kml and kml3d : R Packages to Cluster Longitudinal Data

Article *in* Journal of statistical software · May 2015

DOI: 10.18637/jss.v065.i04

CITATIONS
73

READS
1,786

4 authors, including:

Christophe Genolini
Université Paris Ouest Nanterre La Défense
46 PUBLICATIONS **1,035** CITATIONS
SEE PROFILE

Mariane Sentenac
French Institute of Health and Medical Research
57 PUBLICATIONS **651** CITATIONS
SEE PROFILE

Journal of Statistical Software

May 2015, Volume 65, Issue 4. *http://www.jstatsoft.org/*

kml and kml3d: R Packages to Cluster Longitudinal Data

Christophe Genolini
INSERM U.1027 *and* CeRSM EA 2931

Xavier Alacoque
INSERM U.1027 *and* CHU Toulouse

Mariane Sentenac
INSERM U.1027 *and*
Université Paul-Sabatier, Toulouse

Catherine Arnaud
INSERM U.1027, CHU Toulouse, *and*
Université Paul-Sabatier, Toulouse

Abstract

Longitudinal studies are essential tools in medical research. In these studies, variables are not restricted to single measurements but can be seen as variable-trajectories, either single or joint. Thus, an important question concerns the identification of homogeneous patient trajectories.

kml and **kml3d** are R packages providing an implementation of k-means designed to work specifically on trajectories (**kml**) or on joint trajectories (**kml3d**). They provide various tools to work on longitudinal data: imputation methods for trajectories (nine classic and one original), methods to define starting conditions in k-means (four classic and three original) and quality criteria to choose the best number of clusters (four classic and one original). In addition, they offer graphic facilities to "visualize" the trajectories, either in 2D (single trajectory) or 3D (joint-trajectories). The 3D graph representing the mean joint-trajectories of each cluster can be exported through LaTeX in a 3D dynamic rotating PDF graph (Figures 1 and 9).

Keywords: package, longitudinal data, trajectories, joint-trajectories, k-means, cluster analysis, non-parametric algorithm.

1. Introduction

Longitudinal studies are becoming essential tools in epidemiological research. In these studies, the same variables are measured repeatedly over time. This enables the evolution of a parameter of interest over time to be examined. This kind of variables will be referred to as

"variable-trajectory". Because longitudinal datasets usually include a large number of variables, a key issue is to study the joint evolution of several variable-trajectories. This kind of variable will be referred to as joint-trajectories.

A standard way to work with variable-trajectories is to cluster them into distinct groups of patients with homogeneous characteristics. One strength of these classification methods is that they enable the conversion of several correlated continuous variables into a single categorical variable. The categories obtained can then be used, for instance, in a regression model, either as predictors or as dependent variables. Various methods have been designed to this purpose (Tarpey and Kinateder 2003; Rossi, Conan-Guez, and Golli 2004; Nagin 2005; Muthén and Muthén 2012) but they all present several weaknesses.

1. Most methods cluster according to a single variable-trajectory (except some model-based packages like **mixAK**, see Komárek 2009; Komárek, Hansen, Kuiper, van Buuren, and Lesaffre 2010). To date, joint-trajectories are mainly clustered by (1) clustering each single variable-trajectory independently, and (2) considering the combination obtained by crossing the partitions, which is not a convenient solution because it does not detect any complex interactions occurring between variables.

2. There is no reliable method to determine the "true" number of clusters in a dataset (Everitt, Landau, and Leese 2001). Various indices have been suggested, either non-parametric (Caliński and Harabasz 1974; Ray and Turi 1999; Davies and Bouldin 1979; average silhouette, Rousseeuw 1987, cross validation, ...) or parametric (Schwarz 1978; Akaike 1974; Hurvich and Tsai 1989), but they are not completely satisfactory (Shim, Chung, and Choi 2005; Milligan and Cooper 1985). This issue remains largely unsolved.

3. The problem of missing data is almost ubiquitous in observational research. This is particularly true in longitudinal studies where at least one follow-up assessment is often missing (Little 1993; Hedeker and Gibbons 1997; Mallinckrodt, Lane, Schnell, Peng, and Mancuso 2008; Laird 1988). The appropriate handling of dropout and withdrawal is still a key issue. The classic management techniques may result in considerable loss of information, especially when all patients with at least one missing measure are removed from the analyses. The development of imputation methods, such as imputation according to the mean or the classic LOCF (last observation carried forward) entails a lesser loss of information, but can generate results that are just as biased (Engels and Diehr 2003).

4. Partitioning techniques often rely on mathematical theories whose considerable complexities constitute a barrier for actual implementation. Therefore, algorithms leading to approximate solutions are commonly used. In the case of k-means, given an initial configuration the algorithm converges towards a maximum. But there is no way to be sure that this maximum is the global or a local maximum. One solution can be to run k-means several times from different initial configurations. Then eventually, one configuration will lead to the global maximum. But even so, the user has no certainty about the optimality of the partition obtained.

kml (Genolini 2015a; Genolini and Falissard 2010) and **kml3d** (Genolini 2015b; Genolini *et al.* 2013b) are two R (R Core Team 2015) packages that implement k-means in the context of longitudinal data. **kml** is designed to work specifically on single trajectories while **kml3d**

Figure 1: Example of 3D rotating graph. To move the graph, grab it with the mouse left button, then move the mouse without releasing the button.

clusters joint trajectories. They offer different solutions to the issues tackled above. Both packages are available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=kml and http://CRAN.R-project.org/package=kml3d.

1. Package **kml3d** clusters several variable-trajectories jointly. This summarizes several continuous correlated variables (the joint-trajectories) into a single nominal variable (group) that resumes the information contained in the correlated variables. This makes the use of this information for further statistical analysis much easier. In addition, package **kml3d** provides tools to visualize 3D dynamic graphs in an R session and offers the possibility of exporting them in a 3D dynamic PDF file (Figure 1)[1]. This visualization enables a better representation of the interaction between the two variable-trajectories.

2. The problem of selecting the number of clusters remains thus far unsolved. Nevertheless, various quality criteria have been proposed to choose the "right" number of clusters. As often when several solutions exist, none is fully satisfactory. Packages **kml** and **kml3d**

[1]This feature may not work with all PDF readers. If it does not appear in the current document, try to use Acrobat Reader, version 11.0.07 or later.

therefore offer various quality indices, mainly the ones that show the best performances according to the scientific literature (Shim *et al.* 2005; Milligan and Cooper 1985). Depending on their field of interest and their knowledge, users can choose the one that is the most appropriate for their particular study. Furthermore, when several criteria lead to the same number of clusters, it increases the confidence that one can have in the results.

3. To handle missing values, they both provide eleven standard imputation methods (including LOCF, linear interpolation, imputation either by mean or regression) and one original method: "CopyMean" (Genolini and Falissard 2011; Genolini, Écochard, and Jacqmin-Gadda 2013a).

4. The problem of local or global maximum is also an open question. Packages **kml** and **kml3d** do not solve it, but offer six different ways to set an initial configuration, including two innovative methods.

Finally, packages **kml** and **kml3d** integrate all those tools into a single, "user-friendly" function. They automatically perform the k-means algorithm for a different number of clusters, enabling multiple draws for each of them and varying the initialization methods.

This article presents these two packages. Section 2 introduces the statistical techniques used. Section 3 details the algorithms and the main functions. Section 4 gives examples of use. Section 5 is the discussion.

2. Statistical techniques

2.1. Introduction to k-means

K-means is a hill-climbing algorithm belonging to the EM class (expectation-maximization; Celeux and Govaert 1992). EM algorithms work as follows: initially, each observation is assigned to a cluster. Then the optimal clustering is reached by alternating two phases. During the expectation phase, the centers of the different clusters (known as seeds) are computed. The maximization phase then 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.

2.2. Notations

Let S be a set of n subjects. For each subject, m outcome variables $Y_{..A}$, $Y_{..B}$, ..., $Y_{..M}$ at t different times are measured. $Y_{..A}$ is called a single variable-trajectory (or variable-trajectory). Several variable-trajectories $(Y_{..A}, Y_{..B}, \ldots, Y_{..M})$ considered jointly are called joint variable-trajectories. For subject i, the value of $Y_{..A}$ at time j is denoted as y_{ijA}. The sequence $y_{i.A} = (y_{i1A}, y_{i2A}, \ldots, y_{itA})$ is called a single trajectory (or trajectory). Several single trajectories

$$y_{i..} = \begin{pmatrix} y_{i.A} \\ y_{i.B} \\ \vdots \\ y_{i.M} \end{pmatrix}$$

are called joint trajectories. $y_{i..}$ can also be written as a matrix

$$y_{i..} = \begin{pmatrix} y_{i1A} & y_{i2A} & \cdots & y_{itA} \\ y_{i1B} & y_{i2B} & \cdots & y_{itB} \\ \vdots & & & \\ y_{i1M} & y_{i2M} & \cdots & y_{itM} \end{pmatrix},$$

where lines are single variable trajectories. If j is fixed, the sequence

$$y_{ij.} = \begin{pmatrix} y_{ijA} \\ y_{ijB} \\ \vdots \\ y_{ijM} \end{pmatrix}$$

is called the individual's state at time j. The individual's state at time j is the jth column of the matrix $y_{i..}$. The aim of clustering is to divide S into k homogeneous sub-groups.

2.3. Clustering joint-trajectories

The two main problems encountered in partitioning joint-trajectories are the choice of the distance between individuals and problems of relative weight of variables measured on different scales.

Defining a distance between joint-trajectories

K-means can work with various distances: Euclidean, Manhattan, Minkowski (the generalization of the two previous distances) and many others. Working on joint-trajectories raises the question of the distance between two joint-trajectories. More precisely, considering the joint-trajectories of two individuals $y_{1..}$ and $y_{2..}$, we seek to define $d(y_{1..}, y_{2..})$, the distance between $y_{1..}$ and $y_{2..}$. Strictly speaking, this is the distance between two matrices. Several methods are possible, we will focus on two. The first is to consider the t columns of the two matrices, then to compute t distances between the t couples of columns and finally to combine these t distances using a function that will combine the "column-distances". The second is to consider the m lines of the two matrices, then compute m distances between the m couples of lines and finally to combine these m distances using a function that will combine the "line-distances".

