POLITECNICO DI MILANO

Scuola di Ingegneria dei Sistemi

Corso di Laurea Specialistica in Ingegneria Matematica

Dipartimento di Matematica

Development of the "fdakma" R package for the joint alignment and clustering of functional data: application to neuronal spike trains data

Relatori: Dott.ssa SANGALLI Laura Maria
Dott. VANTINI Simone

Tesi di Laurea di:

Mirco PATRIARCA

Matr. 751737

Anno Accademico 2012–2013

Ebbe ma eh allooooora!
M. Patriarca, Gli anni milanesi

Sommario

In questo lavoro di tesi si presentano le principali caratteristiche dell'analisi di dati funzionali, soffermandosi sugli ultimi sviluppi teorici di questa nuova branca della statistica. In particolare si tratterà il problema della *registrazione* di dati funzionali con l'obiettivo di dare un solido framework teorico al problema. Saranno presentati tre metodi di registrazione, tra i quali il *K-Mean Alignment*, capace di effettuare allo stesso tempo clusterizzazione e allineamento, e un metodo basato sulla metrica di *Fisher-Rao*. Il *K-Mean Alignment* verrà in seguito utilizzato per analizzare un dataset funzionale, in cui sono raccolte le intensità dell'attività neuronale di una scimmia impegnata in un esercizio manuale. Nell'ultima parte di questo lavoro é stato sviluppato il pacchetto R *fdakma*, disponibile sul CRAN. Questo pacchetto riassume il metodo del *K-Mean Alignment* e permette di clusterizzare e allineare dati funzionali usando diverse metriche/semimetriche e diversi tipi di warping functions.

PAROLE CHIAVE: Analisi di dati funzionali; Clustering; Allineamento; Time warping; Similarità; Spike trains.

Abstract

In this work we present the basics of functional data analysis, pointing out the latest developments in this new branch of statistics and in particular in the subfield of functional data registration, in order to give a solid theoretical framework for the solution of the registration problem. Three registration methods are presented, including the *K-Mean Alignment* method, able to jointly cluster and align functional data, and the one based on *Fisher-Rao* metric. The *K-Mean Alignment* technique will be used to analyze a dataset containing spike trains intensities, i.e., data coming from brain impulses of a monkey doing a particular task using his hand. In the last part of this work we develop an R package called *fdakma*. This package, available on CRAN, resumes the *K-Mean Alignment* method and allows user to jointly cluster and align functional data using different metrics/semimetrics and different types of warping functions.

KEYWORDS: Functional data analysis; Clustering; Alignment; Time warping; Similarity; Spike trains.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Framework and motivation

This work aims to present and develop the basic theory behind a new branch of statistics, the *functional data analysis*, whose importance has exponentially grown through the last decades and started to be investigated thanks of the intuition of some pioneers, like Ramsay et Silverman (2005), who proposed this different approach in analyzing data coming from realizations of functional random variables. At the present the statistics community is still looking for a theoretical framework to finally fix the basis of this new domaine, so interesting and useful for its new way to look at data.

It is for these reasons that in November 2012, in The Ohio State University, (Columbus, Ohio) J. S. Marron (UNC), J. O. Ramsay (McGill), L. Sangalli (Politecnico di Milano), A. Srivastava (Florida State) organized a workshop called "Statistics of Time Warpings and Phase Variations". Four datasets have been proposed and the groups who were invited analyzed them and presented their results in this occasion. In this work we present the analysis we performed for this event to one of the available datasets, precisely the "Spike train" dataset.

It is in this framework that this work is born. In particular we will present a new registration method, able to jointly cluster and register functional data (we will present later the concept of "Registration of functional data"). This new method, called *K-Mean Alignment* and described in Sangalli, Secchi, Vantini, et Vitelli (2010), will be applied, as already said, to a dataset containing neuronal spike trains data, which means data deriving from the registration of neuronal activity of a monkey doing a particular task using his hands. Finally, we developed an R package called *fdakma*, which will be available soon on CRAN repository. This package is able to perform the *K-Mean Alignment* algorithm.

1.2 Objectives of the study

The study, as said before, aims at presenting an introduction to FDA (Functional Data Analysis). In chapter 2 we will discuss about the latest development on this domaine, highlighting the properties that the couple *"similarity measure"-"group of warping functions"* (ρ, W) should have. We will explain in detail the new registration method we used, the *K-Mean Alignment* method described in Sangalli et al. (2010) and also some other registration methods, like the one based on the Fisher-Rao metric proposed in Srivastava, Wu, Kurtek, Klassen, et Marron (2011) and the *Minimum eigenvalue* method presented in Ramsay et Silverman (2005).

Then, in chapter 3, we will discuss the results of applying this method to the neuronal "spike train" dataset presented in Wu et Srivastava (2011). We will approach the analysis of this data from two different points of view with the ambitious goal of gaining some informations about how our brain works and exchanges messages through neurons.

After the dataset analysis, in chapter 4 we will present the development of the R package *fdakma*, which resumes the *K-Mean Alignment* method presented in chapter 2. We created this package in order to allow the repetition of the study presented in this work but above all to give to the community the possibility to use this method in other analysis, hoping this will help and improve the state of art of this new fascinating branch of statistics called Function Data Analysis.

Chapter 2

Functional Data Analysis (FDA)

2.1 Introduction to FDA

First we give a definition of FDA. We will try to let the reader understand the cases in which this kind of approach can be useful and revolutionary with respect to other statistics procedures.

Functional data analysis is a branch of statistics that analyzes data providing information about curves, surfaces or anything else varying over a continuum. For example, this continuum can be represented by time, but also by spatial location. The principal strength of this approach concerns the possibility to consider data coming from a function. This gives the chance to analyze different characteristics which are normally difficult to handle.

In particular, functional data analyses can consider informations present in the slopes or curvatures of the functions. This is reflected in their derivatives. First or second derivatives can sometimes reveal important aspects of the processes standing behind the data. For this reason, a fundamental role in functional data analysis is played by curve estimation techniques.

Given that, normally, at the beginning of a functional data analysis, we only dispose of points that are evaluations of some functions, it is crucially important to have some kind of process which enables us to work with true functions and not only with its evaluations. This passage from "discrete points" to a continuum represents the first key step to a good functional data analysis. This procedure is often called *smoothing*. In literature there are many methods able to approximate functions defined by points. They can make use of a simple histogram or more complicated means, like *kernels* but the goal remains the same: to finally have functions defined over a continuum instead of functions defined over a set of points. Obviously there is not a good solution for every situation: sometimes it can be good to interpolate exactly our points with splines, for example, other times the use of more sophisticated methods can bring to light some important features of the phenomenon behind the data.

In this work we will not treat the functional smoothing, but it is important to know that it represents a fundamental starting point of a good functional data analysis. More informations about this preliminary part of FDA can be found in Ramsay et Silverman (2005) and in Hastie, Tibshirani, et Friedman (2009).

In general, models and methods for the analysis of functional may seem similar to those used for the analysis of conventional multivariate data, including linear and nonlinear regression models, principal components analysis, and others. But the possibility of using derivative information considerably extends the power of these techniques, and leads to functional models like those defined by differential equations.

As example, in figure 2.1 we present a classic benchmark of functional data. It shows the heights of 39 men and 54 girls measured at a set of 31 ages in

the Berkeley Growth Study in Tuddenham et Snyder (1954). Note that the ages are not equally spaced: we have four measurements when the child is one year old, one measurement per year from two to eight years, and finally height biannual measurements. Even if each record involves discrete values, these values can be interpreted as part of the realization of a functional random variable, that is we can consider them as evaluations of an *height function*. We can then assume that our dataset is composed by 93 functional observations.

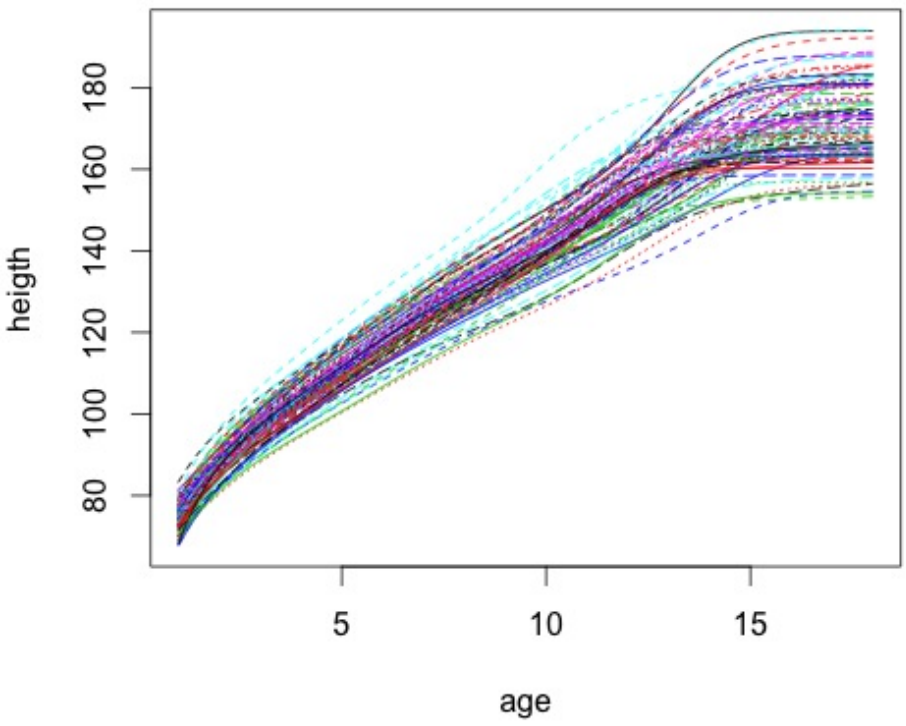

Figure 2.1: Example of functional data: men and women heights in Berkeley study.

2.1.1 Phase and amplitude variability

First of all, let us give the definition of a functional data.

Definition 1. *A random variable* X *is a functional variable if it takes values in a infinite dimensional space F (normed functional space).*

Functional data can be characterized by two types of variability: *phase variability* and *amplitude variability*. In order to understand these two concepts, let us take a look at the following examples.

In figure 2.2 we present two simple cases where 5 functions have been simulated. The black curve, as stated in the legend, is the function $y = sin(x)$. In the first case the other four functions have been generated by changing their amplitude, that is $y = a * sin(x)$ with a taking the following values: $0.8, 0.9, 1.1, 1.2$. In the second case, instead, the other four functions have been generated by changing their phase, that is $y = sin(x + a)$ with a taking the following values: $-0.5, -0.25, 0.25, 0.5$.

We now understand what we mean by phase and amplitude variability. A difference in the evaluations of functions can be interpreted as amplitude variability and, more important for a correct functional analysis, a difference in the occurrence of a particular characteristic of the functions can assessed as phase variability.

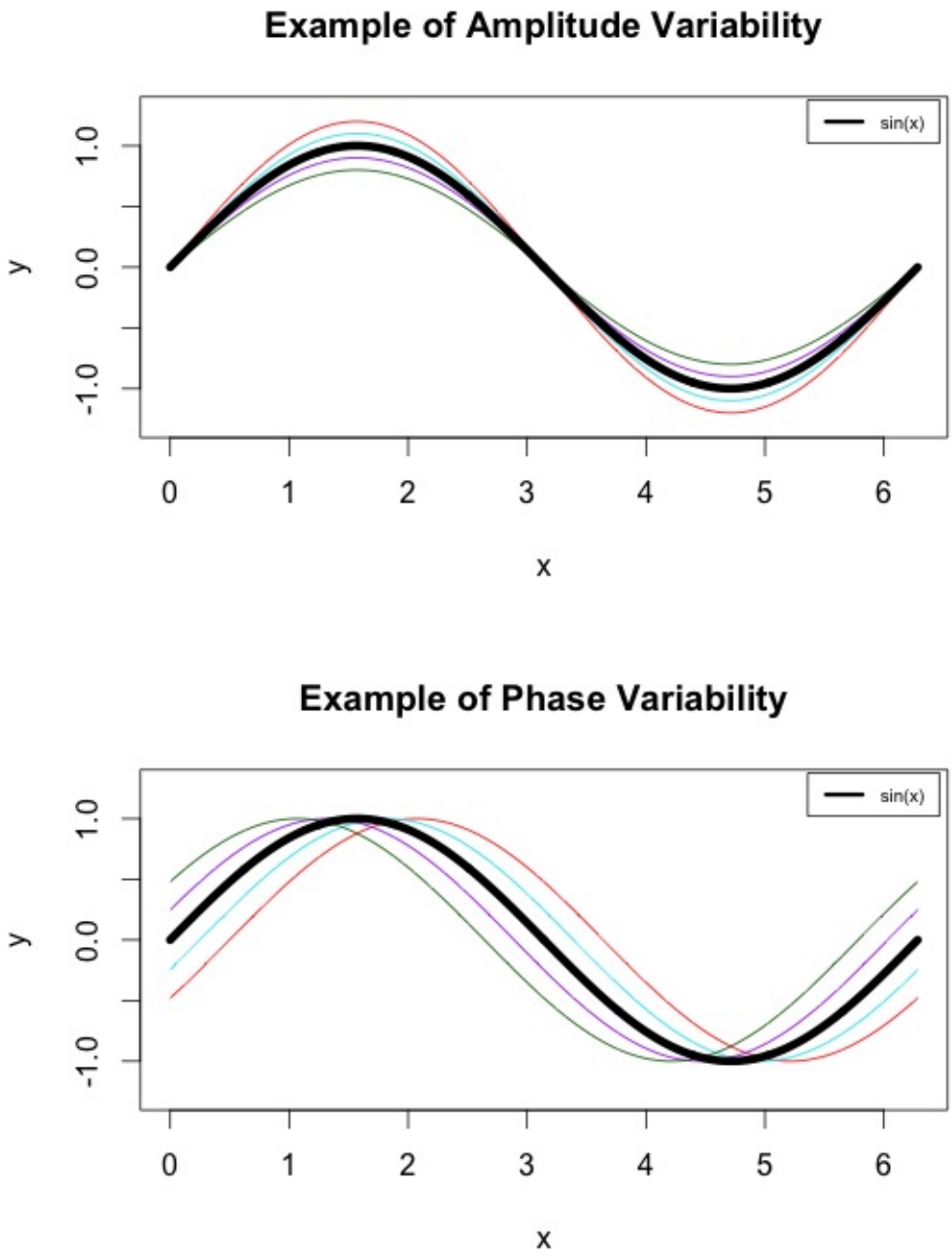

Figure 2.2: Example of amplitude and phase variabilities.

2.2 Registration of functional data

2.2.1 Problem definition

In functional dataset, most of the time, both amplitude and phase variability are present. Before starting any analysis, in order to gain in significance, it is necessary to decouple these two variabilities. We call this procedure *Alignment* or *Registration* of functional data.

More precisely, to align a set of functions means that we have to find the transformation of function independent variable (let us call it t) which allows us to remove phase variability from data.

Formally, if we consider n functions

$$x_1(t), ..., x_n(t), \quad x_i : \mathbb{R} \to \mathbb{R} \; \forall i \in \{1, ..., n\}$$

to be able to align them we need to find n strictly monotonous functions

$$h_1(t), ..., h_n(t), \quad h_i : \mathbb{R} \to \mathbb{R} \; \forall i \in \{1, ..., n\}$$

such that $x_1(h_1(t))$,...,$x_n(h_n(t))$ will have no phase variability. The functions h_1,...,h_n are known as *Warping Functions*.

2.2.2 State of art

Even if FDA represents a "green" way to look at data, many approaches to the problem of registration have been already developed. Here we present three different approaches that solve the problem, each one with his own ability to capture hidden characteristics of functions.

The first one, called *Landmark registration*, is able to line up some important features of our functional data, imposing a nonlinear transformation to the abscissa and allowing to align functions around a key point shared by all of them; obviously the point is known a priori. The second one, known as *Continuous registration*, is a more general method, with the advantage of considering, in the registration process, the entire domain in a continuous way.

Landmark registration

The Landmark registration is for sure the simplest method to align functional data. Despite of his simplicity, it requires a certain knowledge of the phenomenon standing beyond data.

In general, this method allows to align functions in a given point of abscissa, but one has to be sure about the trend of the functions (or of their derivatives in case we are dealing with them) in that point. Substantially, let us suppose that we know where the point with this shared property is; we name it *Landmark point*. Given that, all functions share this characteristic, say a maximum (it can also be a minimum, ora a zero etc..). It is then reasonable to ask all functions to have this maximum in the same abscissa point. In other words, one should align functions so that all of them will show this same characteristic in the Landmark abscissa point. It will be then possible to do a proper analysis.

It is fundamental to specify that, if one wants to choose this kind of approach, it is necessary to know perfectly, without any doubts, what happens to all functions in the *Landmark point*. That is, the choice of the *Landmark point* play a key role in fact we are transforming data and obliging all functions to have that kind of characteristic in that given point. If it is not the case , we will be misled in next analysis by this false hypothesis.

Of course, this reasoning is also valuable when we can find more than one *Landmark point*.

Formally, let us consider n functions defined on the interval [0,T]:

$$x_1(t), ..., x_n(t), \quad x_i : [0, T] \to \mathbb{R} \ \ \forall i \in \{1, ..., n\}$$

Supposing we can find $k \geq 1$ *landmark* points. Let $\tau_1(x_i), ..., \tau_k(x_i)$ be the function landmarks. For each landmark let us compute the mean, i.e., $\tau_j = \mathbb{E}[\tau_j(x_i)]$. We can now estimate the warping functions of the i-th curves simply interpolating these points (and of course the points (0,0) and (T,T)).

We can do this using linear interpolation or more complex methods (*Hermite splines* can also be used, in order to maintain the monotony of warping functions).

In order to better understand this method, we give a simple example, based on functions of figure 2.3. The 5 functions have been simulated modifying a gaussian function so that each one has a unique maximum. Following the procedure just described, we align them so that all curves present the maximum in the same abscissa point. In figure 2.3 we show the original functions and the aligned one using Landmark registration. We can also see the warping functions estimated using a simple linear interpolation.

