6 jumps.

```
R> nJump <- compute_number_jumps(care2, countDuplicated = FALSE)
R> head(nJump)
```

```
15 18 43 48 53 65
 4  1  3  6  1  0
```

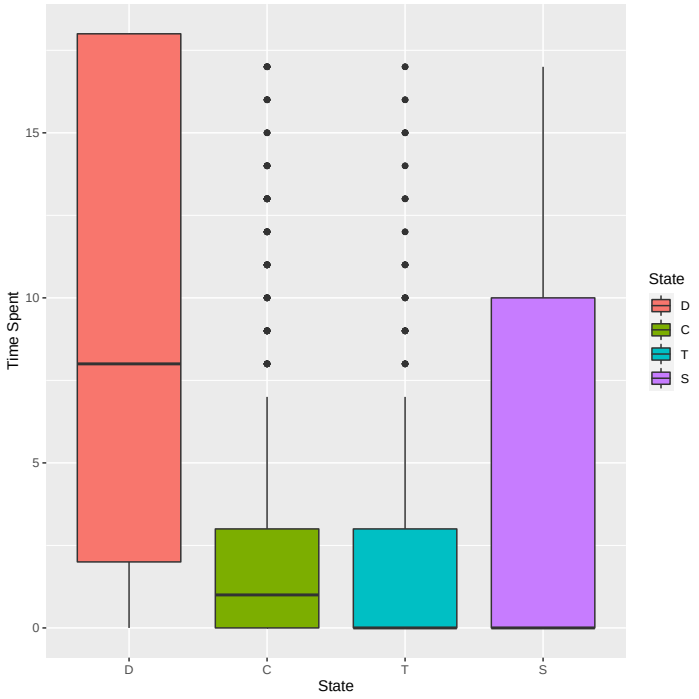


Figure 4: Distribution of time spent per state.

```
R> hist(nJump)
```

The states between which these jumps occur are visible using the `statetable` function that counts the number of transitions between each pair of states. Transitions between identical states can be removed from the output table using `removeDiagonal = TRUE`.

```
R> statetable(care2, removeDiagonal = TRUE)
```

```
    to
from   D   C   T   S
   D   0 697 253 146
   C 271   0 346  97
   T  16  74   0 461
   S  16  91  31   0
```

We note that the state `S` (infection suppressed) is not an absorbing state indicating some patients have relapsed.

Probability to be in a State

The last interesting statistic is the probability to be in a state at a given time which is computed using the `estimate_pt` function. It has two arguments: `data`, the dataset in the right format and `NAafterTmax`. If `NAafterTmax = FALSE`, it considers that the last entry of an individual corresponds to its last change of state, i.e. the individual stays in the last

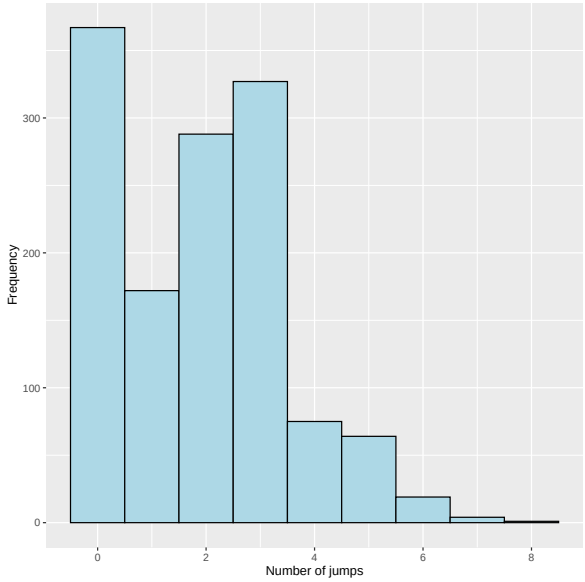


Figure 5: Distribution of number of jumps per individual.

recorded state for any time greater than the last time entry; if `TRUE`, it considers that for any time greater than the last time entry, the records are missing; the default is `FALSE`. This is an important parameter when individuals have different ending time. This function returns a list of two elements: `t`, a vector containing the time values, `pt`, a matrix (with the states in rows and the time values in columns) containing the computed probabilities. A `plot` function is provided to display the results, the first parameter is the output of `estimate_pt` function, the second is `ribbon`. If `ribbon = FALSE`, the probability for a state is displayed with a line, if `TRUE`, with a ribbon (cf. Figure 6). In this figure, we note that the probability of not having a follow-up (`D`) decreases with the time. The probability to be cured (`S`) has an opposite trend.

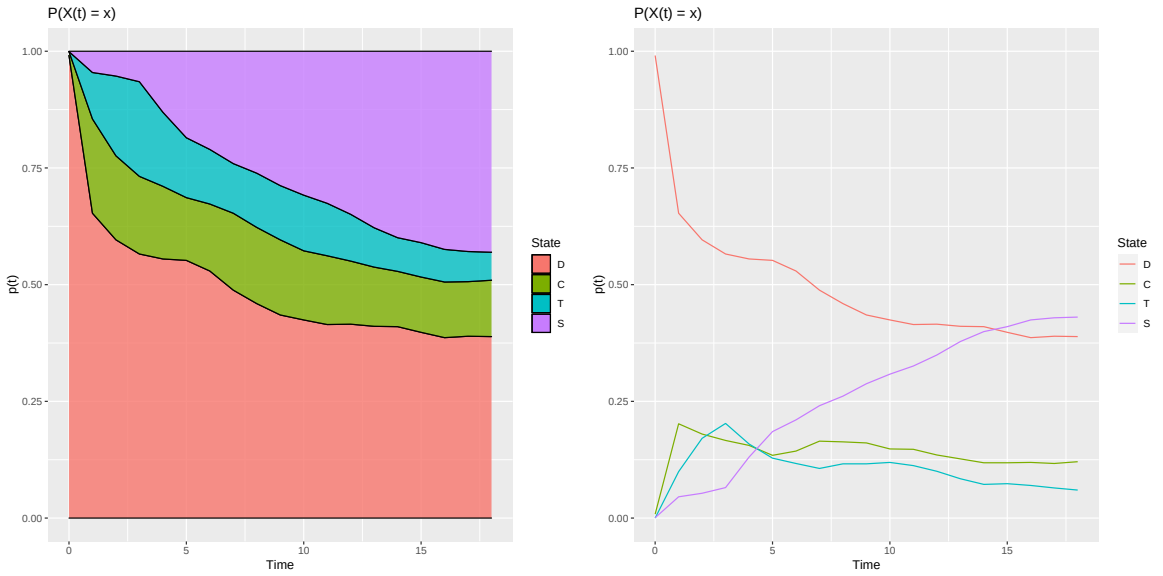
```
R> proba <- estimate_pt(care2)
R> proba

$pt
      0     1     2     3     4     5 ...
D 0.991 0.653 0.596 0.566 0.555 0.552 ...
C 0.008 0.202 0.180 0.166 0.156 0.134 ...
T 0.000 0.099 0.171 0.203 0.159 0.128 ...
S 0.001 0.046 0.053 0.065 0.131 0.185 ...
$t
 [1]  0  1  2  3  4  5  6 ...

R> plot(proba, ribbon = TRUE)
```

Continuous-time Markov Chain

Assume the data come from a continuous-time Markov chain. Note $\mathcal{S}$ the set of states and



(a) `ribbon = TRUE` (b) `ribbon = FALSE`

Figure 6: Probabilities to be in each state with regards to the time.

