

Package: kml (via r-universe)

October 24, 2024

Type Package

Title K-Means for Longitudinal Data

Description An implementation of k-means specifically design to cluster longitudinal data. It provides facilities to deal with missing value, compute several quality criterion (Calinski and Harabatz, Ray and Turie, Davies and Bouldin, BIC, ...) and propose a graphical interface for choosing the 'best' number of clusters.

Version 2.5.0

Date 2024-10-21

Maintainer Christophe Genolini <christophe.genolini@free.fr>

Author Christophe Genolini [cre, aut], Bruno Falissard [ctb], Patrice Kiener [ctb]

License GPL (>= 2)

LazyData no

Collate global.R clusterLongData.R parKml.R parChoice.R kml.R

Depends methods,clv,longitudinalData (>= 2.4)

Encoding UTF-8

NeedsCompilation yes

Date/Publication 2024-10-23 17:30:08 UTC

Repository <https://christophe314.r-universe.dev>

RemoteUrl <https://github.com/cran/kml>

RemoteRef HEAD

RemoteSha dfb9b828d7a1719b7f329466f52260bfa26ec1a9

Contents

kml-package	2
affectFuzzyIndiv	4
affectIndiv	5

calculTrajFuzzyMean	6
calculTrajMean	7
choice	8
clusterLongData	10
ClusterLongData-class	12
epipageShort	14
fuzzyKmlSlow	16
generateArtificialLongData	17
getBestPostProba	19
getClusters	21
kml	22
parKml	24
ParKml-class	25
plot,ClusterLongData	27
plotMeans,ClusterLongData	29
plotTraj,ClusterLongData	30

Index 33

kml-package *~ Overview: K-means for Longitudinal data ~*

Description

This package is a implementation of k-means for longitudinal data (or trajectories). Here is an overview of the package. For the description of the algorithm, see [kml](#).

Details

```

Package:    kml
Type:      Package
Version:   2.4.1
Date:      2016-02-02
License:   GPL (>= 2)
LazyData: yes
Depends:   methods,clv,longitudinalData(>= 2.1.2)
URL:       http://www.r-project.org
URL:       http://christophe.genolini.free.fr/kml

```

Overview

To cluster data, Kml go through three steps, each of which is associated to some functions:

1. Data preparation
2. Building "optimal" partition
3. Exporting results

1. Data preparation

KmL works on object of class `ClusterLongData`. Data preparation therefore simply consists in transforming data into an object `ClusterLongData`. This can be done via function `clusterLongData` (`cld` in short). It converts a `data.frame` or a `matrix` into a `ClusterLongData`.

Instead of working on real data, one can also work on artificial data. Such data can be created with `generateArtificialLongData` (`gald` in short).

2. Building "optimal" partition

Once an object of class `ClusterLongData` has been created, the algorithm `kml` can be run.

Starting with a `ClusterLongData`, `kml` built a `Partition`, a class in package `longitudinalData`. An object of class `Partition` is a partition of trajectories into subgroups. It also contains some information like the percentage of trajectories contained in each group or some quality criterion.

`kml` is a "hill-climbing" algorithm. The specificity of this kind of algorithm is that it always converges towards a maximum, but one cannot know whether it is a local or a global maximum. It offers no guarantee of optimality. To maximize one's chances of getting a quality `Partition`, it is better to run the hill climbing algorithm several times, then to choose the best solution. By default, `kml` executes the hill climbing algorithm 20 times and chooses the `Partition` maximizing the determinant of the matrix between.

Likewise, it is not possible to know *beforehand* the optimum number of clusters. On the other hand, *afterwards*, it is possible to calculate clues that will enable us to choose. In the end, `kml` tests by default 2, 3, 4, 5 et 6 clusters, 20 times each.

3. Exporting results

When `kml` has constructed some `Partition`, the user can examine them one by one and choose to export some. This can be done via function `choice`. `choice` opens a graphic windows showing various information including the trajectories clustered by a specific `Partition`.

When some `Partition` has been selected (the user can select more than 1), it is possible to save them. The clusters are therefore exported towards the file `name-cluster.csv`. Criteria are exported towards `name-criteres.csv`. The graphs are exported according to their extension.

It is also possible to extract a partition from the object `ClusterLongData` using the function `getClusters`.

See Also

Classes : `ClusterLongData`, `Partition` in package `longitudinalData`

Methods : `clusterLongData`, `kml`, `choice`

Plot : `plot(ClusterLongData)`

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

### 1. Data Preparation
```

```

data(epipageShort)
names(epipageShort)
cldSDQ <- cld(epipageShort,timeInData=3:6,time=c(3,4,5,8))

### 2. Building "optimal" clusteration (with only 3 redrawings)
kml(cldSDQ,nbRedrawing=3,toPlot="both")

### 3. Exporting results
### To check the best's cluster numbers
plotAllCriterion(cldSDQ)

# To see the best partition
try(choice(cldSDQ))

### 4. Further analysis
epipageShort$clust <- getClusters(cldSDQ,4)
summary(glm(gender~clust,data=epipageShort,family="binomial"))

### Go back to current dir
setwd(wd)

```

affectFuzzyIndiv ~ *Function: affectFuzzyIndiv* ~

Description

Given some longitudinal data (trajectories) and k cluster's centers, `affectFuzzyIndiv` compute the matrix of individual membership (according to the algorithm fuzzy k-means).

Usage

```
affectFuzzyIndiv(traj, clustersCenter, fuzzyfier=1.25)
```

Arguments

<code>traj</code>	[matrix]: longitudinal data. Each line is an individual, each column is a time measurement.
<code>clustersCenter</code>	[matrix]: cluster's centers. Each line is a cluster's center, each column is a time measurement.
<code>fuzzyfier</code>	[numeric]: value of the fuzzyfier used to compute individual's memberships.

Details

Given a matrix of clusters center `clustersCenter` (each line is a cluster center), the function `affectFuzzyIndiv` compute for each individual and each cluster a "membership".

`affectFuzzyIndiv` used with [calculTrajFuzzyMean](#) simulates one fuzzy k-means step.

Value

Matrix of the membership. Each line is an individual, column are for clusters.

Examples

```
#####
### affectFuzzyIndiv

### Some LongitudinalData
traj <- gald()["traj"]

### 4 clusters centers
center <- traj[runif(4,1,nrow(traj)),]

### Affectation of each individual
affectFuzzyIndiv(traj,center)
```

affectIndiv ~ Functions: *affectIndiv* & *affectIndivC* ~

Description

Given some longitudinal data (trajectories) and k clusters' centers, *affectIndiv* and *affectIndivC* affect each individual to the cluster whose centre is the closest.

Usage