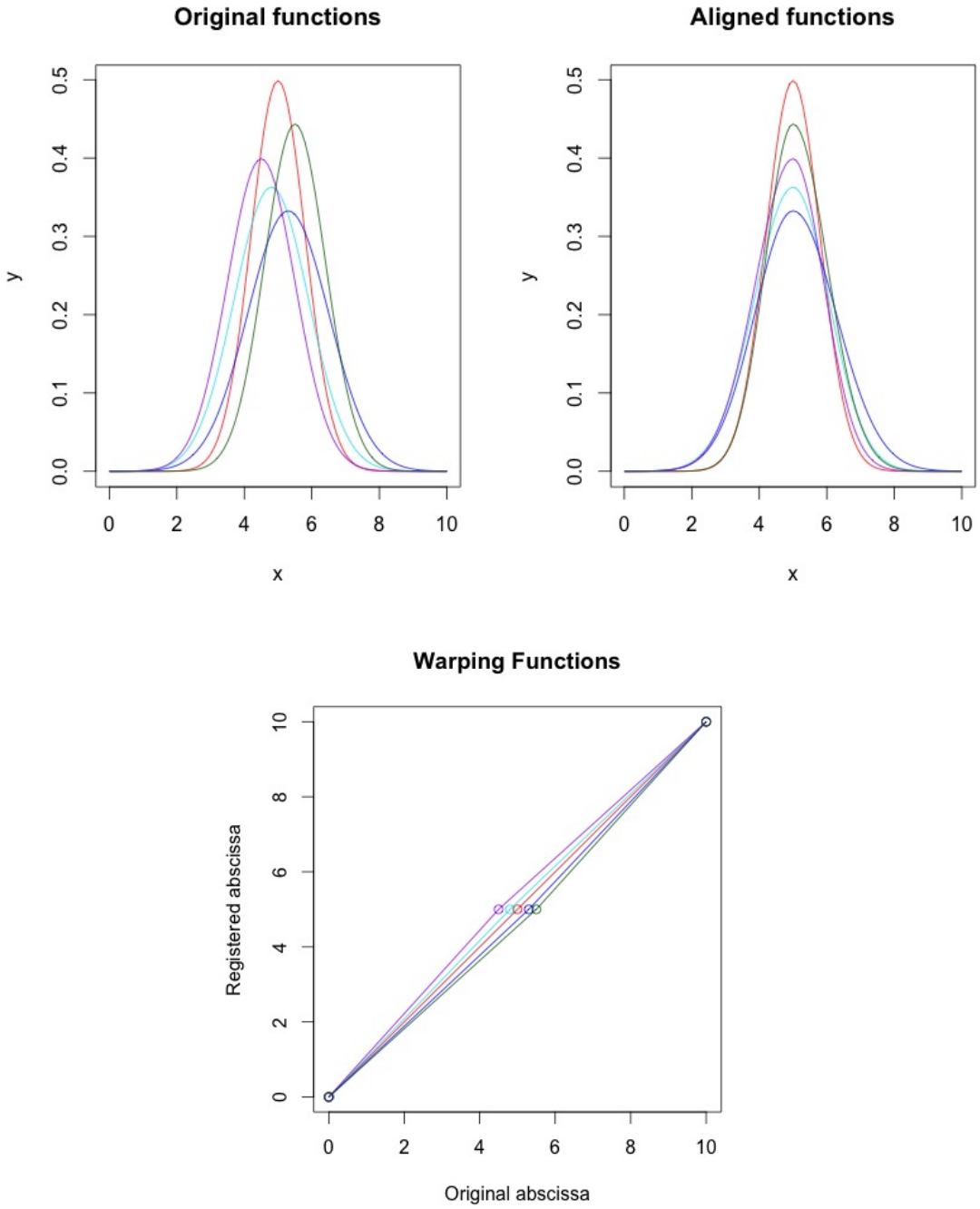

Figure 2.3: Example of Landmark Registration: simulated data. Left panel: original functions; right panel: registered functions. Bottom: warping functions

Continuous registration

Continuous registration represents for sure a more advanced registration methods than the previous one because, as stated before, it takes in account the entire domaine during the registration process.

Suppose we have two functions $x_1(t)$ e $x_2(t)$ and we want to align $x_2(t)$ to $x_1(t)$ (in this case we will refer to $x_1(t)$ as the *center* or *template*). In order to choose the best warping function we will define a similarity/dissimilarity measure which will tell us how much the two functions are similar/dissimilar. The best warping function $h(t)$ will be the one minimizing/maximizing the similarity/dissimilarity measure between $x_1(t)$ e $x_2(h(t))$.

Two natural questions arise:

- which kind of warping functions are we allowed to choose to transform $x_2(t)$?
- which kind of similarity/dissimilarity measure between functions should we consider ?

During the last years, thanks to the works of Sangalli et al. (2010), Vantini (2012) and Srivastava et al. (2011), the statistics community has agreed to the fact that the answer to the first question is strongly linked to the answer to the second question.

Hence, to define a continuous registration technique we need to define two "objects": the similarity/dissimilarity measure and the warping function space. They must have some fundamental properties, which will be presented later. It is important to highlight that choosing these two elements will be a determi-

nant aspect of our analysis. Above all, they will define the meaning of phase and amplitude variability with which we will deal. In general the specification of a space of possible warping functions will define a precise concept of phase variability. The amplitude one, instead, will be defined by choosing a similarity/dissimilarity measure between functions.

In the next three paragraphs we will describe three continuous registration techniques. First we present the one described in Ramsay et Silverman (2005): known as the *Minimum eigenvalue method*, this approach gives a general prospective on how one can deal with the problem of registering one function to another. Then we will describe the so-called *K-Mean Alignment* method presented in Sangalli et al. (2010). Finally, in the last paragraph of this chapter, we will show a technique developed in Srivastava et al. (2011) involving a particular metric, the *Fisher-Rao* metric.

2.3 Minimum eigenvalue method

Ramsay and Silverman solve the problem choosing the following type of warping functions:

$$h(t) = C_0 + C_1 \int_0^t \exp W(u) du$$

where W is a function $W : \mathbb{R} \to \mathbb{R}$ and C_0 e C_1 are fixed assuring that $h(t) = t$ at the lower and upper interval limits of the domaine. Let us suppose we have the evaluations of the two functions $x_1(t)$ e $x_2(t)$ over a fine mesh of n values of t $(t_1, t_2, ..., t_n)$. We consider the pairs of values $(x_1(t_i), x_2(t_i))$ with $i \in 1, 2, ..., n$. We can state that, if the two functions are such that the plot of these couples is a straight line, then they are equal, hence they do not need to be aligned. So the natural technique to analyze their differences can be the Principal Compo-

nent Analysis: if the plot of previous couples is actually a straight line, then the PCA will held to only one positive eigenvalue, that is, the second eigenvalue will tell us how much the two functions have to be registered.

Given that we are dealing with functions, let us consider the functional analogue of the cross-product matrix $\mathbf{X}'\mathbf{X}$:

$$T(x_1, x_2) = \begin{bmatrix} \int x_1(t)^2 dt & \int x_1(t)x_2(t)dt \\ \int x_1(t)x_2(t)dt & \int x_2(t)^2 dt \end{bmatrix}$$

Naturally we end up defining the criterion for stating how much the two functions need to be registered (their "distance"):

$$MINEIG(h) = \mu_2[T(x_1, x_2(h))]$$

where the function μ_2 represents the size of the second eigenvalue. If $MINEIG(h) = 0$ then the two functions are aligned and the corresponding h will be the right warping function.

2.4 K-mean Alignment method

The second method we are going to present is the one developed in Sangalli et al. (2010). It is important to highlight that this method, like the next one (based on the *Fisher-Rao* metric), also represents the first attempt to create a solid framework for the registration problem.

The *K-mean Alignment* method is able to jointly cluster and align functional data. Even if, as we will see in the next, it only enables affine transformations of the abscissa, it allows to differentiate functions and find hidden groups inside the functional dataset.

As said before, to register one function to another one is necessary to define a couple of elements:

- ρ: measure to quantify how "far" the two functions are
- W: warping function space

The elements of the couple (ρ, W) have to be chosen very carefully; in fact they need to satisfy some fundamental coherence properties.

- Coherence properties of the couple (ρ, W)

The measure ρ has to satisfy the following properties:

- $\rho \leq 1$ (boundedness),
- $\rho(x, x) = 1,\ \forall x \in X$ (reflexivity),
- $\rho(x_1, x_2) = \rho(x_2, x_1),\ \forall x_1, x_2 \in X$ (simmetry),
- $[\rho(x_1, x_2) = 1 \wedge \rho(x_2, x_3) = 1] \implies \rho(x_1, x_3) = 1,\ \forall x_1, x_2, x_3 \in X$ (transitivity).

The space of warping functions W has to be:

- a complex vector space,
- a group with respect to the operation of composition $\circ$.

The couple (ρ, W) has to satisfy the following property:

- $\rho(x_1, x_2) = \rho(x_1 \circ h, x_2 \circ h),\ \forall h \in W,\ \forall x_1, x_2 \in X.$

We refer to the last property as the *consistency property*. It is maybe the most important one, in fact it states that, given any couple of elements $x_1,\ x_2 \in X$ and an element $h \in W$, the distance between x_1 and x_2 has to be invariant under the composition of x_1 and x_2 with h.

We understand that at the beginning of a registration process we have to deal with the choice of two elements, ρ and W. There are many couples which respect these properties. Let us introduce the first one, which is the couple we will use to analyze the spike train dataset in the next chapter.

Consider x_1 e $x_2 \in L^2(\mathbb{R}, \mathbb{R})$. We can define then a measure linked to the angle θ between the two functions in this space:

$$\cos(\theta) = \frac{\langle x_1, x_2 \rangle}{\|x_1\| \|x_2\|}$$

The inner product and the norm have to be considered on L^2 :

$$\langle x_1, x_2 \rangle = \int_{\mathbb{R}} x_1(t) x_2(t) dt$$

$$\|x_1\| = \int_{\mathbb{R}} x_1(t)^2 dt$$

Let us consider the functional space

$$X = \{x(t) : \mathbb{R} \to \mathbb{R} \mid x \in L^2, x' \in L^2, x' \neq 0\}$$

We can then define a "distance" between $x_1, x_2 \in X$ considering the cosine of the angle between the two function first derivatives, that is:

$$\rho(\cdot, \cdot) : X \times X \to \mathbb{R} \quad \rho(x_1, x_2) = \frac{\int_{\mathbb{R}} {x_1}'(t) {x_2}'(t)\, dt}{\sqrt{\int_{\mathbb{R}} {x_1}'(t)^2\, dt} \sqrt{\int_{\mathbb{R}} {x_2}'(t)^2\, dt}} \tag{2.1}$$

This measure is also an index, in fact:

$$-1 \leq \rho(x_1, x_2) \leq 1 \quad \forall x_1, x_2 \in X$$

Following this definition we can state that two functions x_1, x_2 are equal iif $\rho(x_1, x_2) = 1$. This means that x_1, x_2 are equal if they differ by a multiplicative or additive factor, i.e., for dilation or translation along the ordinate axis:

$$\rho(x_1, x_2) = 1 \Leftrightarrow \exists m \in \mathbb{R}^+,\ q \in \mathbb{R} : x_1(t) = m * x_2(t) + q$$

If the functions we are considering are multi-dimensional, i.e., $x_1, x_2 \in \mathbb{R}^d$:

$$X = \{x(t) : \mathbb{R} \to \mathbb{R}^d \mid x \in L^2, c' \in L^2, c' \neq 0\}$$

we will define the similarity index between x_1 and x_2 as the mean among the d components:

$$\rho(x_1, x_2) = \frac{1}{d} \sum_{p=1}^{d} \frac{\int_{\mathbb{R}} {x_1}_p'(t) {x_2}_p'(t)\, dt}{\sqrt{\int_{\mathbb{R}} {x_1}_p'(t)^2\, dt} \sqrt{\int_{\mathbb{R}} {x_2}_p'(t)^2\, dt}}$$

where x_{ip} is the p-th component of the function x_i.

Finally, we have to choose the space of warping functions. This will be the space of *affine* transformations:

$$W = \{h : h(s) = ms + q \text{ with } m \in \mathbb{R}^+, q \in \mathbb{R}\} \tag{2.2}$$

We can now have a clear definition of phase variability, which involves, in this particolar case, only translations and dilations of the independent variable t. The warping function that aligns x_2 to x_1 will be the function h^* such that:

$$h^* = \arg\max_{h \in W} \rho(x_1, x_2 \circ h)$$

2.4.1 K-mean Alignment algorithm

Consider the problem of clustering and aligning N functions $x_1(t), ..., x_N(t) \in X$ where $X = \{x(t) : \mathbb{R} \to \mathbb{R}^d \mid x \in L^2, x' \in L^2, x' \neq 0\}$ with respect to k function centers (or templates) $\varphi = \{\varphi_1 ..., \varphi_k\}$ and also consider the similarity index and the space of warping functions just defined.

First some useful definitions.

Definition 2. *We define* domain of attraction *of* φ_j*:*

$$\Delta_j(\varphi) = \{x \in X \ : \ \sup_{h \in W} \rho(\varphi_j, x \circ h) \geq \sup_{h \in W} \rho(\varphi_r, x \circ h),\ \forall r \neq j\} \quad j = 1, ..., k$$

Definition 3. *We define the* labelling function *as follows:*

$$\lambda(\varphi, x) = \min\{r \ : \ x \in \Delta_r(\varphi)\}$$

Hence $\lambda(\varphi, x) = j$ implies that the similarity index obtained by aligning x to φ_j is larger or equal to the similarity index obtained aligning x to any other center. So $\varphi_{\lambda(\varphi,x)}$ will be the center the function x can be best aligned to.

The *K-Mean Alignment* is an iterative method. We have to initialize it choosing k centers. Let us suppose we already run $(q-1)$ iterations of the algorithm. We have now k centers computed at the iteration $(q-1)$:

$$\varphi_{[q-1]} = \varphi_{1[q-1]}, ..., \varphi_{k[q-1]}$$

We also have the registered curves until the last iteration:

$$x_{1[q-1]}(t), ..., x_{N[q-1]}(t)$$

and each function has been assigned to a certain cluster.

The q-th iteration of the algorithm is composed by three steps:

Step 1) *Template identification step*

For all the k clusters we compute the new center using the functions we assigned to that cluster at the last iteration. In order to compute this center we use the Frechét mean:

$$\varphi_{j[q]} = \arg\max_{\varphi \in X} \sum_{i:\lambda_i = j} \rho(\varphi, x_{i[q-1]}) \tag{2.3}$$

We have then the k centers $\varphi_{[q]} = \varphi_{1[q]}, ..., \varphi_{k[q]}$.

Step 2) *Assignment and alignment step*

The functions $x_{1[q-1]}(t), ..., x_{N[q-1]}(t)$ are clustered and aligned to the k centers $\varphi_{1[q]}, ..., \varphi_{k[q]}$. Now for $i = 1, ..., N$:

- the function x_i is assigned to the cluster $\lambda(\varphi_{[q]}, x_i)$,
- the function x_i is aligned to $\varphi_{\lambda(\varphi_{[q]},x)}$, i.e., to the center of the group it is assigned to. In order to have the warping function h^* responsible for this alignment we compute the following optimization:

$$h^* = \arg\max_{h \in W} \rho(\varphi_{\lambda(\varphi_{[q]},x)}, x_i \circ h)$$

Step 3) *Normalization step*

For each cluster we normalize the warping functions of the functions assigned to that cluster so that the mean of the applied warping functions will be the identity function.

At each iteration the algorithm computes the similarity indexes between each function and the center the function is assigned to. The method stops when all functions present an increment of similarity lower than a fixed tolerance. The method stops also if a maximum number of iterations, which can be specified in input, is achieved.

We implemented this method into an R-package called *fdakma*, available on CRAN. In chapter 4 we will present in a detailed way all the parameters which need to be provided for a successful running of the algorithm.

2.5 Other registration approaches

A said before, there are many couples (ρ, W) which satisfy the previous properties. The one presented in the *K-Mean Alignment* is among them. In the

following table we resume the principal ones. All couples but the last one mentioned in this table have been implemented in the package *fdakma*, hence user can choose a couple "measure"-"warping functions" respecting the properties. Note that D is the intersection of the domains of $x_1(t)$ and $x_2(t)$ and all norms and inner product are computed over D.

Metric/Semimetric	Warping functions
$\frac{\langle x_1, x_2 \rangle_{L^2}}{\|x_1\|_{L^2}\|x_2\|_{L^2}}$	y = m*t + q
$\frac{\langle x_1', x_2' \rangle_{L^2}}{\|x_1'\|_{L^2}\|x_2'\|_{L^2}}$	y = m*t + q
$\frac{\|x_1 - x_2\|_{L^2}}{\|D\|}$	y = t + q
$\frac{\|x_1' - x_2'\|_{L^2}}{\|D\|}$	y = t + q
$\frac{\|(x_1 - \bar{x}_1) - (x_2 - \bar{x}_2)\|_{L^2}}{\|D\|}$	y = t + q
$\frac{\|(x_1' - \bar{x}_1') - (x_2' - \bar{x}_2')\|_{L^2}}{\|D\|}$	y= t + q
$\|\text{sign}(x_1')\sqrt{\|x_1'\|_{L^2}} - \text{sign}(x_2')\sqrt{\|x_2'\|}_{L^2}\|_{L^2}$	y = h(t) h(t) diffeomorphism

Table 2.1: Metrics/semimetrics and corresponding warping functions

On the left column of table 2.1 we listed the metric (or semi metrics) considered and on the right column we show the class of warping functions that can be chosen to align data.

2.5.1 Fisher-Rao method

In this paragraph we will present another registration technique based on a very particular metric, the metric of *Fisher-Rao*. This method has been developed in Srivastava et al. (2011) and here we will try to resume the most impor-

tant features of this approach.

Consider the following function space:

$$F = \left\{ f : [0,1] \to \mathbb{R},\ f \text{ absolutely continuous} \right\}$$

Consider also the following space of warping functions:

$$\Gamma = \left\{ \gamma : [0,1] \to [0,1],\ \gamma(0) = 0,\ \gamma(1) = 1,\ \gamma \text{ mon. incr. },\ \gamma \text{ diffeomorphism} \right\}$$

Let us define a new distance on F using the Fisher-Rao metric. This metric is defined on $T_f(F)$, the space tangent to the variety F.