$X_t \in \mathcal{S}$, the state at time t then for $i, j \in \mathcal{S}$ and $t, s \geq 0$, then:

$$\mathsf{P}\left(X_{s+t} = j | X_s = i, \{X_u = x_u : 0 \leq u < s\}\right) = \mathsf{P}\left(X_{s+t} = j | X_s = i\right) = P_{ij}(t).$$

$P_{ij}(t)$ is the probability that the chain will be in state j in t time units, given it is in state i. A continuous-time Markov chain is completely described by its transition matrix $P = (P_{ij})$ and λ_i, $i \in \mathcal{S}$ the parameters of the exponentially distributed sojourn time in state i.

The `estimate_Markov` function estimates the transition matrix (`P`) and the λ parameter (`lambda`) associated with the mean sojourn time spent in each state through `1/lambda`.

```
R> mark <- estimate_Markov(care2)
R> mark
```

```
$P
    to
from          D          C          T          S
   D 0.00000000 0.63594891 0.23083942 0.13321168
   C 0.37955182 0.00000000 0.48459384 0.13585434
   T 0.02903811 0.13430127 0.00000000 0.83666062
   S 0.11594203 0.65942029 0.22463768 0.00000000

$lambda
        D         C         T         S
0.1328033 0.2538578 0.2438443 0.1149503
attr(,"class")
[1] "Markov"
```

The estimated process can be plotted as a diagram with the `plot` function displayed in Figure 7. Each node represents a state with its mean sojourn time. An arrow between two nodes defined a possible transition with its probability.

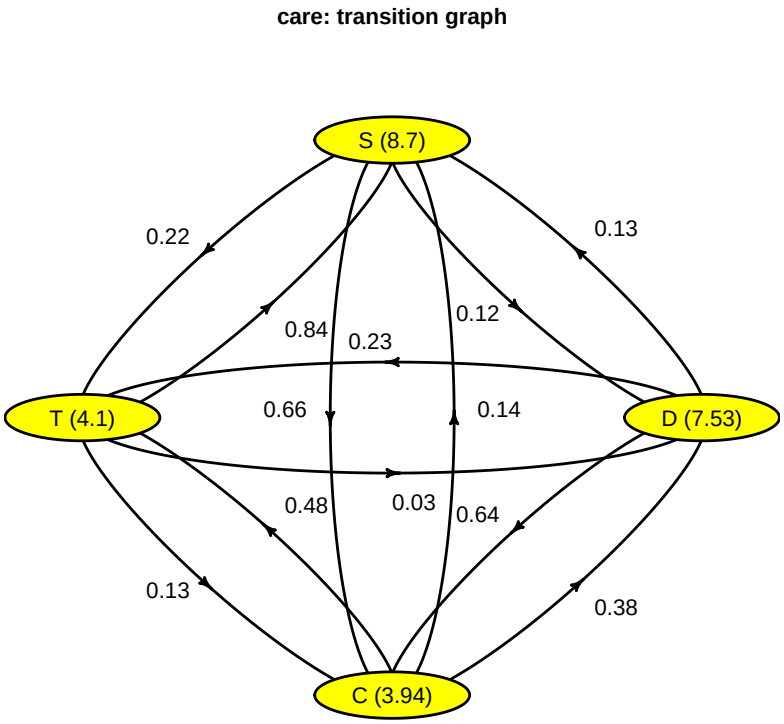


Figure 7: Transition graph displayed using `plot.Markov`.

```
R> plot(mark, main = "care: transition graph")
```

3.3. Optimal Encoding

The main contribution of **cfda** is the computation of an optimal encoding for categorical functional data performed by the `compute_optimal_encoding` function. The two main parameters are `data`, the dataset in the **cfda** format, and `basisobj`, a `basisfd` object created using the different `create.*.basis` functions from the **fda** package. It also performs bootstrap for computing a confidence interval on the computed encoding; associated parameters are `computeCI`, a logical indicating if bootstrap must be performed, `nBootstrap`, the number of bootstrap samples, and `propBootstrap`, the proportion of individuals used for each bootstrap sample. Other parameters are `nCores` the number of cores to use, `verbose`, if `TRUE`, some information are printed during the process. The `compute_optimal_encoding` function uses `integrate` (R Core Team 2020) to compute integrals, parameters for this function can be passed through `...` in particular `subdivisions`, the number of subdivisions to estimate the integral.

```
R> set.seed(42)
```

```
R> basis <- create.bspline.basis(c(0, 18), nbasis = 10, norder = 4)
R> fmca <- compute_optimal_encoding(care2, basis, nCores = 7)

######### Compute encoding #########
Number of individuals: 1317
Number of states: 4
Basis type: bspline
Number of basis functions: 10
Number of cores: 7
---- Compute V matrix:
  |======================================================================| 100%

DONE in 21.78s
---- Compute U matrix:
  |======================================================================| 100%

DONE in 122.42s
---- Compute encoding:
DONE in 0.13s
---- Compute Bootstrap Encoding:
**************************************************
DONE in 1.3s
Run Time: 149.84s
```

The main part of the computation time comes from the computation of V and U, these two steps are performed with parallel computation. Once these matrices computed, bootstrap is performed at a low computational cost. The output object of `compute_optimal_encoding` is a list containing:

eigenvalues eigenvalues of the problem (26)

alpha coefficients of the different encoding for each eigenvector (a list of matrices) (26)

pc principal components for each eigenvector

F F matrix (see equation (24))

V V matrix (see equation (22))

G covariance matrix of V (see equation (23))

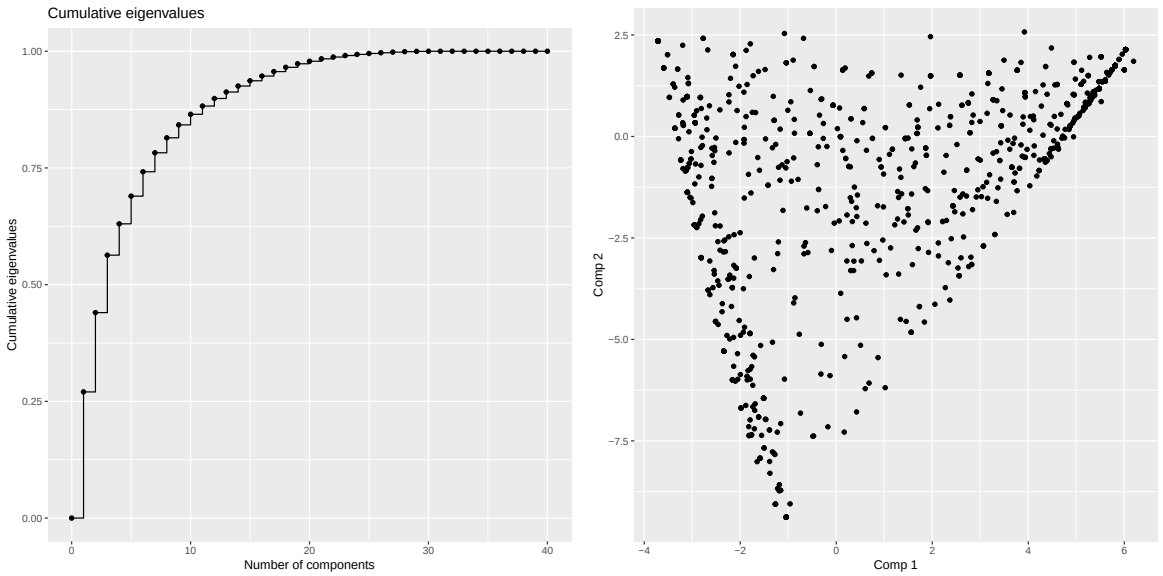
basisobj `basisobj` parameter

bootstrap encoding for each bootstrap sample

varAlpha a list containing $\widehat{\Sigma}_{(i,x)}\ \forall i, \forall x \in \mathcal{S}$, covariance matrix of $\widehat{\alpha}_{i,x} = \left(\widehat{\alpha}_{i,(x,1)}, \ldots, \widehat{\alpha}_{i,(x,m)}\right)$

Plot Functions

Three plot functions are associated with the `compute_optimal_encoding` function, the first argument of these functions is the output of `compute_optimal_encoding`.