```
affectIndiv(traj, clustersCenter, distance = function(x,y){dist(rbind(x, y))})
affectIndivC(traj, clustersCenter)
```

Arguments

traj	[matrix(numeric)]: longitudinal data. Each line is an individual, each column is a time measurement.
clustersCenter	[matrix(numeric)]: clusters centre. Each line is a cluster center, each column is a time measurement.
distance	[numeric <- function(trajectory, trajectory)]: use to estimate the distance between an individual and a clusters center.

Details

Given a matrix of clusters center *clustersCenter* (each line is a cluster center), the function *affectIndiv* affect each individual of the matrix *traj* to the closest clusters (according to distance). *affectIndivC* does the same but assume that the distance is the Euclidean distance. *affectIndivC* is written in C (and is therefor much faster).

affectIndiv used with [calculTrajMean](#) simulates one k-means step.

Value

Object of class `Partition`.

Examples

```
#####
### affectIndiv

### Some trajectories
traj <- gald()["traj"]

### 4 clusters centers
center <- traj[runif(4,1,nrow(traj)),]

### Affectation of each individual
system.time(part <- affectIndiv(traj,center))
system.time(part <- affectIndivC(traj,center))
```

calculTrajFuzzyMean ~ *Function: calculTrajFuzzyMean* ~

Description

Given some longitudinal data and a group's membership, `calculFuzzyMean` computes the mean trajectories of each cluster.

Usage

```
calculTrajFuzzyMean(traj, fuzzyClust)
```

Arguments

`traj` [matrix]: longitudinal data. Each line is an individual, each column is a time measurement.

`fuzzyClust` [matrix(numeric)]: membership matrix of each individual.

Details

Given a matrix of individual membership, the function `calculTrajFuzzyMean` compute the mean trajectory of each clusters.

`affectFuzzyIndiv` used with `calculTrajFuzzyMean` simulates one fuzzy k-means step.

Value

A matrix with `k` line and `t` column containing `k` clusters centers. Each line is a center, each column is a time measurement.

Examples

```
#####
### calculTrajFuzzyMean

### Some LongitudinalData
traj <- gald()["traj"]

### 4 clusters centers
center <- traj[runif(4,1,nrow(traj)),]

### Affectation of each individual
membership <- affectFuzzyIndiv(traj,center)

### Computation of the mean's trajectories
calculTrajFuzzyMean(traj,membership)
```

calculTrajMean ~ Functions: calculTrajMean & calculTrajMeanC ~

Description

Given some longitudinal data and a cluster affectation, calculTrajMean and calculTrajMeanC compute the mean trajectories of each cluster.

Usage

```
calculTrajMean(traj, clust, centerMethod = function(x){mean(x, na.rm =TRUE)})
calculTrajMeanC(traj, clust)
```

Arguments

traj	[matrix(numeric)]: longitudinal data. Each line is an individual, each column is a time measurement.
clust	[vector(numeric)]: affectation of each individual.
centerMethod	[trajectory <- function(matrix(numeric))]: function that compute the mean trajectory of a group of trajectories.

Details

Given a vector of affectation to a cluster, the function calculTrajMean compute the "central" trajectory of each clusters. The "center" can be define using the argument centerMethod. calculTrajMeanC does the same but assume that the center definition is the classic "mean". calculTrajMeanC is written in C (and is therefor much faster).

affectIndiv used with [calculTrajMean](#) simulates one k-means step.

Value

A matrix with k line and t column containing k clusters centers. Each line is a center, each column is a time measurement.

Examples

```
#####
### calculMean

### Some trajectories
traj <- gald()["traj"]

### A cluster affectation
clust <- initializePartition(3,200,"randomAll")

### Computation of the cluster's centers
system.time(centers <- calculTrajMean(traj,clust))
system.time(centers <- calculTrajMeanC(traj,clust))
```

choice

~ Function: choice ~

Description

choice lets the user choose some Partition he wants to export.

Usage

```
choice(object, typeGraph = "bmp")
```

Arguments

object	[ClusterLongData]: Object containing the trajectories and all the Partition found by kml.
typeGraph	[character] for every selected Partition, choice export some graphs. typeGraph set the format that will be used. Possible formats are the ones available for savePlot .

Details

choice is a function that lets the user see the Partition found by kml. At first, choice opens a graphics window (for Linux user, the windows should be explicitly open using `x11(type = "Xlib")`). On the left side, all the Partition contain in Object are plotted by a number (the number of cluster of the Partition). The level of the number is proportionnal to a quality criteria (like Calinski & Harabatz). One Partition is 'active', it is the one marked by a black dot.

On the right side, the trajectories of Object are drawn, according to the active Partition.

From there, choice offers numerous options :

Arrow Change the active Partition.

Space Select/unselect a Partition (the selected Partition are surrounded by a circle).

Return Export all the selected Partition, then quit the function choice.

'e' Change the display (Trajectories alone / quality criterion alone / both)

'd' Change actif criterion.

'c' Sort the Partition according to the actif criterion.

'r' Change the trajectories' style.

'f' Change the means trajectories's style.

'g/t' Change the symbol size.

'y/h' Change the number of symbols.

When 'return' is pressed (or 'm' using Linux), the selected Partition are exported. Exporting is done in a specific named `objectName-Cx-y` where x is the number of cluster and y is the order in the sublist. Four files are created :

objectName-Cx-y-Clusters.csv Table with two columns. The first is the identifier of each trajectory (idAll); the second holds the cluster's affectation of the trajectory.

objectName-Cx-y-Detail.csv Table containing information about the clusteration (percentage of individual in each cluster, various qualities criterion, algorithm used to find the partition and convergence time.)

objectName-Cx-y-Traj.bmp Graph representing the trajectories. All the parameters set during the visualization (color of the trajectories, symbols used, mean color) are used for the export. Note that the 'typeGraph' argument can be used to export the graph in a format different than 'bmp'.

objectName-Cx-y-TrajMean.bmp Graph representing the means trajectories of each clusters. All the parameters set during the visualization (color of the trajectories, symbols used, mean color) are used for the export.

This four file are created for each selected Partition. In addition, two 'global' graphes are created :

objectName-criterionActif.bmp Graph presenting the values of the criterionActifall for all the Partition.

objectName-criterionAll.bmp For each cluster's number, the first Partition is considered. This graph presents on a single display the values of all the criterion for each first Partition. It is helpfull to compare the various qualities criterion.

Value

For each selected Partition, four files are saved, plus two global files.

See Also

Overview: [kml-package](#)

Classes : [ClusterLongData](#), Partition in package longitudinalData

Methods : [kml](#)

Plot : [plot](#)

Examples

```

### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

### Creation of artificial data
cld1 <- gald(25)

### Clusterisation
kml(cld1,3:5,nbRedrawing=2,toPlot='both')

### Selection of the clustering we want
# (note that "try" is for compatibility with CRAN only,
# you probably can use "choice(cld1)")
try(choice(cld1))