Definition 4. *For all $f \in F$ and $v_1, v_2 \in T_f(F)$, the Fisher-Rao metric is defined as:*

$$\langle\langle \cdot, \cdot \rangle\rangle_f : T_f(F) \times T_f(F) \to \mathbb{R} \qquad \langle\langle v_1, v_2 \rangle\rangle_f = \frac{1}{4} \int_0^1 \frac{\dot{v}_1(t)\dot{v}_2(t)}{|\dot{f}(t)|} dt \tag{2.4}$$

The Fisher-Rao distance between two functions f e g is defined as the geodetic distance between f e g on the variety F. To be able to find this distance we should look for the geodetic path which connect f e g with respect to the Fisher-Rao metric. The minimization of this quantity is therefore quite challenging. This is why a new representation of the functions has been introduced.

Definition 5. *The* square-root velocity function *(SRVF) q of a function $f \in F$ is defined in the following way. Let Q be a continuous map:*

$$Q : \mathbb{R} \to \mathbb{R} \qquad Q(x) = \begin{cases} x/\sqrt{\|x|} & \textit{se } |x| \neq 0 \\ 0 & \textit{otherwise} \end{cases}$$

We can write:

$$q : [0,1] \to \mathbb{R} \qquad q(t) = Q(\dot{f}(t)) = \frac{\dot{f}(t)}{\sqrt{|\dot{f}(t)|}} = sign(\dot{f}(t))\sqrt{|\dot{f}(t)|} \tag{2.5}$$

It can be proved that, if f is absolutely continuous, then $q \in L^2$.
Moreover, for all functions $q \in L^2$, there exist a function $f \in F$ whose q is its SRVF. This can be derived from formula 2.5.

$$q(t)|q(t)| = \text{sign}(\dot{f}(t))|\dot{f}(t)| = \dot{f}(t). \tag{2.6}$$

Integrating:

$$f(t) = f(0) + \int_0^t q(s)|q(s)|ds.$$

We can then state that the representation $f \iff (f(0), q)$ is biunique.

In Srivastava et al. (2011) a fundamental property of this metric is shown: under the SRVF representation, the Fisher-Rao metric becomes the L^2 norm.
In L^2 space the geodetic distance is simply the L^2 norm. So we understand that

$$d_{FR}(f_1, f_2) = \|q_1 - q_2\|_{L^2} \ .$$

We want to study the action of the group Γ on the space F. The action of $\gamma \in \Gamma$ on $f \in F$ is represented by the composition of the two functions:

$$(\cdot, \cdot) : F \times \Gamma \to F, \quad (f, \gamma) \mapsto f \circ \gamma.$$

In order to find the corresponding action on the SRVF space we compute the SRVF $\tilde{q}$ of $f \circ \gamma$:

$$\tilde{q}(t) = \frac{\frac{d}{dt}(f \circ \gamma)(t)}{\sqrt{|\frac{d}{dt}(f \circ \gamma)(t)|}} = \frac{(\dot{f} \circ \gamma)(t)\ \dot{\gamma}(t)}{\sqrt{|(\dot{f} \circ \gamma)(t)\ \dot{\gamma}(t)|}} = \frac{(\dot{f} \circ \gamma)(t)}{\sqrt{|(\dot{f} \circ \gamma)(t)|}}\sqrt{\dot{\gamma}(t)} = (q \circ \gamma)(t)\sqrt{\dot{\gamma}(t)}.$$

We can now express the action of $\gamma \in \Gamma$ on the SRVF q:

$$(\cdot, \cdot) : L^2 \times \Gamma \to L^2, \quad (q, \gamma) \mapsto (q \circ \gamma)\sqrt{\dot{\gamma}}. \tag{2.7}$$

Definition 6. *We define the* orbit *as*

$$[\,f\,] = \Big\{(f \circ \gamma) \mid \gamma \in \Gamma\Big\}.$$

In F we have

$$\|f_1 - f_2\|_{L^2} \neq \|f_1 \circ \gamma - f_2 \circ \gamma\|_{L^2}.$$

In SRVF space L^2:

$$\|q_1 - q_2\|_{L^2} = \|(q_1, \gamma) - (q_2, \gamma)\|_{L^2}. \tag{2.8}$$

If we apply a warping function to a SRVF we are in the same orbit. But the important thing is that the distance does not change and the orbits are said to be parallel.

We are able now to define the meanings of phase and amplitude variabilities.

Definition 7. *We define the* phase variability γ_{21} *of* f_1 *with respect to* f_2 *as*

$$\gamma_{21} = \arg\inf_{\gamma \in \Gamma} \|q_1 - (q_2, \gamma)\|.$$

Definition 8. *We define the* amplitude variability d_a *between two functions:*

$$d_a(f_1, f_2) = \inf_{\gamma \in \Gamma} \|q_1 - (q_2, \gamma)\|.$$

The amplitude variability is characterized by the following properties:

- $d_a(f_1, f_2) = d_a(f_2, f_1)$ (simmetry)
- $d_a(f_1, f_3) \leq d_a(f_1, f_2) + d_a(f_2, f_3)$ (triangle inequality)
- $d_a(f_1, f_2) = 0$ if $f_2 \in [f_1]$ (if two functions are in the same orbit, their distance is null)
- $d_a(f_1 \circ \gamma_1, f_2 \circ \gamma_2) = d_a(f_1, f_2)$ with $\gamma_1,\ \gamma_2 \in \Gamma$ (invariance by composition with a warping function)

Thanks to the first three properties we deduce that d_a is a semi metric.

The method aims to compute a center for all the functions. Hence the last step is to find the right warping function able to align them to that center. This last

step is achieved by performing the following optimization:

$$\gamma_i^* = \arg\min_{\gamma \in \Gamma} \|\mu - (q_i \circ \gamma)\sqrt{\dot{\gamma}}\|_{L^2} \ .$$

As it should be clear by now, the *Fisher-Rao* metric has an important "quality". The warping function space play a fundamental role in registration procedure, in fact it determines the type of transformation we want to be possible for the abscissa. In the *K-Mean Alignment* method this space was the *affine* transformation space. In the *Fisher-Rao* method we are able to extend the space of warping functions, allowing almost all kind of transformations, with the only constraint to be a diffeomorphism. This leads to a very large class of transformations and hence, generally, to a better result in aligning functional data.

Although it currently represents the most powerful registration method to align functional data, this enormous quality also represents its biggest limit. With this method we are capable of aligning quite everything, thanks to its very large class of warping functions, but it is important to highlight that it is also possible that two curves, in this framework, will be aligned even if they do not require to be.

Another weak point of this method is represented by the difficulty in interpreting the warping functions. It is a very important aspect, because they store all the informations about the misalignment of data. In the *K-Mean Alignment* method we can simply understand what an affine transformation of abscissa could mean in terms of the physics of the phenomenon. Instead in the *Fisher-Rao* approach it is quite difficult to link a characteristic of the warping functions to something physically meaningful for our functional data.

Chapter 3

Spike Trains Data: K-mean Alignment application

In this chapter we are going to apply the *K-Mean Alignment* algorithm to a particular dataset, provided by Wei Wu (Department of Statistics, Florida State University) for the contest of the Ohio workshop mentioned in the introduction.

3.1 Dataset description

Human and animal brain processes neural informations via spikes (action potentials) from neurons. These spikes are often called the *language of the brain*. So if we would like to understand how the brain works, we should understand how neurons exchange messages among each other, how they decide to send a particular message instead of an other one. This work aims to help neuroscientists in this direction, conscious that we are actually far from the ultimate understanding of this incredibly complex organ called Brain. We will try to extract from data any information who can tell us something about the functioning of the brain when, as we will see, a monkey does a particular job using his hands.

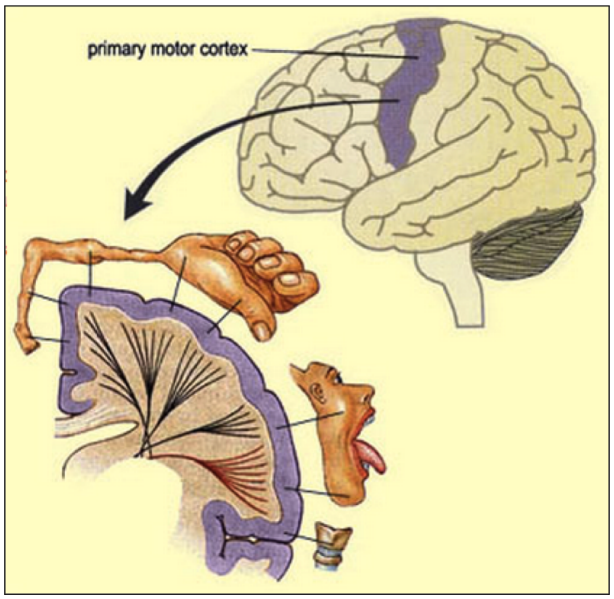

Figure 3.1: Connection between brain and parts of the body: the primary motor cortex is dedicated to the control of arms movement.

In the experiment from which the dataset arises, a rhesus monkey, using his hands, has to follow a light which illuminates buttons located right in front of him. This light follows four precise paths, as described in figure 3.2.

The first path (form button 1 until reached again button 1, indeed closing the cycle clockwise) has been done 60 times by the monkey and the same for the other paths. The electrode implanted in the arm and in the primary motor cortex (the part of the brain dedicated to the control of arms movement, as explained in figure 3.1) aimed to register his neuronal activities (i.e., his action potentials, spikes activity) during the experiment. For a better visualization of the experiment, see figure 3.3.

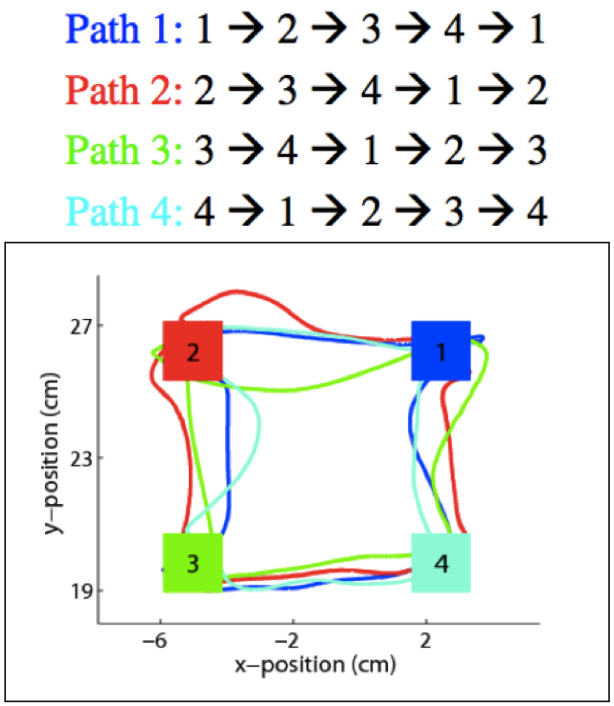

Figure 3.2: The four paths the monkey had to follow

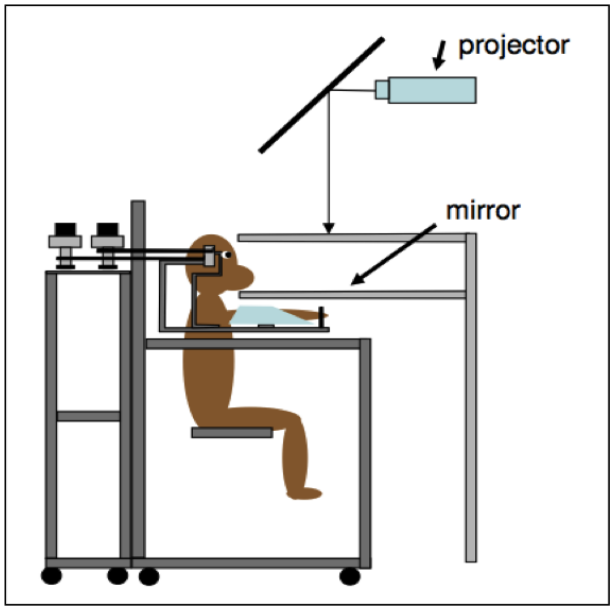

Figure 3.3: Visual representation of the experiment

As we can see in figure 3.4, in each neuron all spike waveforms have nearly the

same shape, and the information is encoded in the firing time of each spike.

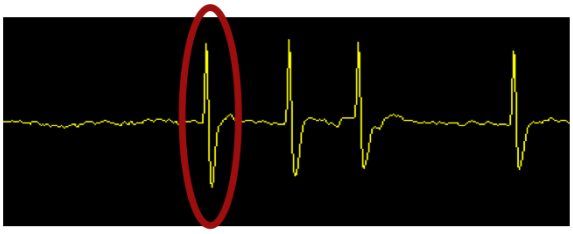

Figure 3.4: Single recorded spike

So at first the dataset was a collection of 1's and 0's indicating that in that moment a spike had been fired or not. In order to analyze spike trains as functional data, they have been smoothed using a gaussian kernel with width σ = 50 ms. This turned the data in an intensity function instead of functions with only 1's and 0's.

So practically, for the analysis, the dataset will consist in a matrix in which each row represents the evaluation of a function (i.e., a statistic unity, a trial in the experiment). This function represents, as said before, the intensity of neuronal activity during one of the four paths. An example of these curves is given in figure 3.5. In this figures we can see a trial, so a function, for each path.

Let us see now the complete dataset. In figure 3.6 we can see all original curves (the path colors of figure 3.2 have been used as convention).

What we can immediately see is that a data registration seems to be quite challenging in this case. The curves are very different from each other and this representation does not help for an initial understanding. Let us draw the curves by paths to allow a better visual primary study (figures 3.7, 3.8).