(a) `plotEigenvalues` (b) `plotComponent`

Figure 8: Plots generated by different graphical functions on the output of the `compute_optimal_encoding` function.

The first one, the `plot` function plots the encodings associated with a given eigenvector (`harm` parameter, by default, the encodings associated with the first eigenvector are plotted). If `compute_optimal_encoding` was run with parameter `computeCI = TRUE`, then the confidence interval can be added on the plot using the parameter `addCI = TRUE`. A subset of the states can be plotted by providing a vector with the state names to the `states` parameter.

The `plotEigenvalues` function plots the computed eigenvalues. It has two extra boolean parameters: `cumulative`, if `TRUE`, the cumulative sum of the eigenvalues is plotted and `normalize`, if `TRUE`, eigenvalues are normalized such that their sum is equal to 1.

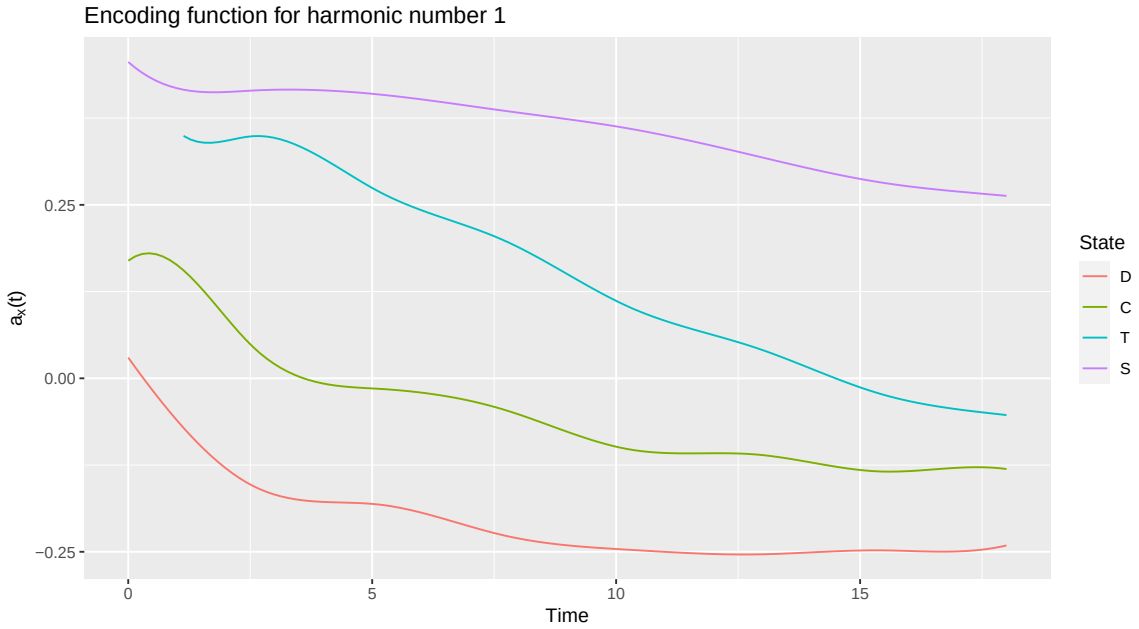
The last one is the `plotComponent` function that plots the individuals using the given components (`comp` parameter, a vector of length 2 containing the components' number). The other arguments are `addNames` that adds the individual's names on the plot and some parameters to adjust the position and size of these names (`nudge_x`, `nudge_y` and `size`).

The plots for the `care` dataset are shown in Figure 8 and 9 and are produced by the following code:

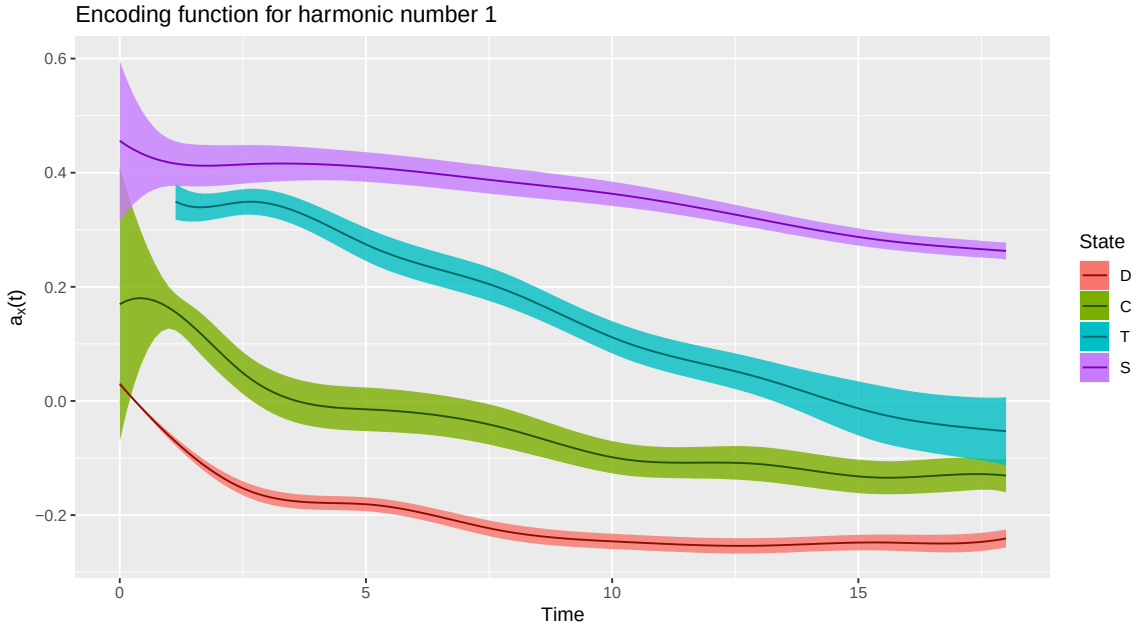
```
R> plotEigenvalues(fmca, cumulative = TRUE, normalize = TRUE)
R> plotComponent(fmca, comp = c(1, 2), addNames = FALSE)
R> plot(fmca)
R> plot(fmca, addCI = TRUE)
```

Extract the Encoding Functions

The computed encoding functions can be extracted using the `get_encoding` function as an object of class `fd` (functional object from **fda**) using `fdObject = TRUE` or as a matrix using



(a) `plot(fmca)`



(b) `plot(fmca, addCI = TRUE)`

Figure 9: Plots generated by the `plot` function on the output of the `compute_optimal_encoding` function.