### Go back to current dir
setwd(wd)

```

clusterLongData ~ Function: clusterLongData (or cld) ~

Description

clusterLongData (or cld in short) is the constructor for [ClusterLongData](#) object.

Usage

```

clusterLongData(traj, idAll, time, timeInData, varNames, maxNA)
cld(traj, idAll, time, timeInData, varNames, maxNA)

```

Arguments

traj	[matrix(numeric)] or [data.frame]: structure containing the trajectories. Each line is the trajectory of an individual. The columns refer to the time during which measures were made.
idAll	[vector(character)]: single identifier for each trajectory (ie each individual). Note that the identifiers are of type character (that allow to deal identifiers like XUK32-612, identifiers that our favorite epidemiologists are so good at providing). If idAll are numeric, they are converted into characters.
time	[vector(numeric)]: time at which measures were made.
timeInData	[vector(numeric)]: precise the column containing the trajectories.
varNames	[character]: name of the variable being measured.
maxNA	[numeric]: maximum number of NA that are tolerates on a trajectory. If a trajectory has more missing than maxNA, then it is remove from the analysis.

Details

clusterLongData construct a object of class [ClusterLongData](#). Two cases can be distinguished:

traj is an array: lines are individual. Column are time of measurment.

If idAll is missing, the individuals are labelled i1, i2, i3,...

If timeInData is missing, all the column are used (timeInData=1:ncol(traj)).

If traj is a data.frame: lines are individual. Column are time of measurement.

If idAll is missing, then the first column of the data.frame is used for idAll

If timeInData is missing and idAll is missing, then all the columns but the first are used for timeInData (the first is omitted since it is already used for idAll): idAll=traj[,1], timeInData=2:ncol(traj).

If timeInData is missing but idAll is not missing, then all the column including the first are used for timeInData: timeInData=1:ncol(traj).

Value

An object of class [ClusterLongData](#).

Author

Christophe Genolini

1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France

2. CeRSME, EA 2931, UFR STAPS, Université de Paris Ouest-Nanterre-La Défense / Nanterre / France

References

[1] C. Genolini and B. Falissard
"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010

[2] C. Genolini and B. Falissard
"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

Overview: [kml-package](#)

Classes : [ClusterLongData](#)

Methods : [choice](#), [kml](#)

Plot : [plot\(ClusterLongData\)](#)

Examples

```
#####  
### From matrix  
  
### Small data
```

```

mat <- matrix(c(1,NA,3,2,3,6,1,8,10),3,3,dimnames=list(c(101,102,104),c("T2","T4","T8")))
clusterLongData(mat)
(ld1 <- clusterLongData(traj=mat,idAll=as.character(c(101,102,104)),time=c(2,4,8),varNames="V"))
plot(ld1)

### Big data
mat <- matrix(runif(1051*325),1051,325)
(ld2 <- clusterLongData(traj=mat,idAll=paste("I-",1:1051,sep=""),time=(1:325)+0.5,varNames="R"))

#####
### From data.frame

dn <- data.frame(id=1:3,v1=c(NA,2,1),v2=c(NA,1,0),v3=c(3,2,2),v4=c(4,2,NA))

### Basic
clusterLongData(dn)

### Selecting some times
(ld3 <- clusterLongData(dn,timeInData=c(1,2,4),varNames=c("Hyp")))

### Excluding trajectories with more than 1 NA
(ld3 <- clusterLongData(dn,maxNA=1))

```

ClusterLongData-class ~ Class: ClusterLongData ~

Description

ClusterLongData is an object containing trajectories and associated Partition (from package LongitudinalData).

Objects from the Class

`kml` is an algorithm that builds a set of Partition from longitudinal data. ClusterLongData is the object containing the original longitudinal data and all the Partition that `kml` finds.

When created, an ClusterLongData object simply contains initial data (the trajectories). After the execution of `kml`, it contains the original data and the Partition which has just been calculated by `kml`.

Note that if `kml` is executed several times, every new Partition is added to the original ones, no pre-existing Partition is erased.

Slots

`idAll` [vector(character)]: Single identifier for each of the trajectory (each individual). Useful for exporting clusters.

`idFewNA` [vector(character)]: Restriction of `idAll` to the trajectories that does not have 'too many' missing value. See `maxNA` for details.

`time` [numeric]: Time at which measures are made.
`varNames` [character]: Name of the variable measured.
`traj` [matrix(numeric)]: Contains the longitudinal data. Each line is the trajectories of an individual. Each column is the time at which measures are made.
`dimTraj` [vector2(numeric)]: size of the matrix `traj` (ie `dimTraj=c(length(idFewNA), length(time))`).
`maxNA` [numeric] or [vector(numeric)]: Individual whose trajectories contain 'too many' missing values are excluded from `traj` and will not be used in the analysis. Their identifier is preserved in `idAll` but not in `idFewNA`. 'too many' is defined by `maxNA`: a trajectory with more missing than `maxNA` is excluded.
`reverse` [matrix(numeric)]: if the trajectories are scaled using the function `scale`, the 'scaling parameters' (probably mean and standard deviation) are saved in `reverse`. This is useful to restore the original data after a scaling operation.
`criterionActive` [character]: Store the criterion name that will be used by functions that need a single criterion (like `plotCriterion` or `ordered`).
`initializationMethod` [vector(character)]: list all the initialization methods that have already been used to find some `Partition` (useful to not run several times a deterministic method).
`sorted` [logical]: are the `Partition` currently held in the object sorted in decreasing order?
`c1` [list(Partition)]: list of `Partition` with 1 cluster.
`c2` [list(Partition)]: list of `Partition` with 2 clusters.
`c3` [list(Partition)]: list of `Partition` with 3 clusters.
...
`c26` [list(Partition)]: list of `Partition` with 26 clusters.

Extends

Class `LongData`, directly. Class `ListPartition`, directly.

Construction

Class `ClusterLongData` objects can be constructed via function `clusterLongData` that turns a `data.frame` or a `matrix` into a `ClusterLongData`. Note that some artificial data can be generated using `gald`.

Methods

`object['xxx']` Get the value of the field `xxx`. Inherit from `LongData` and `ListPartition`.
`object['xxx']<-value` Set the field `xxx` to `value`. `xxx`. Inherit from class `ListPartition`.
`plot` Display the `ClusterLongData` according to a class `Partition`.

Special thanks

Special thanks to Boris Hejblum for debugging the '[' and '['<- operators (the previous version was not compatible with the `matrix` package, which is used by `lme4`).

See Also

Overview: [kml-package](#)
 Classes : `classes Partition, LongData, ListPartition`
 Methods : `clusterLongData, kml, choice`
 Plot : `plot(ClusterLongData), plotCriterion`

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

#####
### Creation of some trajectories

traj <- matrix(c(1,2,3,1,4, 3,6,1,8,10, 1,2,1,3,2, 4,2,5,6,3, 4,3,4,4,4, 7,6,5,5,4),6)

myCld <- clusterLongData(
  traj=traj,
  idAll=as.character(c(100,102,103,109,115,123)),
  time=c(1,2,4,8,15),
  varNames="P",
  maxNA=3
)

#####
### get and set
myCld["idAll"]
myCld["varNames"]
myCld["traj"]

#####
### Creation of a Partition
part2 <- partition(clusters=rep(1:2,3),myCld)
part3 <- partition(clusters=rep(1:3,2),myCld)

#####
### Adding a clusterization to a clusterizLongData
myCld["add"] <- part2
myCld["add"] <- part3
myCld

### Go back to current dir
setwd(wd)
```

Description

A subset of the longitudinal study EPIPAGE.