More formally, let *Dist* be a distance function and $\|\cdot\|$ be a norm. To compute a distance d between $y_{1..}$ and $y_{2..}$ according to the first method, for each fixed j, we define the distance between $y_{1j.}$ and $y_{2j.}$ (distance between the individuals' state at time j) as $d_{j.}(y_{1j.}, y_{2j.}) = Dist(y_{1j.}, y_{2j.})$. This is the distance between column j in matrix $y_{1..}$ and column j in matrix $y_{2..}$. The result is a "vector of t distances"

$$(d_{1.}(y_{11.}, y_{21.}), d_{2.}(y_{12.}, y_{22.}), \ldots, d_{t.}(y_{1t.}, y_{2t.})).$$

Then we combine these t distances using a function that algebraically corresponds to a norm $\|\cdot\|$ of the vector of distance. Finally, the distance between $y_{1..}$ and $y_{2..}$ is

$$d(y_{1..}, y_{2..}) = \|(d_{1.}(y_{11.}, y_{21.}), d_{2.}(y_{12.}, y_{22.}), \ldots, d_{t.}(y_{1t.}, y_{2t.}))\|.$$

To compute a distance d' between $y_{1..}$ and $y_{2..}$ according to the second method, for each variable X, we define the distance between $y_{1.X}$ and $y_{2.X}$ (distance between the individual trajectories X) as $d_{.X}(y_{1.X}, y_{2.X}) = Dist(y_{1.X}, y_{2.X})$. This is the distance between line X in matrix $y_{1..}$ and line X in matrix $y_{2..}$. The result is a "vector of m distances"

$$(d_{.A}(y_{1.A}, y_{2.A}), d_{.B}(y_{1.B}, y_{2.B}), \ldots, d_{.M}(y_{1.M}, y_{2.M})).$$

Then we combine these m distances by considering the norm $\|\cdot\|$ of the vector of distance. Finally,

$$d'(y_{1..}, y_{2..}) = \|(d_{.A}(y_{1.A}, y_{2.A}), d_{.B}(y_{1.B}, y_{2.B}), \ldots, d_{.M}(y_{1.M}, y_{2.M}))\|.$$

The choice of the norm $\|\cdot\|$ and the distance $Dist$ can lead to the definition of a large number of distances. On the contrary, in the case where $\|\cdot\|$ is the standard p-norm and $Dist$ is the Minkowski distance with parameters p, choosing either method d or d' gives the same result: $d(y_{1..}, y_{2..}) = d'(y_{1..}, y_{2..})$.

Proof:

$$\begin{aligned} d(y_{1..}, y_{2..}) &= \sqrt[p]{\sum_j (d_{j.}(y_{1j.}, y_{2j.}))^p} = \sqrt[p]{\sum_j \left(\sqrt[p]{\sum_X |y_{1jX} - y_{2jX}|^p} \right)^p} \\ &= \sqrt[p]{\sum_j \sum_X |y_{1jX} - y_{2jX}|^p} = \sqrt[p]{\sum_X \left(\sqrt[p]{\sum_j |y_{1jX} - y_{2jX}|^p} \right)^p} \\ &= \sqrt[p]{\sum_X (d_{.X}(y_{1.X}, y_{2.X}))^p} = d'(y_{1..}, y_{2..}) \quad (1) \end{aligned}$$

We can therefore define the Minkowski distance between two joint variable-trajectories as:

$$Dist(y_{1..}, y_{2..}) = \sqrt[p]{\sum_{j,X} |y_{1jX} - y_{2jX}|^p}.$$

The Euclidean distance is obtained by setting $p = 2$, the Manhattan distance by setting $p = 1$ and the maximum distance by passing to the limit $p \to +\infty$. In practice, the **kml3d** package uses Euclidean distance as the default distance. But it also allows users to define their own distance.

Data standardization

Since longitudinal studies deal with several different kinds of variables, the joint variables can be measured on different scales. This problem has already been extensively discussed in the classic (non-trajectory) situation (Everitt *et al.* 2001). A possible solution is to normalize the data. This can also be done with trajectories. As a slight difference to the classic situation, each variable-trajectory is not normalized at each time but in its entirety: let $\overline{y_{..X}}$ and $s_{..X}$ be respectively the mean and the standard deviation of $y_{..X}$. Then, the "global normalization" will turn the measure y_{ijX} into $y'_{ijX} = \frac{y_{ijX} - \overline{y_{..X}}}{s_{..X}}$. The normalized joint trajectory $y'_{i..}$ is obtained by normalization of its single trajectories $y'_{i.A}$, ..., $y'_{i.M}$ one by one.

2.4. Imputation methods for longitudinal data

In their founding documents, Rubin and Little distinguished three kinds of missing values (Rubin 1976; Little and Rubin 1987). Let Y_{TRUE} be the trajectories without missing values (unavailable data); let Y_{OBS} be trajectories with missing values (available measured longitudinal data); let R denote the Boolean matrix of the location of a missing value; let Y_{MISS} be the missing part of Y_{TRUE}. Thus, $Y_{TRUE} = Y_{OBS} + Y_{MISS}$. Their classification of missing values is then based on a potential link between R and Y_{TRUE}, Y_{OBS}, and Y_{MISS} as follows:

- **MCAR**: A value is Missing Completely At Random if the probability of missingness for observation y_{ijX} is independent of Y_{TRUE}.
- **MAR**: A value is Missing At Random if the probability of missingness for observation y_{ijX} is independent of Y_{MISS}, but may be dependent on the observed values Y_{OBS}. For example, if patients who performed badly at time $j-1$ decide to miss time j, the missing data will be MAR.
- **MNAR**: A value is Missing Not At Random if the probability of missingness for observation y_{ijX} is independent of its own value, i.e., ofY_{MISS}. Typically, the probability for an observation y_{ijX} to be missing at time j depends on the current value of y at time j. For example, if patients who think that they are going to perform badly at time j refuse to be tested at time j, the data will be MNAR.

The impact of the mechanism of missingness on the imputation of the missing values was examined by Molenberghs *et al.* (2004). In the particular case of longitudinal data, the missingness mechanisms were classified according to the position of the missing values within the trajectory:

- **Intermittent missing data** are data missing within a trajectory. Formally, y_{ijX} is an intermittent missing value if there exists a and b, $a < j < b$, such that y_{iaX} and y_{ibX} are not missing.
- **Monotone missing data** are data missing either at the beginning or at the end of a trajectory. This includes the case of left – or right – truncated variables. If a value is missing, then all the following (or, conversely, preceding) values are also missing. Formally, y_{ijX} is a right monotone missing value if, for all $d > j$, y_{idX} is missing. Conversely, y_{ijX} is a left monotone missing value if, for all $d < j$, y_{idX} is missing.

Orthogonally, imputation methods are grouped according to the information necessary for their implementation. **Cross-sectional** methods impute y_{ijX} using data collected at time j, that is according to the values of the other individuals at the same time j:

1. **Cross Mean** replaces y_{ijX} by the mean values of variable X observed at time j.
2. **Cross Median** replaces y_{ijX} by the median value of variable X observed at time j.
3. **Cross Hot Deck** replaces y_{ijX} by a value chosen randomly among all the values of variable X observed at time j.

Longitudinal methods impute y_{ijA} using the non-missing data of variable X and subject i. The imputation is performed independently from the data of other individuals:

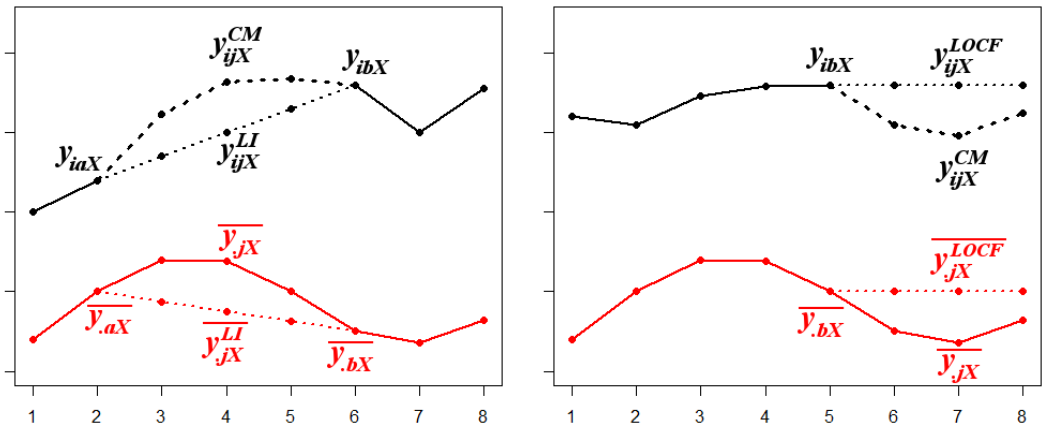
4. **Traj Mean** replaces y_{ijX} by the average of the values of trajectory $y_{i.X}$.
5. **Traj Median** replaces y_{ijX} by the median value of trajectory $y_{i.X}$.
6. **Traj Hot Deck** replaces y_{ijX} by a value chosen randomly among the values of trajectory $y_{i.X}$.
7. **Last Occurrence Carried Forward** (LOCF) replaces y_{ijX} by the previous non-missing value.
8. **Linear Interpolation** imputes y_{ijX} by drawing a line between the two non-missing values that immediately precede and follow the missing value. Let y_{iaX} and y_{ibX} be the closest preceding and following non-missing values of y_{ijX}; then $y_{ijX} = y_{iaX} + \frac{y_{ibX}-y_{iaX}}{b-a}(j-a)$.
9. **Spline Interpolation** imputes y_{ijA} by drawing a cubic spline between the two non-missing values that immediately precede and follow the missing value. For mathematical details, see Fritsch and Carlson (1980).

Cross-sectional & Longitudinal methods use both longitudinal and cross-sectional information:

10. **Copy Mean** is a new method. The main idea is to impute using linear interpolation, and then to add a variation to make the trajectory follow the "shape" of the population's mean trajectory. Formally, if y_{ijX} is missing, let y_{iaX} and y_{ibX} be the closest preceding and following non-missing values of y_{ijX} (all the notation introduced here is illustrated in Figure 2(a)); let $\overline{y_{..X}} = (\overline{y_{.1X}}, \ldots, \overline{y_{.tX}})$ denote the mean trajectory of the population S; let y^{LI}_{ijX} be the value obtained by imputing y_{ijX} using linear interpolation; and finally, let $\overline{y^{LI}_{.jX}}$ be the value obtained by applying a linear interpolation between a and b on the mean trajectory $\overline{y_{..X}}$, this is $\overline{y^{LI}_{.jX}} = \overline{y_{.aX}} + \frac{\overline{y_{.bX}}-\overline{y_{.aX}}}{b-a}(j-a)$. Then the average variation at time j is the difference between $\overline{y_{.jX}}$ and $\overline{y^{LI}_{.jX}}$, this is $AV_{jX} = \overline{y_{.jX}} - \overline{y^{LI}_{.jX}}$. Finally, Copy Mean imputes y_{ijX} by adding the average variation AV_{jX} to the result of the linear interpolation: $y^{CM}_{ijX} = y^{LI}_{ijX} + AV_{jX}$. Figure 2 shows an example of a trajectory imputed using Copy Mean. The efficiency of this method has been shown in Genolini *et al.* (2013a).

All these methods work both on intermittent and on monotone missing values except for two: (1) linear interpolation needs to know values surrounding the missing value, it cannot therefore impute monotone missing values. And since (2) Copy Mean uses Linear Interpolation, it inherits this weakness. To overcome this problem, a modified version of Copy Mean is dedicated to the imputation of monotone missing values:

10' **Copy Mean for monotone missing values**: As with intermittent missing data, the main idea is first to impute using a longitudinal method, then to add a variation.