`fdObject = FALSE`. In the latter case, an extra parameter `nx` specifies the number of time points to extract.

```
R> encodingFd <- get_encoding(fmca, fdObject = TRUE)
R> str(encodingFd)

List of 3
 $ coefs  : num [1:10, 1:4] 0.0299 -0.0543 -0.1965 -0.1645 -0.2371 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:4] "D" "C" "T" "S"
 $ basis  :List of 10
  ..$ call       : language basisfd(type = type, | __truncated__
  ..$ type       : chr "bspline"
  ..$ rangeval   : num [1:2] 0 18
  ..$ nbasis     : num 10
  ..$ params     : num [1:6] 2.57 5.14 7.71 10.29 12.86 ...
  ..$ dropind    : NULL
  ..$ quadvals   : NULL
  ..$ values     : list()
  ..$ basisvalues: list()
  ..$ names      : chr [1:10] "bspl4.1" "bspl4.2" "bspl4.3" "bspl4.4" ...
  ..- attr(*, "class")= chr "basisfd"
 $ fdnames:List of 3
  ..$ args: chr "time"
  ..$ reps: chr [1:4] "reps 1" "reps 2" "reps 3" "reps 4"
  ..$ funs: chr "values"
 - attr(*, "class")= chr "fd"

R> encodingMat <- get_encoding(fmca, fdObject = FALSE, nx = 19)
R> encodingMat

$x
 [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18

$y
                D           C          T         S
 [1,]  0.02986969  0.169492601  0.50380590 0.4559043
 [2,] -0.06073672  0.163315622  0.35643506 0.4180230
 [3,] -0.12970105  0.088328506  0.34225746 0.4126089
 [4,] -0.16758420  0.020411074  0.34651825 0.4159033
 [5,] -0.17812958 -0.007566828  0.31652186 0.4149973
 [6,] -0.18096760 -0.014569119  0.27436391 0.4100582
 [7,] -0.19348217 -0.020533054  0.24249695 0.4020949
 [8,] -0.21335627 -0.032358522  0.21809774 0.3925044
 [9,] -0.23081796 -0.051930578  0.18880144 0.3828197
[10,] -0.24069960 -0.077085840  0.15014660 0.3734934
```

```
[11,] -0.24597159 -0.098545473  0.11162958 0.3630747
[12,] -0.25009513 -0.107580629  0.08297660 0.3500902
[13,] -0.25321503 -0.107980977  0.06221397 0.3347482
[14,] -0.25359641 -0.110631333  0.04073503 0.3182122
[15,] -0.25084388 -0.121299335  0.01379572 0.3018281
[16,] -0.24813344 -0.132111070 -0.01301242 0.2873926
[17,] -0.24881502 -0.133707940 -0.03278505 0.2766485
[18,] -0.24943891 -0.128922501 -0.04469775 0.2692082
[19,] -0.24091335 -0.130703127 -0.05297104 0.2629170
```

Interpreting the Encoding Functions

First have a look at the plot of the encoding functions associated with the harmonic number 1 (cf. Figure 9a).

```
R> plot(fmca, harm = 1)
```

The lowest curve is for state `"D"`, this indicates that individuals with a large negative value for component number 1 tend to spend more time in this state. Similarly, individuals with a large positive value tend to spend more time in the state `"S"`. To check these statements, individuals with extreme values on the first component are plotted using the `plotData` function with the `group` parameter. A group variable is created with two different values: `"min"` for the individuals with the 5% lowest value, and `"max"` for the individuals with 5% highest value.

```
R> minpc1 <- names(which(fmca$pc[,1] <= quantile(fmca$pc[,1], 0.05)))
R> maxpc1 <- names(which(fmca$pc[,1] >= quantile(fmca$pc[,1], 0.95)))
R> ids <- unique(care2$id)
R> group <- factor(rep(NA, length(ids)), levels = c("min", "max"))
R> group[ids %in% minpc1] = "min"
R> group[ids %in% maxpc1] = "max"
R> plotData(care2, group = group, addId = FALSE, addBorder = FALSE,
+           sort = TRUE) +
+  ggplot2::labs(title = "Extreme individuals on component 1")
```

The result is visible in Figure 10 and confirms our suppositions. Clearly, individuals in the `"min"` group spend 18 months in the `"D"` state (without medical follow-up) whereas individuals in the `"max"` group spend most of their time in the state `"S"` (infection suppressed).

Application to Clustering

The proposed method produces numerical encoding for categorical functional data. These encoding can be used for classical statistical methods such that regression or clustering. In the following, we perform a hierarchical clustering to find a structure in the `care` dataset.

The clustering is performed with the first principal components explaining at least 90% of the variance. The associated tree is displayed in Figure 11.

```
R> nPc90 <- which(cumsum(prop.table(fmca$eigenvalues)) > 0.9)[1]
R> hc <- hclust(dist(fmca$pc[, 1:nPc90]), method = "ward.D2")
R> plot(hc, labels = FALSE)
```

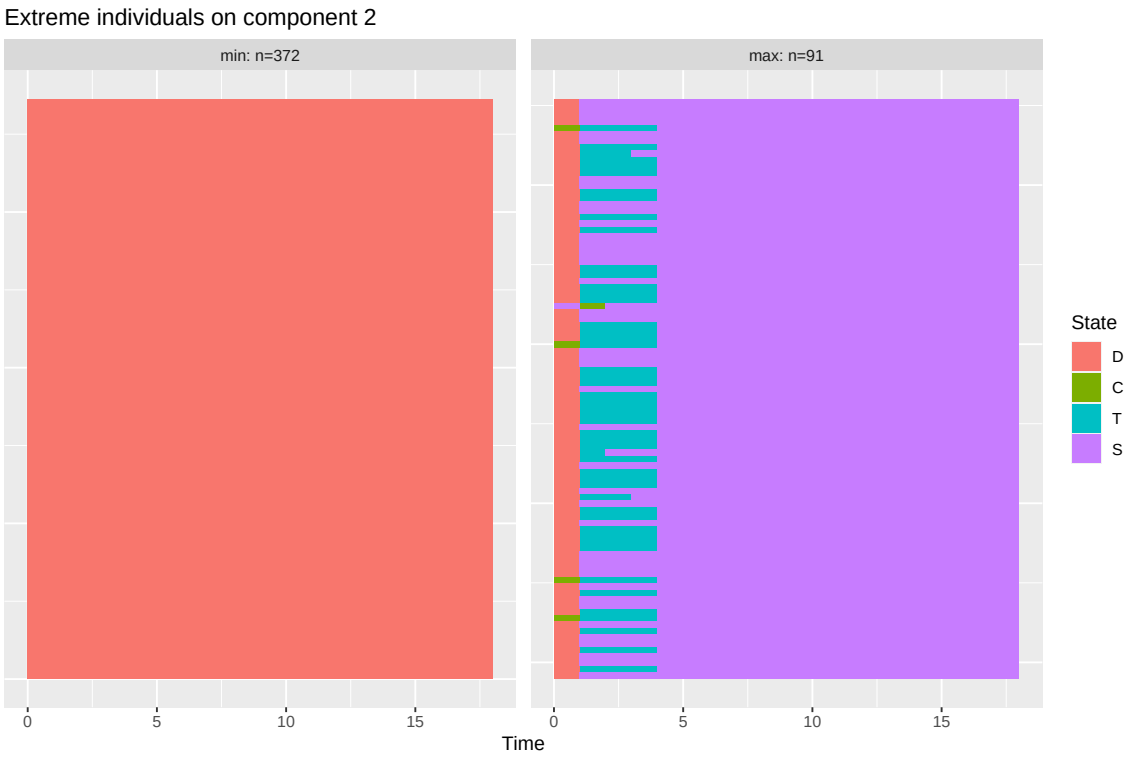


Figure 10: Individuals with extreme negative value (`min`) and extreme positive value (`max`) on the component 1.