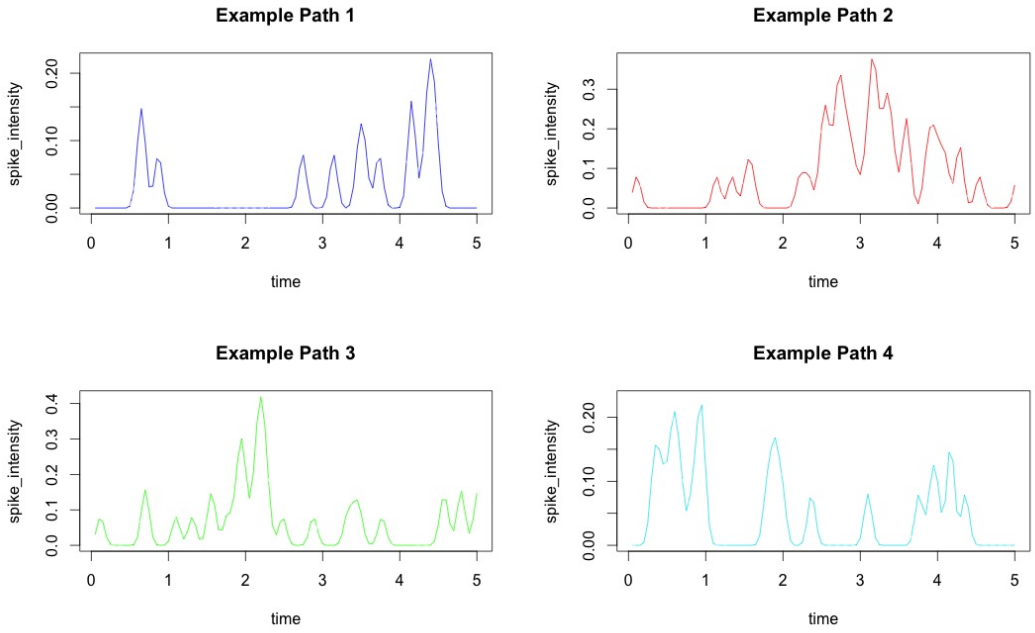

Figure 3.5: Examples of the four paths

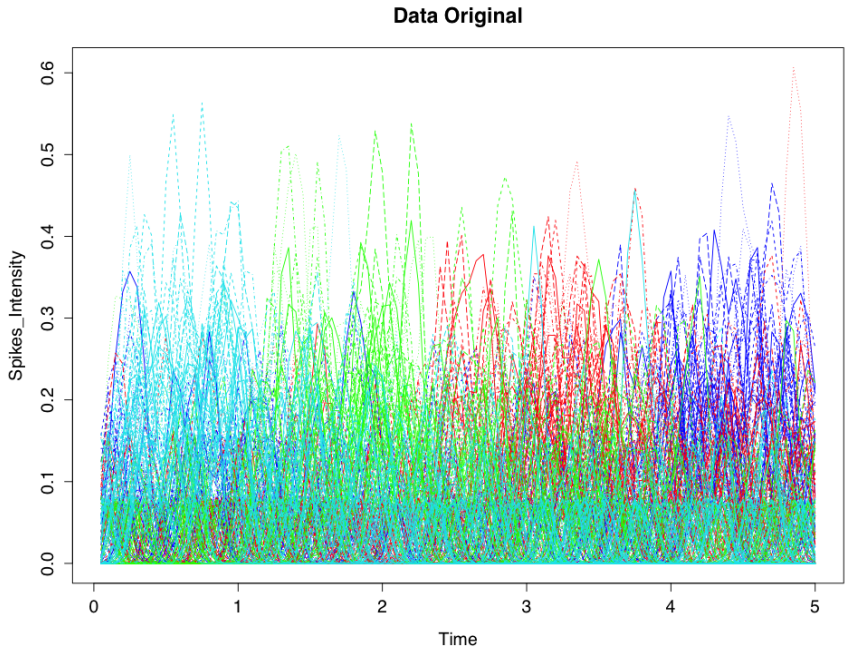

Figure 3.6: All 240 original curves

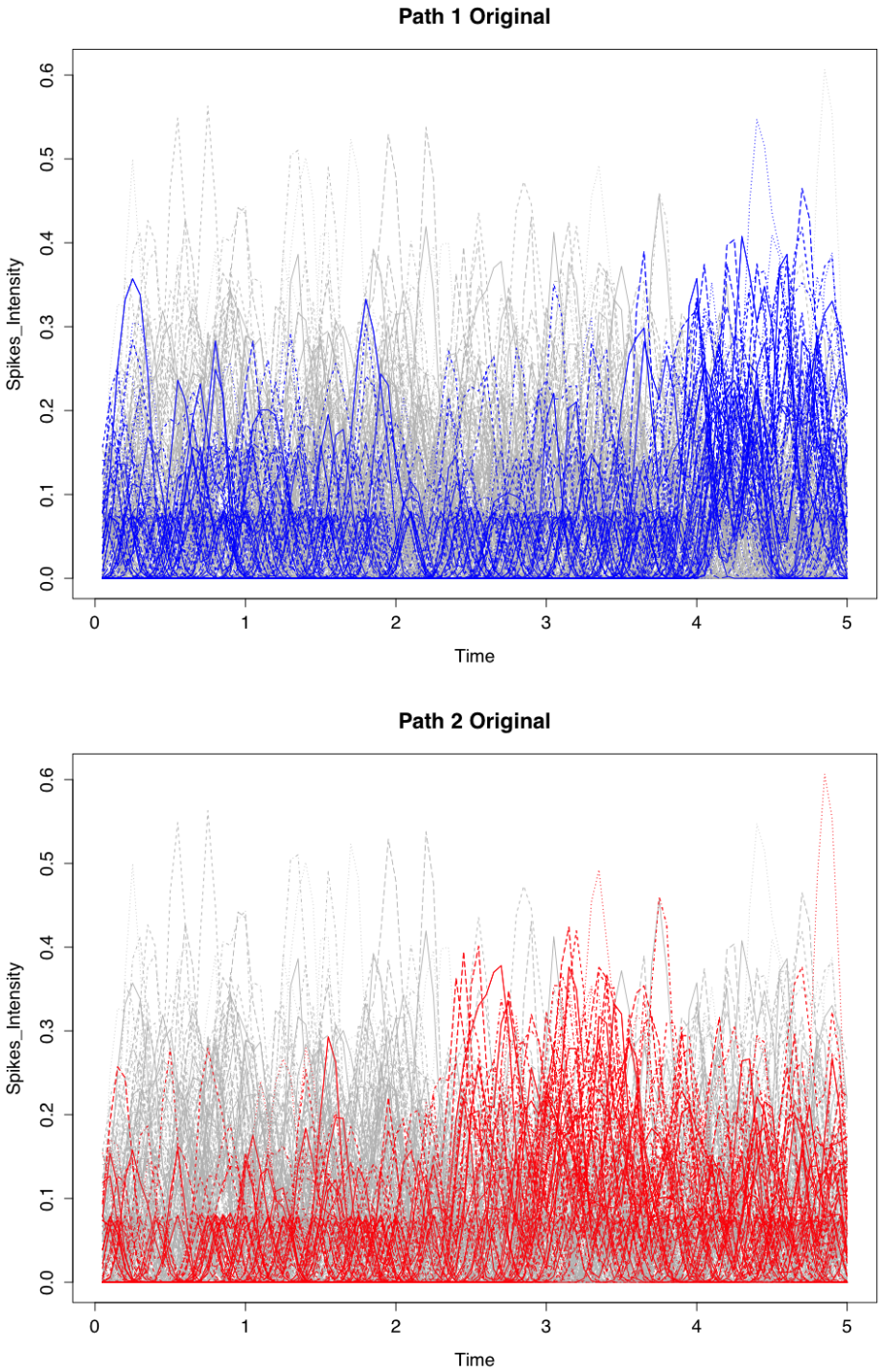

Figure 3.7: Original curves divided by paths: path 1,2

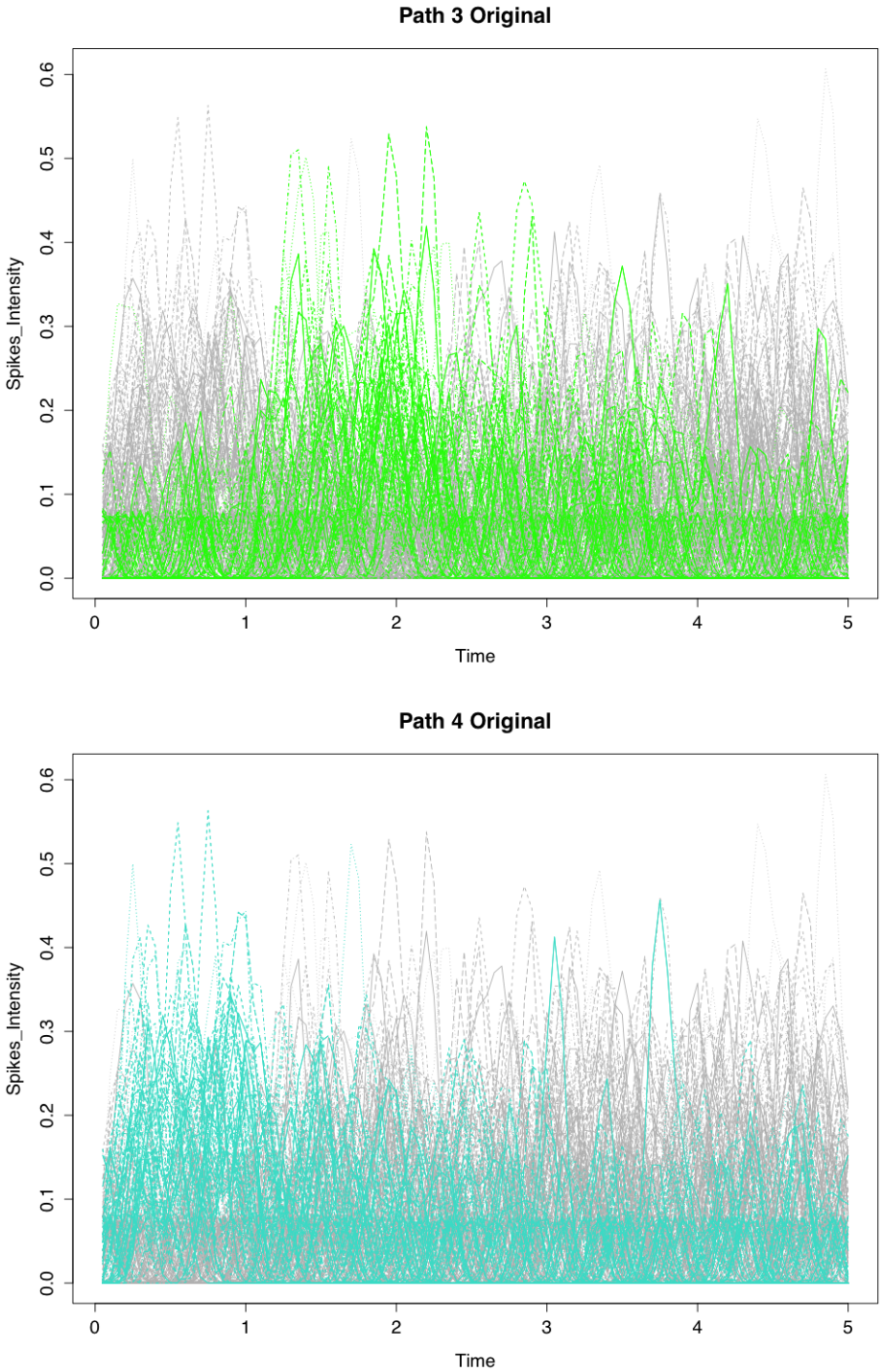

Figure 3.8: Original curves divided by paths: path 3,4

3.2 First approach: non-periodic data

To start the analysis we use an useful function, called *kma.compare*, implemented in the *R* package *fdakma*. This function allows the user to run the *kma* function for several numbers of clusters and types of warping. We run this function with *n.clust* set to *c(1,2,3,4,5)* and *warping.method* set to *c('NOalignment', 'shift', 'dilation', 'affine')*. The variable which controls the similarity measure is *similarity.method* and for the analysis of this chapter we will set it to *'d0.pearson'* which is the first measure listed in table 2.1 in chapter 2.

We report the output of the *kma.compare* function in figure 3.9, that is a graphic containing the means of the similarity indexes of all the functions of the dataset in all cases, i.e., for all elements of *n.clust* and *warping.method*.

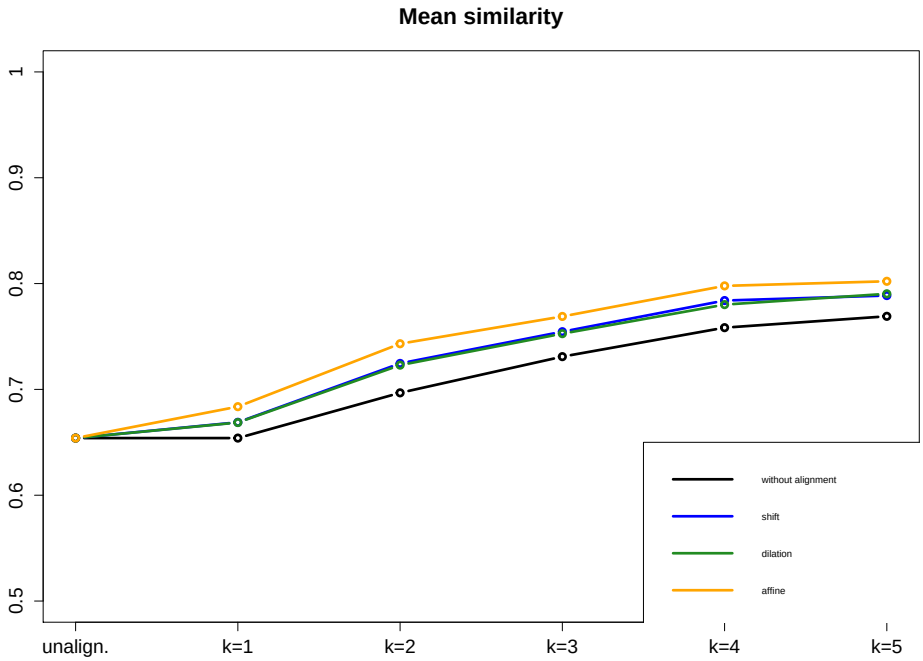

Figure 3.9: Mean similarity indexes with different numbers of clusters. Results of kma.compare function

We can see that in quite all kind of warping method, the elbow is present for $k = 4$. This means that the number of cluster suggested for our data is 4. For this reason in the next we will draw and analyze the results for $k = 4$ in two cases:

- k-mean clustering, without registration ($warping.method =' NOalignment'$)
- affine registration ($warping.method =' affine'$)

3.2.1 k-mean clustering $k = 4$

Given the type of experiment behind the data, it is reasonable to ask whether it is possible to classify the curves. The reason is evident: the curves come from different paths, so it would a first success to be able to predict which path the curve belongs to, that is, disposing of the intensity of his neuronal activity, we can deduce which path the monkey is following.

We present the results of the *K-Mean Alignment* algorithm, that is, the *kma* function of *fdakma* package with the parameters *warping.method* set to *NOalignment* and *n.clust* set to *4*. In order to only perform clustering of data, we did not allow for any transformation. The labels assigned by the *kma* function to the 240 functions are represented in the following confusion matrix (table 3.1). The results of the clustering seem to be very good.

The graphical results can be seen in figure 3.10, where each function has been colored with respect to the cluster it has been assigned to.

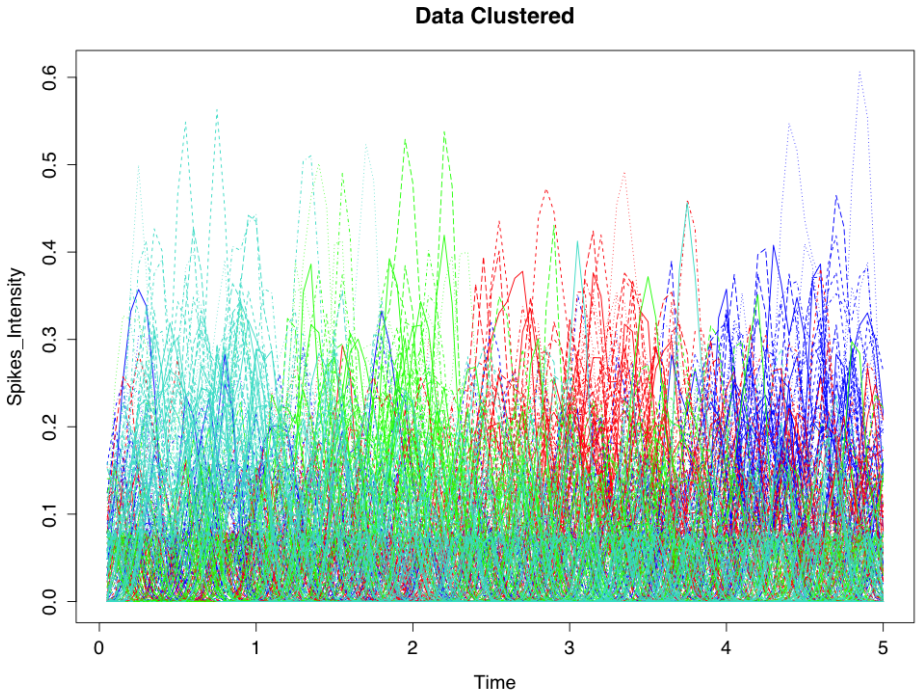

Figure 3.10: K-mean Clustering without alignment: result

		labels			
		1	**2**	**3**	**4**
true labels	**1**	56	2	1	1
	2	6	53	1	0
	3	0	4	55	1
	4	0	0	0	60

Table 3.1: K-mean clustering without alignment: confusion matrix

It is now clear that, in order to classify the data and predict which path the monkey is following, there is no need for registration. The k-mean clustering is sufficient to cluster data.

3.2.2 Affine warping $k = 4$

Given that our goal is not only to classify data, but also to study the phase variability, we try to align the curves allowing affine warping. In this case we present the results of the *kma* function with the parameters *warping.method* set to *affine* and *n.clust* set to *4*.

The first results, in figure 3.11, are not very exciting from a registration point of view. To be more clear, we also draw separately the results of the four clusters given by *kma* output in figures 3.12 and 3.13.