(a) Copy Mean for intermittent missing values. (b) Copy Mean for monotone missing values.

Figure 2: Copy Mean methods. In full black lines, the trajectory that should be imputed. In red, the population's mean trajectory. In dotted lines, values imputed either by linear interpolation (LI, Fig. a) or by LOCF (Fig. b). In dashed lines, values imputed by Copy Mean (CM).

The only difference is that since the longitudinal method Linear Interpolation is not available on monotone missing values, the LOCF method will be used instead. Formally, if y_{ijX} is a right monotone missing value, let y_{ibX} $(b < j)$ be the last known value of the trajectories $y_{i.X}$ (all the notation introduced here is illustrated in Figure 2(b)); let $\overline{y_{..X}} = (\overline{y_{.1X}}, \ldots, \overline{y_{.tX}})$ denote the mean trajectory of the population S; let y^{LOCF}_{ijX} be the value obtained by imputing y_{ijX} using LOCF: $y^{LOCF}_{ijX} = y_{iaX}$; and finally, let $\overline{y^{LOCF}_{ijX}}$ be the value obtained by applying LOCF on the mean trajectory $\overline{y_{..X}}$, this is $\overline{y_{.jX}} = \overline{y_{.aX}}$. Then the average variation at time j is the difference between $\overline{y_{.jX}}$ and $\overline{y^{LOCF}_{jX}}$, that is $AV_{jX} = \overline{y_{.jX}} - \overline{y^{LOCF}_{.jX}}$. Finally, Copy Mean imputes y_{ijX} by adding the average variation AV_{jX} to the result of the LOCF imputation: $y^{CM}_{ijX} = y^{LOCF}_{ijX} + AV_{jX}$.

For the imputation of the left monotone missing value, NOCB (next occurrence carried backward) is used instead LOCF. The rest of the method remains the same.

2.5. Quality criteria

Quality criteria are indices associated with a partition. These take high values for partitions of "high quality", low values otherwise (or the reverse, depending on the criterion). Different definitions of the "high" and "low" quality partitions lead to different indices, but the operating principles are mostly similar: a "good" partition is a partition where clusters are (1) compact and (2) well separated from each other. So most of the indices calculate some kind of "within-cluster compactness index" and "between-cluster spacing index"; they then divide one by the other. More specifically:

- Calinski & Harabasz criterion (Caliński and Harabasz 1974): $C(k) = \frac{Trace(B)}{Trace(W)} \cdot \frac{n-k}{k-1}$ where B is the between-cluster covariance matrix (so high values of $Trace(B)$ denote

well-separated clusters) and W is the within-cluster covariance matrix (so low values of $Trace(W)$ correspond to compact clusters).

- Calinski & Harabasz, Kryszczuk variant (Kryszczuk and Hurley 2010)) : $C_K(k) = \frac{Trace(B)}{Trace(W)} \cdot \frac{n-1}{n-k}$ where B is the between-cluster covariance matrix and W is the within-cluster covariance matrix.
- Calinski & Harabasz, Genolini variant: $C_G(k) = \frac{Trace(B)}{Trace(W)} \cdot \frac{n-k}{\sqrt{k-1}}$ where B is the between-cluster covariance matrix and W is the within-cluster covariance matrix.
- Ray & Turi criterion (Ray and Turi 1999):

$$R(k) = \frac{V_{intra}}{V_{inter}}$$

with $V_{intra} = \sum_x (dist(x, center(x)))$ and $V_{inter} = \min_{i,j} \left(dist(center_i, center_j)^2\right)$.

- Davies & Bouldin criterion (Davies and Bouldin 1979):

$$D(k) = mean(Proximite(cluster_i, cluster_j))$$

with

$$Proximite(i,j) = \frac{DistInterne(i) + DistInterne(j)}{DistExterne(i,j)},$$

where *DistInterne* is a cluster compactness measure (for example, maximum distance between two points of the cluster) and *DistExterne* is a separation cluster measure (for example, distance between cluster's centers).

These five criteria are non-parametric. They can be computed without making any hypothesis. Considering extra hypothesis, some parametric criteria like BIC (Schwarz 1978), AIC (Akaike 1974), AIC with correction for finite sample size (Hurvich and Tsai 1989) or the global posterior probability (Bolstad 2007) can also be computed. This computation needs two kinds of information: the likelihood and the number of measures.

The former can be found assuming that for each cluster C and each time, the single trajectory variable $Y_{.X}$ follows a normal law with variance $\sigma^2 = var_{i \in S, j \in \{1,\dots,t\}}(Y_{ijX})$ (homoscedastic, one variance for all the times and all the groups) and $mean_{jC} = mean_{i \in C}(Y_{ijX})$ (the mean trajectories can change any time and for any group). Using these hypotheses, it is possible to compute the posterior probabilities of each individual trajectory, and then the likelihood.

The number of measures is more problematic: in a classic study, the number of independent observations is equal to the sample size. In longitudinal studies, the sample size is n while the number of observation is $N = t \cdot n$. But these $t \cdot n$ observations are not independent measures. Therefore, deciding whether the number of measures should be n or N is not straightforward. Packages **kml** and **kml3d** compute criteria using both definitions:

- $BIC = 2 \times \log(L) - h \times \log(n)$
- $BIC2 = 2 \times \log(L) - h \times \log(N)$

- $AIC = 2 \times \log(L) - 2 \times h$
- $AICc = AIC + \frac{2h(h+1)}{n-h-1}$
- $AICc2 = AIC + \frac{2h(h+1)}{N-h-1}$

where h is the number of parameters in the model. Note that although these parametric criteria can always be computed, they are only pertinent if the hypothesis is true (the variable follows a normal law).

Maximization or minimization?

Among all these criteria, some should be maximized (high value denoting good partition), others should be minimized (low value denoting good partition). This might be confusing, in particular for the comparison of several criteria all together. To avoid this confusion, packages **kml** and **kml3d** compute the criteria that should be maximized, and compute *the opposite* of the criteria that should be minimized. As such, all the criteria proposed by packages **kml** and **kml3d** should be maximized. This enables the representation of several criteria on the same graph, making it easier to compare them (see Figure 6).

2.6. Initialization of k-means

The first step of the k-means algorithm is to choose an initial configuration, which is a set of k cluster centers. This choice has a dual importance: (1) on it depends the partition towards which the algorithm will converge (local or global maximum); (2) on it also depends the convergence time. Ultimately, if a method is able to choose an initial configuration fairly close to the best partition, k-means would converge fast to the optimal solution. Some authors have therefore proposed various initialization methods (Pena, Lozano, and Larranaga 1999; Khan and Ahmad 2004; Redmond and Heneghan 2007; Steinley and Brusco 2007). Most of these methods try to build an initial configuration in which the initial centers are as distant as possible from each other. The idea is that individuals that are clearly distant probably belong to different clusters, which is a guarantee of quick convergence to a good partition. However, whatever the method chosen, it is nevertheless not certain that the initial configuration enables convergence to the best partition. Thus most methods include a non-deterministic process that allows the user to run the method several times. Since the runs start from different initial configurations, they might converge to different maxima. Eventually, one of them will reach the global maximum. The packages **kml** and **kml3d** offer seven methods for choosing initial configurations:

1. **randomK**: k individuals are chosen randomly. They are the initial cluster centers.
2. **randomAll**: All individuals are randomly assigned to a cluster. The mean of each cluster is the cluster center.
3. **maxDist**: This method is incremental. First, it selects the two individuals that are the most distant and considers them as the two first centers. Then it adds the individual that is the farthest away from the list of centers already preselected. More precisely:
   (a) Compute the matrix of the distance between all points.

(b) Choose the two farthest points as the first two centers c_1 and c_2.

(c) Start a loop:

i. For each data point x, consider $D(x)$, the distance between x and the nearest center that has already been chosen.

ii. Choose as a new center c_i, the data point for which $D(c_i)$ is maximum.

iii. Repeat steps 3(c)i and 3(c)ii until k centers have been chosen.

4. **kmeans+**: The maxDist method is more effective than randomAll or randomK (see Genolini and Falissard 2011) since the initial centers are distant from each other. But it is also time-consuming with a complexity in $o(n^2)$ (computation of the matrix of all the distances). kmeans+ is an improvement of maxDist. It is based on a similar principle: distant centers are added to the list of already preselected centers. This ensures that the property of having initial centers distant from each other is retained. The only difference is that the first center is chosen randomly. Thus, it is no longer necessary to compute the matrix of all distances between individuals, the calculation of distances between each individual and the centers already selected (up to k) being sufficient. Thus the complexity is $o(nk)$:

(a) Choose a center c_1 at random (uniformly).

(b) Start a loop:

i. For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.

ii. As a new center c_i choose the data point for which $D(c_i)$ is maximum.

iii. Repeat steps 4(b)i and 4(b)ii until k centers have been chosen.

5. **kmeans-**: kmeans+ is a maxDist improvement with respect to the time costs $o(kn)$. On the other hand, its efficiency is slightly worse. Indeed, the first point selected may be a "bad" choice (e.g., a point located between two clusters). A solution to improve this is to choose the first and the second centers according to kmeans+ rules, then to remove the first center (the one that may be a bad choice) from the list. The second point cannot be a bad choice since it is at least distant from the first one. After the first center has been removed, the second center becomes the new first center. Then the other centers are added one by one as in kmeans+. Formally:

(a) Choose one center c_0 uniformly at random from among the data points.

(b) For each data point x, compute $D(x)$, the distance between x and c_0.

(c) Choose as second point c_1 for which $D(c_1)$ is maximum.

(d) Remove c_0 from the list of centers. c_1 is now the "new" first center.

(e) Start a loop:

i. For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.

ii. As a new center c_i, choose the data point for which $D(c_i)$ is maximum.

iii. Repeat steps 5(e)i and 5(e)ii until k centers have been chosen.

6. **kmeans--:** kmeans- has the same advantages of maxDist in terms of center repartition (distant one from the other) and the same advantages of kmeans+ in terms of complexity (small time complexity, $o(kn)$). But except for the initial draw, kmeans- is a deterministic method. This greatly impairs the advantage of multiple draws, which is one of the strengths of k-means. A way to solve this problem is to randomly choose the centers that are added to the list of the previously selected centers. To maintain a high probability of obtaining centers that are distant from each other, the probability for an individual x to be added to the selected center list will be proportional to the square of the distance between x and the selected centers. Formally, kmeans-- is the same algorithm as kmeans- except for step 3 and 5b:

   (a) Choose one center c_0 uniformly at random from among the data points.
   (b) For each data point x, compute $D(x)$, the distance between x and c_0.
   (c) Choose one new center c_1 at random using a weighted probability distribution proportional to $D(x)^2$.
   (d) Remove c_0 from the list of the centers.
   (e) Start a loop:
      i. For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.
      ii. Randomly choose a data point as the new center c_i, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.
      iii. Repeat steps 6(e)i and 6(e)ii until k centers have been chosen.

   kmeans-- combines all the advantages of the previous methods: the centers are distant from each other, the first center cannot be a bad choice, the time complexity is $o(kn)$ and the method is non-deterministic.