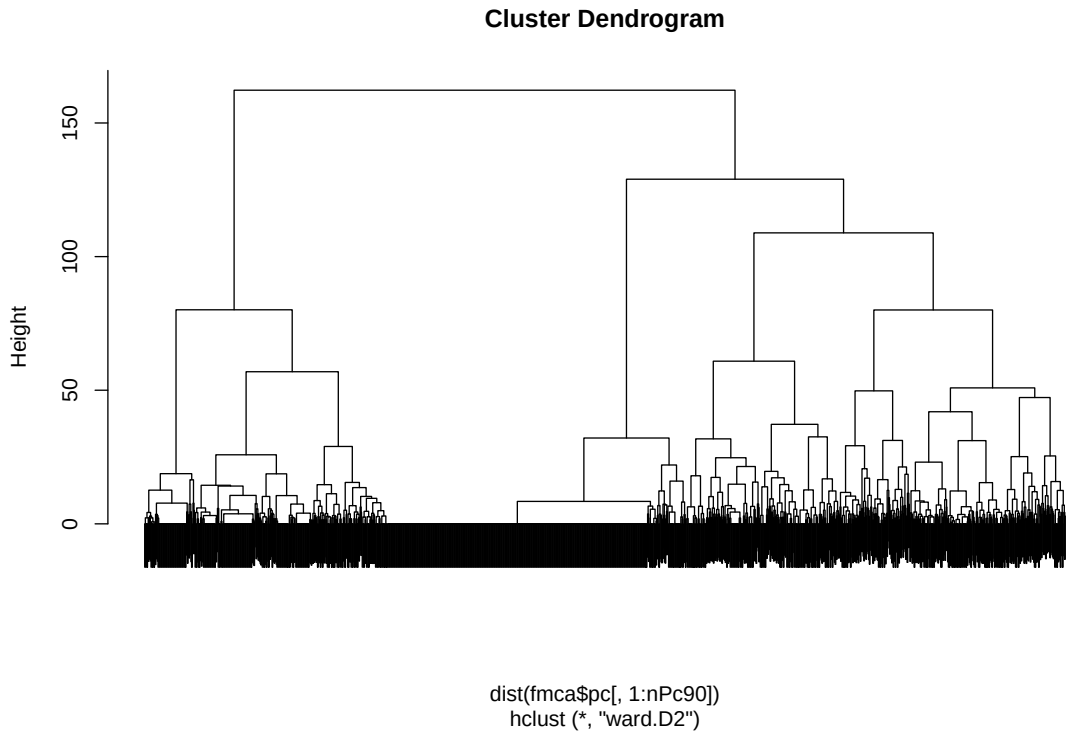


Figure 11: Hierarchical tree obtained using the principal components.

We decide to keep 4 clusters regarding the heights of the tree. The resulting clusters can be displayed using the `plotData` function with the `group` argument.

```
R> class <- cutree(hc, k = 4)
R> plotData(care2, group = class, addId = FALSE, addBorder = FALSE, +
+           sort = TRUE)
```

The different clusters are associated with the time spent in the different states after leaving the state `"D"` (cf. Figure 12). For example, the cluster number 1 corresponds to individuals that have spent most of their time (after `"D"`) in the `"C"` state.

4. Simulation Study

4.1. Birth-and-death Process

Data are simulated under the simple model of birth-and-death process presented in Saporta (1981). The process is defined on the interval time $[0, 1]$ by

$$X_t = \begin{cases} 0, & \text{if } t < \theta, \\ 1, & \text{if } t \geq \theta, \end{cases} \tag{31}$$

where θ is a random variable uniformly distributed on $[0, 1]$.

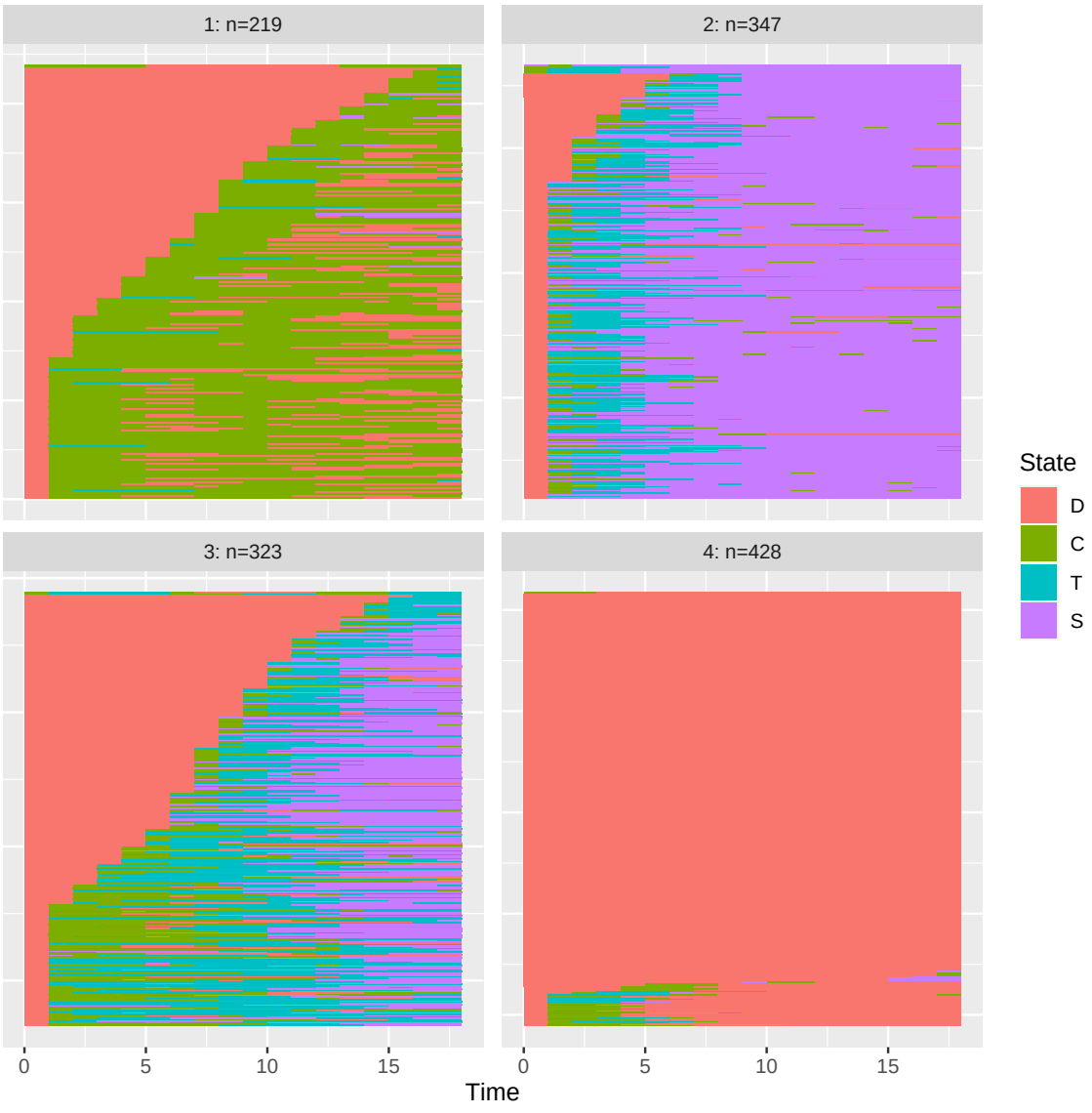


Figure 12: Content of the different clusters.