Usage

```
data(epipageShort)
```

Format

id unique identifier for each patient.

gender Male or Female.

sdq3 score of the Strengths and Difficulties Questionnaire at 3 years old.

sdq4 score of the Strengths and Difficulties Questionnaire at 4 years old.

sdq5 score of the Strengths and Difficulties Questionnaire at 5 years old.

sdq8 score of the Strengths and Difficulties Questionnaire at 8 years old.

Details

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 behavioural 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 and then to the age of 8.

The SDQ is a behavioral questionnaire for children and adolescents ages 4 through 16 years old. It measures the severity of the disability (higher score indicate higher disability).

The database belongs to the INSERM unit U953 (P.Y. Ancel). which has agreed to include the variable SDQ in the library.

References

- [lar08] Larroque B, Ancel P, Marret S, Marchand L, André M, Arnaud C, Pierrat V, Rozé J, Messer J, Thiriez G, et al. (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.
- [lau11] Laurent C, Kouanfack C, Laborde-Balen G, Aghokeng A, Mbougua J, Boyer S, Carrieri M, Mben J, Dontsop M, Kazé S, et al. (2011). "Monitoring of HIV viral loads, CD4 cell counts, and clinical assessments versus clinical monitoring alone for antiretroviral therapy in rural district hospitals in Cameroon (Stratall ANRS 12110/ESTHER): a randomised non-inferiority trial." *The Lancet Infectious Diseases*, 11(11), 825-833.

Examples

```
data(epipageShort)
str(epipageShort)
```

fuzzyKmlSlow

~ Algorithm fuzzy kml: Fuzzy k-means for Longitudinal data ~

Description

fuzzyKmlSlow is a new implementation of fuzzy k-means for longitudinal data (or trajectories).

Usage

```
fuzzyKmlSlow(traj, clusterAffectation, toPlot = "traj",
             fuzzyfier = 1.25, parAlgo = parALGO())
```

Arguments

traj	[matrix(numeric)]: Matrix holding the longitudinal data
clusterAffectation	[vector(numeric)]: Initial starting condition
toPlot	[character]: if "traj", then the trajectories are plot. If "none", there is no graphical display (faster).
fuzzyfier	[numeric]: value of the fuzzy k-means algorithm fuzzyfier.
parAlgo	[ParKml]: default parameters for the algorithm.

Details

fuzzyKmlSlow is a new implementation of fuzzy k-means for longitudinal data (or trajectories). To date, it is written in R (and not in C, this explain the "slow")

Value

The matrix of the individual membership.

See Also

[kml](#)

Examples

```
### Data generation
traj <- gald(25)["traj"]
partInit <- initializePartition(3,100,"kmeans--",traj)

### fuzzy Kml
partResult <- fuzzyKmlSlow(traj,partInit)
```

```
generateArtificialLongData
```

~ Function: generateArtificialLongData (or gald) ~

Description

This function build up an artificial longitudinal data set (single variable-trajectory) an turn it into an object of class `ClusterLongData`.

Usage

```
gald(nbEachClusters=50, time=0:10, varNames="V",
     meanTrajectories=list(function(t){0}, function(t){t},
                           function(t){10-t}, function(t){-0.4*t^2+4*t}),
     personalVariation=function(t){rnorm(1, 0, 2)},
     residualVariation=function(t){rnorm(1, 0, 2)},
     decimal=2, percentOfMissing=0)
```

```
generateArtificialLongData(nbEachClusters=50, time=0:10, varNames="V",
                           meanTrajectories=list(function(t){0}, function(t){t},
                                                  function(t){10-t}, function(t){-0.4*t^2+4*t}),
                           personalVariation=function(t){rnorm(1, 0, 2)},
                           residualVariation=function(t){rnorm(1, 0, 2)},
                           decimal=2, percentOfMissing=0)
```

Arguments

`nbEachClusters` [numeric] or [vector(numeric)]: number of trajectories that each cluster must contain. If a single number is given, it is duplicated for all groups.

`time` [vector(numeric)]: time at which measures are made.

`varNames` [character]: name of the variable.

`meanTrajectories` [list(function)]: lists the functions define the average trajectories of each cluster.

`personalVariation` [function] or [list(function)]: lists the functions defining the personal variation between an individual and the mean trajectories of its cluster. Note that these function should be constant function (the personal variation can not evolve with time). If a single function is given, it is duplicated for all groups (see detail).

`residualVariation` [function] or [list(function)]: lists the functions generating the noise of each trajectory within its own cluster. If a single function is given, it is duplicated for all groups (see detail).

`decimal` [numeric]: number of decimals used to round up values.

percentOfMissing

[numeric]: percentage (between 0 and 1) of missing data generated in each cluster. If a single value is given, it is duplicated for all groups. The missing values are Missing Completely At Random (MCAR).

Details

`generateArtificialLongData` (`gald` in short) is a function that construct a set of artificial longitudinal data. Each individual is considered as belonging to a group. This group follows a theoretical trajectory, function of time. These functions (one per group) are given via the argument `meanTrajectories`.

Within a group, the individual undergoes individual variations. Individual variations are given via the argument `residualVariation`.

The number of individuals in each group is given by `nbEachClusters`.

Finally, it is possible to add missing values randomly (MCAR) striking the data thanks to `percentOfMissing`.

Value

An object of class `ClusterLongData`.

Author

Christophe Genolini

1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France

2. CeRSME, EA 2931, UFR STAPS, Université de Paris Ouest-Nanterre-La Défense / Nanterre / France

References

[1] C. Genolini and B. Falissard
"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010

[2] C. Genolini and B. Falissard
"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

`ClusterLongData`, `clusterLongData`

Examples

```
par(ask=TRUE)
```

```
#####  
### Default example
```

```

(ex1 <- generateArtificialLongData())
plot(ex1)
plot(ex1,parTraj=parTRAJ(col=rep(2:5,each=50)))

#####
### Three diverging lines

ex2 <- generateArtificialLongData(meanTrajectories=list(function(t)0,function(t)-t,function(t)t))
plot(ex2,parTraj=parTRAJ(col=rep(2:4,each=50)))

#####
### Three diverging lines with high variance, unbalance groups and missing value

ex3 <- generateArtificialLongData(
  meanTrajectories=list(function(t)0,function(t)-t,function(t)t),
  nbEachClusters=c(100,30,10),
  residualVariation=function(t){rnorm(1,0,3)},
  percentOfMissing=c(0.25,0.5,0.25)
)
part3 <- partition(rep(1:3,c(100,30,10)))
plot(ex3,parTraj=parTRAJ(col=rep(2:4,c(100,30,10))))

#####
### Four strange functions

ex4 <- generateArtificialLongData(
  nbEachClusters=c(300,200,100,100),
  meanTrajectories=list(function(t){-10+2*t},function(t){-0.6*t^2+6*t-7.5},
    function(t){10*sin(t)},function(t){30*dnorm(t,2,1.5)}),
  residualVariation=function(t){rnorm(1,0,3)},
  time=0:10,decimal=2,percentOfMissing=0.3)
plot(ex4,parTraj=parTRAJ(col=rep(2:5,c(300,200,100,100))))

#####
### To get only longData (if you want some artificial longData
###   to deal with another algorithm), use the getteur ["traj"]

ex5 <- gald(nbEachCluster=3,time=1:3)
ex5["traj"]

par(ask=FALSE)

```

Description

Given a [ClusterLongData](#) object that hold a Partition, this function extract the best posterior probability of each individual.

Usage

```
getBestPostProba(xCld, nbCluster, clusterRank = 1)
```

Arguments

`xCld` [ClusterLongData]: object from who a cluster should be extracted.
`nbCluster` [integer]: number of cluster of the desired cluster.
`clusterRank` [integer]: rank of the partition in the clusters list.

Details

Given a [ClusterLongData](#) object that hold a Partition, this function extract the best posterior probability of each individual.

Value

A vector of numeric.

See Also

[ClusterLongData](#)

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

### Creation of an object ClusterLongData
myCld <- gald(20)

### Computation of some partition
kml(myCld,2:4,3)

### Extraction the best posterior probabilities
### form the list of partition with 3 clusters of the second clustering
getBestPostProba(myCld,3,2)

### Go back to current dir
setwd(wd)
```

getClusters ~ Function: getClusters ~

Description

This function extract a cluster affectation from an [ClusterLongData](#) object.

Usage

```
getClusters(xCld, nbCluster, clusterRank = 1, asInteger = FALSE)
```

Arguments

xCld	[ClusterLongData]: object from who a cluster should be extracted.
nbCluster	[integer]: number of cluster of the desired cluster.
clusterRank	[integer]: rank of the partition in the clusters list.
asInteger	[logical]: should the cluster be given as a vector of integer ? If FALSE, a vector of LETTERS is return.

Details

This function extract a clusters from an object [ClusterLongData](#). It is almost the same as `xCld[paste("c", nbCluster, sep="")]` except that the individual with too many missing value (and thus excludes from the analysis) will be noted by some NA values.

Value

A vector of numeric or a LETTER, according to the value of `asInteger`.

See Also

[ClusterLongData](#)

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

### Creation of an object ClusterLongData
myCld <- gald(20)

### Computation of some partition
kml(myCld,2:4,3)

### Extraction form the list of partition with 3 clusters
### of the second clustering
getClusters(myCld,3,2)
```

```
### Go back to current dir
setwd(wd)
```

kml

~ Algorithm kml: K-means for Longitudinal data ~

Description

kml is a implementation of k-means for longitudinal data (or trajectories). This algorithm is able to deal with missing value and provides an easy way to re roll the algorithm several times, varying the starting conditions and/or the number of clusters looked for.

Here is the description of the algorithm. For an overview of the package, see [kml-package](#).

Usage

```
kml(object, nbClusters=2:6, nbRedrawing=20, toPlot="none", parAlgo=parALGO())
```

Arguments

object	[ClusterLongData]: contains trajectories to cluster as well as previous Partition.
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 partitions with respectively 2, then 3, ... up to 6 clusters. Maximum number of cluster is 26.
nbRedrawing	[numeric]: Sets the number of time that k-means must be re-run (with different starting conditions) for each number of clusters.
toPlot	[character]: either 'traj' for plotting trajectories alone, 'criterion' for plotting criterion alone, 'both' for plotting both or 'none' for not display anything (faster).
parAlgo	[ParKml]: parameters used to run the algorithm. They can be change using the function parKml . Option are mainly 'saveFreq', 'maxIt', 'imputationMethod', 'distance' and 'startingCondition'. See ParKml for details.

Details

kml works on object of class ClusterLongData. For each number included in nbClusters, kml computes a Partition then stores it in the field cX of the object ClusterLongData according to the number of clusters 'X'. The algorithm starts over as many times as it is told in nbRedrawing. By default, it is executed for 2, 3, 4, 5 and 6 clusters 20 times each, namely 100 times.

When a Partition has been found, it is added to the corresponding slot c1, c2, c3, ... or c26. The sublist cX stores the all Partition with X clusters. Inside a sublist, the Partition can be sorted from the biggest quality criterion to the smallest (the best are stored first, using `ordered, ListPartition`), or not.

Note that `Partition` are saved throughout the algorithm. If the user interrupts the execution of `kml`, the result is not lost. If the user run `kml` on an object, then running `kml` again on the same object will add some new `Partition` to the one already found.

The possible starting conditions are defined in [initializePartition](#).

Value

A `ClusterLongData` object, after having added some `Partition` to it.

Optimisation

Behind `kml`, there are two different procedures :

1. Fast: when the parameter `distance` is set to "euclidean" and `toPlot` is set to 'none' or 'criterion', `kml` call a C compiled (optimized) procedure.
2. Slow: when the user defines its own distance or if he wants to see the construction of the clusters by setting `toPlot` to 'traj' or 'both', `kml` uses a R non compiled programmes.

The C procedure is 25 times faster than the R one.

So we advice to use the R procedure 1/ for trying some new method (like using a new distance) or 2/ to "see" the very first clusters construction, in order to check that every thing goes right. Then it is better to switch to the C procedure (like we do in [Example](#) section).

If for a specific use, you need a different distance, feel free to contact the author.