7. **kmeans++:** In the same way, kmeans++ (Arthur and Vassilvitskii 2007) is the non-deterministic version of kmeans+:

   (a) Choose one center c_0 uniformly at random from among the data points.
   (b) Start a loop:
      i. For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.
      ii. Randomly choose a data point as the new center c_i, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.
      iii. Repeat steps 7(b)i and 7(b)ii until k centers have been chosen.

3. Algorithm and codes

3.1. Overview

An overview of the packages and the algorithm is presented in Figure 3.

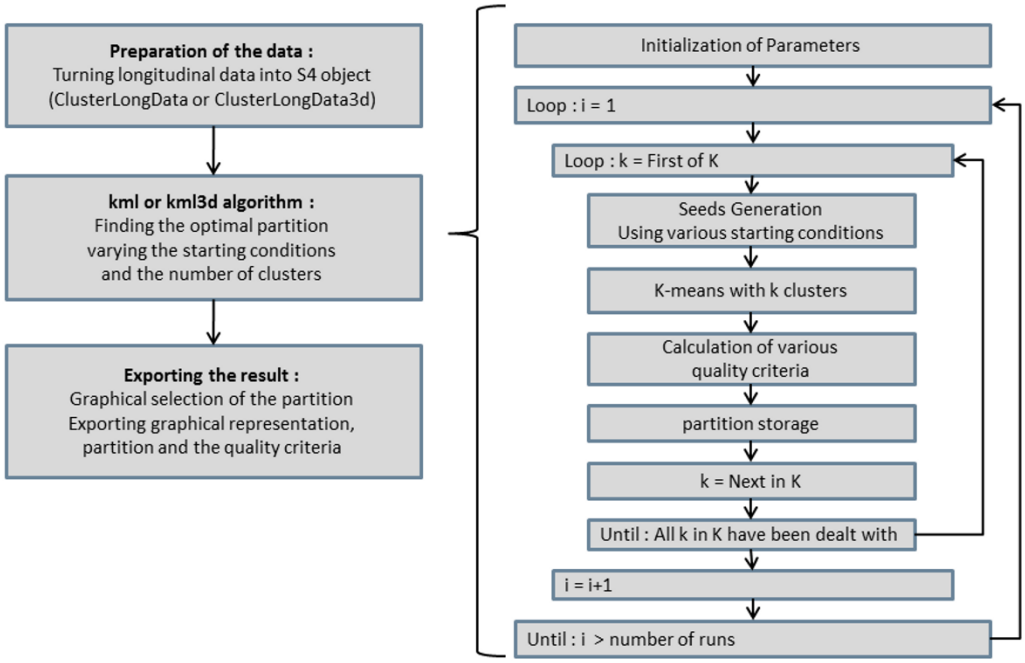


Figure 3: On the left, **kml** and **kml3d** packages. On the right, `kml` and `kml3d` algorithms.

3.2. Preparation of the data

Packages **kml** and **kml3d** cluster longitudinal data. One of their properties is that they "memorize" all the clusters that they find. To do this, they use an S4 structure (Chambers 2008; Genolini 2008). '`Partition`' objects contain clusters. They also contain some information such as the size of each cluster, as well as quality criteria (see Section 2.5). The second object, '`ClusterLongData`' (or '`ClusterLongData3d`' for **kml3d**), mainly contains a field `traj` that stores the trajectory, and fields `c2`, `c3`, `c4`, ..., `c26` that store lists of '`Partition`' objects with respectively 2, 3, 4, ..., 26 clusters. Data preparation therefore simply consists in transforming longitudinal data into a '`ClusterLongData`' or '`ClusterLongData3d`' object.

In package **kml**, this can be done using function `cld()`. `cld()` turns data frames or matrix into a '`ClusterLongData`' object. It uses the following argument (the type of the argument is given in brackets):

- `traj` ([`matrix(numeric)`] or [`data.frame(numeric)`]): contains the longitudinal data. Each line is the trajectory of an individual. The columns refer to the time at which measures were performed.
- `idAll` ([`vector(character)`]): single identifier for each individual (each trajectory).
- `time` ([`vector(numeric)`]): time at which measures are performed.
- `timeInData` ([`vector(numeric)`]): column numbers (in the matrix or the data frame) that contain the trajectories.
- `varNames` ([`character`]): name of the variable measured.

- `maxNa` ([`numeric`]): maximum number of missing values that is tolerated on a trajectory. If a trajectory has more missing values than `maxNa`, then it will be removed from the analysis.

Package **kml3d** works in nearly the same way: the object '`ClusterLongData3d`' contains the same fields as the object '`ClusterLongData`'. They are built using function `cld3d()` from a `data.frame` or a three-dimensional `array`.

- `traj` ([`array(numeric)`] or [`data.frame(numeric)`]): contains the joint-longitudinal data. For an array `A`, each line (`A[i, , ]`) is the joint trajectory of an individual. Each column (`A[, j, ]`) refers to the time at which measures were performed. Each third dimension (`A[, , l]`) contains variables. For a data frame `D`, each line `D[i, ]` is an individual, the variable trajectories are in various columns (which should be specified by the argument `timeInData`, see below).
- `idAll` ([`vector(character)`]): a single identifier for each individual (each trajectory).
- `time` ([`vector(numeric)`]): time at which measures are performed.
- `timeInData` ([`list(vector(numeric))`]): if `traj` is a data frame, `timeInData` specifies columns that contain the trajectories. The list labels are the names of the variables. The vectors of numbers associated with each variable are the column numbers in the data frame that store the variable. For example, `timeInData = list(A = c(2, 4, 6), B = c(3, 5, 9))` defines a joint-trajectory composed of two variables `A` and `B`, the trajectories of `A` are contained in columns 2, 4 and 6, the trajectories of `B` are in 3, 5 and 9.

  This argument is not used if `traj` is an array.
- `varNames` ([`character`]): names of the variables measured.
- `maxNa` ([`numeric`] or [`vector(numeric)`]): maximum number of missing values that is tolerated on a trajectory. If a trajectory has more missing values than `maxNa`, then it will be removed from the analysis. In the 3D case, `maxNa` can be a single numeric (same value for all the variable) or a vector of numeric values (one value per variable).

3.3. Finding the optimal partition

Once an object of the class '`ClusterLongData`' has been created, the `kml()` function can be called. `kml()` runs k-means several times, varying starting conditions and the number of clusters. On each run, it finds a '`Partition`' and stores it in the appropriate field. The starting condition can be `"randomAll"`, `"randomK"`, `"maxDist"`, `"kmeans++"`, `"kmeans+"`, `"kmeans--"` or `"kmeans-"` as described in Section 2.6. In addition, it can also use two specific values: `"all"` stands for `c("maxDist", "kmeans-")` followed by an alternation of `"kmeans--"` and `"randomK"`; `"nearlyAll"` stands for `"kmeans-"` followed by an alternation of `"kmeans--"` and `"randomK"`. By default, `kml()` runs k-means for $k \in \{2, 3, 4, 5, 6\}$ clusters 20 times each using `"nearlyAll"`.

The k-means version used here is the Hartigan and Wong version (Hartigan and Wong 1979). The default distance is the Euclidean distance with Gower adjustment (Gower 1966). `kml()`

can also work with user-defined distances using the optional argument `distance`. If specified, `distance` should be a function that takes two trajectories and returns a number, letting the user compute a non-classic distance (such as the adaptive dissimilarity index, Hennig and Hausdorf 2006; dynamic time warping, Rath and Manmatha 2003; Fréchet distance, Fréchet 1905; ...).

There are two implementations of `kml()`. The first one (`kmlSlow`) is programmed in R. It is called by `kml()` when the user asks for some graphical display or when he/she wants to use some non-classical user-defined distance function. The second (`kmlFast`) is optimized in C. It is run only when all the parameters are standard. This second version is around 20 times faster than the first one. The dispatching to `kmlSlow` / `kmlFast` is done automatically by `kml`, according to the options defined by the user.

Every '`Partition`' found by `kml()` is stored in the '`ClusterLongData`' object. Fields `c2`, `c3`, ..., `c26` are lists storing partitions with a specific number of clusters (for example; the sublist `c3` stores all the '`Partition`' with 3 clusters). 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 (this can be specified by the user) to guard against any system interruption that may occur when the algorithm has to run for a long time (up to several days).

The main arguments of `kml()` are:

- `object` (['`ClusterLongData`']): contains trajectories to clusters and all the partitions already found.
- `nbClusters` ([`vector(numeric)`]): contains 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, 4, 5 up to 6 clusters.
- `nbRedrawing` ([`numeric`]): sets the number of times that k-means must be run (with different starting conditions) for each number of clusters. The default value is 20.
- `toPlot` ([`character`]): while it is running, `kml()` can display two graphs. The first displays the quality criterion of all the partitions already found. The second represents the evolution of the clustering process (the different steps in k-means). According to the value of `toPlot`, the first (`toPlot = "criterion"`), the second (`toPlot = "traj"`), none (`toPlot = "none"`) or both (`toPlot = "both"`) will be displayed.
- `parAlgo` (['`ParKml`']): some more advanced options can be specified using the argument `parAlgo`. `parAlgo` takes as value an object of class '`ParKml`'. Objects of class '`ParKml`' contain different fields that detail the parameters that `kml()` should use (such as a user-defined distance, the starting condition, the save frequency, ...).

The `kml3d()` function from package **kml3d** works on the same principle. The only difference is that the argument `object` belongs to the class '`ClusterLongData3d`' instead of '`ClusterLongData`'.