In this case, $K = 2$, $\mathcal{S} = \{s_1 = 0, s_2 = 1\}$ and

$$p^x(t) = \mathsf{P}(X_t = x) = \begin{cases} t, & \text{if } x = 1 \\ 1 - t, & \text{if } x = 0. \end{cases}$$

For $t < s$ we have

$$p^{x,y}(t,s) = \mathsf{P}(X_t = x, X_s = y) = \begin{cases} 1 - s, & \text{if } x = 0, y = 0, \\ s - t, & \text{if } x = 0, y = 1, \\ 0 & \text{if } x = 1, y = 0, \\ t & \text{if } x = 1, y = 1, \end{cases}$$

and for $t > s$,

$$p^{x,y}(t,s) = \mathsf{P}(X_t = x, X_s = y) = \begin{cases} 1 - t, & \text{if } x = 0, y = 0, \\ 0 & \text{if } x = 0, y = 1, \\ t - s & \text{if } x = 1, y = 0, \\ s & \text{if } x = 1, y = 1. \end{cases}$$

From (12) and (14), the authors provide explicit formulas for the eigenvalues $\{\lambda_i\}_{i\geq 1}$, the optimal encoding functions $\{a_i^x\}_{i\geq 1}$, $x \in \mathcal{S}$, and the principal components $\{z_i\}_{i\geq 1}$ as follows:

- the eigenvalues are given by:

$$\lambda_i = \frac{1}{i(i+1)}, \quad i \geq 1. \tag{32}$$

- if $P_n(u) = \dfrac{1}{2^n n!}\dfrac{d^n}{du^n}(u^2 - 1)^n$ is the Legendre polynomials of order n, then the principal components z_i corresponding to λ_i, are given, up to a constant, by:

$$z_i = P_i(2\theta - 1), \quad i \geq 1.$$

In particular, for $i = 1, 2$,

$$z_1 = \sqrt{6}\left(\theta - \frac{1}{2}\right)$$

is uniformly distributed on $\left[-\sqrt{\frac{3}{2}}; \sqrt{\frac{3}{2}}\right]$ and

$$z_2 = \sqrt{30}\left(\theta^2 - \theta + \frac{1}{6}\right).$$

Let observe that z_1 and z_2 are linearly uncorrelated but related by

$$z_2 = \sqrt{\frac{5}{6}}\left(z_1^2 - \frac{1}{2}\right), \tag{33}$$

showing some regularity of the 2-D representation of data throughout the plot $\{(z_1(\omega), z_2(\omega), \omega \in \Omega\}$.

- for $i = 1, 2$, the optimal encoding functions are given by:

$$a_1^x(t) = \begin{cases} \sqrt{6}t, & \text{if } x = 0, \\ \sqrt{6}(t-1), & \text{if } x = 1, \end{cases} \tag{34}$$

and

$$a_2^x(t) = \begin{cases} \sqrt{120}\left(t^2 - \frac{t}{2}\right), & \text{if } x = 0, \\ \sqrt{120}\left(t^2 - \frac{3}{2}t + \frac{1}{2}\right), & \text{if } x = 1. \end{cases} \tag{35}$$

4.2. Results

We simulate data from the above process with different number of trajectories (individuals), $n = 50, 100, 200, 500$ and a B-spline basis functions of order 4 with different number of basis functions $m = 5, 10, 20$ (equidistant knots). Simulation results are compared to the theoretical results presented in Section 4.1 to ensure the well-behaviour of the implemented method.

Eigenvalues

The first five eigenvalues for the different settings are compared to the eigenvalues from (32) in the table 1. The estimations are presented together with the associated standard errors in order to measure the impact of the choice of the sample size (n) and the dimension of the basis (m).

Encoding Functions

Figure 13 shows the mean over 100 samples of the first and second encoding functions for the state 0 for $m = 5$. The true encoding functions (34) and (35) are displayed in solid black line. As for the eigenvalues, the best estimates are achieved with $n = 200$ and $n = 500$. The same conclusion holds for $m = 10$ and $m = 20$ but the number of basis does not seem to influence the accuracy.

Principal Components

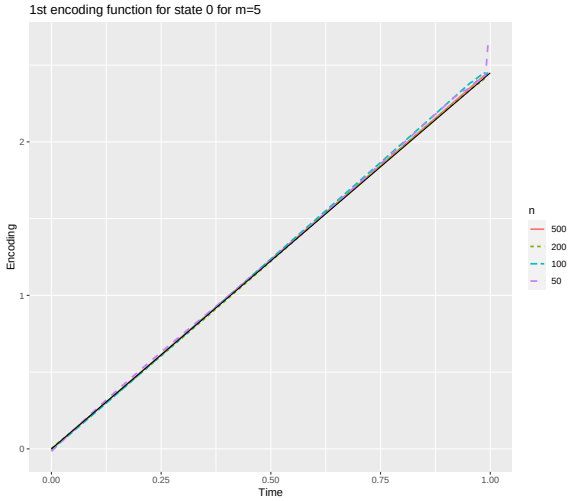
In Figure 14, we check the relation between the first and second principal component (33). The theoretical equation is displayed in black whereas the computed principal components for a sample with $n = 500$ and $m = 20$ are in red. We note the closeness of the computed components with the theoretical equation.

In Figure 15, the cumulative distribution functions (cdf) for the two first principal components are displayed as well as their empiric equivalent for $n = 500$ and $m = 20$. As described above, z_1 follow an uniform distribution between $-\sqrt{\frac{3}{2}}$ and $\sqrt{\frac{3}{2}}$, the empiric cdf (in red) is closed to the theoretical one. The same representation is made for z_2.

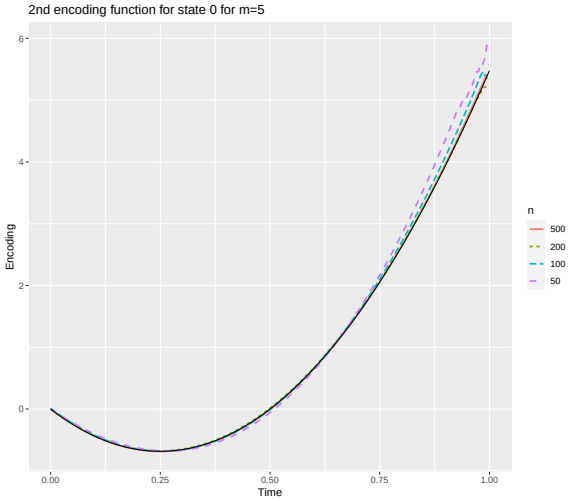
Computational details

Table 1: True and estimated eigenvalues for the birth-and-death process. The estimated values are the mean over 100 samples. In brackets, the standard error is displayed.

