

Package ‘longitudinalData’

March 28, 2012

Type Package

Title Longitudinal Data

Version 2.0

Date 2011-04-01

Author Christophe Genolini

Maintainer Christophe Genolini <genolini@u-paris10.fr>

Description Tools for Longitudinal Data and Longitudinal Data 3D

License GPL (>= 2)

Lazyload yes

LazyData yes

Depends methods,clv,rgl,misc3d

URL <http://www.r-project.org>

Collate global.r function.r partition.r listPartition.r parWindows.r
parLongData.r longData.r longData3d.r longDataPlot.r
imputCross.R imputTraj.R imputLinearInterpol.R imputCopyMean.R
imputation.r distanceFrechet.r clean.r

Encoding latin1

Repository CRAN

Date/Publication 2012-03-28 11:09:21

R topics documented:

longitudinalData-package	2
Constants	4
distFrechet	5
distTraj	6
expandParLongData	8
generateArtificialLongData	10
generateArtificialLongData3d	13
imputation	15
initializePartition	22
ListPartition-class	25
longData	28
LongData-class	29
longData3d	32
LongData3d-class	34
longDataFrom3d	37
longDataTo3d	38
makeLatexFile	39
ordered(ListPartition)	40
parLongData	42
ParLongData-class	44
partition	45
Partition-class	47
parWindows	49
ParWindows-class	51
pathFrechet	52
plot,LongData	54
plot3d,LongData	56
plot3dPdf	58
plotCriterion	60
qualityCriterion	61
regroup	64
restaureRealData	65
saveTrianglesAsASY	67
scale	68
windowsCut	70
Index	72

longitudinalData-package

~ Package overview: longitudinalData ~

Description

longitudinalData provide some tools to deal with the clusterization of longitudinal data.

Details

Package: longitudinalData
Type: Package
Version: 2.0
Date: 2012-04-01
License: GPL (>= 2)
Lazyload: yes
Depends: methods,clv,rgl,misc3d
URL: <http://www.r-project.org>

Overview

longitudinalData provide some tools to deal with the clusterization of longitudinal data, mainly:

1. [plot](#)
2. [imputation](#)
3. [qualityCriterion](#)

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] Christophe M. Genolini and Bruno Falissard
"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010

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

See Also

Classes: [LongData](#), [Partition](#)
Methods: [longData](#), [partition](#), [ordered](#)
Plot: [plot\(LongData\)](#), [plot3d\(LongData\)](#)
Imputation: [imputation](#)
Criterion: [qualityCriterion](#)

Examples

```
### Generation of artificial longData
data <- gald3d(percentOfMissing=0.3)
part <- partition(rep(1:3,each=50))
plot3d(data,part)

### Imputation
data1 <- imputation(data,method="copyMean")

### Quality criterion
# qualityCriterion(data,part)
```

Constants

~ Constants define in the package ~

Description

Constants define in the package ~

Usage

```
MAX_CLUSTERS
CLUSTERS_NAMES
CRITERION_MIN_OR_MAX
CRITERION_NAMES
DISTANCE_METHODS
CHOICE_STYLE
```

Value

```
MAX_CLUSTERS = 26
CLUSTER_NAMES = paste("c",2:MAX_CLUSTERS,sep="")
CRITERION_NAMES <- c("Calinski.Harabatz","Kryszczuk.Calinski","Genolini.Calinski","Ray.Turi","Davies.Bouldin","ra
DISTANCE_METHODS = c("manhattan", "euclidean", "minkowski", "maximum", "canberra",
"binary")
CHOICE_STYLE = list( typeTraj=c("l","l","n"), colTraj=c("clusters","black","black"), typeMean=c("b","b","b","b","l","l","
colMean=c("clusters","black","clusters","black","clusters","black","black"), pchMean=c("letters","letters","symbols","symb
)
```

Examples

```
### Maximum number of clusters that kml can deal with
MAX_CLUSTERS

### Names of the field that save clusters in object 'ClusterLongData'
cat(CLUSTER_NAMES,"\n")
```

```

### List of the available criterion
CRITERION_NAMES

### Distance available
DISTANCE_METHODS[2]

### Define the style use by choice
CHOICE_STYLE[['typeTraj']][2]

```

distFrechet ~ Function: Frechet distance ~

Description

Compute Frechet distance between two trajectories.

Usage

```
distFrechet(P, Q, method = "max", Fdist = dist)
```

Arguments

P	[vector(numeric)] First trajectories.
Q	[vector(numeric)] First trajectories.
method	[character] Method used. Can be either 'max' or 'sum'
Fdist	[numeric <- function(numeric,numeric)] Frechet distance between two trajectories use a distance that is use the compute the distance between points of the trajectories. Fdist can be used to define a specific distance. The special value "2D" is used for "euclidean" distance. The special value "1D" can be use for considering that the trajectories are in 1D.

Details

Given two curve P and Q and a distance d, Frechet distance between P and Q is define as $\inf_{a,b} \max_{t} d(P(a(t)), Q(b(t)))$. It's computation is a NP-complex problem. When P and Q are trajectories (discrete curve), the problem is polynomial (and quite simple). The Frechet distance can also be define using a sum instead of a max: $\inf_{a,b} \sum_{t} d(P(a(t)), Q(b(t)))$

The function `distFrechetRec` use the recursive algorithm define by Thomas Eiter and Heikki Mannila. The function `distFrechetR` use a more efficient algorithm in R. The function `distFrechet` use the efficient algorithm in C (and is thus much faster than the two other). Note that `distFrechet` (the fastest) can only use the "2D" and "1D".

Value

A numeric value.

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

Thomas Eiter & Heikki Mannila: "Computing Discrete Fréchet Distance"

[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

distTraj

Examples

```
P <- rnorm(7)
Q <- rnorm(6)

### Function from Eiter and Mannila compiled in C
distFrechet(P,Q)

### Frechet using sum instead of max.
distFrechet(P,Q,method="sum")

### Frechet using "manhattan" distance
distFrechet(P,Q,Fdist=function(x)dist(x,method="manhattan"))
```

distTraj

~ Function: distance for trajectories ~

Description

This function computes and returns the distance computed by using the specified distance measure between two trajectorye.

Usage

```
distTraj(x, y, method = "euclidean", p = 2)
```

Arguments

x	[vector(numeric)]: first trajectories
y	[vector(numeric)]: second trajectories
method	[character]: the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Unambiguous substring can not be given.
p	[numeric]: The power of the Minkowski distance.

Details

This function compute the same distances than the `dist` function but is optimized for trajectories. It can compute only a single distance at a time (whereas `dist` can return a matrix of distances) but on a single couple of trajectories, it is around 5 times faster.

Available distance measures are (written for two vectors x and y):

- 'euclidean': Usual square distance between the two vectors (2 norm).
- 'maximum': Maximum distance between two components of x and y (supremum norm)
- 'manhattan': Absolute distance between the two vectors (1 norm).
- 'canberra': $\sum(|x_i - y_i| / (|x_i| + |y_i|))$. Terms with zero numerator and denominator are omitted from the sum and treated as if the values were missing.
- 'binary': (aka `_asymmetric binary_`): The vectors are regarded as binary bits, so non-zero elements are 'on' and zero elements are 'off'. The distance is the `_proportion_` of bits in which only one is on amongst those in which at least one is on.
- 'minkowski': The p norm, the pth root of the sum of the pth powers of the differences of the components.

Missing values are allowed, and are excluded from all computations involving the column within which they occur. Further, when 'Inf' values are involved, all pairs of values are excluded when their contribution to the distance gave 'NaN' or 'NA'.

If some columns are excluded in calculating a Euclidean, Manhattan, Canberra or Minkowski distance, the sum is scaled up proportionally to the number of columns used (Gower adjustment). If all pairs are excluded when calculating a particular distance, the value is 'NA'.

Value

A numeric value.

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

Brian Everitt, Sabine Landau & Morven Leese : "Cluster Analysis"

[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

[dist](#)

Examples

```
x <- -1+rnorm(25);x[floor(runif(5,1,26))] <- NA
y <- 1+rnorm(25);y[floor(runif(5,1,26))] <- NA

# plot(x,type="b",col=2,ylim=c(-5,5))
# lines(y,type="b",col=3)

system.time(for(i in 1:10000)dist(rbind(x,y)))
system.time(for(i in 1:10000)distTraj(x,y))

system.time(for(i in 1:10000)dist(rbind(x,y),method="maximum"))
system.time(for(i in 1:10000)distTraj(x,y,method="maximum"))

system.time(for(i in 1:10000)dist(rbind(x,y),method="manhattan"))
system.time(for(i in 1:10000)distTraj(x,y,method="manhattan"))
```

expandParLongData ~ Function: expandParLongData ~

Description

Prepare the values of an object [ParLongData](#) to make them being usable by a plotting function.

Usage

```
expandParLongData(xParLongData, y)
```

Arguments

xParLongData [ParLongData]: The object to expand.
 y [Partition] or [numeric]: see detail.

Details

`ParLongData` object can hold values that are easy to specify (like `col="clusters"` or `pch="symbol"`) but that can not be directly used by graphical functions `plot` and `plot3d`. This function modify theses values to make them fit with `plot` and `plot3d` expectations.

The field `col` and `pch` are the ones concern by this function.

If `y` is a `Partition`, `col` and `pch` are extended to fit with the number of individual. If `y` is a number of clusters, `col` and `pch` are extended to fit with the number of clusters.

If `col='clusters'`, a color is affected to each clusters. Then the field `col` receive a vector of color such that each individual (if `y` is a `Partition`) or each clusters (if `y` is a number of clusters) get its corresponding color.

If `pch='letters'`, a letters is affected to each clusters. Then the field `pch` receive a vector of letters such that each individual (if `y` is a `Partition`) or each clusters (if `y` is a number of clusters) get its corresponding letters.

Same if `pch='symbols'`.

Value

An object of class `ParLongData`

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

Examples

```
#####  
### Some parameters for trajectories  
(paramTraj <- parTRAJ(col="clusters"))  
  
### Expand to a small partition with 3 clusters  
part <- partition(LETTERS[rep(1:3,4)])  
expandParLongData(paramTraj,part)
```

```
#####
### Some parameters for the mean trajectories
paramMean <- parMEAN()

### If there is 3 clusters :
expandParLongData(paramMean,3)

### If there is 5 clusters :
expandParLongData(paramMean,5)
```

```
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 [LongData](#).

Usage

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

generateArtificialLongData(nbEachClusters=50,time=0:10,varNames="V",
                           functionClusters=list(function(t){0},function(t){t},function(t){10-t},function(t){-0.4*t^2+4*t}),
                           constantPersonal=function(t){rnorm(1,0,2)},
                           functionNoise=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.

functionClusters [list(function)]: lists the functions defining the average trajectories of each cluster.

constantPersonal [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).

`functionNoise` [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 `functionClusters`.

Even if it belong to a clusters, individual does not perfectly follow the mean trajectories. So a personal variation is added via the argument `constantPersonal`. This personal variation is constant over time.

Then some residual noise is added to all the trajectories via the argument `functionNoise`.

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 `LongData`.

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

[LongData](#), [longData](#), [generateArtificialLongData3d](#)

Examples

```

par(ask=TRUE)

#####
### Default example

(ex1 <- generateArtificialLongData())
plot(ex1)
part1 <- partition(rep(1:4,each=50))
plot(ex1,part1)

#####
### Three diverging lines

ex2 <- generateArtificialLongData(functionClusters=list(function(t)0,function(t)-t,function(t)t))
part2 <- partition(rep(1:3,each=50))
plot(ex2,part2)

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

ex3 <- generateArtificialLongData(
  functionClusters=list(function(t)0,function(t)-t,function(t)t),
  nbEachClusters=c(100,30,10),
  functionNoise=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,part3)

#####
### Four strange functions

ex4 <- generateArtificialLongData(
  nbEachClusters=c(300,200,100,100),
  functionClusters=list(function(t){-10+2*t},function(t){-0.6*t^2+6*t-7.5},function(t){10*sin(t)},function(t){
  functionNoise=function(t){rnorm(1,0,3)},
  time=0:10,decimal=2,percentOfMissing=0.3)
part4 <- partition(rep(1:4,c(300,200,100,100)))
plot(ex4,part4)