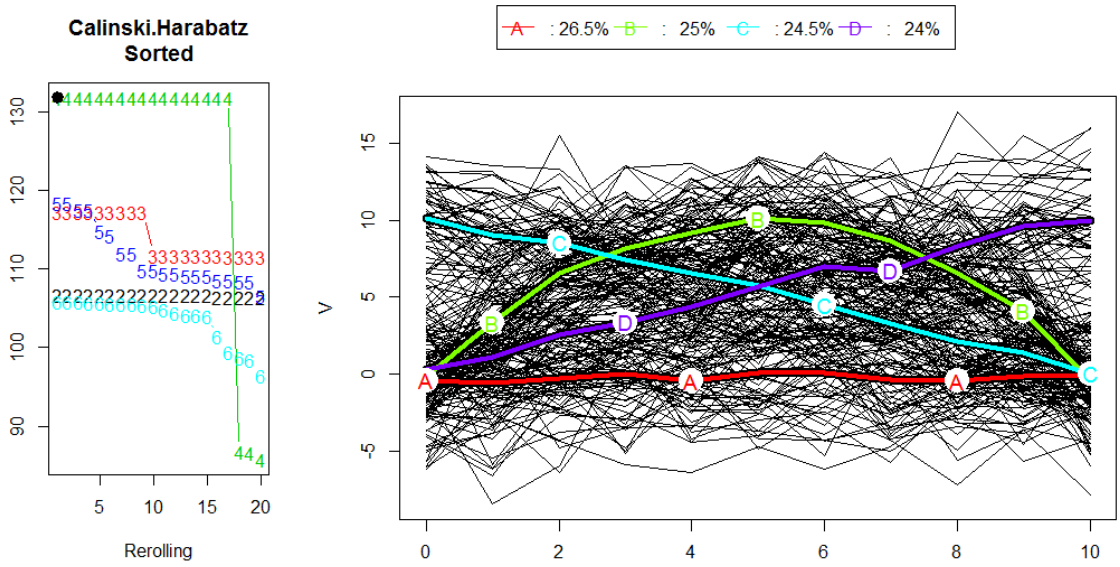


Figure 4: `kml()` results, displayed by the function `choice()`. On the left, every partition is presented by its cluster number. One of them (marked by a black dot) is selected. The mean trajectories of the clusters, according to the selected partition are represented on the right-hand part of the display.

3.4. Exporting results

After `kml()` has found some partitions, the user can "visualize" them, and then can decide to export some of them. This can be done via the function `choice()`. `choice()` opens a graphic window split in two parts (see Figure 4 for example). On the left, all the partitions that have been found by `kml()` are represented. A partition is represented by a number that is the number of its clusters. The height of the points represents the value of a specific quality criterion (presented in Section 2.5) called the "active criterion". The name of the active criterion appears above the graph. Thus this graph gives an overall view of all partitions which have been found.

Among all the partitions plotted on the left-hand graph, one is "selected" (the one highlighted with a black dot, see Figure 4). This partition is graphically plotted on the right-hand part of the graphic window. More precisely, the right-hand graph plots the longitudinal data and highlights the cluster structure of the selected partition using colors. According to the user's choice, trajectories belonging to a specific cluster may or may not be plotted, colorized, or labeled (ditto for mean trajectories of each groups).

This is a dynamic process: the user can change the selected partition (by moving the black dot); he/she can also change the active criterion. Clusters and mean trajectories shown on the right side of the graphic window are amended accordingly. This allows the user to visualize various partitions, but also to check if the best partition based on a criterion remains the best one according to another criterion.

Finally, the user can choose to export some partitions by selecting them (by simply pressing the space bar when the partition is shown). When he/she has made his/her choices, clusters and related information (quality criteria, frequency of individuals in each cluster, ...) are exported into `.csv` files. Graphs are also exported in a format compatible with `savePlot()`.

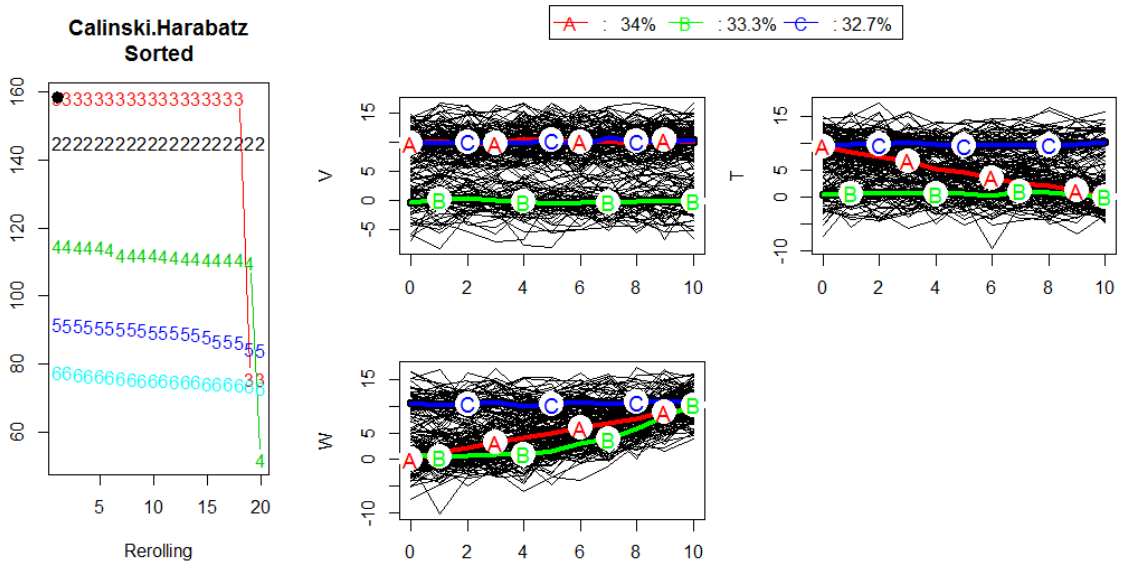


Figure 5: `kml3d()` results. On the left, every partition is presented by its cluster number. One of them (marked by a black dot) is selected. The mean trajectories of the clusters, according to the selected partition, are represented on the right-hand part of the display (one graph for each variable).

For joint trajectories, the process is similar. The only difference is that the right-hand graph shows every single variable trajectory (one graph for each) that are part of the joint-variable trajectories (see Figure 5)

More precisely, the `choice()` function arguments are:

- `object` (['`ClusterLongData`']): Object containing the trajectories and all the partitions found by `kml()` or by `kml3d()` that the user wants to see.
- `typeGraph` ([`character`]): For every selected partition, `choice()` can export graphs. `typeGraph` sets the format that will be used. The possible formats are those available for `savePlot`.

3.5. 3D rotating graph in a PDF file

For joint trajectories, **kml3d** provides tools to visualize trajectories. The `plot3d()` function displays the joint-trajectories in a three-dimensional space where the x-axis represents time, the y-axis is the first variable, and the z-axis is the second one. This graph function is built using the **rgl** package (Adler and Murdoch 2014). It is therefore possible to move the plotted graph in three dimensions using the mouse. This helps to visualize the joint evolution for trajectories fairly accurately.

Moreover, recent versions of the PDF standard accept dynamic graphs. In association with LaTeX and **Asymptote** (Hammerlindl, Bowman, and Prince 2014)[2], **kml3d** make it possible to

[2]Windows configuration: MiKTeX V0.9, Asymptote V1.96. Note that some PDF readers cannot support the 3D rotating graphs. If Figures 1 and 9 seem to be missing in the current document, try Adobe Acrobat reader, version 11.0.07 or later.

export 3D graphic representations of mean trajectories inside a `.pdf` document. This is done using the **misc3d** package (Feng and Tierney 2008). The 3D trajectories can thus easily be sent to a colleague, or can be dynamically included in an article (like the present article, see Figures 1 and 9). The procedure for producing 3D dynamic graphs is the following:

1. `plot3dPdf()`: Create a scene, which is a collection of triangle objects that represent a 3D image.

2. `saveTrianglesAsASY()`: Export the scene to an `.asy` file.

3. The `.asy` file cannot be included in a LaTeX file. LaTeX can read only `.pre` files. So the next step is to use **Asymptote** to convert an `.asy` file into a `.pre` file. This is done by the command:

   `asy -inlineimage -tex pdflatex scene.asy`.

4. The previous step produced a file `scene+0.prc` that can be included in a LaTeX file. `makeLatexFile()` creates a LaTeX file (default name: `"main.tex"`) that is directly compilable using `pdflatex`.

Then compiling `main.tex` produces a `.pdf` document that contains the 3D dynamic graph.

3.6. Other functions

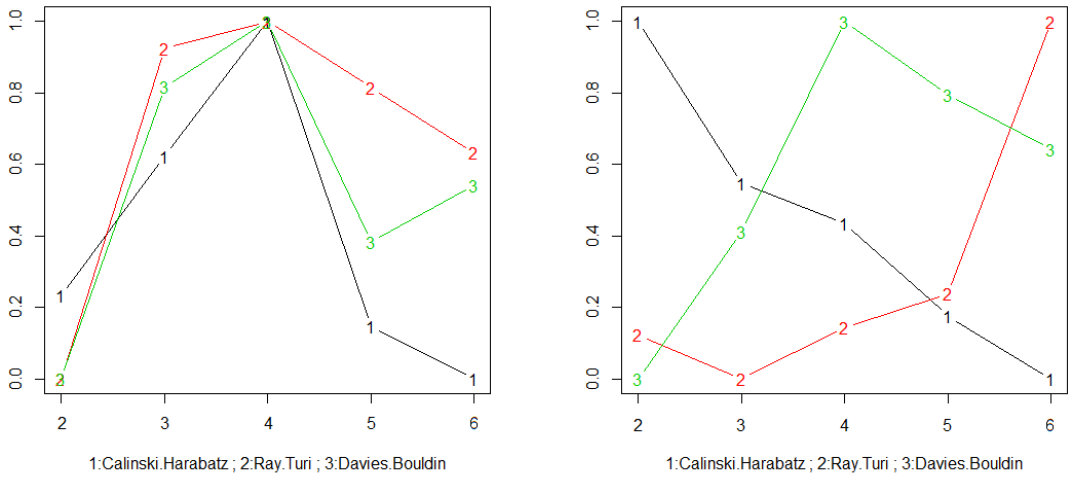
The four previous functions are probably sufficient for most users. Nevertheless, the following functions may also be helpful in some specific cases.

*Scaling then restoring the original data (package **kml3d** only)*

When working on joint-trajectories, variables may be measured on different scales. This can give too much weight to one of the variables at the expense of another. A possible solution to overcome this problem is to normalize the data. The `scale()` function can either normalize the data globally (global normalization as defined in Section 2.3) or change the scales according to values defined by the user. Whatever the changes made by the `scale()` function, the function `restoreRealData()` can restore the data in their original form. By default, the scaling (normalization) and restoring options are automatically performed by `kml3d()` but this can be switched off (using the argument `parAlgo`), then the `scale()` function can be used to specify values manually.

Quality criteria comparison

As we noted in the introduction, quality criteria used to select the "correct" number of clusters are not always efficient. Using several of them might strengthen the reliability of the results. The function `plotAllCriterion()` displays several criteria on the same graph. In order to make them comparable, `kml()` and `kml3d()` compute the opposite of the criterion that should be minimized. Thus all criteria have to be maximized. In addition, criteria are mapped into `[0, 1]`. This is more convenient for comparing them. Figure 6 gives an example of concordant and discordant criteria.



(a) Concordant criteria, criteria agree on 4 clusters. (b) Discordant criteria, these advise either 2, 4 or 6 clusters.

Figure 6: Calinski & Harabasz, Ray & Turi, Davies & Bouldin criteria on the same graph.

Fuzzy k-means

The **kml** package also provides a fuzzy version of k-means (`fuzzyKmlSlow()`). It has the same syntax and offers the same facilities as the function `kml()`.