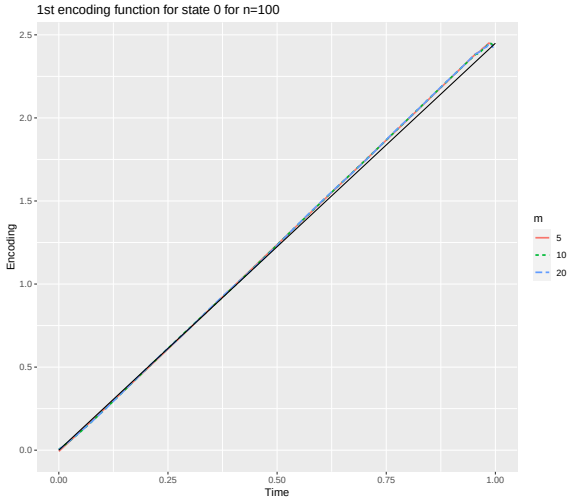
			m=5		
	true	**n=50**	**n=100**	**n=200**	**n=500**
1	0.5000	0.5117 (6.4e-4)	0.5013 (4.4e-4)	0.5000 (3.1e-4)	0.5009 (1.8e-4)
2	0.1667	0.1680 (3.3e-4)	0.1672 (2.0e-4)	0.1679 (1.5e-4)	0.1664 (0.9e-4)
3	0.0833	0.0824 (1.9e-4)	0.0835 (1.4e-4)	0.0834 (0.9e-4)	0.0835 (0.5e-4)
4	0.0500	0.0455 (1.3e-4)	0.0490 (0.9e-4)	0.0494 (0.6e-4)	0.0492 (0.4e-4)
5	0.0333	0.0184 (0.6e-4)	0.0205 (0.4e-4)	0.0211 (0.3e-4)	0.0215 (0.2e-4)
			m=10		
	true	**n=50**	**n=100**	**n=200**	**n=500**
1	0.5000	0.5124 (6.4e-4)	0.5016 (4.3e-4)	0.5002 (3.1e-4)	0.5009 (1.8e-4)
2	0.1667	0.1692 (3.4e-4)	0.1677 (2.0e-4)	0.1682 (1.5e-4)	0.1665 (0.9e-4)
3	0.0833	0.0841 (2.0e-4)	0.0843 (1.4e-4)	0.0839 (0.9e-4)	0.0837 (0.5e-4)
4	0.0500	0.0486 (1.3e-4)	0.0510 (0.9e-4)	0.0508 (0.7e-4)	0.0501 (0.4e-4)
5	0.0333	0.0317 (0.8e-4)	0.0335 (0.6e-4)	0.0334 (0.4e-4)	0.0336 (0.2e-4)
			m=20		
	true	**n=50**	**n=100**	**n=200**	**n=500**
1	0.5000	0.5128 (6.4e-4)	0.5018 (4.4e-4)	0.5003 (3.1e-4)	0.5010 (1.8e-4)
2	0.1667	0.1699 (3.4e-4)	0.1681 (2.0e-4)	0.1683 (1.5e-4)	0.1666 (0.9e-4)
3	0.0833	0.0849 (2.0e-4)	0.0847 (1.4e-4)	0.0841 (0.9e-4)	0.0837 (0.5e-4)
4	0.0500	0.0495 (1.3e-4)	0.0514 (0.9e-4)	0.0510 (0.6e-4)	0.0502 (0.4e-4)
5	0.0333	0.0327 (0.8e-4)	0.0340 (0.6e-4)	0.0337 (0.5e-4)	0.0337 (0.2e-4)



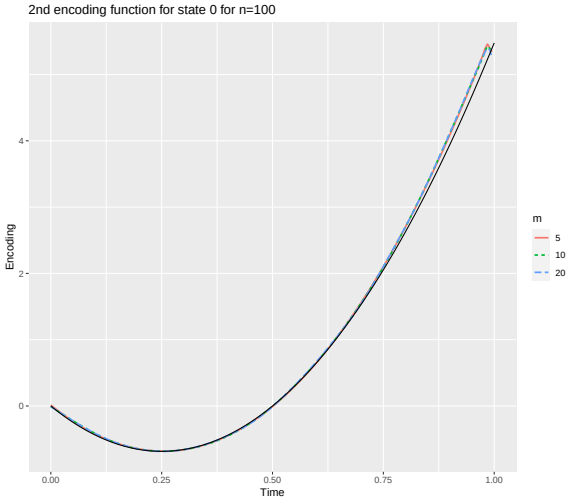
(a) First encoding function for state 0 ($m = 5$)



(b) Second encoding function for state 0 ($m = 5$)



(c) First encoding function for state 0 ($n = 100$)



(d) Second encoding function for state 0 ($n = 100$)

Figure 13: True (solid black) and estimated encoding functions for state 0 of the birth-and-death process. The estimated encoding functions are the mean of 100 samples.

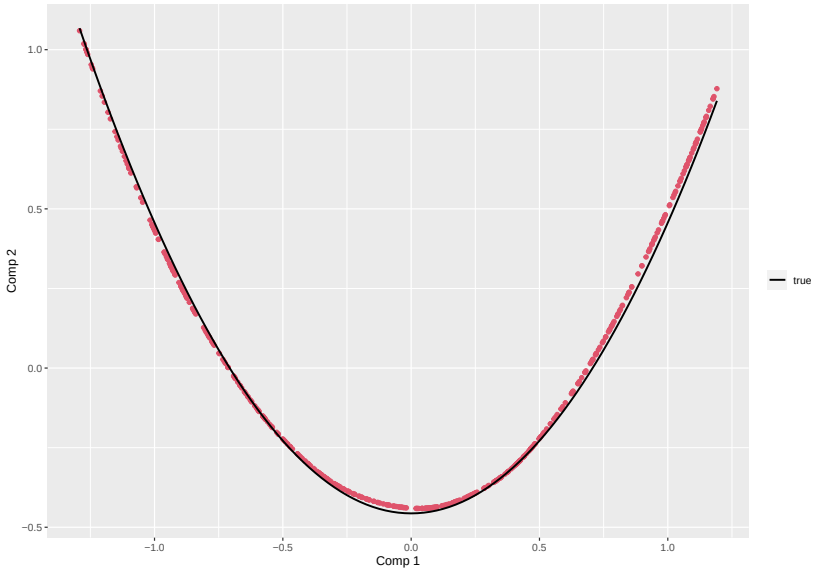
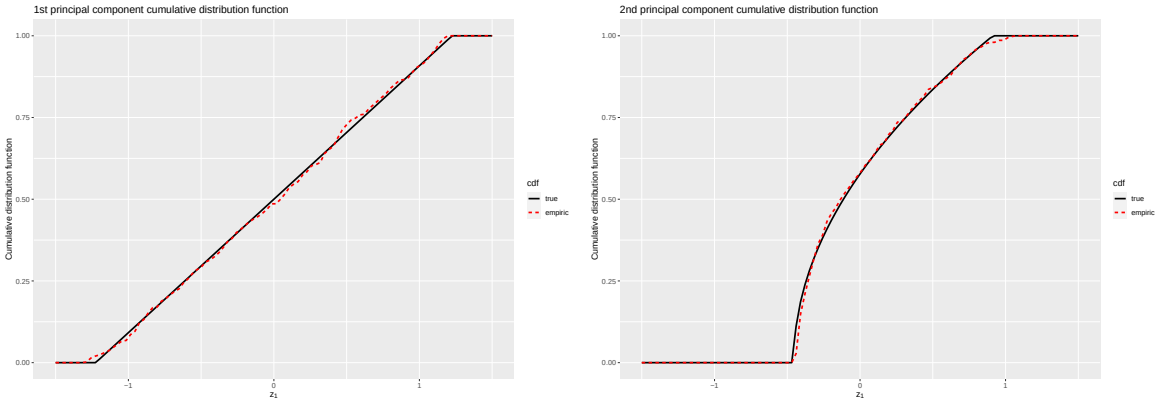


Figure 14: In red, first and second principal components for every individual of the birth-and-death process for $n = 500$ and $m = 20$. In solid black, the theoretical relation between these two components.



(a) Cumulative distribution function for z_1 (b) Cumulative distribution function for z_2

Figure 15: Empirical (resp. theoretical) cumulative distribution function for the first (z_1) and second (z_2) principal components of the birth-and-death process for $n = 500$ and $m = 20$.