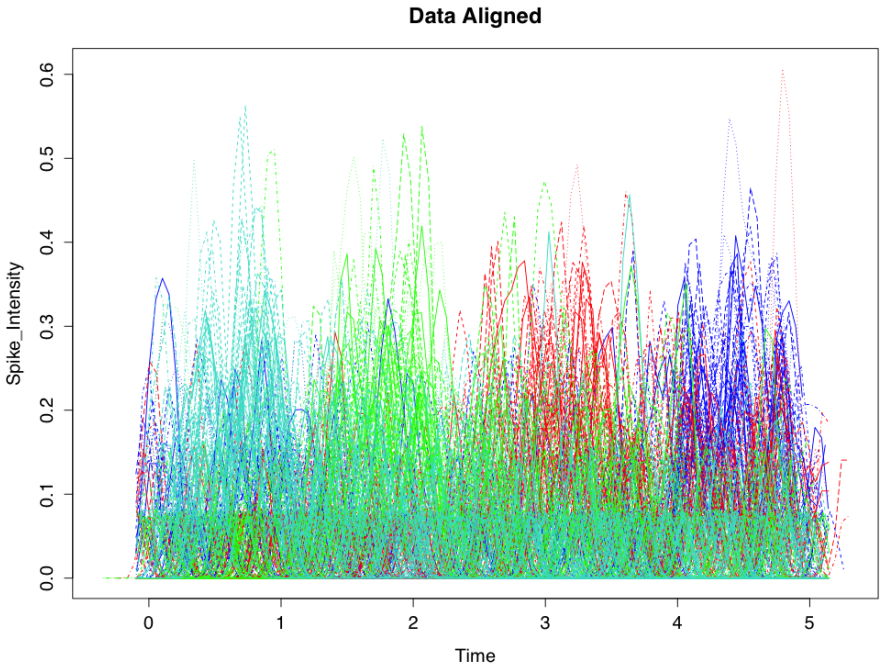

Figure 3.11: K-Mean Alignment, affine registration: results

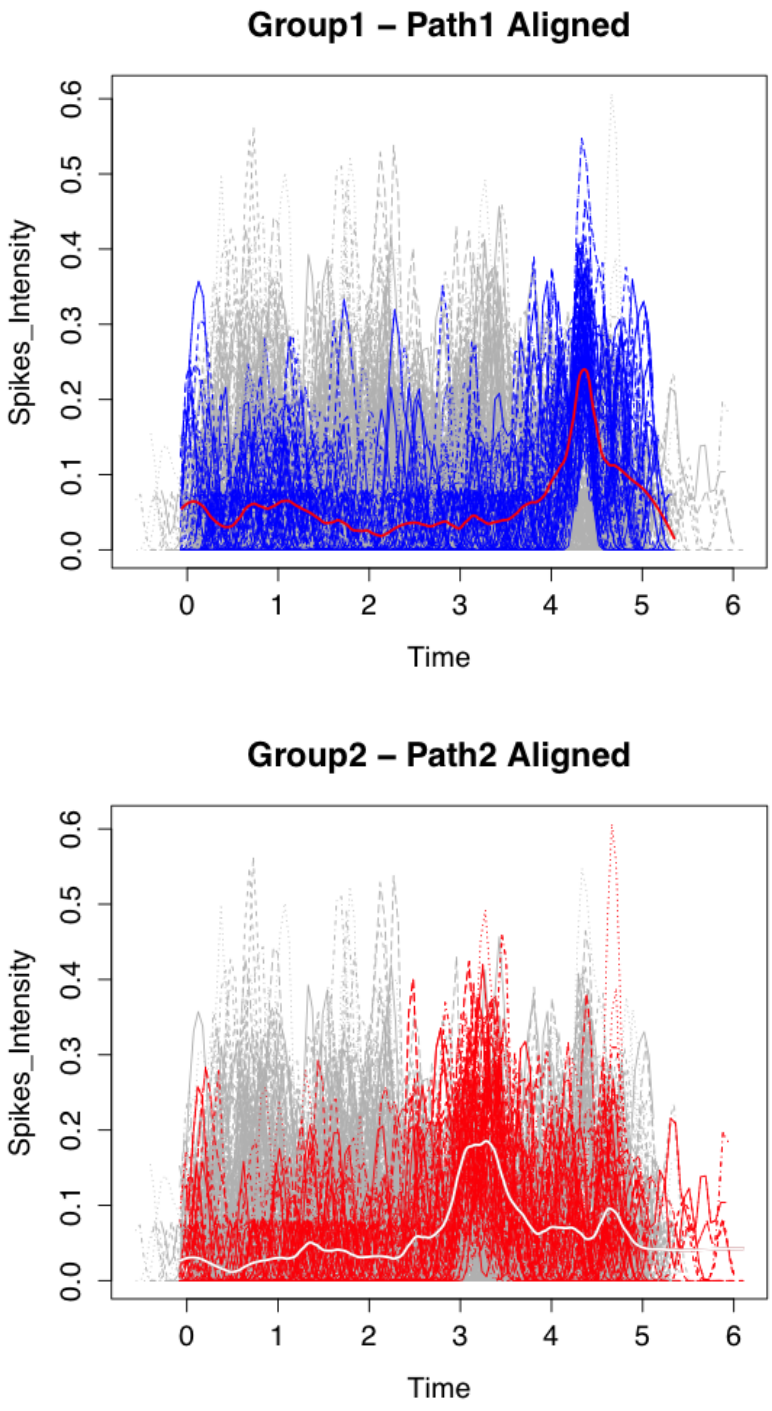

Figure 3.12: K-Mean Alignment, affine registration: results, paths 1-2

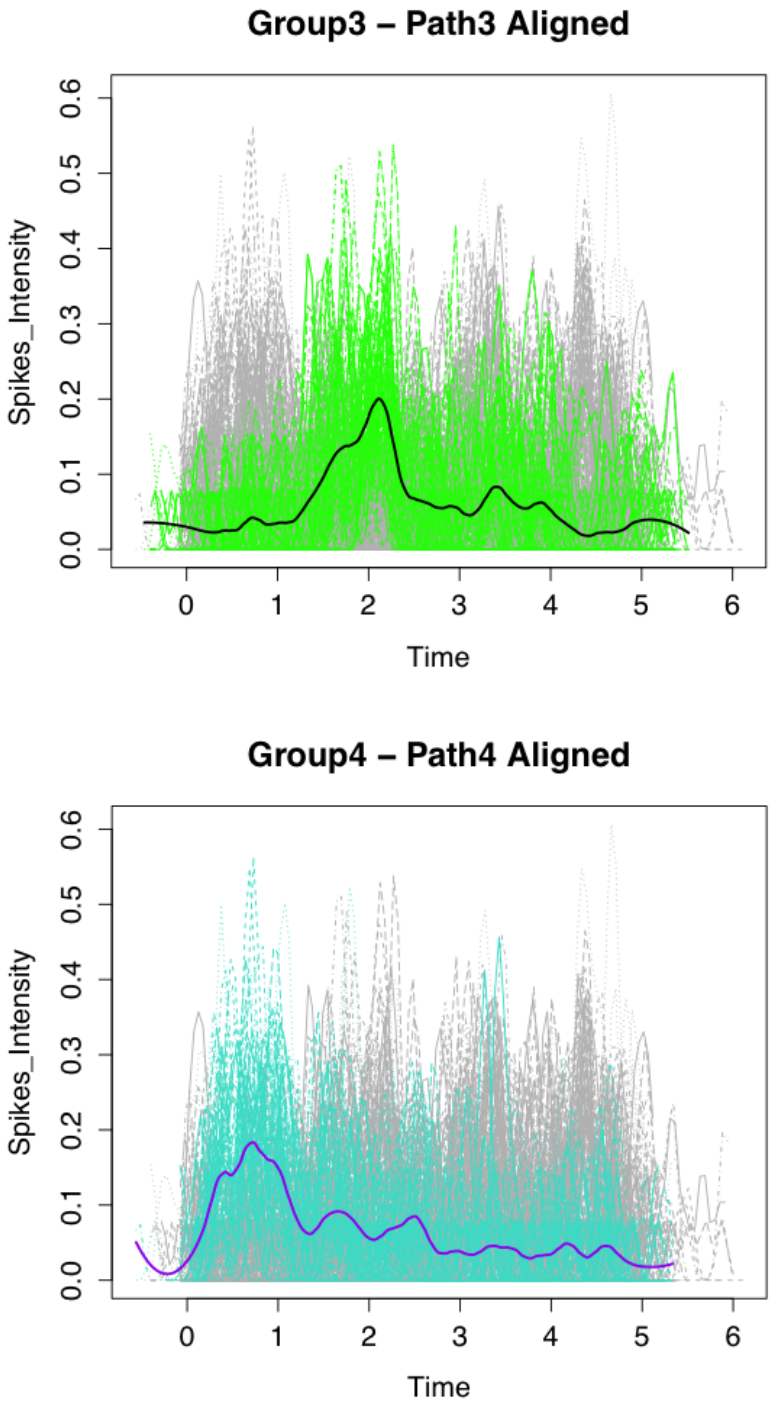

Figure 3.13: K-Mean Alignment, affine registration: results, paths 3-4

As before, we report the confusion matrix in this case (table 3.2), in which we can see that there is no gain in classification using affine warping compared to the confusion matrix obtained without warping and presented in table 3.1.

		labels			
		1	**2**	**3**	**4**
true labels	**1**	57	2	0	1
	2	4	54	1	1
	3	1	5	52	2
	4	1	0	2	57

Table 3.2: K-mean alignment, affine warping: confusion matrix

During the running of the K-Mean Alignment algorithm on the spike trains data, a need for a control of "anomalous" updates of the warping functions have been introduced. This was due to the very high amplitude variability of data. The problem has been solved with an extra control at the end of each iteration. More precisely, when a shift or dilation value represented an outlier with respect to the warping values of the other curves computed in the same iteration, they were re-optimized in a new fence, that is with new lower and upper bounds, represented by the first and third quartile of the distributions

of shifts and dilations computed in the same iteration. In this way, we avoided the possibility to have anomalous warpings.

The package *fdakma* already has this control, but it can be deactivated setting the input variable *fence* to *FALSE*. It is important to notice that, in case the algorithm find a lot of outlighers, it will repeat the optimization step many times, leading to an increase of complexity of the algorithm. It is then suggested to set it to *FALSE* when the dimension of the problem is considerable.

Finally, to resume the results of the non-periodic approach we found an absence of phase variability, so we do not need to align the functions in order to cluster. Moreover the form of the cluster centers seems to tell us that, starting from the upward movement, the hand job for the monkey seems to be always less difficult to do, in fact the neuronal activity slightly decreases.

3.3 Second approach: periodic data

In a second attempt to find new informations from data, we thought it would be interesting to interpret data as periodical. So we artificially reproduced the functions five times in the domaine [-10,15] making them periodical with period equal to 5, that is the original domain of our curves.

We repeated the analyses on this new dataset, but this time we allowed only a shift warping, so the output graphics of the *kma.compare* function (figure 3.14) will present only two curves, the first relative to *warping.method='NOalignment'* and the other one to *warping.method='shift'*.

The similarity measure, in this case, has been computed in the domain of interest, that is [0,5]. Moreover the amount of total shift for each curve was bounded, precisely it had to be in the interval [-2.5,2.5].

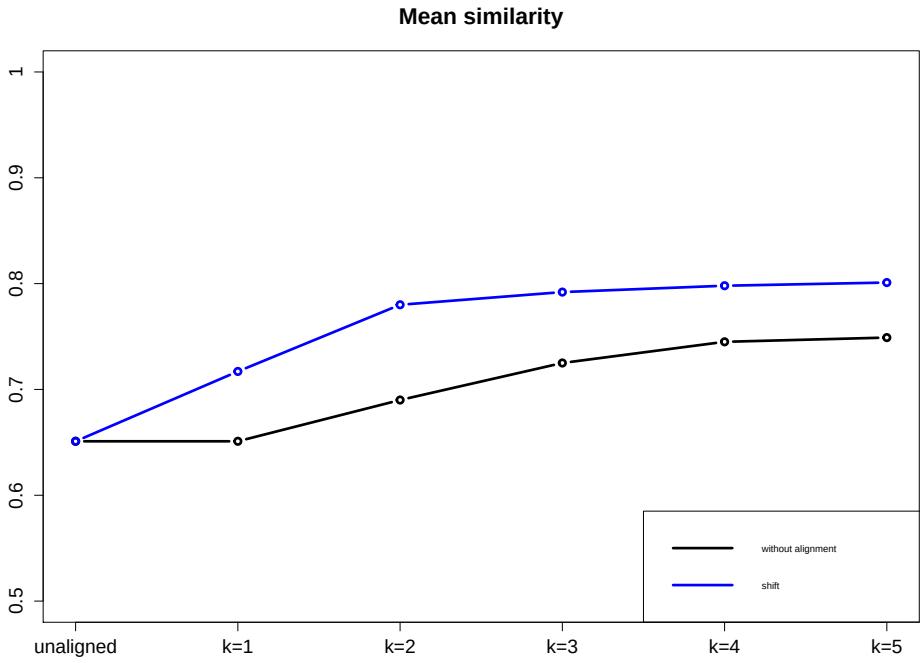

Figure 3.14: Mean similarity indexes with different numbers of clusters and 'shift' as warping method. Results of kma.compare function.

The black line, the one relative to k-mean clustering procedure, is obviously the same than before, but the blue line, relative to the shift warping, suggests to consider two clusters. However, let us present the results of the analyses for $k = 1$ and $k = 2$.

3.3.1 Shift warping $k = 1$

In figure 3.15 we can see all the curves aligned with a shift warping, the center of the unique cluster ($k = 1$) is represented with a black line. To better visualize the original clusters (the four paths), we plot them in four different plots (figures 3.16, 3.17).

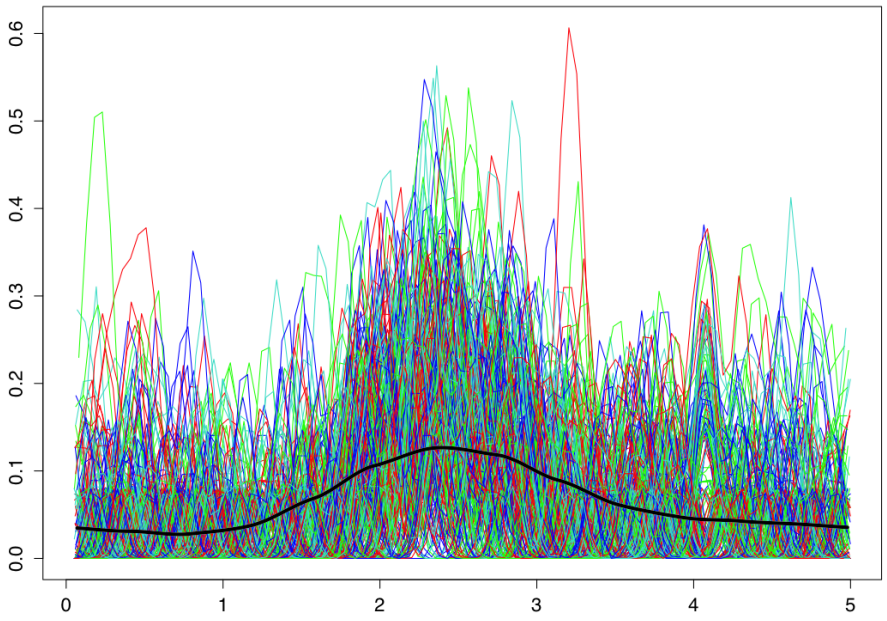

Figure 3.15: Periodic approach: shfit registration, $k = 1$

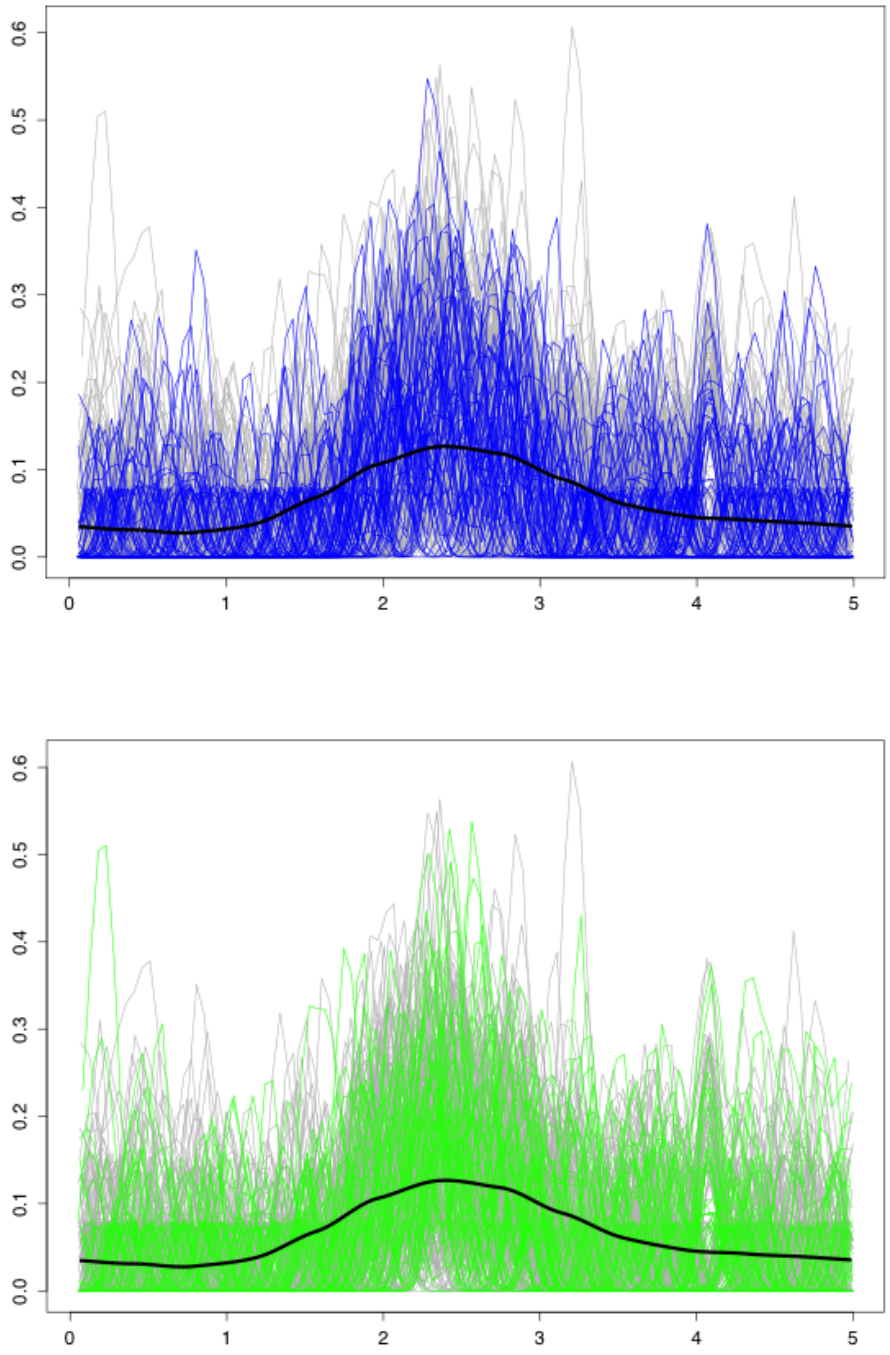

Figure 3.16: Periodic approach: shift registration, $k = 1$, paths 1-2

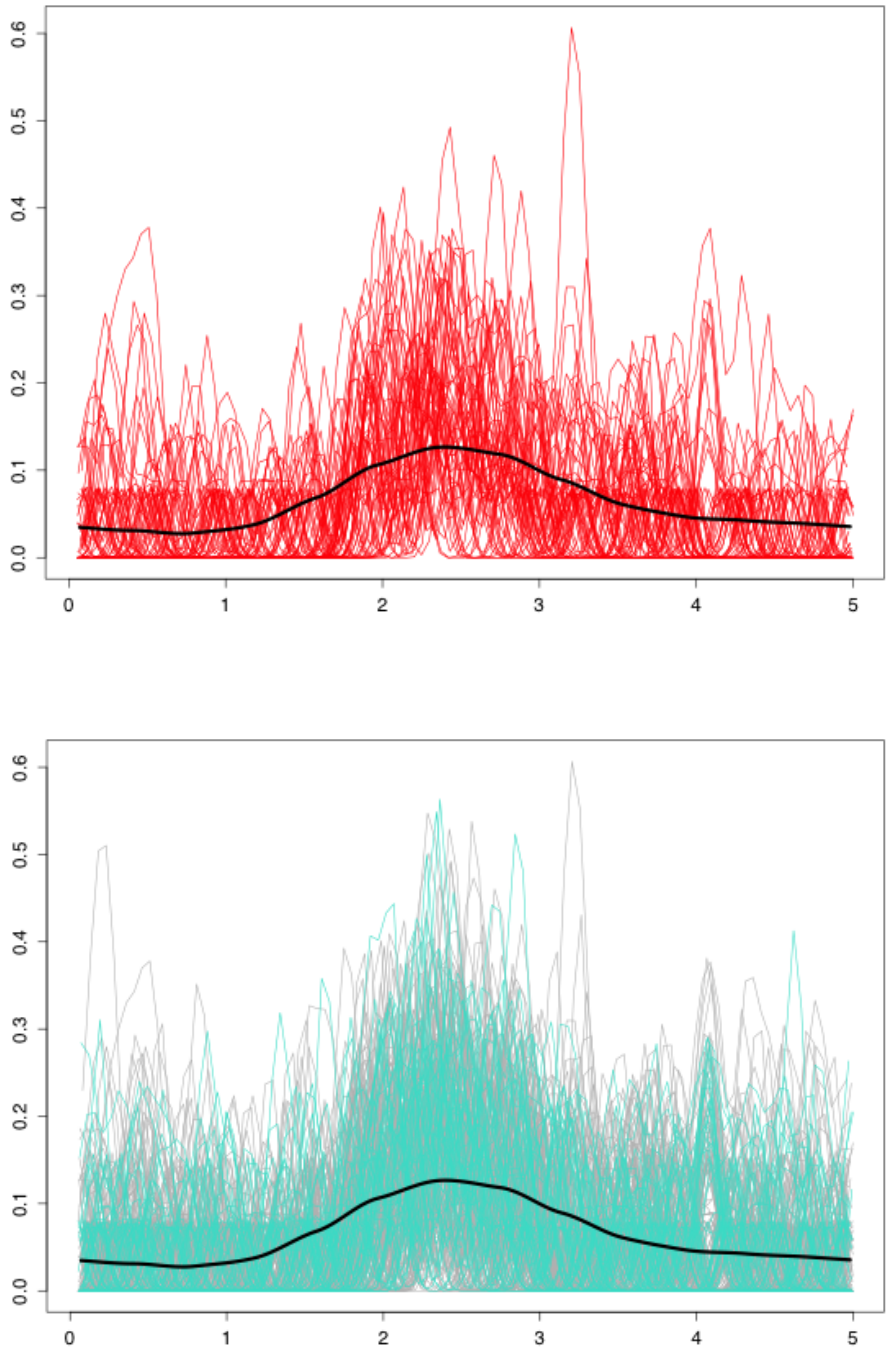

Figure 3.17: Periodic approach: shift registration, $k = 1$, paths 3-4

As largely explained in the second chapter, in a registration procedure it is important to analyze the warping functions. Given that we are allowing only for shift registration, we can represent the warping functions in the way proposed in figure 3.18.