Generating artificial longitudinal data

When researchers want to test new algorithms or methods, they usually first work on artificial data. These data have the advantage of being well known, and thus providing the optimal solution. A way to demonstrate the effectiveness of a new method is to prove that it is better than the others at finding the optimal solution on artificial data.

Package **kml** (resp. **kml3d**) provides a function that generates artificial longitudinal data. The random generation method works as follows: A data set shape is defined by a number of groups k and k real functions from $\mathbb{R}$ to $\mathbb{R}$ (resp. k functions from $\mathbb{R}$ to $\mathbb{R}^i$), one for each group. This defines the typical single trajectories (resp. typical joint trajectories) that follow individuals in the group. Data sets are then created from the data set shape. Initially, a number of individuals per group is set. The trajectory of an individual is obtained by adding a personal variation and a residual variation to the typical trajectory of its group. An individual's personal variation is a constant over time (it represents the individual's specificities) while the residual variation can change at each time. Finally, a percentage of missing values (Missing Completely at Random according to the Rubin classification) can be added to each cluster.

4. Examples of use

4.1. Using package kml

Package **kml** has already been used in various studies (Touchette *et al.* 2008; Pryor *et al.* 2011;

Pingault *et al.* 2011, 2014). The following example is included in the package.

Dataset description

The EPIPAGE cohort, funded by INSERM and the French general health authority, is a multi-regional French follow-up survey of severely premature children. It included more than 4000 children born at less than 33 weeks gestational age, and two control samples of children, respectively born at 33–34 weeks of gestational age and born full term. The general objectives were to study short and long term motor, cognitive and behavioral outcomes in these children, and to determine the impact of medical practice, care provision and organization of perinatal care, environment, family circle and living conditions on child health and development. About 2600 children born severely premature and 400 and 600 controls respectively were followed up to the age of 5 years (Larroque *et al.* 2008) and then to the age of 8 (Larroque *et al.* 2011). The database belongs to the INSERM unit U953 (P.Y. Ancel) which has agreed to include some of the data in the package.

Code

A part of the EPIPAGE Database has been included in the package, mainly the gender and the "Strengths and Difficulties Questionnaire" (SDQ) score at age 3, 4, 5 and 8. The SDQ is a behavioral questionnaire for children and adolescents aged 4 through 16 years old. It measures the severity of the disability (a higher score indicates greater disability). We present here a complete example code for analyzing the data. Data is loaded simply using the `data()` instruction:

```
R> library("kml")
R> data("epipageShort", package = "kml")
R> head(epipageShort)

  id gender     sdq3 sdq4     sdq5 sdq8
1 S1   Male 24.00000   20 16.00000 22.0
2 S2   Male 32.00000   24 22.00000 16.0
3 S3 Female 29.47368   46 17.77778   NA
4 S4   Male 32.00000   NA 24.00000 17.5
5 S5   Male 34.00000   34       NA 14.0
6 S6 Female 35.78947   40 12.00000   NA
```

Since there are some missing values in the `data.frame`, the user may want to impute them, either using the default method `"copyMean"`:

```
R> imputation(as.matrix(epipageShort[, 3:6]))
```

or using one of the other methods:

```
R> imputation(as.matrix(epipageShort[, 3:6]), method = "linearInterpol")
```

Trajectories of `sdq` are in columns 3 to 6 of the `data.frame`. So we can build the object of class '`cld`':

```
R> cldSDQ <- cld(epipageShort, timeInData = 3:6)
R> cldSDQ

   ~~~ Class: ClusterLongData ~~~
      ~ Sub-Class: LongData ~
~ idAll       = [697] S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 ...
~ idFewNA     = [697] S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 ...
~ varNames    = [1] V
~ time        = [4] 1 2 3 4
~ maxNA       = [1] 2
~ reverse     = [2x1]
    - mean    = 0
    - SD      = 1

~ traj = [697x4] (limited to 5x10)  :
         t1 t2       t3   t4
S1 24.00000 20 16.00000 22.0
S2 32.00000 24 22.00000 16.0
S3 29.47368 46 17.77778   NA
S4 32.00000 NA 24.00000 17.5
S5 34.00000 34       NA 14.0
... ...

    ~ Sub-Class: ListPartition ~
 ~ criterionActif          =  Calinski.Harabatz
 ~ initializationMethod    =
 ~ sorted                  =
 ~ criterion values (Calinski.Harabatz):
    <no Partition>
```

Partitioning the data is then performed using the function `kml()`. At first, we want to "see" the clustering process. Since it might be slow, we will ask only for two redrawings with the graph display.

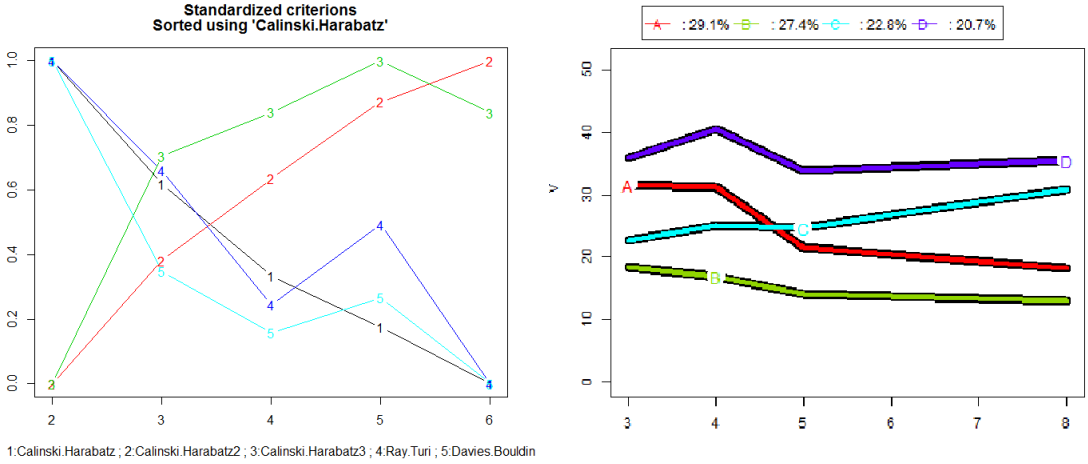
```
R> kml(cldSDQ, nbRedraw = 2, toPlot = "both")

 ~ Slow kml ~
**********
```

Then we want more rerolling, without the graphical display (that slow down the process):

```
R> kml(cldSDQ)

 ~ Fast kml ~
**************************************************************************
**********************
```



(a) Cluster numbers.

(b) Trajectories.

Figure 7: Trajectories of SDQ between ages of 3 and 8.

We can also cluster the data by changing parameters in the `kml` function. This can be done using the `parAlgo` argument. This argument lets the user control various parameters like the automatic save frequency, the distance used to cluster the data or the way to compute the clusters' center at each iteration. For example, `kml` can cluster with the correlation distance with a save frequency of 10:

```
R> kml(cldSDQ, 4, parAlgo = parALGO(distance = function(x, y)
+    cor(x, y), saveFreq = 10))
```

We can explore all the partitions found by the algorithm using the function `choice()`, or with `plotAllCriterion()`:

```
R> choice(cldSDQ)
R> plotAllCriterion(cldSDQ)
```

The result of `choice()` is presented in Figure 4, the result of `plotAllCriterion()` is in Figure 7(a). According to the latter, there is no clear evidence on any "best cluster number". According to expert opinion, the partition with four clusters seems to be the most relevant. According to this partition, four profiles can be found in the population: (A) high then decreasing, (B) low, (C) low then increasing and (D) high trajectories (see Figure 7(b)).

From a medical standpoint, this result is particularly pertinent: at three years of age, groups (A) and (D) are close. Then the health of some of these children does not change while others will see an improvement in their condition. The opposite occurs for groups (B) and (C): close at three years of age, some children keep a good health while some other see their condition deteriorate.

Finding the determinants that explain the divergence between (A) and (D) (or (B) and (C)) is therefore a major issue. This can be done by trying to explain the membership to the groups (A) or (D) using classical analysis. In our example, we try to see if the gender has an impact:

```
R> epipageShort$clusters <- getClusters(cldSDQ, 4)

  [1] A C C C C C B D D B D A A B D C B D C B C A A A B A B A D D A A A C B A
 [37] C B C B D A D A A A C B B D C C B A A C B B D D B A B D D C C A A B B A
 [73] D A B C C C D D B B ...

R> epipageGroupAD <- epipageShort[epipageShort$clusters %in% c("A", "D"), ]
R> summary(glm(clusters ~ gender, data = epipageGroupAD,
+    family = "binomial"))

Call:
glm(formula = clusters ~ gender, family = "binomial", data = epipageGroupAD)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.8884  -0.8884  -0.7585   1.4971   1.6651

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.0986     0.1952  -5.629 1.82e-08 ***
genderMale    0.3727     0.2507   1.487    0.137
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 392.04  on 323  degrees of freedom
Residual deviance: 389.80  on 322  degrees of freedom
AIC: 393.8

Number of Fisher Scoring iterations: 4
```

In our example, we see that gender has no influence on belonging to groups A or D.

4.2. Using package kml3d

Package **kml3d** has already been used in various studies (Pingault *et al.* 2012; Wahl *et al.* 2014). The following example is included in the package.

Presentation of the data

The QUIDEL database aims to gain better knowledge of hormone profiles among women who have no fertility problem. This database has been described as the largest existing database on hormone profiles in the normal human menstrual cycle, involving ultrasound scan on the day of ovulation (Ecochard 2006). It involves 107 women and 283 cycles in all, with identification of the day of ovulation and daily titration of the levels of the four main hormones in the ovulation cycle. It has already been the subject of numerous publications

(including Ecochard and Gougeon 2000; Ecochard, Boehringer, Rabilloud, and Marret 2001). The database belongs to the laboratory in charge of the analysis of hormone trajectories (CNRS 5558, René Ecochard) which has agreed to include some of the data in the package.

Code

In this analysis, we investigate the joint evolution of pregnanediol and temperature. The data are included in the package **kml3d**.

```
R> library("kml3d")
R> data("pregnandiol", package = "kml3d")
R> head(pregnandiol)
```

```
   id temperature1 pregnandiol1 temperature2 pregnandiol2 temperature3  ...
1 I-1         36.8         1.78         36.4         2.84         37.0  ...
2 I-2         36.8         1.34         36.8         1.36         36.7  ...
3 I-3           NA         3.13           NA         1.17           NA  ...
4 I-4         36.8        11.50         36.8         3.95         36.5  ...
5 I-5           NA        17.90           NA         7.35           NA  ...
6 I-6           NA           NA           NA         4.59           NA  ...
```

The longitudinal variable pregnanediol is contained in the odd-number columns between 3 and 61. Temperature is in the even-number columns between 2 and 60. The code to build an object of class '`ClusterLongData3d`' is:

```
R> cldPreg <- cld3d(pregnandiol, timeInData = list(preg = 1:30 * 2 + 1,
+    temp = 1:30 * 2))
R> cldPreg
```

```
   ~~~ Class: ClusterLongData3d ~~~
      ~ Sub-Class: LongData3d ~
~ idAll       = [80] I-1 I-2 I-3 I-4 I-5 I-6 I-7 I-8 I-9 I-10 ...
~ idFewNA     = [80] I-1 I-2 I-3 I-4 I-5 I-6 I-7 I-8 I-9 I-10 ...
~ varNames    = [2] preg temp
~ time        = [30] 1 2 3 4 5 6 7 8 9 10 ...
~ maxNA       = [2] 28 28
~ reverse     = [2x2]
    - mean    = 0 0
    - SD      = 1 1

~ traj = [80x30x2] (limited to 5x10x3)  :

 preg :
       t1   t2    t3   t4   t5   t6   t7   t8   t9   t10 more
I-1  1.78 2.84    NA 2.38 2.02 2.35 2.67 2.23 2.16  3.05  ...
I-2  1.34 1.36  1.67 0.61 0.69 1.08 1.22 2.16 0.87  1.09  ...
I-3  3.13 1.17  1.13 0.75 0.41 0.69 0.88 0.70 0.77  0.27  ...
```

```
I-4 11.50 3.95  3.15 2.74 3.65 3.38 3.94 3.54 3.18 11.40  ...
I-5 17.90 7.35 12.70 3.23 6.08 4.83 7.68 3.02 2.33  6.71  ...
... ...

 temp :
      t1   t2   t3   t4   t5   t6   t7   t8   t9  t10 more
I-1 36.8 36.4 37.0 37.0 36.5 36.3 36.5 36.5 36.5 36.5  ...
I-2 36.8 36.8 36.7 36.7 36.6 36.7 36.4 36.6 36.6 36.5  ...
I-3   NA   NA   NA   NA   NA   NA   NA 36.5 36.5 36.5  ...
I-4 36.8 36.8 36.5 36.7 36.6 36.7 36.7 36.7 36.4 36.6  ...
I-5   NA   NA   NA   NA 36.3 36.8 36.5 36.6 36.5 36.3  ...
... ...

    ~ Sub-Class: ListPartition ~
 ~ criterionActif          =  Calinski.Harabatz
 ~ initializationMethod    =
 ~ sorted                  =
 ~ criterion values (Calinski.Harabatz):
    <no Partition>
```

Partitioning these data is done using the function `kml3d()`:

```
R> kml3d(cldPreg)

 ~ Fast kml ~
**************************************************************************
**********************
```

Then, classically, we can explore the set of partitions found using `choice()`.

```
R> choice(cldPreg)
```

There is no clear evidence on any "best cluster number" (see Figure 8).

If, according to expert opinion, we consider the partition with 4 clusters, we can export this in a 3D PDF dynamic graph. We shall first build an `.asy` file and create a LaTeX main document.

```
R> scene <- plot3dPdf(cldPreg, 3)
R> saveTrianglesAsASY(scene)
R> makeLatexFile()
```

Then, using **Asymptote** (in a console), we can build a `.prc` file that can be included in a `.pdf` file using LaTeX.

```
Cons> asy -inlineimage -tex pdflatex scene.asy
Cons> pdflatex main.tex
```

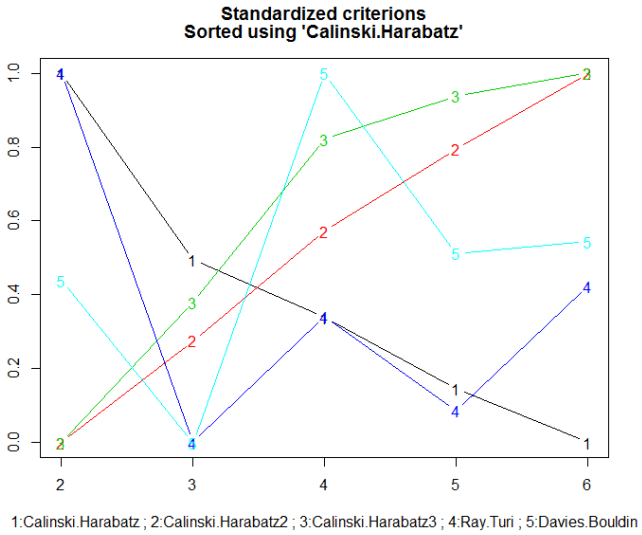


Figure 8: Quality criteria for pregnandiol/temperature joint trajectories.

Figure 9: Joint evolution of pregnandiol (y-axis) and temperature (z-axis).

	$t = 4$	$t = 11$	$t = 31$	$t = 101$	$t = 301$	$t = 1001$	$t = 3001$	$t = 10001$
$n = 12$	3	3	3	3	5	34	283	5 842
$n = 40$	3	3	3	3	7	81	715	NA
$n = 120$	3	5	5	7	14	257	2 388	NA
$n = 400$	11	9	12	20	47	1 115	10 252	NA
$n = 1200$	32	33	41	68	482	5 721	47 684	NA
$n = 4000$	135	147	211	750	6 996	54 903	NA	NA
$n = 12000$	634	784	1 579	5 403	NA	NA	NA	NA
$n = 40000$	5 463	7 715	14 175	73 177	NA	NA	NA	NA

Table 1: Execution time (in second) using `fastKml`, 2 to 6 clusters, 20 reroll each. n is the number of individuals, t is the length of the trajectories. NA stands for the R error that R is unable to allocate a vector of this size.

The result is presented in Figure 9[3].

4.3. Adjusting parameters

Default values

There is a huge number of k-means variants in the literature. Different authors propose to modify initial conditions, center calculation methods, distances between individuals, distances between clusters, or stopping criteria, while others sequentialize the cluster allocation or randomly disrupt some individuals to escape from local minima (see Liao 2005 for a survey of all these techniques). It is obviously not possible to test all these variants, and even less their combinations. In packages **kml** and **kml3d**, we decided to define some default settings for the non-expert, and to allow the expert user to modify a large number of parameters. We chose the default values in two ways: when there is an article showing the superiority of one method over the other (kmeans++, Copy Mean, Calinski & Harabatz criterion), then it is used as the default method. Otherwise, we took the method that most closely matches the original k-means algorithm (Hartigan and Wong 1979): the distance between the trajectories is the Euclidean distance, the cluster centers are the mean of each cluster, the distance between two clusters is the Euclidean distance between their centers. This version of k-means is also the one that was used in Genolini and Falissard (2010), an article showing that in some cases **kml** outperforms **Traj Proc**.

Limitations

Table 1 shows the limitations in terms of data size and, when appropriate, the average execution time. The computation was performed using `fastKml` with the default settings (number of clusters: 2–6; 20 rerolls each). The tests were conducted under Windows on a laptop equipped with an Intel (R) Core (TM) i5-2520M processor CPU@2.50GHz with 8GB of RAM.

On small data sizes, k-means has a clear advantage over mixture models: it always converges. Thus, the limitation in terms of size is the meaning that one can give to results. Is it reasonable to present the average trajectory of a group of three or four individuals? Can we consider

[3]This feature may not work with all PDF readers. If this figure does not appear in the current document, try to use Adobe Acrobat reader, version 11.0.07 or later.

that only two or three repeated measurements define a trajectory? The answer depends on the specificities of each scientific field. In all cases, packages **kml** and **kml3d** can work well with very small datasets.

5. Discussion

This article presents the packages **kml** and **kml3d**, versions of k-means adapted to the analysis of single and joint variable trajectories respectively. They are able to deal with missing values, provide an easy way to run the algorithm several times and their plotting facilities help the user to choose the appropriate number of clusters when criteria traditionally devoted to this task are not efficient. Package **kml3d** also provides devices for displaying and exporting 3D dynamic graphs.

5.1. Limitations

The limitations of packages **kml** and **kml3d** are those inherent in all clustering algorithms. These techniques are mainly exploratory; they cannot statistically test the reality of cluster existence. Packages **kml** and **kml3d** are not model-based, which can be an advantage (non parametric, more flexible) but also a disadvantage (no scope for testing goodness-of-fit). The determination of the optimal number of clusters is still an unsettled issue. The EM algorithm can also be particularly sensitive to the problem of local maxima. Finally, the **kml3d** package provides some tools making it possible to "see" the trajectories in 3D, but these tools can only display two variables at the same time. This might be a problem for clustering data using more than two joint variables.

5.2. Strengths

Packages **kml** and **kml3d** provide non-parametric algorithms. They do not need any prior information. They enable the clustering of trajectories that do not follow polynomial or parametric trajectories. They avoid the issues related to model selection. Package **kml3d** provides a way to cluster data according to several joint trajectories. This can help to highlight complex relationships between variable-trajectories, it also combines the information of several strongly-correlated variable-trajectories into a single nominal variable.

5.3. Perspectives

A number of unsolved problems still need further investigation. In the context of joint trajectories, the optimization of cluster number is becoming an increasingly important issue, since it is not possible to graphically represent the result of the partitioning process if there are more than two variables. It is possible that the particular situation of joint longitudinal data could lead to an efficient solution not yet found in the general context of cluster analysis. Another interesting approach would be to compare the efficiency of the initialization methods. `kmeans++` is more efficient than the classic `randomAll`, but other methods have not yet been formally evaluated. It would be interesting to study their respective performances. Another interesting context would be to extend the package to binary data clustering, which is already used in Subtil, Boussari, Bastard, Etard, Ecochard, and Genolini (2015).

Acknowledgments

We would especially like to thank Prof. René Écochard for lending us his database Pregnandiol and Prof. Pierre Yves Ancel for lending us his database EPIPAGE. This research was founded by the ANR grant 'IDOL : ANR-12-BSV1-0036'.

References

Adler D, Murdoch D (2014). ***rgl**: 3D Visualization Device System (OpenGL)*. R package version 0.95.1201, URL http://CRAN.R-project.org/package=rgl.

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723.

Arthur D, Vassilvitskii S (2007). "k-means++: The Advantages of Careful Seeding." In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics.

Bolstad WM (2007). *Introduction to Bayesian Statistics*. John Wiley & Sons.

Caliński T, Harabasz J (1974). "A Dendrite Method for Cluster Analysis." *Communications in Statistics – Theory and Methods*, **3**(1), 1–27.

Celeux G, Govaert G (1992). "A Classification EM Algorithm for Clustering and Two Stochastic Versions." *Computational Statistics & Data Analysis*, **14**(3), 315–332.

Chambers J (2008). *Software for Data Analysis: Programming With R*. Springer-Verlag.

Davies DL, Bouldin DW (1979). "A Cluster Separation Measure." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **1**(2), 224–227.

Ecochard R (2006). "Heterogeneity in Fecundability Studies: Issues and Modelling." *Statistical Methods in Medical Research*, **15**(2), 141–160.

Ecochard R, Boehringer H, Rabilloud M, Marret H (2001). "Chronological Aspects of Ultrasonic, Hormonal, and Other Indirect Indices of Ovulation." *BJOG: An International Journal of Obstetrics & Gynaecology*, **108**(8), 822–829.