- The approximation of optimal encoding functions in a basis of functions is based on the computation of random variables $V_{(x,i)}$ and $U_{(x,i),(x,j)}$ defined in (22) and (25) respectively. The computation of integrals involved in the definition of these random variables uses the `inprod` function of the **fda** R package which, at its turn, calls the function `eval.fd`. For n and K fixed, this step is the most computational in terms of time resources and it depends of the number of basis functions, m, considered for the approximation (19). As the computation is at done for every ω in $(\omega_1, \ldots, \omega_n)$, parallel computation in performed.
- The F matrix (24) can be singular in some specific situations, namely when there exists an interval $I \subset [0, T]$ and some state x such that $p^x(t) = 0, \forall t \in I$. In this case the hypothesis H_2 is not satisfied. For $t \in I$, the operator E_t is degenerated, however, the eigenvalue equation (5) is still valid. From (12), the optimal encodings a^x function is not defined for $t \in I$. From a computational point of view, if ϕ_i is some element of the basis functions $\{\phi_1, \ldots, \phi_m\}$ with support in I then the random variables $V(x, i)$ and $U_{(x,i),(.,.)}$ are zero-constant and therefore, the row and column corresponding to (x, i) in the F and G matrices are zero vectors. Thus, the element $\alpha_{(x,i)}$ of the expansion coefficients vector α_x is not defined. Dropping the rows and columns from F and G corresponding to (x, i) allows to solve the eigen-problem in (26). Notice that the constraints (27) are fulfilled.

The results in this paper were obtained using R 4.0.3 with the **cfda** 0.9.7 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

5. Summary and discussion

Categorical functional data is represented by paths of a continuous-time stochastic process with values in a finite set of states. Less popular than the real-valued functional data, that is yet another kind of infinite dimensional object. The analysis of categorical functional data is presented in this paper as an extension of the multiple correspondence analysis for the finite dimensional setting. principal components, optimal encoding functions and optimal representations are presented. A simulation study and a real data application illustrate the methodology implemented in the **cfda** R package.

References

Boumaza R (1980). *Contribution à l'Étude Descriptive d'une Fonction Aléatoire Qualitative*. Ph.D. thesis, Université Paul Sabatier, Toulouse.

Cardot H, Lecuelle G, Schlich P, Visalli M (2019). "Estimating Finite Mixtures of Semi-Markov Chains: an Application to the Segmentation of Temporal Sensory Data." *Journal of the Royal Statistical Society C*, **68**(5), 1281–1303. doi:10.1111/rssc.12356.

Deville J, Saporta G (1983). "Correspondence Analysis with an Extension towards Nominal Time Series." *Journal of Econometrics*, **22**, 169–189. doi:10.1016/0304-4076(83)90098-2.

Deville JC (1974). "Méthodes Statistiques et Numériques de l'Analyse Harmonique." *Annales de l'INSEE*, **15**, 5–101. doi:10.2307/20075177.

Deville JC (1982). "Analyse de Données Chronologiques Qualitatives : Comment Analyser des Calendriers ?" *Annales de l'INSEE*, **45**, 45–104. doi:10.2307/20076433.

Escofier B (1978). "Analyse Factorielle et Distances Répondant au Principe d'Équivalence Distributionnelle." *Revue de Statistiques Appliquées*, **26**, 29–37.

Ferraty F, Vieu P (2006). *Nonparameric Functional Data Analysis. Theory and Practice*. Springer-Verlag New York. ISBN 978-0-387-36620-3. doi:10.1007/0-387-36620-2.

Gabadinho A, Studer M, Müller N, Bürgin R, Fonta PA, Ritschard G (2019). ***TraMineR**: Trajectory Miner: a Toolbox for Exploring and Rendering Sequences*. R package version 2.0-13, URL https://CRAN.R-project.org/package=TraMineR.

Heijden P, Teunissen J, van Orlé C (1997). "Multiple Correspondence Analysis as a Tool for Quantification or Classification of Career Data." *Journal of Educational and Behavioral Statistics*, **22**, 447–477. doi:10.3102/10769986022004447.

Hotelling H (1936). "Relations Between Two Sets of Variates." *Biometrika*, **28**, 321–377. doi:10.2307/2333955.

Jackson CH (2011). "Multi-State Models for Panel Data: The **msm** Package for R." *Journal of Statistical Software*, **38**(8), 1–29. doi:10.18637/jss.v038.i08.

Melnykov V (2016). "**ClickClust**: An R Package for Model-Based Clustering of Categorical Sequences." *Journal of Statistical Software*, **74**(9), 1–34. ISSN 1548-7660. doi:10.18637/jss.v074.i09.

Mercer J (1909). "Functions of Positive and Negative Type and their Connection with the Theory of Integral Equations." *Philosophical Transactions of the Royal Society A*, **209**, 441–458. doi:10.1098/rsta.1909.0016.

Microsoft, Weston S (2019). ***foreach**: Provides Foreach Looping Construct*. R package version 1.4.7, URL https://CRAN.R-project.org/package=foreach.

Nath R, Pavur R (1985). "A New Statistic in the One Way Multivariate Analysis of Variance." *Computational Statistics and Data Analysis*, **2**, 297–315. doi:10.1016/0167-9473(85)90003-9.

Preda C (1998). "Analyse Harmonique Qualitative des Processus Markoviens des Sauts Stationnaires." *Scientific Annals of Computer Science*, **7**, 5–18. URL https://www.info.uaic.ro/en/sacs_articles/analyse-harmonique-qualitative-des-processus-markoviens-des-sauts-stationnaires/.

Ramsay J, Silverman B (2005). *Functional Data Analysis*. Springer-Verlag New York. ISBN 978-0-387-22751-1. doi:10.1007/b98888.

Ramsay JO, Wickham H, Graves S, Hooker G (2018). ***fda**: Functional Data Analysis*. R package version 2.4.8, URL https://CRAN.R-project.org/package=fda.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Saporta G (1981). *Méthodes Exploratoires d'Analyse de Données Temporelles*. Université Pierre et Marie Curie, Paris, France. URL http://www.numdam.org/item/BURO_1981__37-38__7_0/.

Scholz M (2016). "R Package **clickstream**: Analyzing Clickstream Data with Markov Chains." *Journal of Statistical Software*, **74**(4), 1–17. doi:10.18637/jss.v074.i04.

Studer M (2013). "**WeightedCluster** Library Manual: A Practical Guide to Creating Typologies of Trajectories in the Social Sciences with R." *Technical report*, LIVES Working Papers 24. doi:10.12682/lives.2296-1658.2013.24.

Tierney L, Rossini AJ, Li N, Sevcikova H (2018). ***snow**: Simple Network of Workstations*. R package version 0.4-3, URL https://CRAN.R-project.org/package=snow.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org.

Affiliation:

Cristian Preda
Université de Lille,
Laboratoire de Mathématiques Paul Painlevé, UMR CNRS 8524
59655 Villeneuve d'Ascq, France
and
MΘDAL team, Inria Lille-Nord Europe
40 avenue Halley
59650 Villeneuve-d'Ascq, France
and
Institute of Statistics and Applied Mathematics of the Romanian Academy
050711 Bucharest, Romania
E-mail: cristian.preda@univ-lille.fr

Quentin Grimonprez
MΘDAL team, Inria Lille-Nord Europe, France
40 avenue Halley
59650 Villeneuve-d'Ascq, France
E-mail: quentin.grimonprez@inria.fr

Vincent Vandewalle
Université de Lille
ULR 2694 - METRICS
59000 Lille, France
and
MΘDAL team, Inria Lille-Nord Europe, France
40 avenue Halley
59650 Villeneuve-d'Ascq, France
E-mail: vincent.vandewalle@univ-lille.fr

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd