journal homepage: www.intl.elsevierhealth.com/journals/cmpb

Kml: A package to cluster longitudinal data

Christophe Genolini[a,b,c,*], *Bruno Falissard*[a,b,d]

[a] *Inserm, U669, Paris, France*
[b] *Univ Paris-Sud and Univ Paris Descartes, UMR-S0669, Paris, France*
[c] *Modal'X,Univ Paris-Nanterre, UMR-S0669, Paris, France*
[d] *AP-HP, Hôpital de Bicêtre, Service de Psychiatrie, Le Kremlin-Bicêtre, France*

ARTICLE INFO

Article history:
Received 21 May 2010
Received in revised form 4 May 2011
Accepted 25 May 2011

Keywords:
Package presentation
Longitudinal data
k-Means
Cluster analysis
Non-parametric algorithm

ABSTRACT

Cohort studies are becoming essential tools in epidemiological research. In these studies, measurements are not restricted to single variables but can be seen as trajectories. Thus, an important question concerns the existence of homogeneous patient trajectories.

KmL is an R package providing an implementation of *k*-means designed to work specifically on longitudinal data. It provides several different techniques for dealing with missing values in trajectories (classical ones like linear interpolation or LOCF but also new ones like copyMean). It can run *k*-means with distances specifically designed for longitudinal data (like Frechet distance or any user-defined distance). Its graphical interface helps the user to choose the appropriate number of clusters when classic criteria are not efficient. It also provides an easy way to export graphical representations of the mean trajectories resulting from the clustering. Finally, it runs the algorithm several times, using various kinds of starting conditions and/or numbers of clusters to be sought, thus sparing the user a lot of manual re-sampling.

1. Introduction

Cohort studies are becoming essential tools in epidemiological research. In these studies, measurements collected for a single subject can be seen as trajectories. Thus, an important question concerns the existence of homogeneous patient trajectories. From a statistical point of view many methods have been developed to deal with this issue [1–4]. In its survey [5] Warren-Liao divide these methods into five families: partitioning methods construct *k* clusters containing at least one individual; hierarchical methods work by grouping data objects into a tree of clusters; density-based methods make clusters grow as long as the density in the "neighborhood" exceeds a certain threshold; grid-based methods quantize the object space and perform the clustering operation on the resulting finite grid structure; model-based methods assume a model for each cluster and look for the best fit of data to the model.

The pros and cons of these approaches are regularly discussed [6,7] even if there is little data to show which method is indeed preferable in which situation. In this paper, we consider *k*-means, a well-known partitioning method [8,9]. In favor of an algorithm of this type the following points can be cited: (1) it does not require any normality or parametric assumptions within clusters (although it might be more efficient under certain assumptions). This might be of great interest when the aim is to cluster data on which no prior information is available; (2) it is likely to be more robust as regards numerical convergence; (3) in the particular

* *Corresponding author at*: Inserm, U669, 97 Bd Port Royal, 75014 Paris, France. Tel.: +33 6 21 48 47 84.
E-mail address: genolini@u-paris10.fr (C. Genolini).
0169-2607/$ – see front matter © 2011 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2011.05.008

context of longitudinal data, it does not require any assumption regarding the shape of the trajectory (this is likely to be an important point: the clustering of longitudinal data is basically an exploratory approach); (4) also in the longitudinal context, it is independent from time scaling.

On the other hand, it also suffers from some drawbacks: (1) formal tests cannot be used to check the validity of the partition; (2) the number of clusters needs to be known a priori; (3) the algorithm is not deterministic, the starting condition is often determined at random. So it may converge to a local optimum and one cannot be sure that the best partition has been found; (4) the estimation of a quality criterion cannot be performed if there are missing values in the trajectories.

Regarding software, numerous versions of *k*-means exist, some with a traditional approach [10,11], some with variations [12–17]. They however have several weaknesses: (1) they are not able to deal with missing values. (2) Since determining the number of clusters is still an open question, they require the user to manually re-run the *k*-means several times.

KmL is a new implementation of *k*-means specifically designed to analyze longitudinal data. Our package is designed for R platform and is available on CRAN [18]. It is able to deal with missing values; it also provides an easy way to run the algorithm several times, varying the starting conditions and/or the number of clusters looked for; its graphical interface helps the user to choose the appropriate number of clusters when the classic criterion is not efficient.

Section 2 presents theoretical aspects of KmL: the algorithm, different solutions to deal with missing values and quality criteria to select the best number of clusters. Section 3 gives a description of the package. Section 4 compares the impact of the different starting conditions. Section 5 is the discussion.

2. Theoretical background

2.1. Introduction to k-means

k-Means is a hill-climbing algorithm [7] belonging to the EM class (Expectation–Maximization) [11]. EM algorithms work as follow: Initially, each observation is assigned to a cluster; then the optimal partition is reached by alternating two phases called respectively "Expectation" and Maximization". During the Expectation phase, the center of each cluster is determined. Then the Maximization consists in assigning each observation to its "nearest cluster". The alternation of the two phases is repeated until no further changes occur in the clusters.

More precisely, consider a set S of n subjects. For each subject, an outcome variable Y at t different times is measured. The value of Y for subject i at time l is noted as y_{il}. For subject i, the sequence y_{il} is called a trajectory, it is noted $y_i = (y_{i1}, y_{i2}, \ldots, y_{it})$. The aim of clustering is to divide S into k homogeneous sub-groups. The notion of the "nearest cluster" is strongly related to the definition of distance. Traditionally, *k*-means can be run using several distances. Euclidean distance is defined as $Dist^E(y_i, y_j) = \sqrt{\sum_{l=1}^{t}(y_{il} - y_{jl})^2}$. Manhattan distance $Dist^M(y_i, y_j) = \sum_{l=1}^{t}|y_{il} - y_{jl}|$ is more robust to outliers [10]. KmL can also work using distances specific to longitudinal data like the Frechet distance or dynamic time warping [19]. Finally, it can work with some user-defined distances thus opening many possibilities.