Ecochard R, Gougeon A (2000). "Side of Ovulation and Cycle Characteristics in Normally Fertile Women." *Human Reproduction*, **15**(4), 752–755.

Engels JM, Diehr P (2003). "Imputation of Missing Longitudinal Data: A Comparison of Methods." *Journal of Clinical Epidemiology*, **56**(10), 968–976.

Everitt BS, Landau S, Leese M (2001). *Cluster Analysis*. 4th edition. Arnold, London.

Feng D, Tierney L (2008). "Computing and Displaying Isosurfaces in R." *Journal of Statistical Software*, **28**(1), 1–24. URL http://www.jstatsoft.org/v28/i01/.

Fréchet M (1905). "Sur l'Écart de Deux Courbes et Sur Les Courbes Limites." *Transactions of the American Mathematical Society*, **6**(4), 435–449.

Fritsch FN, Carlson RE (1980). "Monotone Piecewise Cubic Interpolation." *SIAM Journal on Numerical Analysis*, **17**(2), 238–246.

Genolini C (2008). "A (Not So) Short Introduction to S4." URL http://christophe.genolini.free.fr/webTutorial/notSoShort.php.

Genolini C (2015a). ***kml**: K-Means for Longitudinal Data*. R package version 2.3, URL http://CRAN.R-project.org/package=kml.

Genolini C (2015b). ***kml3d**: K-Means for Joint Longitudinal Data*. R package version 2.3, URL http://CRAN.R-project.org/package=kml3d.

Genolini C, Écochard R, Jacqmin-Gadda H (2013a). "Copy Mean: A New Method to Impute Intermittent Missing Values in Longitudinal Studies." *Open Journal of Statistics*, **3**(4A), 26–40.

Genolini C, Falissard B (2010). "**kml**: *K*-Means for Longitudinal Data." *Computational Statistics*, **25**(2), 317–328.

Genolini C, Falissard B (2011). "**kml**: A Package to Cluster Longitudinal Data." *Computer Methods and Programs in Biomedicine*, **104**(3), e112–e121.

Genolini C, Pingault JB, Driss T, Côté S, Tremblay R, Vitaro F, Arnaud C, Falissard B (2013b). "KmL3D: A Non-Parametric Algorithm for Clustering Joint Trajectories." *Computer Methods and Programs in Biomedicine*, **109**(1), 104–111.

Gower JC (1966). "Some Distance Properties of Latent Root and Vector Methods Used in Multivariate Analysis." *Biometrika*, **53**(3–4), 325–338.

Hammerlindl A, Bowman J, Prince T (2014). *Asymptote: The Vector Graphics Language*. URL http://asymptote.sourceforge.net/.

Hartigan JA, Wong MA (1979). "Algorithm AS 136: A *K*-Means Clustering Algorithm." *Journal of the Royal Statistical Society C*, **28**(1), 100–108.

Hedeker D, Gibbons RD (1997). "Application of Random-Effects Pattern-Mixture Models for Missing Data in Longitudinal Studies." *Psychological Methods*, **2**(1), 64–78.

Hennig C, Hausdorf B (2006). "Design of Dissimilarity Measures: A New Dissimilarity Between Species Distribution Areas." In V Batagelj, HH Bock, A Ferligoj, A Žiberna (eds.), *Data Science and Classification*, pp. 29–37. Springer-Verlag.

Hurvich CM, Tsai CL (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika*, **76**(2), 297–307.

Khan SS, Ahmad A (2004). "Cluster Center Initialization Algorithm for k-Means Clustering." *Pattern Recognition Letters*, **25**(11), 1293–1302.

Komárek A (2009). "A New R Package for Bayesian Estimation of Multivariate Normal Mixtures Allowing for Selection of Number of Components and Interval-Censored Data." *Computational Statistics & Data Analysis*, **53**(12), 3932–3947.

Komárek A, Hansen BE, Kuiper EMM, van Buuren HR, Lesaffre E (2010). "Discriminant Analysis Using a Multivariate Linear Mixed Model with a Normal Mixture in the Random Effects Distribution." *Statistics in Medicine*, **29**(30), 3267–3283.

Kryszczuk K, Hurley P (2010). "Estimation of the Number of Clusters Using Multiple Clustering Validity Indices." In N El-Gayar, J Kittler, F Roli (eds.), *Multiple Classifier Systems: Proceedings of the 9th International Workshop, MCS 2010, Cairo, Egypt, April 7–9, 2010*, pp. 114–123. Springer-Verlag.

Laird NM (1988). "Missing Data in Longitudinal Studies." *Statistics in Medicine*, **7**(1–2), 305–315.

Larroque B, Ancel PY, Marchand-Martin L, Cambonie G, Fresson J, Pierrat V, Rozé JC, Marpeau L, Thiriez G, Alberge C, others (2011). "Special Care and School Difficulties in 8-Year-Old Very Preterm Children: The Epipage Cohort Study." *PloS ONE*, **6**(7), e21361.

Larroque B, Ancel PY, Marret S, Marchand L, André M, Arnaud C, Pierrat V, Rozé JC, Messer J, Thiriez G, others (2008). "Neurodevelopmental Disabilities and Special Care of 5-Year-Old Children Born Before 33 Weeks of Gestation (the EPIPAGE Study): A Longitudinal Cohort Study." *The Lancet*, **371**(9615), 813–820.

Liao TW (2005). "Clustering of Time Series Data – A Survey." *Pattern Recognition*, **38**(11), 1857–1874.

Little RJA (1993). "Pattern-Mixture Models for Multivariate Incomplete Data." *Journal of the American Statistical Association*, **88**(421), 125–134.

Little RJA, Rubin DB (1987). *Statistical Analysis With Missing Data*, volume 4. John Wiley & Sons, New York.

Mallinckrodt CH, Lane PW, Schnell D, Peng Y, Mancuso JP (2008). "Recommendations for the Primary Analysis of Continuous Endpoints in Longitudinal Clinical Trials." *Drug Information Journal*, **42**(4), 303–319.

Milligan GW, Cooper MC (1985). "An Examination of Procedures for Determining the Number of Clusters in a Data Set." *Psychometrika*, **50**(2), 159–179.

Molenberghs G, Thijs H, Jansen I, Beunckens C, Kenward MG, Mallinckrodt C, Carroll RJ (2004). "Analyzing Incomplete Longitudinal Clinical Trial Data." *Biostatistics*, **5**(3), 445–464.

Muthén LK, Muthén B (2012). ***Mplus*** *User's Guide*. 7th edition. Muthén & Muthén, Los Angeles.

Nagin D (2005). *Group-Based Modeling of Development*. Harvard University Press.

Pena JM, Lozano JA, Larranaga P (1999). "An Empirical Comparison of Four Initialization Methods for the k-Means Algorithm." *Pattern Recognition Letters*, **20**(10), 1027–1040.

Pingault JB, Côté S, Galéra C, Genolini C, Falissard B, Vitaro F, Tremblay R (2012). "Childhood Trajectories of Inattention, Hyperactivity and Oppositional Behaviors and Prediction of Substance Abuse/Dependence: A 15-Year Longitudinal Population-Based Study." *Molecular Psychiatry*, **18**(7), 806–812.

Pingault JB, Côté S, Vitaro F, Falissard B, Tremblay R, Genolini C (2014). "The Developmental Course of Chilhood Inattention Symptoms Uniquely Predicts Educational Attainment: A 16-Year Longitudinal Study." *Psychiatry Research*, **219**(3), 707–709.

Pingault JB, Tremblay RE, Vitaro F, Carbonneau R, Genolini C, Falissard B, Côté SM (2011). "Childhood Trajectories of Inattention and Hyperactivity and Prediction of Educational Attainment in Early Adulthood: A 16-Year Longitudinal Population-Based Study." *American Journal of Psychiatry*, **168**(11), 1164–1170.

Pryor L, Tremblay R, Boivin M, Touchette E, Dubois L, Genolini C, Liu X, Falissard B, Côté S (2011). "Developmental Trajectories of Body Mass Index in Early Childhood and their Risk Factors: An 8-Year Longitudinal Study." *Archives of Pediatrics & Adolescent Medicine*, **165**(10), 906–912.

Rath TM, Manmatha R (2003). "Word Image Matching Using Dynamic Time Warping." In *Proceedings of the 2003 IEEE Computer Society Conference On Computer Vision and Pattern Recognition, 2003*, volume 2.

Ray S, Turi RH (1999). "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*, pp. 137–143.

R Core Team (2015). *R A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Redmond SJ, Heneghan C (2007). "A Method for Initialising the k-Means Clustering Algorithm Using kd-Trees." *Pattern Recognition Letters*, **28**(8), 965–973.

Rossi F, Conan-Guez B, Golli AE (2004). "Clustering Functional Data with the SOM Algorithm." In *Proceedings of ESANN*, pp. 305–312.

Rousseeuw PJ (1987). "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis." *Journal of Computational and Applied Mathematics*, **20**, 53–65.

Rubin DB (1976). "Inference and Missing Data." *Biometrika*, **63**(3), 581–592.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464.

Shim Y, Chung J, Choi IC (2005). "A Comparison Study of Cluster Validity Indices Using a Nonhierarchical Clustering Algorithm." In *International Conference on Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, volume 1, pp. 199–204.

Steinley D, Brusco MJ (2007). "Initializing k-Means Batch Clustering: A Critical Evaluation of Several Techniques." *Journal of Classification*, **24**(1), 99–121.

Subtil F, Boussari O, Bastard M, Etard JF, Ecochard R, Genolini C (2015). "An Alternative Classification to Mixture Modeling for Longitudinal Counts or Binary Measures." *Statistical Methods in Medical Research*. doi:10.1177/0962280214549040. Forthcoming.

Tarpey T, Kinateder KKJ (2003). "Clustering Functional Data." *Journal of Classification*, **20**(1), 93–114.

Touchette E, Petit D, Tremblay R, Boivin M, Falissard B, Genolini C, Montplaisir J (2008). "Associations Between Sleep Duration Patterns and Overweight/Obesity at Age 6." *Sleep*, **31**(11), 1507–1514.

Wahl S, Krug S, Then C, Kirchhofer A, Kastenmüller G, Brand T, Skurk T, Claussnitzer M, Huth C, Heier M, others (2014). "Comparative Analysis of Plasma Metabolomics Response to Metabolic Challenge Tests in Healthy Subjects and Influence of the FTO Obesity Risk Allele." *Metabolomics*, **10**(3), 386–401.

Affiliation:

Christophe Genolini
INSERM U.1027
37 allee Jules Guesdes
31000 Toulouse, France
and
CeRSM EA 2931
UFR STAPS
Université Paris Ouest
E-mail: christophe.genolini@u-paris10.fr
URL: http://christophe.genolini.free.fr/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 65, Issue 4 *Submitted:* 2012-11-20
May 2015 *Accepted:* 2014-08-18