See Also

Overview: [kml-package](#)

Classes : `ClusterLongData`, `Partition` in package `longitudinalData`

Methods : `clusterLongData`, `choice`

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

### Generation of some data
cld1 <- generateArtificialLongData(25)

### We suspect 3, 4 or 6 clusters, we want 3 redrawing.
### We want to "see" what happen (so printCal and printTraj are TRUE)
kml(cld1,c(3,4,6),3,toPlot='both')

### 4 seems to be the best. We want 7 more redrawing.
### We don't want to see again, we want to get the result as fast as possible.
kml(cld1,4,10)

### Go back to current dir
setwd(wd)
```

parKml *~ Function: parKml ~*

Description

parKml and parALGO are constructor for the object [ParKml](#).

Usage

```
parKml(saveFreq,maxIt,imputationMethod,distanceName,power,distance,
       centerMethod,startingCond,nbCriterion,scale)
```

```
parALGO(saveFreq=100,maxIt=200,imputationMethod="copyMean",
        distanceName="euclidean",power=2,distance=function(){} ,
        centerMethod=meanNA,startingCond="nearlyAll",nbCriterion=1000,scale=TRUE)
```

Arguments

saveFreq	[numeric]: Long computations can take several days. So it is possible to save the object ClusterLongData on which works kml once in a while. saveFreq defines the frequency of the saving process. The ClusterLongData is saved every saveFreq clustering calculations. The object is saved in the file objectName.Rdata in the curent folder. If saveFreq is set to Inf, the object is never saved.
maxIt	[numeric]: Set a limit to the number of iteration if convergence is not reached.
imputationMethod	[character]: the calculation of quality criterion can not be done if some value are missing. imputationMethod define the method use to impute the missing value. See imputation for detail.
distanceName	[character]: name of the distance used by k-means. If the distanceName is one of "manhattan", "euclidean", "minkowski", "maximum", "canberra" or "binary", a compiled optimized version specificaly design for trajectories version is used. Otherwise, the function define in the slot distance is used.
power	[numeric]: If distanceName="minkowski", this define the power that will be used.
distance	[numeric <- function(trajA,trajB)]: function that computes the distance between two trajectories. If no function is specified, the Euclidian distance with Gower adjustment (to deal with missing value) is used.
centerMethod	[numeric <- function(vector(numeric))]: k-means algorithm computes the centers of each cluster. It is possible to personalize the definition of "center" by defining a function "centerMethod". This function should take a vector of numeric as argument and return a single numeric -the center of the vector-.
startingCond	[character]: specifies the starting condition. Should be one of "randomAll", "randomK", "maxDist", "kmeans++", "kmeans+", "kmeans-" or "kmeans-" (see initializePartition for details). It also could take two specifics values: "all"

	stands for c("maxDist","kmeans-") then an alternance of "kmeans-" and "randomK" while "nearlyAll" stands for "kmeans-" then an alternance of "kmeans-" and "randomK".
nbCriterion	[numeric]: set the maximum number of quality criterion that are display on the graph (since displaying a high criterion number an slow down the overall process). The default value is 100.
scale	[logical]: if TRUE, then the data will be automatically scaled (using the function scale with default values) before the execution of k-means on joint trajectories. Then the data will be restore (using the function restoreRealData) just before the end of the function kml3d . This option has no effect on kml .

Details

parKml is the constructor of object [ParKml](#).

Value

An object [ParKml](#).

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

### Generation of some data
cld1 <- generateArtificialLongData()

### Setting two different set of option :
(option1 <- parALGO())
(option2 <- parALGO(distanceName="maximum", centerMethod=function(x)median(x, na.rm=TRUE)))

### Running kml We suspect 3, 4 or 5 clusters, we want 3 redrawing.
kml(cld1, 3:5, 3, toPlot="both", parAlgo=option1)
kml(cld1, 3:5, 3, toPlot="both", parAlgo=option2)

### Go back to current dir
setwd(wd)
```

ParKml-class

~ Class: "ParKml" ~

Description

ParKml is an object containing some parameter used by [kml](#).

Slots

- saveFreq** [numeric]: Long computations can take several days. So it is possible to save the object `ClusterLongData` on which works `kml` once in a while. `saveFreq` defines the frequency of the saving process. The `ClusterLongData` is saved every `saveFreq` clustering calculations. The object is saved in the file `objectName.Rdata` in the current folder. If `saveFreq` is set to `Inf`, the object is never saved.
- maxIt**: [numeric]: Set a limit to the number of iteration if convergence is not reached.
- imputationMethod**: [character]: the calculation of quality criterion can not be done if some value are missing. `imputationMethod` define the method use to impute the missing value. See [imputation](#) for detail.
- distanceName**: [character]: name of the distance used by k-means. If the `distanceName` is one of "manhattan", "euclidean", "minkowski", "maximum", "canberra" or "binary", a compiled optimized version specifically design for trajectories version is used. Otherwise, the function define in the slot `distance` is used.
- power**: [numeric]: If `distanceName="minkowski"`, this define the power that will be used.
- distance**: [numeric <- function(trajA, trajB)]: function that computes the distance between two trajectories. This field is used only if 'distanceName' is not one of the classical function.
- centerMethod**: [numeric <-function(vector(numeric))]: k-means algorithm computes the centers of each cluster. It is possible to personalize the definition of "center" by defining a function "centerMethod". This function should take a vector of numeric as argument and return a single numeric -the center of the vector-.
- startingCond**: [character]: specifies the starting condition. Should be one of "randomAll", "randomK", "maxDist", "kmeans++", "kmeans+", "kmeans-" or "kmeans-" (see [initializePartition](#) for details). It also could take two specific values: "all" stands for `c("maxDist","kmeans-")` then an alternance of "kmeans-" and "randomK" while "nearlyAll" stands for "kmeans-" then an alternance of "kmeans-" and "randomK".
- nbCriterion** [numeric]: set the maximum number of quality criterion that are display on the graph (since displaying a high criterion number slow down the overall process). The default value is 100.
- scale** [logical]: if TRUE, then the data will be automatically scaled (using the function [scale](#) with default values) before the execution of k-means on joint trajectories. Then the data will be restore (using the function [restoreRealData](#)) just before the end of the function `kml3d`. This option has no effect on `kml`.

Methods

`object['xxx']` Get the value of the field xxx.