2.2. Choosing an optimal number of clusters

An unsettled problem with *k*-means is the need to know a priori the number of clusters. A possible solution is to run *k*-means varying the initial number of seeds, and then to select the "best" number of clusters according to some quality criterion.

KmL uses mainly the Calinski & Harabatz criterion $C(k)$ [20]. It has interesting properties, as shown by several authors [21,22]. The Calinski & Harabatz criterion can be defined as follows: let n_m be the number of trajectories in cluster m; $\overline{y_m}$ the mean trajectories of clusters m; $\overline{y}$ the mean trajectory of the whole set S. Let v' denotes the transposition of vector v. The between-clusters covariance matrix is $B = \sum_{m=1}^{k} n_m(\overline{y_m} - \overline{y})(\overline{y_m} - \overline{y})'$. If $trace(B)$ designates the sum of the diagonal coefficients of B, high values of $trace(B)$ denote well-separated clusters, while low values of $trace(B)$ indicate clusters close to each other. The within-cluster covariance matrix is $W = \sum_{m=1}^{k}\sum_{l=1}^{n_m}(y_{ml} - \overline{y_m})(y_{ml} - \overline{y_m})'$. Low values of $trace(W)$ correspond to compact clusters while high values of $trace(W)$ correspond to heterogeneous groups (see [7] for details). The Calinski & Harabazt criterion combines the within and between matrices to evaluate the quality of the partition. The optimal number of clusters corresponds to the value of k that maximizes $C(k) = (trace(B)/trace(W))(n - k/k - 1)$.

If the Calinski & Harabazt criterion can help to select the optimal number of clusters, it has been shown that it does not always find the correct solution [22]. In practice, users often like to have several criteria at their disposal so that their concordance will strengthen the reliability of the result. In addition to the Calinski & Harabatz criterion, KmL calculates two other criteria: Ray & Turi [23] and Davies & Bouldin [24]. Both indices are regularly presented with interesting properties [22].

Since both Shim and Milligan suggest that Calinski & Harabatz is the index with the most interesting properties [21,22], KmL uses it as the main selection criterion. The other two (Ray & Turi and Davies & Bouldin) are available for checking consistency.

2.3. Avoiding a local maximum

One major weakness of hill-climbing algorithms is that they may converge to a local maximum that does not correspond to the best possible partition in terms of homogeneity [8,25]. To overcome this problem, different solutions have been proposed. Some authors [26,27,10] compare several methods of determination of the initial cluster seeds in terms of efficiency. Vlachos et al. [15] propose a "wavelet" *k*-means: an initial clustering is performed on trajectories reduced to a single point (the mean of each trajectory). The results obtained from this "quick and dirty" clustering are used to initialize clustering at a slightly finer level of approximation (each trajectory is reduced to two points). This process is repeated until the finest level of "approximation", the full trajectory, is reached. Sugar

and Hand [28,29] suggest running the algorithm several times, retaining the best solutions. KmL mixes the solutions obtained from different starting conditions and several runs. It proposes three different ways to choose the initial seeds:

- **randomAll**: all the individuals are randomly assigned to a cluster with at least one individual in each cluster. This method produces initial seeds that are close to each other (see Fig. 1(b)).
- **randomK**: k individuals are randomly assigned to a cluster, the other individuals are not assigned. Each seed is a single individual and not an average of several individuals. This method produces initial seeds that are not close to each other, so that this method may produce initial seeds that are from different clusters (possibly one seed in each cluster) which will speed up the convergence (see Fig. 1(c)).
- **maxDist**: k individuals are chosen incrementally. The first two are the individuals that are the farthest from each other. The following individuals are added one at a time and are the individuals farthest from those that are already selected. The "farthest" is the individual with the greatest distance from the selected individuals. If D is the set of the individuals already selected, then the individual to be added is individual i for who $s(i) = \min_{j \in D}(Dist(i, j))$ is maximum (see Fig. 1(d)).

Different starting conditions can lead to different partitions. As for the number of clusters, the "best" solution is the one that maximizes the between-matrix variance and minimizes the within-matrix variance.

2.4. *Dealing with missing values*

There are very few papers that propose cluster analysis methods that deal with missing values [30]. The simplest way to handle missing data is to exclude trajectories for which some data are missing. This can however severely reduce the sample size since longitudinal data are especially vulnerable to missing values. In addition, individuals with particular patterns of missing data can constitute a particular cluster, for example an "early drop-out" group.

KmL deals with missing data at two different stages. First, during clustering, it is necessary to calculate the distance between two trajectories and this calculation can be hampered by the presence of missing data in one of them. To tackle this problem, one can either impute missing values (using methods that we define in the next paragraph) or use classic distances with Gower adjustment [31]: given y_i and y_j, let $w_{ijl}w_{ijl}$ be 0 if y_{il} or y_{jl} or both are missing, and 1 otherwise; the Euclidian distance with Gower adjustment between y_i and y_j is

$$Dist^{E}_{NA}(y_i, y_j) = \sqrt{\frac{t}{\sum_{l=1}^{t} w_{ijl}} \sum_{l=1}^{t} (y_{il} - y_{jl})^2 . w_{ijl}}$$

The second problematic step is the calculation of quality criteria which help in the determination of the optimal partition. At this stage, it can be necessary to impute missing values. The classic "imputation by the mean" is not recommended because it neglects the longitudinal nature of the data and is thus likely to annihilate the cluster structure. KmL proposes several methods. Each deals with one of the three particular situations: missing values at the start of the trajectory (the first values are missing), at the end (the last values are missing) or in the middle (the missing values are surrounded by non-missing values).

The different methods can be described as follows:

- **LOCF** (Last Occurrence Carried Forward): The values missing in the middle and at the end are imputed from the previous non-missing values. For values missing at the start, the first non-missing value is duplicated backwards.
- **FOCB** (First Occurrence Carried Backward): The values missing in the middle and at the start are imputed from the next non-missing values. For values missing at the end, the last non-missing value is duplicated forward.

The next four imputation methods all use linear interpolation for missing values in the middle; they differ for the imputation of values missing at the start or at the end. Linear interpolation imputes by drawing a line between the non-missing values surrounding the missing one(s). If y_{il} is missing, let y_{ia} and y_{ib} be the closest preceding and following non-missing values of y_{ik}; then y_{ik} is imputed by $y_{il} = (y_{ib} - y_{ia})/(b - a)(l - a) + y_{ia}$.

For missing values at the start and at the end, different options are possible:

- **Linear interpolation, OCBF** (Occurrences Carried Backward and Forward): Missing values at the start and at the end are imputed using FOCB and missing values at the end are imputed using LOCF (see Fig. 2(a)).
- **Linear interpolation, Global**: The values missing at the start and the end are imputed on a line joining the first and the last non-missing values (dotted line in Fig. 2(b)). If y_{il} is missing at the start or the end, let y_{is} and y_{ie} be the first and the last non-missing values; then y_{il} is imputed by $y_{il} = (y_{ie} - y_{is})/(e - s) \cdot (l - s) + y_{is}$.
- **Linear interpolation, Local**: Missing values at the start and at the end are imputed locally "in continuity" (prolonging the closest non-missing values, see Fig. 2(c)). If y_{il} is missing at the start, let y_{is} and y_{ia} be the first and the second non-missing values; then y_{il} is imputed by $y_{il} = (y_{ia} - y_{is})/(a - s) \cdot (l - s) + y_{is}$. If y_{il} is missing at the end, let y_{ie} and y_{ib} be the last and the penultimate non-missing value; then y_{il} is imputed by $y_{il} = (y_{ie} - y_{ib})/(e - b) \cdot (l - b) + y_{ib}$.
- **Linear interpolation, Bisector**: the method **Linear interpolation, Local** is very sensitive to the firsts and lasts values; the method **Linear interpolation, Global** ignores developments close to the end or to the start. **Linear Interpolation, Bisector** offers a mixed solution by considering an intermediate line, the bisector between the global and the local lines (see Fig. 2(d)).

Last method, **copyMean** is an imputation method that is available only when clusters are known. The main idea is to impute using linear interpolation, then to add a variation to make the trajectory follow the "shape" of the mean trajectories. If y_{il} is missing in the middle, let y_{ia} and y_{ib} be the closest preceding and following non-missing values

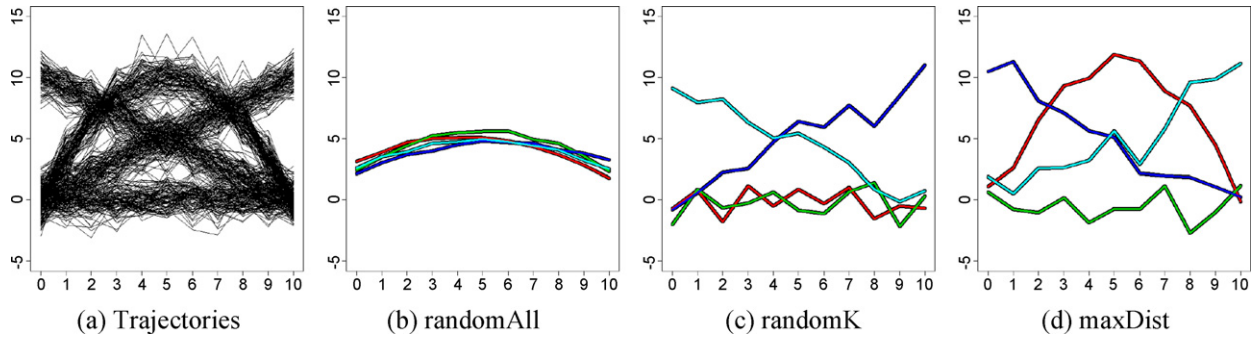

Fig. 1 – Examples of different ways to choose initial seeds. (a) Shows the trajectories (there are obviously four clusters). (b) Uses the `randomAll` method, the seeds are the mean of several individuals so they are close to each other; (c) uses the `randomK` method, the seeds are single individuals. The four initial seeds are in three different clusters. (d) Uses the `maxDist` method, the seeds are individuals far away from each other.

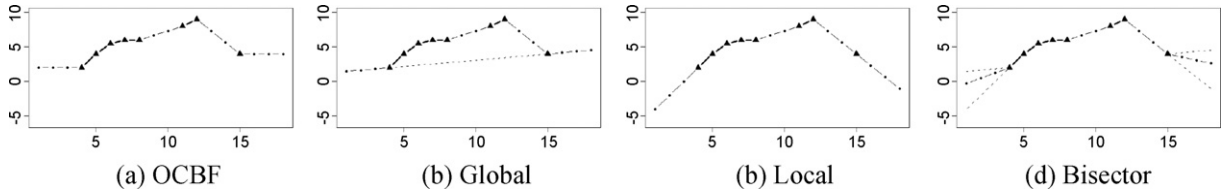

Fig. 2 – Different variations of linear interpolation. Triangles are known values and dots are imputed values.

of y_{il}; let $\overline{y_m} = (\overline{y_{m_1}}, \ldots, \overline{y_{m_t}})$ denote the mean trajectory of y_i cluster. Then $y_{il} = (y_{ib} - y_{ia})/(\overline{y_{mb}} - \overline{y_{ma}}) \cdot (\overline{y_{ml}} - \overline{y_{ma}}) + y_{ia}$. If the first values are missing, let y_{is} be the first non-missing value. Then $y_{il} = \overline{y_{ml}} + (y_{is} - \overline{y_{ms}})$. If the last values are missing, let y_{ie} be the last non-missing value. Then $y_{il} = \overline{y_{ml}} + (y_{ie} - \overline{y_{me}})$.

Fig. 3 gives examples of mean shape-copying imputation (the mean trajectory $\overline{y_m} = (\overline{y_{m_1}}, \ldots, \overline{y_{m_t}})$ is drawn with white circles).

3. Package description

In this section, the content of the package is presented (see Fig. 4).

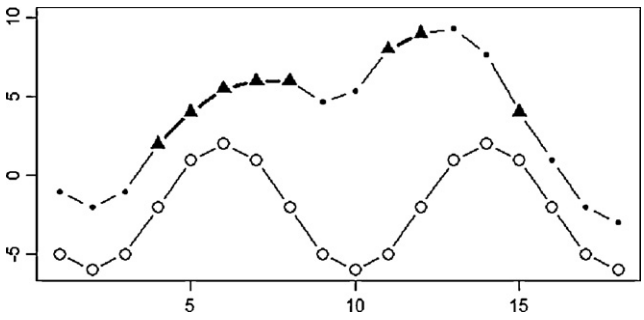

Fig. 3 – copyMean imputation method; triangles are known values, dots are imputed value and circles are the mean trajectory.

3.1. *Preparation of the data*

One advantage of KmL is that the algorithm memorizes all the clusters that it finds. To do this, it works on a S4 structure called `ClusterizLongData`. A `ClusterizLongData` object has two main fields: `traj` stores the trajectories; `clusters` is a list of all the partitions found. Data preparation therefore simply consists in transforming longitudinal data into a `ClusterizLongData` object. This can be done via function `cld()` or `as.cld()`. The first lets the user build data, the second converts a `data.frame` "wide" format (each line corresponds to one individual, each column is one time) into a `ClusterizLongData` object. `cld()` uses the following arguments (the type of the argument is given in brackets):

- `traj` [array of numeric]: contains the longitudinal data. Each line is the trajectory of an individual. The columns refer to the time at which measures were performed.
- `id` [character]: single identifier for each individual (each trajectory).
- `time` [numeric]: time at which measures were performed.
- `varName` [character]: name of the variable, for the graphic output.
- `trajMinSize` [numeric]: Trajectories whose values are partially missing can either be excluded by processing, or included. `trajSizeMin` sets the minimum number of values that a trajectory must contain not to be excluded. For example, if trajectories have 7 measurements (time = 7) and `trajSizeMin` is set to 3, the trajectory (5,3,na,4,na,na,na) will be included in the calculation while (2,na,na,na,4,na,na) will be excluded.

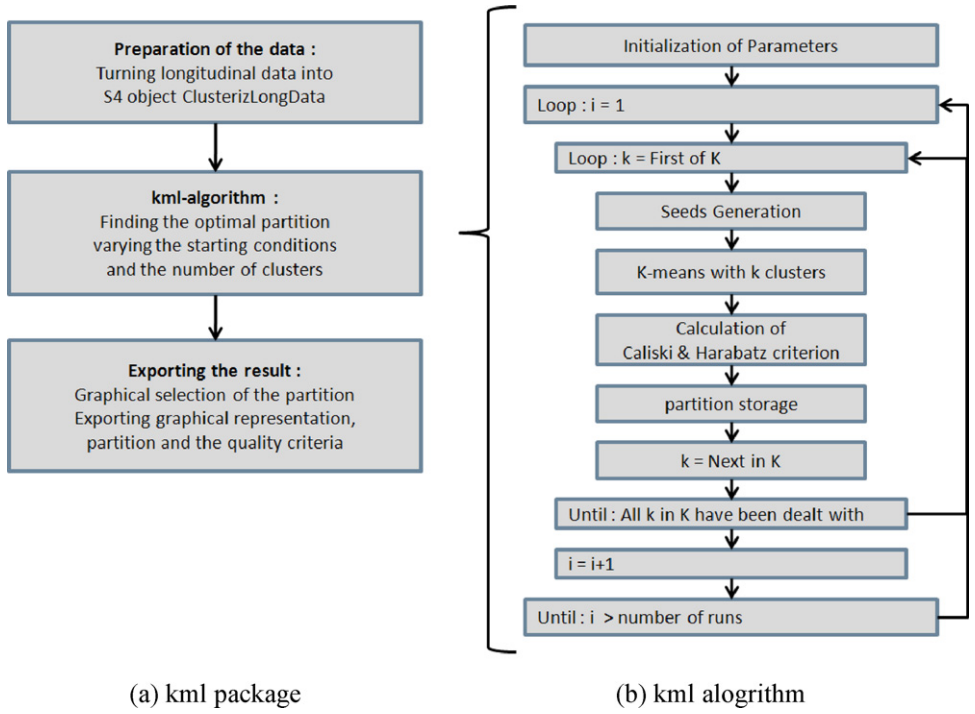

Fig. 4 – Package description.

3.2. *Finding the optimal partition*

Once an object of class `ClusterizLongData` has been created, the `kml()` function can be called. `kml()` runs *k*-means several times varying starting conditions and the number of clusters. The starting condition can be `randomAll`, `randomK` or `maxDist` as described in Section 2.3. In addition, the `allMethods` method combines the three previous ones by running one `maxDist`, one `randomAll` and then `randomK` for all the other iterations.

By default, `kml()` runs *k*-means for $k = 2, 3, 4, 5, 6$ clusters, 20 times each using `allMethods`.

The *k*-means version used here is the Hartigan and Wong version (1979). The default distance is the Euclidean distance with Gower adjustment. The six distances defined in the `dist()` function ("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski") and the Frechet distance are also available. Finally, `kml()` can work with user-defined distances through the optional argument `distance`. If provided, `distance` should be a function that takes two trajectories and returns a number, letting the user compute a non-classical distance (like the adaptive dissimilarity index or dynamic time warping [19]).

Every partition found by `kml()` is stored in the `ClusterizLongData` object. The field `Cluster` is a list of sublists, each sublist corresponding to a specific number of clusters (for example; the sublist `c3` stores all the partitions with 3 clusters). The storage is performed in real time. If `kml()` is interrupted before the computation ends, the partitions already found are not lost. When `kml()` is re-run several times on the same data, the new partitions found are added to the previous ones. This is convenient when the user asks for several runs, then realizes that the result is not optimal and asks for further runs.

In addition, `kml()` saves all the partitions on the hard disc at frequent intervals to guard against any system interruption that may occurs when the algorithm is run for a long time (days or weeks).

The main options of `kml()` are:

- `Object` [ClusterizLongData]: contains trajectories to cluster and all the partition already found.
- `nbClusters` [vector(numeric)]: Vector containing the number of clusters with which `kml()` must work. By default, nbClusters is 2:6 which indicates that `kml()` must search for partitions starting from 2, then 3, ... up to 6 clusters.
- `nbRedrawing`: [numeric]: Sets the number of times that *k*-means must be re-run (with different starting conditions) for each number of clusters.
- `saveFreq`: [numeric]: Long computations can take several days. So it is possible to save the object `ClusterizLongData` at intervals. `saveFreq` defines the frequency of the saving process.
- `maxIt` [numeric]: Sets a limit to the number of iterations if convergence is not reached.
- `imputationMethod`: [character]: the calculation of the quality criterion cannot be performed if some values are missing. `imputationMethod` defines the method used to impute the missing values. It should be one of "LOCF", "FOCB", "LI-Global", "LI-Local", "LI-Bisector", "LI-OCFB" or "copyMean" as presented in Section 2.4.
- `distance` [numeric ← function(trajectory,trajectory)]: function that computes the distance between two trajectories. If

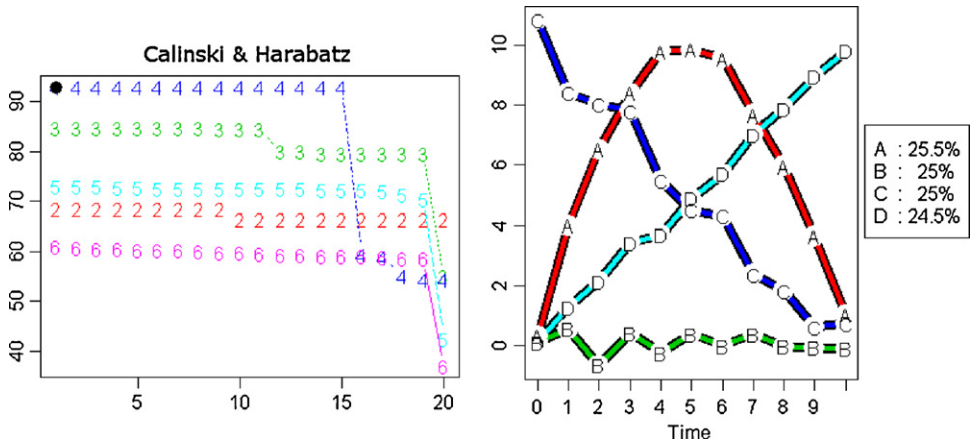

Fig. 5 – Function `choice()`.

no function is specified, the Euclidian distance with Gower adjustment is used.

- `startingCond` [character]: specifies the starting condition. It should be one of "maxDist", "randomAll", "randomK" or "allMethods" as presented in Section 2.3.

3.3. *Exporting results*

When `kml()` has found some partitions, the user can decide to select and export some of them. This can be done via the function `choice()`. `choice()` opens a graphic windows showing information: on the left, all the partitions stored in the object are represented by a number (the number of clusters it comprises). Partitions with the same cluster number are sorted according to the Calinski & Harabatz criterion in decreasing order, the best coming first. From all the partitions, one is selected (black dot). The means trajectories it defines are presented on the right-hand side of the windows (see Fig. 5).

The user can decide to export the partition with the highest Calinski & Harabatz criterion value. But since quality criteria are not always as efficient as one might expect, he can also visualize different partitions and decide which he wants to export, according to some other criterion.

When partitions have been selected (the user can select any number), `choice()` saves them. The clusters are therefore exported to a `csv` file; the Calinski & Harabatz criterion, the percentage of individuals in each cluster and various other parameters are exported towards a second file. Graphical representations are exported in the format specified by the user. `choice()` arguments are:

- `Object` [ClusterizLongData]: Object containing the trajectories and all the partitions found by `kml()` from which the user want to make an export.
- `typeGraph` [character]: For every selected partition, `choice()` exports some graphs, type sets the format that will be used. Possible formats are those available for `savePlot()`: "wmf", "emf", "png", "jpg", "jpeg", "bmp", "tif", "tiff", "ps", "eps" and "pdf".

3.4. *Reliability of results*

As we noted in Section 1, quality criteria are not always efficient. Using several of them might strengthen the reliability of the results. The function `plotAllCriterion` displays the three criteria estimated by the algorithm (Calinsky & Harabatz, Ray & Turi, Davies & Bouldin). In order to plot them on the same graph, they are mapped into [0,1]. In addition, while the Davies & Bouldin is a criterion that should be minimized, `plotAllCriterion` considers its opposite (and thus it becomes a criterion that has to be maximized, like the other two). Fig. 6 gives an example of concordant and discordant criteria.

4. Sample runs and example

4.1. *Artificial data sets*

To test `kml()` and compare the efficiency of its various options, we used simulated longitudinal data. We constructed the data as follows: a data set is the mixture of several sub-groups. A subgroup m is defined by a function $f_m(x)$ called the *theoretical trajectory*. Each subject i of a sub-group follows the theoretical trajectory of its subgroup plus a personal variation $\varepsilon_i(x)$. The mixture of different theoretical trajectories is called the data set shape. The final construction is performed with the function `gald()` (Generate Artificial Longitudinal Data). It takes the cluster sizes, the number of time measurements, functions that define the theoretical trajectories and noise for each cluster. In addition, for each cluster, it is possible to decide that a percentage of values will be missing.

To test kml, 5600 data sets were formed varying the data set shape, the number of subjects in each cluster and the personal variations. We defined four data set shapes:

1. **(a) Three diverging lines** is defined by $f_A(x) = -x$; $f_B(x) = 0$; $f_C(x) = x$ with x in [0:10].
2. **(b) Three crossing lines** is defined by $f_A(x) = 2$; $f_B(x) = 10$; $f_C(x) = 12 - 2x$ with x in [0:6].
3. **(c) Four normal laws** is defined by $f_A(x) = \mathcal{N}(x - 20, 4)$; $f_B(x) = \mathcal{N}(x - 25, 4)$; $f_C(x) = \mathcal{N}(x - 30, 4)$; $f_D(x) = \mathcal{N}(x - 25, 16)/2$ with

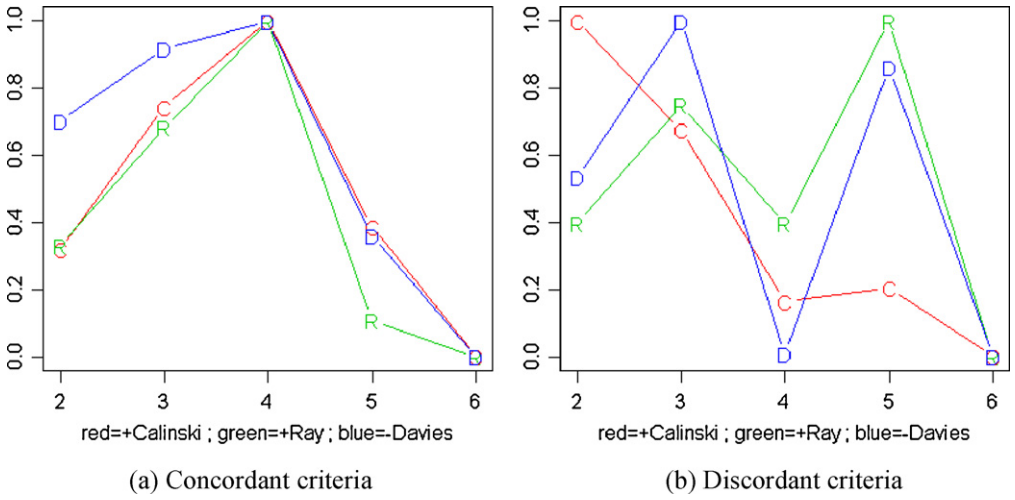

Fig. 6 – Function `plotAllCriterion()`: (a) all the criteria suggest 4 clusters and (b) the criteria does not agree on the number of clusters.

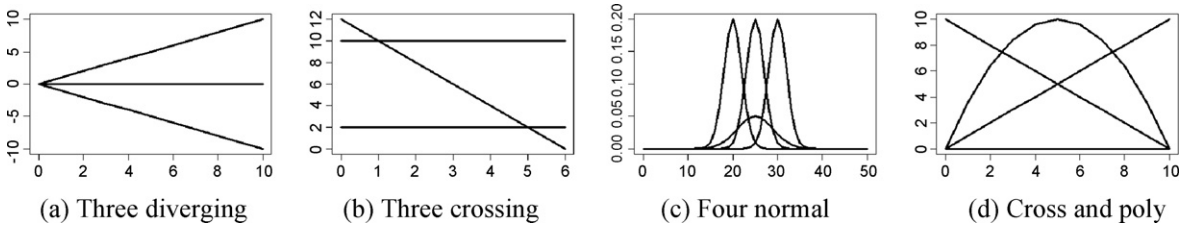

Fig. 7 – Trajectory shapes.

x in [0:50] (where $\mathcal{N}(\mu, \sigma^2)$ is the normal law of mean μ and standard deviation σ).

4. **(d) Crossing and polynomial** is defined by $f_A(x) = 0$; $f_B(x) = x$; $f_C(x) = 10 - x$; $f_D(x) = -0.4x^2 + 4x$ with x in [0:10].

Trajectory shapes are presented Fig. 7.

They were chosen either to correspond to three clearly identifiable clusters (set (a)), or to present a complex structure (every trajectory intersecting all the others (set (d))), or to copy some real data (sets (b) and (c)). Personal variations $\varepsilon_i(x)$ are randomised and follow the normal law $\mathcal{N}(0, \sigma^2)$. Standard deviations step from $\sigma = 0.5$ to $\sigma = 8$ (by steps of 0.01). Since the distance between two theoretical trajectories is around 10, $\sigma = 0.5$ provides some "easily identifiable and distinct clusters" whereas $\sigma = 8$ give some "markedly overlapping groups".

The number of subjects in each cluster is set at either 50 or 200.

In all, 4 (data set shape) × 750 (variance) × 2 (number of subjects) = 6000 data sets were created.

4.2. *Results*

Let P be a partition found by the algorithm and T the "true" partition (since we are working on artificial data, T is known). The correct classification rate (CCR) is the percentage of trajectories that are in the same cluster in P and T [32] that is the percentage of subjects for whom an algorithm makes the right decision. To compare the starting conditions, we modelled the impact on the various factors (starting conditions, number of iterations, standard deviation, size of the groups and data set shapes) on the CCR using a linear regression.

Of the three starting methods, `maxDist` is the most efficient (see Fig. 8(a)). On the other hand, `maxDist` is a deterministic method. As such, it can be run only once, whereas the two other starting conditions can be run several times. Between `randomAll` and `randomK`, the former gives better results but the difference tends to disappear as the number of runs increases (see Fig. 8(b)). Overall, `allMethod` which combines the three other starting conditions is the most efficient method.

4.3. *Application to real data*

We also tested KmL on real data. To assess its performance, we compared it to Proc Traj, semi-parametric procedure widely used to cluster longitudinal data [33]. The first example is derived from [34]. In a sample of 1492 children, daily sleep duration was reported by the children's mothers at ages 2.5, 3.5, 4, 5, and 6. The aim of the study was to investigate the associations between longitudinal sleep duration patterns and behavioural/cognitive functioning at school entry. On this data, KmL finds an optimal solution for a partition into four clusters, as does Proc Traj. The partitionings found by the two procedures are very close (see Fig. 9). The average distance between observed trajectories found by Proc Traj and by KmL

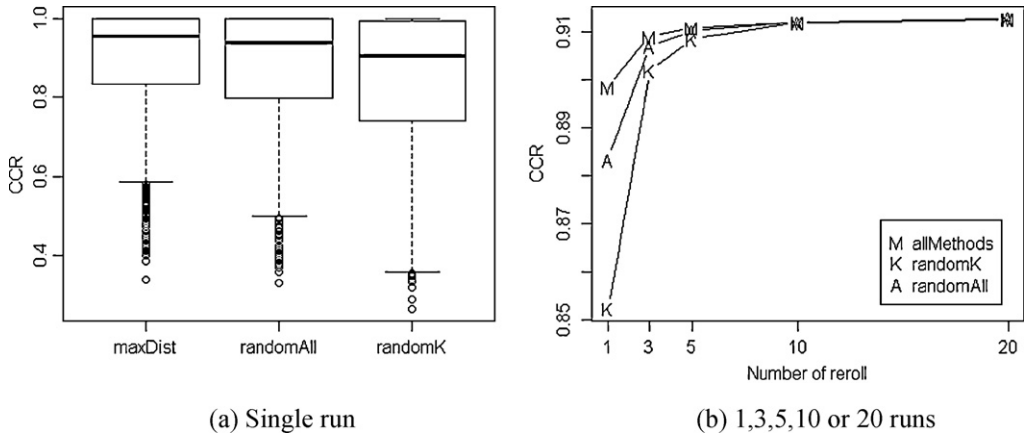

Fig. 8 – CCR according the different starting methods.

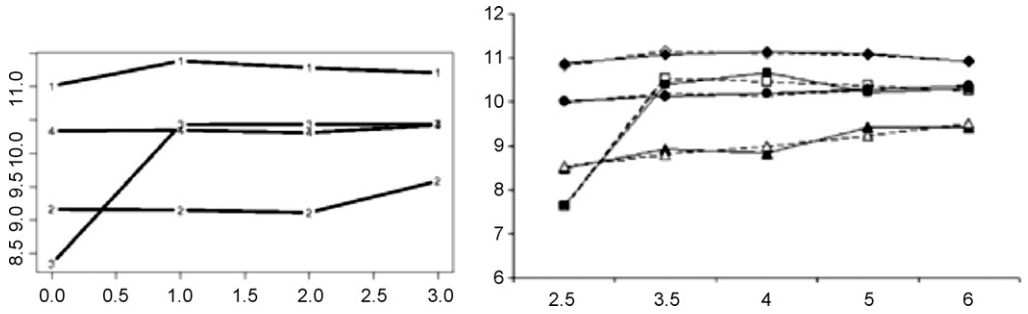

Fig. 9 – Mean trajectories found by KmL (on the left) and Proc Traj (on the right).

is 0.31, which is rather small considering the range of the data (0;12).

The second example comes from an epidemiological survey on anorexia nervosa. The survey, conducted by Dr. Nathalie Godard, focuses on 331 patients hospitalized for anorexia. Patients were followed for 0–26 years retrospectively at their first admission, and prospectively thereafter. One of the variables of interest is the annual time spent in hospital. The authors sought to determine whether there were homogeneous subgroups of patients for this variable. On these data, KmL found an optimal solution for a partition into three clusters. Depending on the number of clusters specified in the program, Proc Traj either stated a "false convergence" or gave incoherent results. The trajectories found by KmL are presented in Fig. 10.

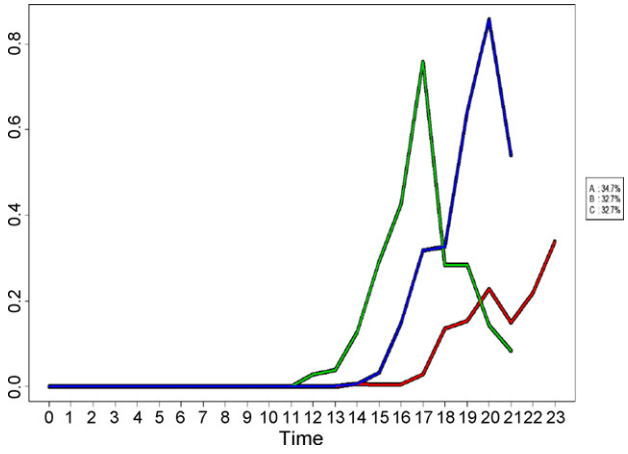

Fig. 10 – Trajectories of the evolution of hospitalisation length.

5. Discussion

5.1. *Overview*

KmL is a new implementation of *k*-means specifically designed to cluster longitudinal data. It can work either with classical distance (Euclidean, manhattan, Minkovski, etc.), with a distance dedicated to longitudinal data (Frechet, dynamic time warping) or with any user-defined distance. It is able to deal with missing values, using either using Gower adjustment or several imputation methods that are provided. It also provides an easy way to run the algorithm several times, varying the starting conditions and/or the number of clusters looked for. As *k*-means is non-deterministic, varying the starting condition increases the chances of finding a global maximum. Varying the number of clusters enables selection of the correct number of clusters. For this purpose, KmL provides three quality criteria whose effectiveness has been demonstrated several times. These are Calinsky & Harabatz, Ray & Turi and Davies & Bouldin. Finally, the graphical interface makes it possible to visualize (and export) the dif-

ferent partitions found by the algorithm. This gives the user the possibility of choosing the appropriate number of clusters when classic criteria are not efficient.

5.2. *Limitations*

KmL nevertheless suffers from a number of limitations inherent to the *k*-means, non-parametric algorithms and partitioning algorithms. First, clusters found by KmL are spherical. Consequently, these clusters all have more or less the same variance. In the case of a population composed of groups with different variances, KmL would have difficulty identifying the correct clusters. In addition, KmL is nonparametric, which is an advantage in some circumstances, but also a weakness. Indeed, it is not possible to test the fit between the partition found and a theoretical model, nor to calculate a likelihood. Finally, like all partitioning algorithms, KmL is unable to give truly reliable and accurate information on the number of clusters.

5.3. *Perspectives*

A number of unsolved problems need investigation. The optimization of cluster number is a long-standing and important question. Perhaps the particular situation of longitudinal data and the strong correlation between different measurements could lead to an efficient solution which is still lacking in the general context of cluster analysis. Another interesting point is the generalization of KmL to problems of higher dimension. At this time, KmL deals only with longitudinal trajectories for a single variable. It would be interesting to develop it for multidimensional trajectories, considering several facets of a patient jointly.

Conflict of interest statement

The author Genolini declare that he had certified that potential conflicts about this manuscript do not exist, have no relevant financial interests in this manuscript, and had full access to all the real data used in the study and take the responsibility for the integrity of the data analysis.

REFERENCES

[1] T. Tarpey, K. Kinateder, Clustering functional data, Journal of Classification 20 (2003) 93–114.
[2] F. Rossi, B. Conan-Guez, A.E. Golli, Clustering functional data with the SOM algorithm, in: Proceedings of ESANN, 2004, pp. 305–312.
[3] C. Abraham, P. Cornillon, E. Matzner-Lober, N. Molinari, Unsupervised curve clustering using B-splines, Scandinavian Journal of Statistics 30 (2003) 581–595.
[4] G. James, C. Sugar, Clustering for sparsely sampled functional data, Journal of the American Statistical Association 98 (2003) 397–408.
[5] T. Warren-Liao, Clustering of time series data—a survey, Pattern Recognition 38 (2005) 1857–1874.
[6] J. Magidson, J.K. Vermunt, Latent class models for clustering: a comparison with K-means, Canadian Journal of Marketing Research 20 (2002) 37.
[7] B.S. Everitt, S. Landau, M. Leese, Cluster Analysis, 4th edn, A Hodder Arnold Publication, 2001.
[8] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, vol. 1, 1966, pp. 281–296.
[9] J. Hartigan, M. Wong, A K-means clustering algorithm, Journal of the Royal Statistical Society Series C—Applied Statistics 28 (1979) 100–108.
[10] Rousseeuw, Kaufman, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, 1990.
[11] G. Celeux, G. Govaert, A classification EM algorithm for clustering and two stochastic versions, Computational Statistics and Data Analysis 14 (1992) 315–332.
[12] S. Tokushige, H. Yadohisa, K. Inada, Crisp and fuzzy k-means clustering algorithms for multivariate functional data, Computational Statistics 22 (2007) 1–16.
[13] T. Tarpey, Linear transformations and the k-means clustering algorithm: applications to clustering curves, The American Statistician 61 (2007) 34.
[14] L.A. García-Escudero, A. Gordaliza, A proposal for robust curve clustering, Journal of Classification 22 (2005) 185–201.
[15] M. Vlachos, J. Lin, E. Keogh, D. Gunopulos, A wavelet-based anytime algorithm for K-means clustering of time series, in: 3rd SIAM International Conference on Data Mining, May 1–3, 2003, Workshop on Clustering High Dimensionality Data and Its Applications, San Francisco, CA, 2003.
[16] P.D. Urso, Fuzzy C-means clustering models for multivariate time-varying data: different approaches, International Journal of Uncertainty Fuzziness and Knowledge Base Systems 12 (2004) 287–326.
[17] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, S.J. Brown, Incremental genetic K-means algorithm and its application in gene expression data analysis, BMC Bioinformatics 5 (2004).
[18] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2009, ISBN 3-900051r-r07-0.
[19] A.D. Chouakria, P.N. Nagabhushan, Adaptive dissimilarity index for measuring time series proximity, Advances in Data Analysis and Classification 1 (2007) 5–21.
[20] T. Calinski, J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics 3 (1974) 1–27.
[21] G.W. Milligan, M.C. Cooper, An examination of procedures for determining the number of clusters in a data set, Psychometrika 50 (1985) 159–179.
[22] Y. Shim, J. Chung, I. Choi, A comparison study of cluster validity indices using a nonhierarchical clustering algorithm, in: Proceedings of CIMCA-IAWTIC'05-Volume 01, IEEE Computer Society, Washington, DC, 2005, pp. 199–204.
[23] S. Ray, R. Turi, Determination of number of clusters in k-means clustering and application in colour image segmentation, in: Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99), Calcutta, India, 1999, pp. 137–143.
[24] D. Davies, D. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (1979) 224–227.
[25] S. Selim, M. Ismail, K-means-type algorithms: a generalized convergence theorem and characterization of local optimality, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (1984) 81–86.
[26] J. Peña, J. Lozano, P. Larrañaga, An empirical comparison of four initialization methods for the K-means algorithm, Pattern Recognition Letters 20 (1999) 1027–1040.
[27] E. Forgey, Cluster analysis of multivariate data: efficiency vs. interpretability of classification, Biometrics 21 (1965) 768.

[28] C. Sugar, G. James, Finding the number of clusters in a dataset: an information-theoretic approach, Journal of the American Statistical Association 98 (2003) 750–764.
[29] D. Hand, W. Krzanowski, Optimising k-means clustering results with standard software packages, Computational Statistics and Data Analysis 49 (2005) 969–973.
[30] L. Hunt, M. Jorgensen, Mixture model clustering for mixed data with missing information, Computational Statistics and Data Analysis 41 (2003) 429–440.
[31] J. Gower, A general coefficient of similarity and some of its properties, Biometrics 27 (1971) 857–871.
[32] T.P. Beauchaine, R.J. Beauchaine, A comparison of maximum covariance and K-means cluster analysis in classifying cases into known taxon groups, Psychological Methods 7 (2002) 245–261.
[33] D.S. Nagin, Analyzing developmental trajectories: a semiparametric, group-based approach, Psychological Methods 4 (1999) 139–157.
[34] E Touchette, D. Petit, J. Seguin, M. Boivin, R. Tremblay, J. Montplaisir, Associations between sleep duration patterns and behavioral/cognitive functioning at school entry, Sleep 30 (2007) 1213–1219.