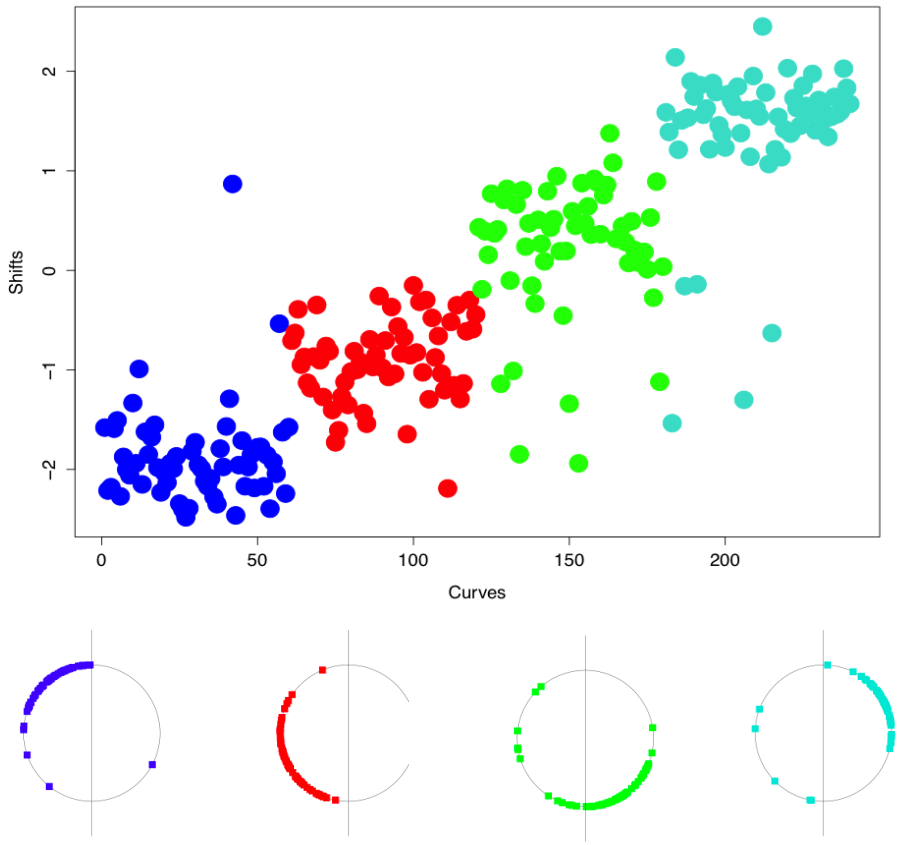

Figure 3.18: Periodic approach: shift registration, $k = 1$. Warping functions.

About the top panel of the graphic, we can clearly see a strong association between shifts and paths. It is normal if we think at the experiment: the monkey has to follow four different paths, but touching the same buttons. It does not seem strange at all that, allowing a shift warping to all curves, they will be aligned so that the values of the first 60 curves corresponding to the

movement "button D" - "button A" will be at the same place that, for example, the values of the second 60 curves corresponding to the same movement.

To understand the meaning of the second part of figure 3.18, we introduce a new representation model for this kind of warping. In figure 3.19 we can see that the value of the shifts imposed to the curves by the registration process can be drawn in a circle.

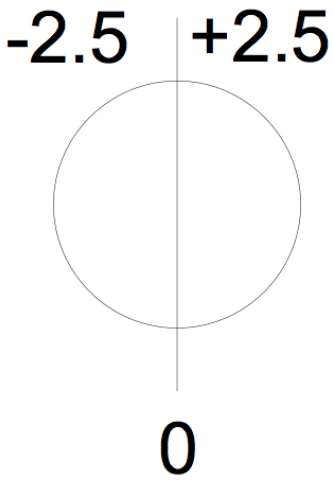

Figure 3.19: Representation model of warping functions.

The meaning of this representation is the following: a shift of *2.5s* has to be interpreted exactly like a shift of *-2.5s*, because we are dealing with periodic functions with a domain of length *5s*. For the same reason a shift of *1.25s* has to be considered completely opposite to a shift of *-1.25s*. Even with this representation it is clear how each function is aligned so that the movements of each path correspond to the same abscissa values.

3.3.2 Shift warping $k = 2$

Searching for a number of cluster different from *1* means that we are looking for something that can differentiate the four paths from each other. The figure 3.14 seems to suggest a number of cluster equal to *2*. Let us draw the registered curves and the centers of the two clusters found with the *kma* function (figure 3.20).

As before, the plot is not perfectly clear, because of the high number of functions involved. For a better visualization of the results for each path, let us draw them separately (figures 3.21, 3.22).

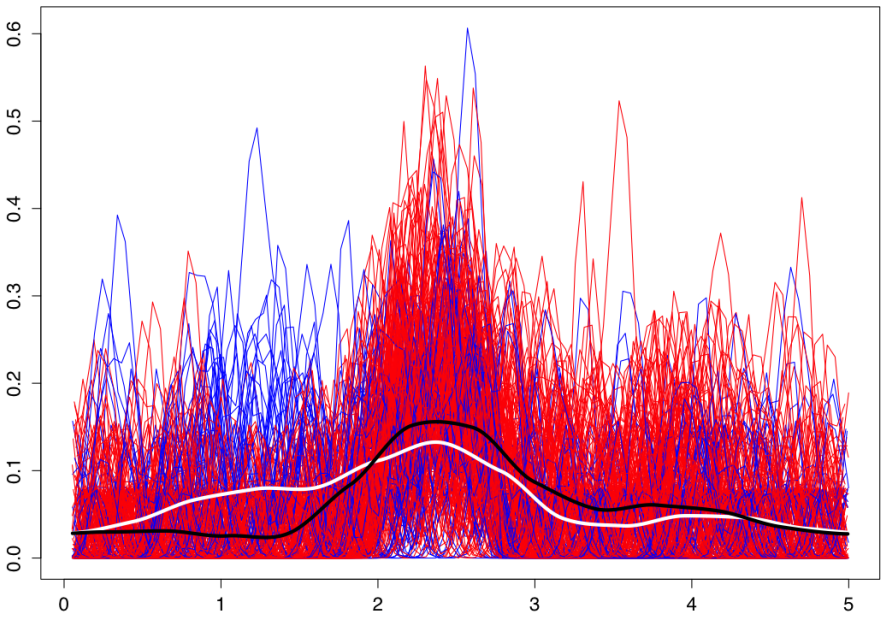

Figure 3.20: Periodic approach: shift registration, $k = 2$

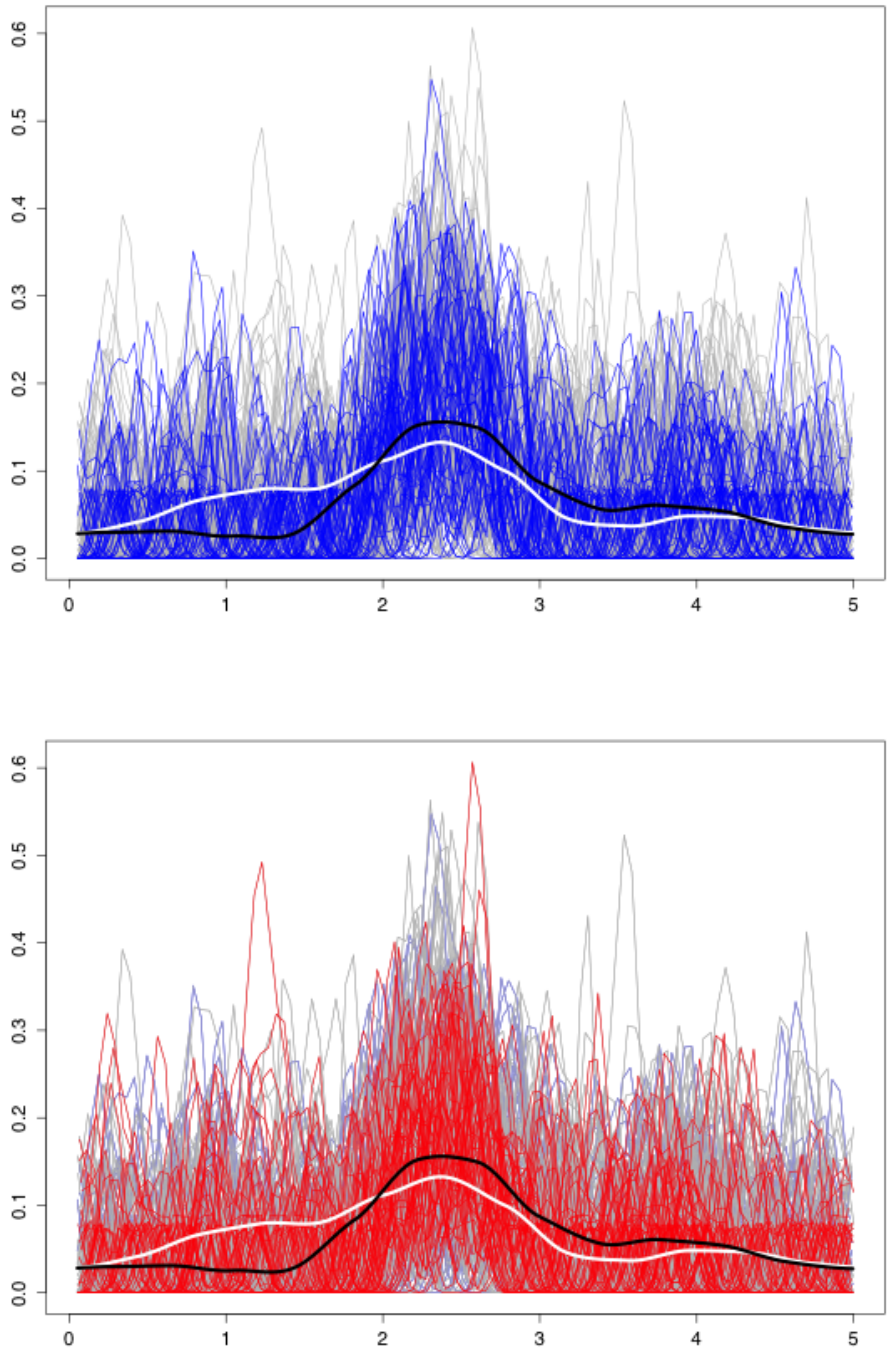

Figure 3.21: Periodic approach: shift registration, $k = 2$, paths 1-2

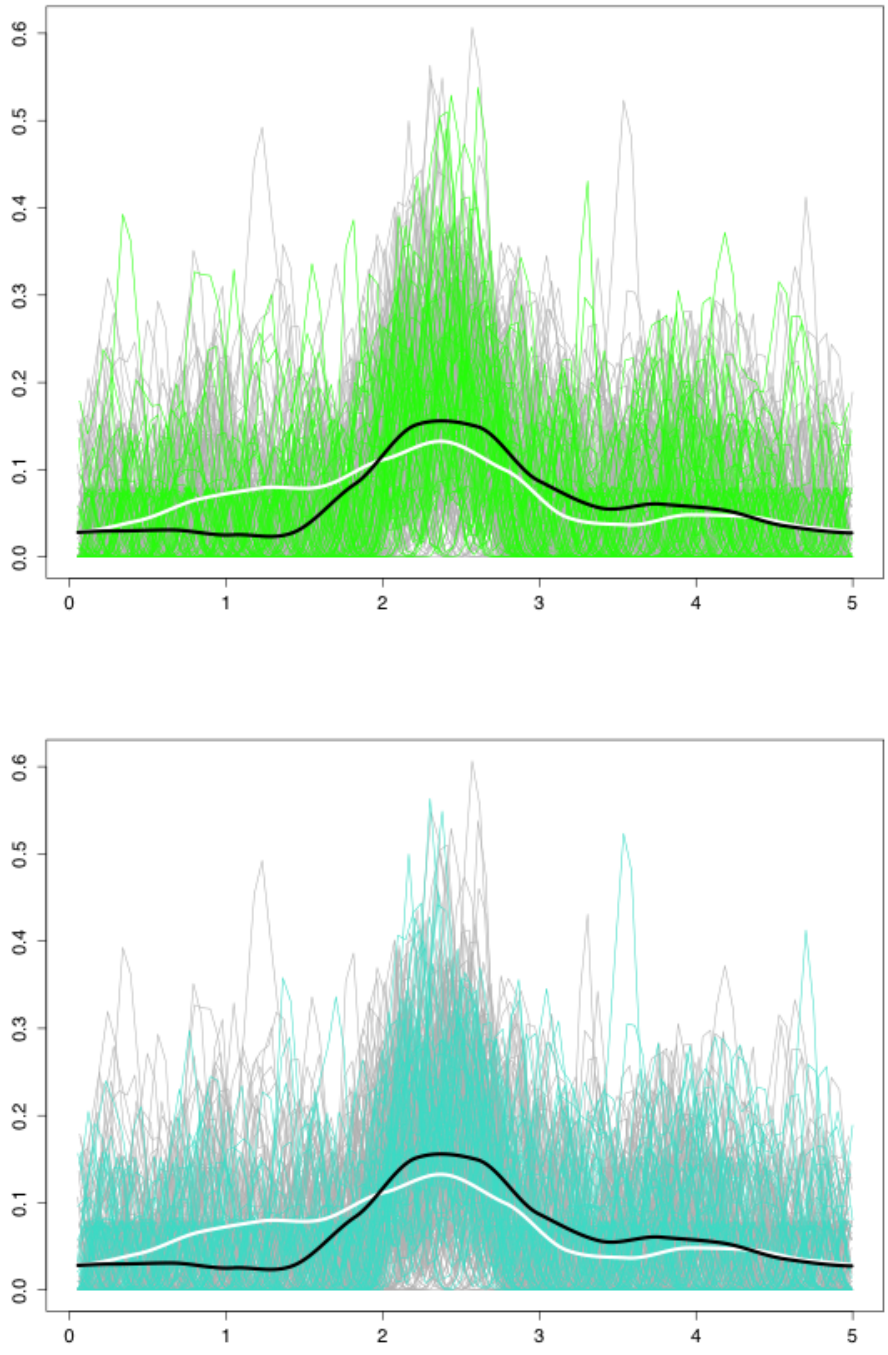

Figure 3.22: Periodic approach: shift registration, $k = 2$, paths 3-4

Let us now analyze the warping functions in figure 3.23. They seem to be quite similar to the previous case ($k = 1$), in fact we can notice the strong association between shifts and paths.

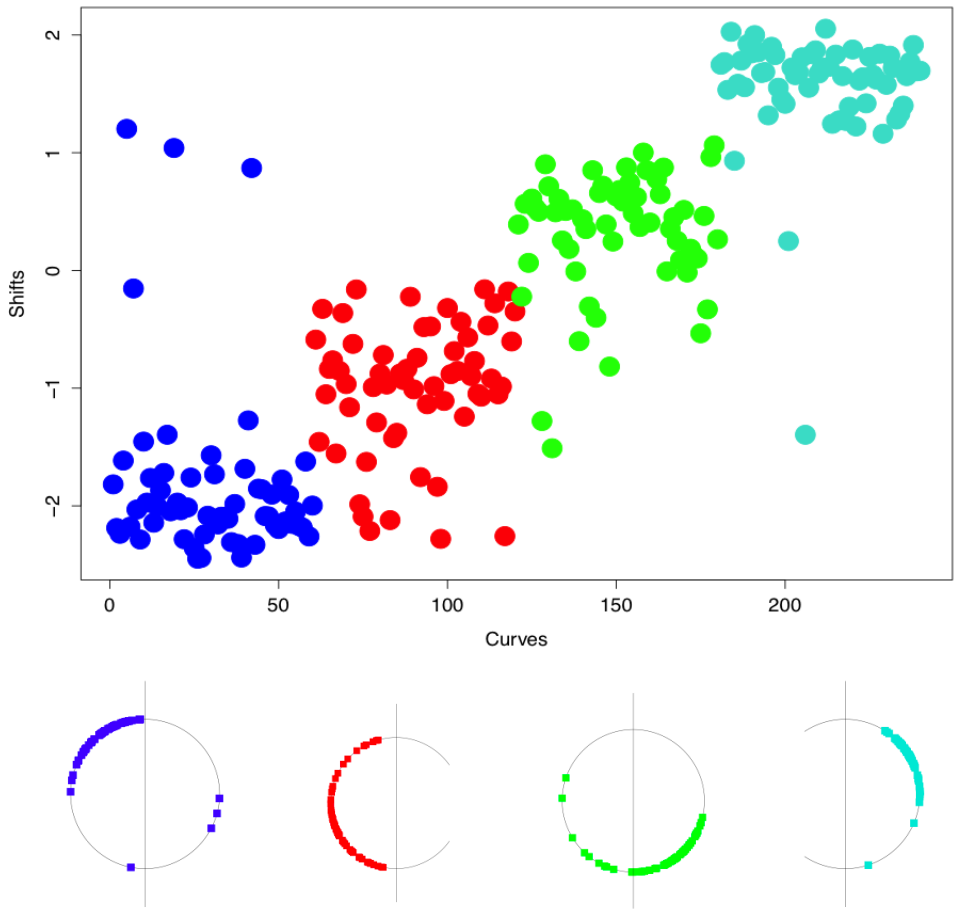

Figure 3.23: Periodic approach: shift registration, $k = 2$. Warping functions

Looking at the confusion matrix (table 3.3) we can see that the two clusters found by the algorithm are not related to the original four paths.

		labels	
		1	**2**
true labels	**1**	42	18
	2	51	9
	3	45	15
	4	43	17

Table 3.3: Periodic approach, shift warping, $k = 2$: confusion matrix

A confirmation of the last result is given by the analysis of figure 3.24. What we have done is a simple change of colors. In figure 3.23 the colors are the same than the original paths, but we can not see the two clusters found by the *kma* function. To visualize them we have colored the warping functions considering the clusters the functions belong to.