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

### Building data
myCld <- gald()
```

```

### Standard kml
kml(myCld,,3,toPlot="both")

### Using median instead of mean
parWithMedian <- parALGO(centerMethod=function(x){median(x,na.rm=TRUE)})
kml(myCld,,3,toPlot="both",parAlgo=parWithMedian)

### Using distance max
parWithMax <- parALGO(distanceName="maximum")
kml(myCld,,3,toPlot="both",parAlgo=parWithMax)

### Go back to current dir
setwd(wd)

```

plot,ClusterLongData ~ Function: plot for ClusterLongData ~

Description

plot the trajectories of an object ClusterLongData relatively to a Partition.

Usage

```

## S4 method for signature 'ClusterLongData,ANY'
plot(x,y=NA,parTraj=parTRAJ(),parMean=parMEAN(),
     addLegend=TRUE, adjustLegend=-0.12,toPlot="both",criterion=x["criterionActif"],
     nbCriterion=1000, ...)

```

Arguments

x	[ClusterLongData]: Object containing the trajectories to plot.
y	[numeric] or [vector(numeric)]: Give the Partition to represent. If y is missing, the Partition with the highest quality criterion (the actif one) is selected. If y is a number, the first Partition of the sublist c-y is selected. If y is a couple of numeric, the y[2]th Partition of the sublist c-y[1] is selected.
parTraj	[ParLongData]: Specification of the plotting parameters of the individual trajectories. Fields that can be changes are 'type','col','pch','xlab' and 'ylab'. In addition to the standard possible values, the option col="clusters" can be use to color the individual trajectories according to their clusters (exemple: parTraj=parTRAJ(type="o",col="clusters")). See ParLongData in package longitudinalData for details.
parMean	[ParLongData]: Specification of the plotting parameters of the mean trajectories. Fields that can be changes are 'type','col','pch','pchPeriod' and 'cex'. See ParLongData in package longitudinalData for details.

toPlot	[character]: either 'traj' for plotting trajectories alone, 'criterion' for plotting criterion alone, 'both' for plotting both or 'none' for not display anything (faster).
criterion	[character] or [vector(character)]: criterion to display (only if 'toPlot' is 'criterion' or 'both'). If a single criterion is given, it will be display for all the Partition. If several criterion are used, they will be display for the first Partition for each clusters' numbers.
nbCriterion	[numeric]: if a single criterion is given to criterion (and thus is displayed for 'all' the Partition), this slot allows to fix a limit on the number of points that will be display.
addLegend	[logical]: should the legend be displayed?
adjustLegend	[numeric]: fix the hight of the legend
...	Some other parameters can be passed to the method (like "xlab" or "ylab").

Details

plot the trajectories of an object [ClusterLongData](#) relatively to the 'best' Partition, or to the Partition define by y.

Graphical option concerning the individual trajectory (col, type, pch and xlab) can be change using `parTraj`. Graphical option concerning the cluster mean trajectory (col, type, pch, pchPeriod and cex) can be change using `parMean`. For more detail on `parTraj` and `parMean`, see object of class `ParLongData` in package `longitudinalData`.

See Also

Overview: [kml-package](#)
 Classes : [ClusterLongData](#)
 Plot : [plot: overview](#), [plotCriterion](#)

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

#####
### Construction of the data

ld <- gald()

### Basic plotting
plot(ld)

#####
### Changing graphical parameters 'par'

kml(ld,3:4,1)
```

```

### No letters on the mean trajectories
plot(ld,3,parMean=parMEAN(type="l"))

### Only one letter on the mean trajectories
plot(ld,4,parMean=parMEAN(pchPeriod=Inf))

### Color individual according to its clusters (col="clusters")
plot(ld,3,parTraj=parTRAJ(col="clusters"))

### Mean without individual
plot(ld,4,parTraj=parTRAJ(type="n"))

### No mean trajectories (type="n")
### Color individual according to its clusters (col="clusters")
plot(ld,3,parTraj=parTRAJ(col="clusters"),parMean=parMEAN(type="n"))

### Only few trajectories
plot(ld,4,nbSample=10,parTraj=parTRAJ(col='clusters'),parMean=parMEAN(type="n"))

### Go back to current dir
setwd(wd)

```

```
plotMeans, ClusterLongData
```

```
~ Function: plotMeans for ClusterLongData ~
```

Description

plotMeans plots the means' trajectories of an object ClusterLongData relatively to a Partition.

Usage

```

## S4 method for signature 'ClusterLongData,ANY'
plotMeans(x,y,parMean=parMEAN(),
  parWin=windowsCut(x['nbVar'],addLegend=TRUE),...)

```

Arguments

x	[ClusterLongData]: Object containing the trajectories to plotMeans.
y	[numeric] or [vector(numeric)]: Give the Partition to represent. If y is a number, the first Partition of the sublist c-y is selected. If y is a couple of numeric, the y[2]th Partition of the sublist c-y[1] is selected (so y=c(2,3) select the partition with 2 clusters, the third one).
parMean	[ParLongData]: Specification of the plotting parameters of the mean trajectories. Fields that can be changes are 'type','col','pch','pchPeriod' and 'cex'. See ParLongData in package longitudinalData for details.

parWin [parWindows]: Set the graphical display of the windows. See ParWindows in package longitudinalData for details.

... Some other parameters can be passed to the method.

Details

plotMeans plots the means' trajectories of an object [ClusterLongData](#) relatively to the 'best' Partition, or to the Partition define by y.

Graphical option (col, type, pch, pchPeriod and cex) can be change using parMean. For more detail on parTraj and parMean, see object of class ParLongData in package longitudinalData.

See Also

Overview: [kml-package](#)
 Classes : [ClusterLongData](#)
 PlotMeans : [plotMeans: overview](#), [plotCriterion](#)

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

#####
### Construction of the data

ld <- gald(10)
kml(ld,3:4,2)

### Basic plotMeansting
plotMeans(ld,3)

### Go back to current dir
setwd(wd)
```

plotTraj, ClusterLongData

~ Function: plotTraj for ClusterLongData ~

Description

plotTraj plot the trajectories of an object ClusterLongData relatively to a Partition.

Usage

```
## S4 method for signature 'ClusterLongData,ANY'
plotTraj(x,y,parTraj=parTRAJ(col="clusters"),
         parWin=windowsCut(x['nbVar'],addLegend=TRUE),nbSample=1000,...)
```

Arguments

x	[ClusterLongData]: Object containing the trajectories to plotTraj.
y	[numeric] or [couple(numeric)]: Give the Partition to represent. If y is missing, the Partition with the highest quality criterion (the actif one) is selected. If y is a number, the first Partition of the sublist c-y is selected. If y is a couple of numeric, the y[2]th Partition of the sublist c-y[1] is selected (so y=c(2,3) select the partition with 2 clusters, the third one).
parTraj	[ParLongData]: Specification of the plotting parameters of the individual trajectories. Fields that can be changes are 'type', 'col', 'pch', 'xlab' and 'ylab'. In addition to the standard possible values, the option col="clusters" (the default) can be use to color the individual trajectories according to their clusters (exemple: parTraj=parTRAJ(type="o", col="clusters")). See ParLongData in package longitudinalData for details.
parWin	[parWindows]: Set the graphical display of the windows. See ParWindows in package longitudinalData for details.
nbSample	[numeric]: Graphical display of huge sample can be time consuming. This parameters fixe the maximum number of trajectories (randomly chosen) that will be drawn.
...	Some other parameters can be passed to the method.

Details

plotTraj the trajectories of an object [ClusterLongData](#) relatively to the 'best' Partition, or to the Partition define by y.

Graphical option (col, type, pch and xlab) can be change using parTraj. For more detail on parTraj, see object of class ParLongData in package longitudinalData .

See Also

Overview: [kml-package](#)

Classes : [ClusterLongData](#)

PlotTraj : [plotTraj: overview](#), [plotCriterion](#)

Examples

```
### Move to tempdir
wd <- getwd()
setwd(tempdir()); getwd()

#####
### Construction of the data

ld <- gald()
kml(ld,3:4,1)

### Basic plotTrajting
plotTraj(ld,3)
```

```
### Go back to current dir  
setwd(wd)
```


Index

- * **chron**
 - choice, 8
 - ClusterLongData-class, 12
 - kml, 22
 - kml-package, 2
 - plot, ClusterLongData, 27
 - plotMeans, ClusterLongData, 29
 - plotTraj, ClusterLongData, 30
- * **classes**
 - clusterLongData, 10
 - ClusterLongData-class, 12
 - ParKml-class, 25
- * **classif**
 - choice, 8
 - ClusterLongData-class, 12
 - fuzzyKmlSlow, 16
 - kml, 22
 - kml-package, 2
 - plot, ClusterLongData, 27
 - plotMeans, ClusterLongData, 29
 - plotTraj, ClusterLongData, 30
- * **cluster**
 - choice, 8
 - ClusterLongData-class, 12
 - fuzzyKmlSlow, 16
 - generateArtificialLongData, 17
 - kml, 22
 - kml-package, 2
 - plot, ClusterLongData, 27
 - plotMeans, ClusterLongData, 29
 - plotTraj, ClusterLongData, 30
- * **datagen**
 - generateArtificialLongData, 17
- * **datasets**
 - epiPageShort, 14
- * **documentation**
 - epiPageShort, 14
- * **dplotMeans**
 - plotMeans, ClusterLongData, 29
- * **dplotTraj**
 - plotTraj, ClusterLongData, 30
- * **dplot**
 - kml, 22
 - kml-package, 2
 - plot, ClusterLongData, 27
- * **iplotMeans**
 - plotMeans, ClusterLongData, 29
- * **iplotTraj**
 - plotTraj, ClusterLongData, 30
- * **iplot**
 - choice, 8
 - kml-package, 2
 - plot, ClusterLongData, 27
- * **models**
 - kml-package, 2
- * **nonparametric**
 - choice, 8
 - ClusterLongData-class, 12
 - fuzzyKmlSlow, 16
 - kml, 22
 - kml-package, 2
- * **package**
 - kml-package, 2
- * **robust**
 - fuzzyKmlSlow, 16
 - kml, 22
 - kml-package, 2
- * **spatial**
 - choice, 8
 - ClusterLongData-class, 12
 - kml, 22
 - kml-package, 2
 - plot, ClusterLongData, 27
 - plotMeans, ClusterLongData, 29
 - plotTraj, ClusterLongData, 30
- * **ts**
 - choice, 8
 - ClusterLongData-class, 12

- fuzzyKmlSlow, 16
- generateArtificialLongData, 17
- kml, 22
- kml-package, 2
- plot, ClusterLongData, 27
- plotMeans, ClusterLongData, 29
- plotTraj, ClusterLongData, 30
- [, ClusterLongData-method
 - (ClusterLongData-class), 12
- [, ParChoice-method (choice), 8
- [, ParKml, ANY, ANY-method (parKml), 24
- [, ParKml-method (ParKml-class), 25
- [<- , ClusterLongData, character, missing, missing-method
 - (ClusterLongData-class), 12
- [<- , ParKml, ANY, ANY, ANY-method (parKml), 24
- [<- , ParKml-method (ParKml-class), 25
- affectFuzzyIndiv, 4, 6
- affectIndiv, 5
- affectIndivC (affectIndiv), 5
- calculTrajFuzzyMean, 4, 6
- calculTrajMean, 5, 7, 7
- calculTrajMeanC (calculTrajMean), 7
- choice, 3, 8, 11, 14, 23
- choice, ClusterLongData-method (choice), 8
- cld, 3
- cld (clusterLongData), 10
- ClusterLongData, 3, 9–11, 17, 18, 20, 21, 23, 24, 26, 28, 30, 31
- clusterLongData, 3, 10, 13, 14, 18, 23
- clusterLongData, ANY, ANY, ANY, ANY, ANY, ANY-method
 - (clusterLongData), 10
- clusterLongData, missing, missing, missing, missing, missing, missing-method
 - (clusterLongData), 10
- ClusterLongData-class, 12
- epipageShort, 14
- fuzzyKmlSlow, 16
- gald, 3, 13
- gald (generateArtificialLongData), 17
- generateArtificialLongData, 3, 17
- getBestPostProba, 19
- getClusters, 21
- imputation, 24, 26
- initializePartition, 23, 24, 26
- is.na, ClusterLongData-method
 - (ClusterLongData-class), 12
- kml, 2, 3, 9, 11, 12, 14, 16, 22, 24–26
- kml, ClusterLongData-method (kml), 22
- kml-package, 2, 22
- kml3d, 25, 26
- ordered, 13
- parALGO (parKml), 24
- ParKml, 16, 22, 24, 25
- parKml, 22, 24
- ParKml-class, 25
- Partition, 6
- plot, 9, 13
- plot (plot, ClusterLongData), 27
- plot (ClusterLongData), 3, 11, 14
- plot, ClusterLongData, 27
- plot, ClusterLongData, ANY-method
 - (plot, ClusterLongData), 27
- plot, ClusterLongData, missing-method
 - (plot, ClusterLongData), 27
- plot, ClusterLongData, numeric-method
 - (plot, ClusterLongData), 27
- plot, ClusterLongData, Partition-method
 - (plot, ClusterLongData), 27
- plotCriterion, 13, 14, 28, 30, 31
- plotMeans (plotMeans, ClusterLongData), 29
- plotMeans, ClusterLongData, 29
- plotMeans, ClusterLongData, ANY-method
 - (plotMeans, ClusterLongData), 29
- plotMeans, ClusterLongData, missing-method
 - (plotMeans, ClusterLongData), 29
- plotMeans, ClusterLongData, numeric-method
 - (plotMeans, ClusterLongData), 29
- plotMeans, ClusterLongData, Partition-method
 - (plotMeans, ClusterLongData), 29
- plotTraj (plotTraj, ClusterLongData), 30
- plotTraj, ClusterLongData, 30
- plotTraj, ClusterLongData, ANY-method
 - (plotTraj, ClusterLongData), 30
- plotTraj, ClusterLongData, numeric-method
 - (plotTraj, ClusterLongData), 30
- restoreRealData, 25, 26
- savePlot, 8

scale, [13](#), [25](#), [26](#)