We can clearly see that the clustering is orthogonal to paths, in the sense that the two found clusters contain functions from all the paths in a indistinct way. This means that we can cluster data in two groups, that have nothing to do

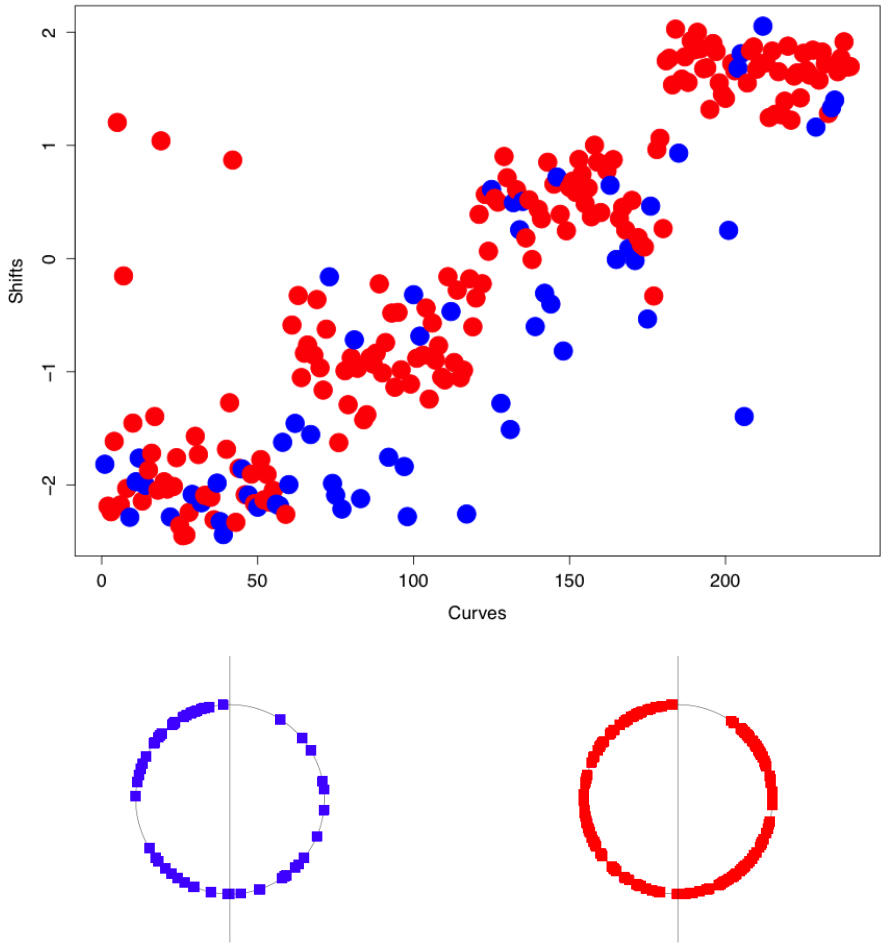

Figure 3.24: Periodic approach: shift registration, $k = 2$. Warping functions colored by group.

with the four different paths.

We advanced some hypothesis which can explain this phenomenon but, of course, a confirmation by a specialist in the sector should be necessary. At first we thought the reason could be strictly related to the experiment itself, while other hypothesis concern the physics of the subject. Unfortunately we do not

dispose of such informations and for the moment these hypothesis can not be confirmed.

Chapter 4

R package: *fdakma*

4.1 Introduction to the package

In the last chapter of this work we are going to present the conception of an R package called *fdakma*, whose aim is to resume the *K-Mean Alignment* method for functional data registration described in chapter 2.

The package will be available on CRAN, so that it can be used to repeat the analysis done in this work and also be useful for any functional data registration and/or clustering purpose. In order to build the package, we followed the guide lines given in R Development Core Team (2013) and in Team (2010). Some inspiring concepts also came from Ramsay, Hooker, et Graves (2005) and Ramsay, Jim O. and Silverman, Bernard (2002).

Using the package the user will quickly understand its usefulness, but despite all the good thing, some problems are still present and a lot of improvement can be done.

For example, like every k-means procedure, there is a certain dependence on the choice of the initial curves. In the *K-Mean Alignment* algorithm, having

also the alignment goal, the problem seems to be even more delicate.

Another important feature that one has to consider when using for the first time this package is the complexity and hence the time required for the complete execution of the algorithm. If the functions present a lot of noise and/or a lot of variations, it is important to understand the meaning of the parameter *fence*. This parameter, if set to *TRUE*, activates a control at the end of each iteration. This control can cause an increase of time needed for the completion of the running. This happens because other optimizations are run for the curves which present values of shift or dilation that are outlier with respect to ones of the other curves.

4.2 R documentation

In the following pages we report the *R help* which will also be available on CRAN repository. All the details of the package are explained, as well as all the input and output parameters. The *R* documentation also includes an example for each function of the package, so that one can easily learn how to use it on his own data.

Package 'fdakma'

June 24, 2013

Type Package

Title Clustering and alignment of a functional dataset

Version 1.0

Date 2013-06-23

Author Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

Maintainer Simone Vantini <simone.vantini@polimi.it>

Description The package fdakma jointly performs clustering and alignment of a multidimensional or unidimensional functional dataset.

License GPL (>= 3)

R topics documented:

`fdakma-package` *Functional Data Analysis: K-Mean Alignment*

Description

fdakma jointly performs clustering and alignment of a functional dataset (multidimensional or unidimensional functions).

Details

Package:	fdakma
Type:	Package
Version:	1.0
Date:	2013-05-04
License:	GPL-3

Author(s)

Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

References

Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*. Computational Statistics and Data Analysis, 54, 1219-1233.

See Also

kma.compare, kma.similarity, kma.data, kma, kma.show.results

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')

# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')

# Example: result of kma function with 2 clusters,
# allowing affine transformation for the abscissas
# and considering 'd1.pearson' as similarity.method.
fdakma_example <- kma (
  x=x, y0=y0, y1=y1, n.clust = 2,
  warping.method = 'affine',
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21)
)

kma.show.results(fdakma_example)

names(fdakma_example)

# Labels assigned to each function
fdakma_example$labels

# Total shifts and dilations applied to the original
# abscissa to obtain the aligned abscissa
fdakma_example$shift
fdakma_example$dilation
```

`kma` *Clustering and alignment of functional data*

Description

kma jointly performs clustering and alignment of a functional dataset (multidimensional or unidimensional functions). To run kma function with different numbers of clusters and/or different alignment methods see kma.compare.

Usage

```
kma(x, y0 = NULL, y1 = NULL, n.clust = 1, warping.method = "affine",
similarity.method = "d1.pearson", center.method = "k-means", seeds = NULL,
optim.method = "L-BFGS-B", span = 0.15, t.max = 0.1, m.max = 0.1, n.out = NULL,
tol = 0.01, fence = TRUE, iter.max = 100, show.iter = 0)
```

Arguments

`x`	matrix *n.func* X *grid.size* or vector *grid.size*: the abscissa values where each function is evaluated. *n.func*: number of functions in the dataset. *grid.size*: maximal number of abscissa values where each function is evaluated. The abscissa points may be unevenly spaced and they may differ from function to function. `x` can also be a vector of length *grid.size*. In this case, `x` will be used as abscissa grid for all functions.
`y0`	matrix *n.func* X *grid.size* or array *n.func* X *grid.size* X *d*: evaluations of the set of original functions on the abscissa grid `x`. *n.func*: number of functions in the dataset. *grid.size*: maximal number of abscissa values where each function is evaluated. *d*: (only if the sample is multidimensional) number of function components, i.e. each function is a *d*-dimensional curve. Default value of `y0` is `NULL`. The parameter `y0` must be provided if the chosen `similarity.method` concerns original functions.
`y1`	matrix *n.func* X *grid.size* or array *n.func* X *grid.size* X *d*: evaluations of the set of original functions first derivatives on the abscissa grid `x`. Default value of `y1` is `NULL`. The parameter `y1` must be provided if the chosen `similarity.method` concerns original function first derivatives.
`n.clust`	scalar: required number of clusters. Default value is `1`. Note that if `n.clust=1` kma performs only alignment without clustering.
`warping.method`	character: type of alignment required. If `warping.method='NOalignment'` kma performs only k-mean clustering (without alignment). If `warping.method='affine'` kma performs alignment (and possibly clustering) of functions using linear affine transformation as warping functions, i.e., *x.final* = *dilation*x* + *shift*. If `warping.method='shift'` kma allows only shift, i.e., *x.final* = *x* + *shift*. If `warping.method='dilation'` kma allows only dilation, i.e., *x.final* = *dilation*x*. Default value is `'affine'`.

`similarity.method`
: character: required similarity measure.
Possible choices are: `'d0.pearson'`, `'d1.pearson'`, `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'`, `'d1.L2.centered'`. Default value is `'d1.pearson'`. See kma.similarity for details.

`center.method`
: character: type of clustering method to be used.
Possible choices are: `'k-means'` and `'k-medoids'`. Default value is `'k-means'`.

`seeds`
: scalar or vector: indexes of the functions to be used as initial centers.
If `seeds=NULL` centers are randomly chosen among the *n.func* original functions. Default value of `seeds` is `NULL`.

`optim.method`
: character: optimization method chosen to find the best warping functions at each iteration.
Possible choices are: `'L-BFGS-B'` and `'SANN'`. See optim function for details. Default method is `'L-BFGS-B'`.

`span`
: scalar: the span to be used for the loess procedure in the center estimation step when `center.method='k-means'`.
Default value is 0.15. If `center.method='k-medoids'` value of `span` is ignored.

`t.max`
: scalar: `t.max` controls the maximal allowed shift, at each iteration, in the alignment procedure with respect to the range of curve domains.
`t.max` must be such that *0<t.max<1* (e.g., `t.max=0.1` means that shift is bounded, at each iteration, between *-0.1*range(x)* and *+0.1*range(x)*). Default value is `0.1`. If `warping.method='dilation'` value of `t.max` is ignored.

`m.max`
: scalar: `m.max` controls the maximal allowed dilation, at each iteration, in the alignment procedure.
`m.max` must be such that *0<m.max<1* (e.g., `m.max=0.1` means that dilation is bounded, at each iteration, between *1-0.1* and *1+0.1*). Default value is `0.1`. If `warping.method='shift'` value of `m.max` is ignored.

`n.out`
: scalar: the desired length of the abscissa for computation of the similarity indexes and the centers. Default value is `round(1.1*grid.size)`.

`tol`
: scalar: the algorithm stops when the increment of similarity of each function with respect to the corrispondent center is lower than `tol`.
Default value is `0.01`.

`fence`
: boolean: if `fence=TRUE` a control is activated at the end of each iteration. The aim of the control is to avoid shift/dilation outlighers with respect to their computed distributions.
If `fence=TRUE` the running time can increase considerably. Default value of `fence` is TRUE.

`iter.max`
: scalar: maximum number of iterations in the k-mean alignment cycle. Default value is `100`.

`show.iter`
: boolean: if `show.iter=TRUE` kma shows the current iteration of the algorithm.
Default value is `FALSE`.

Value

The function output is a list containing the following elements:

`iterations`
: scalar: total number of iterations performed by kma function.

`x` as input.

`y0` as input.

`y1` as input.

`n.clust` as input.

`warping.method` as input.

`similarity.method`
as input.

`center.method` as input.

`x.center.orig` vector *n.out*: abscissa of the original center.

`y0.center.orig` matrix *1* X *2*grid.size*: the unique row contains the evaluations of the original function center.
If `warping.method='k-means'` there are two scenarios:
- if `similarity.method='d0.pearson'` or `'d0.L2'` or `d0.L2.centered` the original function center is computed via loess procedure applied to original data;
- if `similarity.method='d1.pearson'` or `'d1.L2'` or `d1.L2.centered` it is computed by integration of first derivatives center `y1.center.orig`.
If `warping.method='k-medoids'` the original function center is the medoid of original functions.

`y1.center.orig` matrix *1* X *2*grid.size*: the unique row contains the evaluations of the original function first derivatives center.
If `warping.method='k-means'` the original center is computed via loess procedure applied to original function first derivatives.
If `warping.method='k-medoids'` the original center is the medoid of original functions.

`similarity.orig`
vector: original similarities between the original functions and the original center.

`x.final` matrix *n.func* X *grid.size*: aligned abscissas.

`n.clust.final` scalar: final number of clusters.
Note that `n.clust.final` may differ from initial number of clusters (i.e., from `n.clust`) if some clusters are found to be empty. In this case a warning message is issued.

`x.centers.final`
vector *n.out*: abscissas of the final function centers and/or of the final function first derivatives centers.

`y0.centers.final`
matrix *n.clust.final* X *2*grid.size*: rows contain the evaluations of the final functions centers.
`y0.centers.final` is `NULL` if `y0` is not given as input.

`y1.centers.final`
matrix *n.clust.final* X *2*grid.size*: rows contains the evaluations of the final derivatives centers.
`y1.centers.final` is `NULL` if the chosen similarity measure does not concern function first derivatives.

`labels` vector: cluster assignments.

`similarity.final`
vector: similarities between each function and the center of the cluster the function is assigned to.

`dilation.list`	list: dilations obtained at each iteration of kma function.
`shift.list`	list: shifts obtained at each iteration of kma function.
`dilation`	vector: dilation applied to the original abscissas `x` to obtain the aligned abscissas `x.final`.
`shift`	vector: shift applied to the original abscissas `x` to obtain the aligned abscissas `x.final`.

Author(s)

Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

References

Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*. Computational Statistics and Data Analysis, 54, 1219-1233.

See Also

`kma.compare`, `kma.similarity`, `fdakma`, `kma.data`, `kma.show.results`

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')

# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')


# Example: result of kma function with 2 clusters,
# allowing affine transformation for the abscissas
# and considering 'd1.pearson' as similarity.method.
kma_example <- kma (
  x=x, y0=y0, y1=y1, n.clust = 2,
  warping.method = 'affine',
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21)
  )

kma.show.results(kma_example)

names(kma_example)

# Labels assigned to each function
kma_example$labels
```

```
# Total shifts and dilations applied to the original
# abscissa to obtain the aligned abscissa
kma_example$shift
kma_example$dilation
```

`kma.compare` *kma.compare runs kma with different numbers of clusters and different warping methods.*

Description

In kma.compare the user can specify multiple values for n.clust and warping.method. kma.compare runs the K-Mean Alignment algorithm (kma function) for all couples of specified values of `n.clust` and `warping.method`.

Usage

```
kma.compare(x, y0 = NULL, y1 = NULL, n.clust = c(1, 2),
warping.method = c("NOalignment", "shift", "dilation", "affine"),
similarity.method = "d1.pearson", center.method = "k-means", seeds = NULL,
optim.method = "L-BFGS-B", span = 0.15, t.max = 0.1, m.max = 0.1, n.out = NULL,
tol = 0.01, fence = TRUE, iter.max = 100, show.iter = 0, plot.graph = 0)
```

Arguments

`x`	matrix *n.func* X *grid.size* or vector *grid.size*: the abscissa values where each function is evaluated. *n.func*: number of functions in the dataset. *grid.size*: maximal number of abscissa values where each function is evaluated. The abscissa points may be unevenly spaced and they may differ from function to function. `x` can also be a vector of length *grid.size*. In this case, `x` will be used as abscissa grid for all functions.
`y0`	matrix *n.func* X *grid.size* or array *n.func* X *grid.size* X *d*: evaluations of the set of original functions on the abscissa grid `x`. *n.func*: number of functions in the dataset. *grid.size*: maximal number of abscissa values where each function is evaluated. *d*: (only if the sample is multidimensional) number of function components, i.e. each function is a *d*-dimensional curve. Default value of `y0` is `NULL`. The parameter `y0` must be provided if the chosen `similarity.method` concerns original functions.
`y1`	matrix *n.func* X *grid.size* or array *n.func* X *grid.size* X *d*: evaluations of the set of original functions first derivatives on the abscissa grid `x`. Default value of `y1` is `NULL`. The parameter `y1` must be provided if the chosen `similarity.method` concerns original function first derivatives.
`n.clust`	vector: `n.clust` contains the numbers of clusters with which kma.compare runs kma function. Default value is `c(1,2)`. See details.

`warping.method` vector: `warping.method` contains the types of alignment with which kma.compare runs kma function. See details.

`similarity.method`
character: required similarity measure.
Possible choices are: `'d0.pearson'`, `'d1.pearson'`, `'d0.L2'`, `'d1.L2'`, `'d1.L2.centered'`, `'d0.L2.centered'`. Default value is `'d1.pearson'`. See kma.similarity for details.

`center.method` character: type of clustering method to be used.
Possible choices are: `'k-means'` and `'k-medoids'`. Default value is `'k-means'`.

`seeds` vector *max(n.clust)*: indexes of the functions to be used as initial centers.
Length of `seeds` must be equal to *max(n.clust)*. If `seeds=NULL`, centers are randomly chosen among the *n.func* original functions. Default value of `seeds` is `NULL`.

`optim.method` character: optimization method chosen to find the best warping functions at each iteration.
Possible choices are: `'L-BFGS-B'` and `'SANN'`. See optim function for details. Default method is `'L-BFGS-B'`.

`span` scalar: the span to be used for the loess procedure in the center estimation step when `center.method='k-means'`.
Default value is 0.15. If `center.method='k-medoids'` value of `span` is ignored.

`t.max` scalar: `t.max` controls the maximal allowed shift, at each iteration, in the alignment procedure with respect to the range of curve domains.
`t.max` must be such that *0<t.max<1* (e.g., `t.max=0.1` means that shift is bounded, at each iteration, between *-0.1*range(x)* and *+0.1*range(x)*). Default value is `0.1`. If `warping.method='dilation'` value of `t.max` is ignored.

`m.max` scalar: `m.max` controls the maximal allowed dilation, at each iteration, in the alignment procedure.
`m.max` must be such that *0<m.max<1* (e.g., `m.max=0.1` means that dilation is bounded, at each iteration, between *1-0.1* and *1+0.1*). Default value is `0.1`. If `warping.method='shift'` value of `m.max` is ignored.

`n.out` scalar: the desired length of the abscissa for computation of the similarity indexes and the centers. Default value is `round(1.1*grid.size)`.

`tol` scalar: the algorithm stops when the increment of similarity of each function with respect to the corrispondent center is lower than `tol`.
Default value is `0.01`.

`fence` boolean: if `fence=TRUE` a control is activated at the end of each iteration. The aim of the control is to avoid shift/dilation outlighers with respect to their computed distributions.
If `fence=TRUE` the running time can increase considerably. Default value of `fence` is TRUE.

`iter.max` scalar: maximum number of iterations in the k-mean alignment cycle. Default value is `100`.

`show.iter` boolean: if `show.iter=TRUE` kma shows the current iteration of the algorithm.
Default value is `FALSE`.

`plot.graph` boolean: if `plot.graph=TRUE`, kma.compare plots a graphic with the means of similarity indexes as ordinate and the number of clusters as abscissa.
Default value is `FALSE`.

Details

Example of use: if `n.clust=c(1,2,3)` and `warping.method=c('shift','affine')`, kma.compare runs kma function with number of clusters equal to 1, 2 and 3 using `warping.method='shift'` and `warping.method='affine'`. Note that the kma.compare function always runs with the required number of clusters and `warping.method='NOalignment'` as well, even if not specified by users.

Value

The function output is a list containing the following elements:

`Result.NOalignment`
: list of outputs of kma function with `warping.type='NOalignment'`.
The sublist `Result.NOalignment[[k]]` corresponds to the results when number of clusters is `n.clust[k]`.

`Result.shift`
: list of outputs of kma function with `warping.type='shift'`.
The sublist `Result.shift[[k]]` corresponds to the results when number of clusters is `n.clust[k]`. Note that if `'shift'` is not chosen as `warping.type`, then `Result.shift` will be `NULL`.

`Result.dilation`
: list of outputs of kma function with `warping.type='dilation'`.
The sublist `Result.dilation[[k]]` corresponds to the results when number of clusters is `n.clust[k]`. Note that if `'dilation'` is not chosen as `warping.type`, then `Result.dilation` will be `NULL`.

`Result.affine`
: list of outputs of kma function with `warping.type='affine'`.
The sublist `Result.affine[[k]]` corresponds to the results when number of clusters is `n.clust[k]`. Note that if `'affine'` is not chosen as `warping.type`, then `Result.affine` will be `NULL`.

`n.clust`
: as input.

`mean.similarity.NOalignment`
: vector: mean similarity indexes of functions after running kma function with all elements of `n.clust` and `warping.type='NOalignment'`.
`mean.similarity.NOalignment` contains the ordinates of the black curve ("without alignment" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

`mean.similarity.shift`
: vector: mean similarity indexes of curves after running kma function with all elements of `n.clust` and `warping.type='shift'`.
`mean.similarity.shift` contains the ordinates of the blue curve ("shift" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

`mean.similarity.dilation`
: vector: mean similarity indexes of curves after running kma function with all elements of `n.clust` and `warping.type='dilation'`.
`mean.similarity.dilation` contains the ordinates of the green curve ("dilation" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

`mean.similarity.affine`
: vector: mean similarity indexes of curves after running kma function with all elements of `n.clust` and `warping.type='affine'`.
`mean.similarity.affine` contains the ordinates of the orange curve ("affine" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

Author(s)

Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

References

Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*. Computational Statistics and Data Analysis, 54, 1219-1233.

See Also

kma, kma.similarity, fdakma, kma.data, kma.show.results

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')

# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')

# Example: results of kma function with 3 different
# numbers of clusters (1,2,3) combined with four alignment
# methods ('NOalignment' by default, 'shift', 'dilation',
# 'affine') and considering 'd1.pearson' as similarity.method.
kma.compare_example <- kma.compare (
  x=x, y0=y0, y1=y1, n.clust = 1:3,
  warping.method = c('affine'),
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21,30),
  plot.graph=1)

names (kma.compare_example)

# To see results for kma function with n.clust=2
# and warping.method='affine'.
kma.show.results (kma.compare_example$Result.affine[[2]])

# Labels assigned to each function for the
# kma function with n.clust=2 and warping.method='affine'.
kma.compare_example$Result.affine[[2]]$labels
```

`kma.data` *Simulated Data*

Description

kma.data is a functional dataset displaying both amplitude and phase variability.

Usage

```
data(kma.data)
```

Format

List of 3 elements:

$x : abscissa values where each function is evaluated

$y0: evaluations of the original functions on the abscissa grid `kma.data$x`

$y1: evaluations of the original function first derivatives on the abscissa grid `kma.data$x`.

References

Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*. Computational Statistics and Data Analysis, 54, 1219-1233.

See Also

`kma.compare`, `kma.similarity`, `fdakma`, `kma`, `kma.show.results`

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')

# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')
```

`kma.show.results` *Auxiliary function plotting results of kma function.*

Description

kma.show.results graphically shows the output results of kma function or one of the outuput results of kma.compare function.

Four or six plots are generated (depending on the presence of `y0` and/or `y1` among the inputs):

- Plot of original functions with center (if `y0` is given as input in kma/kma.compare function).

- Plot of aligned functions with centers (if `y0` is given as input in kma/kma.compare function).

- Plot of original function first derivatives with center (if `y1` is given as input and the chosen `similarity.method` concerns function first derivatives).

- Plot of aligned function first derivatives with centers (if `y1` is given as input and the chosen `similarity.method` concerns function first derivatives).

- Plot of warping functions.

- Boxplot of similarity/dissimilarity indexes of original and aligned functions.

Usage

```
kma.show.results(Result)
```

Arguments

`Result`	list: output of kma function or one of the outputs of kma.compare functions.

Author(s)

Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

References

Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*. Computational Statistics and Data Analysis, 54, 1219-1233.

See Also

`kma`, `kma.compare`, `kma.similarity`, `fdakma`, `kma.data`

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

# kma function with 2 clusters, allowing affine
# transformation for the abscissas and considering
# 'd1.pearson' as similarity.method.
kma.show.results_example1 <- kma (
```

```
  x=x, y0=y0, y1=y1, n.clust = 2,
  warping.method = 'affine',
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21)
)

# Example: kma.show.results shows the results of kma function
kma.show.results(kma.show.results_example1)

## Not run:
# Example using outputs of kma.compare function

# Results of kma function with 3 different
# numbers of clusters (1,2,3) combined with four alignment
# methods ('NOalignment' by default, 'shift', 'dilation',
# 'affine') and considering 'd1.pearson' as similarity.method.
kma.show.results_example2 <- kma.compare (
  x=x, y0=y0, y1=y1, n.clust = 1:3,
  warping.method = c('affine'),
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21,30),
  plot.graph=1)

names (kma.show.results_example2)

# To see results for kma function with n.clust=2
# and warping.method='affine'.
kma.show.results (kma.show.results_example2$Result.affine[[2]])

# Labels assigned to each function for the
# kma function with n.clust=2 and warping.method='affine'.
kma.show.results_example2$Result.affine[[2]]$labels

## End(Not run)
```

`kma.similarity` *Similarity/dissimilarity index between two functions*

Description

kma.similarity computes a similarity/dissimilarity measure between two functions f and g. Users can choose among different types of measures.

Usage

```
kma.similarity(x.f = NULL, y0.f = NULL, y1.f = NULL,
x.g = NULL, y0.g = NULL, y1.g = NULL, similarity.method, unif.grid = TRUE)
```

Arguments

`x.f`	vector: abscissa grid where function f and his first derivatives f' is evaluated. `x.f` must always be provided.
`y0.f`	vector: evaluations of function f on the abscissa grid `x.f`. Default value of `y0.f` is `NULL`. The vector `y0.f` must be provided if the chosen `similarity.method` concerns original functions.
`y1.f`	vector: evaluations of f first derivative, i.e., f', on the abscissa grid `x.f`. Default value of `y1.f` is `NULL`. The vector `y1.f` must be provided if the chosen `similarity.method` concerns function first derivatives.
`x.g`	vector: abscissa grid where function g and his first derivatives g' is evaluated. `x.g` must always be provided.
`y0.g`	vector: evaluations of function g on the abscissa grid `x.g`. Default value of `y0.g` is `NULL`. The vector `y0.g` must be provided if the chosen `similarity.method` concerns original functions.
`y1.g`	vector: evaluations of g first derivative, i.e., g', on the abscissa grid `x.g`. Default value is of `y1.g` `NULL`. The vector `y1.g` must be provided if the chosen `similarity.method` concerns function first derivatives.
`similarity.method`	character: similarity/dissimilarity between f and g. Possible choices are: `'d0.pearson'`, `'d1.pearson'`, `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'`, `'d1.L2.centered'`. Default value is `'d1.pearson'`. See details.
`unif.grid`	boolean: if equal to `TRUE` the similarity measure is computed over an uniform grid built in the intersection domain of the two functions, that is an additional discretization is performed. If equal to `FALSE` the additional discretization is not performed, so the functions are supposed to be already defined on the same abscissa grid and the grid is supposed to be fine enough to well compute similarity.

Details

We report the list of the currently available similarities/dissimilarities. Note that all norms and inner products are computed over D, that is the intersection of the domains of f and g. $\overline{f}$ and $\overline{g}$ denote the mean value, respectively, of functions f and g.

1. `'d0.pearson'`: this similarity measure is the cosine of the angle between the two functions f and g.

$$\frac{< f, g >_{L^2}}{\|f\|_{L^2}\|g\|_{L^2}}$$

2. `'d1.pearson'`: this similarity measure is the cosine of the angle between the two function derivatives f' and g'.

$$\frac{< f', g' >_{L^2}}{\|f'\|_{L^2}\|g'\|_{L^2}}$$

3. `'d0.L2'`: this dissimilarity measure is the L2 distance of the two functions f and g normalized by the length of the common domain D.

$$\frac{\|f-g\|_{L^2}}{|D|}$$

4. `'d1.L2'`: this dissimilarity measure is the L2 distance of the two function first derivatives f' and g' normalized by the length of the common domain D.

$$\frac{\|f'-g'\|_{L^2}}{|D|}$$

5. `'d0.L2.centered'`: this dissimilarity measure is the L2 distance of $f-\overline{f}$ and $g-\overline{g}$ normalized by the length of the common domain D.

$$\frac{\|(f-\overline{f})-(g-\overline{g})\|_{L^2}}{|D|}$$

6. `'d1.L2.centered'`: this dissimilarity measure is the L2 distance of $f'-\overline{f'}$ and $g'-\overline{g'}$ normalized by the length of the common domain D.

$$\frac{\|(f'-\overline{f'})-(g'-\overline{g'})\|_{L^2}}{|D|}$$

For multidimensional functions, if `similarity.method='d0.pearson'` or `'d1.pearson'` the similarity/dissimilarity measure is computed via the average of the indexes in all directions.

The coherence properties specified in Sangalli et al. (2010) imply that if `similarity.method` is set to `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'` or `'d1.L2.centered'`, value of `warping.method` must be `'shift'` or `'NOalignment'`. If `similarity.method` is set to `'d0.pearson'` or `'d1.pearson'` all values for `warping.method` are allowed.

Value

scalar: similarity/dissimilarity measure between the two functions f and g computed via the similarity/dissimilarity measure specified.

Author(s)

Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

References

Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*, Computational Statistics and Data Analysis, 54, 1219-1233, 2010.

See Also

`kma`, `kma.compare`, `kma.show.results`, `fdakma`, `kma.data`

Examples

```
data(kma.data)

x.f <- kma.data$x # abscissas of f and f'
x.g <- kma.data$x # abscissas of g and g'

y0.f <- kma.data$y0[1,] # evaluations of f on the abscissa grid x.f
y1.f <- kma.data$y1[1,] # evaluations of f' on the abscissa grid x.f
y0.g <- kma.data$y0[3,] # evaluations of g on the abscissa grid x.g
y1.g <- kma.data$y1[3,] # evaluations of g' on the abscissa grid x.g


# Plot of the two functions f and g
plot(t(x.f),t(y0.f), type='l', xlab='x', ylab='y')
points(t(x.g),t(y0.g), type='l', col='red')
title ('f and g')
legend('bottomleft', legend=c('f','g'),
       col=c('black','red'), lty=c(1,1), cex = 0.5)

# Example: 'd0.pearson'
kma.similarity (x.f=x.f, y0.f=y0.f, x.g=x.g,  y0.g=y0.g, similarity.method='d0.pearson')

# Example: 'd0.L2'
kma.similarity (x.f=x.f, y0.f=y0.f, x.g=x.g,  y0.g=y0.g, similarity.method='d0.L2')


# Plot of the two function first derivatives f' and g'
plot(t(x.f),t(y1.f), type='l', xlab='x', ylab='y')
points(t(x.g),t(y1.g), type='l', col='red')
title ("f' and  g'")
legend('bottomleft', legend=c("f'","g'"),
       col=c('black','red'), lty=c(1,1), cex = 0.5)

# Example: 'd1.pearson'
kma.similarity (x.f=x.f, y1.f=y1.f, x.g=x.g,  y1.g=y1.g, similarity.method='d1.pearson')

# Example: 'd1.L2'
kma.similarity (x.f=x.f, y1.f=y1.f, x.g=x.g,  y1.g=y1.g, similarity.method='d1.L2')
```

Index

Chapter 5

Conclusions

In this work we analyzed in detail the problem of registration of functional data. We tried to capture the qualities of the different continuous registration methods present in literature with the ambitious goal of building solid theoretical foundations for this young and fascinating area of statistics called Functional Data Analysis.

At first we presented the different approaches that solve this problem. Then we studied in deep the continuous approach, describing the *K-Mean Alignment* method presented in Sangalli et al. (2010). We compared it to the other registration methods present in literature, finding that the *K-Mean Alignment* procedure is the only method allowing to jointly cluster and align functional data, that is a remarkable quality with respect to the other methods.

We applied the *K-Mean Alignment* method to the neuronal "spike train" dataset. The aim was to align the neuronal intensities of a monkey doing a particular task with his hand. We finally discovered that there is no need for registration if we want to cluster the four different paths. The k-mean procedure allows to reach this goal in a very satisfying way. In a second approach, we considered data as periodic functions. With this hypothesis we allowed only for shift

registration. We finally found a strong correlation between shifts and paths when searching for one cluster. In the case of two clusters (i.e., the correct number of clusters suggested by the *kma.compare* function of the R package *fdakma*), we discovered that they are not related to the paths. Of course, for an explanation from a neuroscientific point of view, we would require the intervention of a specialist in the sector.

Finally, we developed the R package called *fdakma*, which will be soon available on CRAN repository. When we started this work, a little implementation of the algorithm was already present. We then extended the possibilities of the algorithm: it is now possible to align functional data with several numbers of clusters and types of warping functions and to automatically plot the results of the clustering/alignment procedure. The estimation of the cluster(s) center(s) has also been implemented, so that, in all cases, users can easily recognize the general trend of the functions given by the shape of the final cluster(s) center(s).

As a conclusive remark, let me point out that a lot of improvements can still be done to the *fdakma* package, as well as to the *K-Mean Alignment* procedure.

The *K-Mean Alignment* scheme gives the possibility to insert other couples *measure-warping functions* respecting the rules listed in chapter 2. For example, it would be interesting to implement the *Fisher-Rao* metric in it. In fact the flexibility regarding the space of warping functions of the *Fisher-Rao* metric makes this measure naturally insertable in the framework of Sangalli et al. (2010). At the same time, the *K-Mean Alignment* procedure does not depend on the chosen metric or the space of warping functions we allow for our functions. Hence it is naturally thinkable to extend the *K-Mean Alignment*

algorithm allowing the choice of the *Fisher-Rao* metric as measure to register functional data.

In a future prospective, it would also be useful to work on the complexity of the algorithm. For example, the R package *optim* does all the optimization jobs, but, given that the optimization step represents an important part of the algorithm, we could gain a lot in term of execution time improving its efficiency. Always in this prospective, a parallelization of the code would also be an important amelioration. This is possible because many parts of the code are independent from each other, hence they can easily be parallelized. Of course this would considerably reduce the complexity of the algorithm.

References

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction* (2e éd.). Springer.

R Development Core Team. (2013). R: A Language and Environment for Statistical Computing [Manuel de logiciel]. Vienna, Austria. (ISBN 3-900051-07-0)

Ramsay, J. O., Hooker, G., & Graves, S. (2005). *Functional data analysis with R and MATLAB*. Dordrecht ; Springer, c2009.

Ramsay, J. O., & Silverman, B. (2005). *Functional data analysis*. New York : Springer, c1997.

Ramsay, Jim O. and Silverman, Bernard. (2002). Applied Functional Data Analysis.

Sangalli, L. M., Secchi, P., Vantini, S., & Vitelli, V. (2010). K-Mean Alignment for Curve Clustering. *Computational Statistics & Data Analysis*, *54*(5), 1219–1233.

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., & Marron, J. S. (2011). Registration of Functional Data Using Fisher-Rao Metric.

Team, R. D. C. (2010). Writing R Extensions [Manuel de logiciel].

Tuddenham, R., & Snyder, M. (1954). Physical growth of California boys and girls from birth to age 18. *Univ. Calif. Publ. in Child Develop.*.

Vantini, S. (2012). On the definition of phase and amplitude variability in functional data analysis. *TEST*, *21*(4), 676.

Wu, W., & Srivastava, A. (2011). An information-geometric framework for statistical inferences in the neural spike train space. *Journal of Computational Neuroscience*, *31*(3), 725–48.