#####
### 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)
```

```
generateArtificialLongData3d
```

```
~ Function: generateArtificialLongData3d (or gald3d) ~
```

Description

This function build up an artificial longitudinal data set (joint trajectories) an turn them into an object of class [LongData](#).

Usage

```
gald3d(nbEachClusters=50,time=0:10,varNames=c("V","T"),
      functionClusters=list(function(t){c(0,0)},function(t){c(10,10)},function(t){c(10-t,10-t)}),
      constantPersonal=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
      functionNoise=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
      decimal=2,percentOfMissing=0)
```

```
generateArtificialLongData3d(nbEachClusters=50,time=0:10,varNames=c("V","T"),
                             functionClusters=list(function(t){c(0,0)},function(t){c(10,10)},function(t){c(10-t,10-t)}),
                             constantPersonal=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
                             functionNoise=function(t){c(rnorm(1,0,2),rnorm(1,0,2))},
                             decimal=2,percentOfMissing=0)
```

Arguments

- nbEachClusters** [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** [vector(character)]: names of the variables.
- functionClusters** [list(function)]: lists the functions that define the average trajectories of each cluster. Each functions shall return a vector containing one value for each variable of **varNames**.
- constantPersonal** [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).
- functionNoise** [function] or [list(function)]: lists the functions generating the noise of each trajectory within its own cluster. Each functions shall return a vector containing one value for each variable of **varNames**. If a single function is given, it is duplicated for all groups.

`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

`generateArtificialLongData3d` (`gald3d` in short) is a function that construct a set of artificial joint 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 `functionClusters`.

Even if it belong to a clusers, individual does not perfectly follow the mean trajectories. So a personal variation is added via the argument `constantPersonal`. This personal variation is constant over time.

Then some residual noise is added to all the trajectories via the argument `functionNoise`.

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

Object of class [LongData](#).

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

[LongData3d](#), [longData3d](#), [generateArtificialLongData](#)

Examples

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

ex1 <- generateArtificialLongData3d()
plot3d(ex1)
part1 <- partition(rep(1:3,each=50))
plot3d(ex1,part1)

#####
### 4 lines with unbalanced groups

ex2 <- generateArtificialLongData3d(
  nbEachClusters=c(5,10,20,40),
  functionClusters=list(
    function(t)c(t,t^3/100),
    function(t)c(0,t),
    function(t)c(t,t),
    function(t)c(0,t^3/100)
  ),
  functionNoise = function(t){c(rnorm(1,0,1),rnorm(1,0,1))}
)
plot3d(ex2)
part2 <- partition(rep(1:4,time=c(5,10,20,40)))
plot3d(ex2,part2)
```

imputation

~ Function: imputation ~

Description

imputation is a function that offer different methods to impute missing value of a [LongData](#) (or a matrix).

Usage

```
imputation(traj,method="copyMean",lowerBound="min",upperBound="max")
```

Arguments

traj	[LongData] or [matrix]: trajectories to impute.
method	[character]: Name of the imputation method (see detail)
lowerBound	[character] or [numeric]: fixes the smallest value that an imputed value can take. If a single value is given, it is duplicate for all the column. The special value 'min' means that the lower bound will be the smallest value of the column. The special value 'globalMin' means that the lower bound will be the overall smallest value (of each variable if there is several variable-trajectories). The special value 'NA' can be used to impute without using a lower bound.

`upperBound` [character] or [numeric]: fixes the biggest value that an imputed value can take. If a single value is given, it is duplicate for all the column. The special value 'max' means that the upper bound will be the biggest value of the column. The special value 'globalMax' means that the upper bound will be the overall biggest value (of each variable if there is several variable-trajectories). The special value 'NA' can be used to impute without using an upper bound.

Details

`imputation` is a function that impute missing value of a `LongData` or a matrix. Several imputation methods are available. A brief description follows. For a fully detailed description, see [gen12] (submitted). Illustrating examples showing strengths and weakness of methods are presented section "examples".

For each method, the imputation has to deal with monotone missing value (at start and at end of the trajectories) and intermitant (in the middle). Here is a brief description of each methods.

- 'linearInterpol.locf' (linear interpolation, locf)
 - Intermitant: values immediatly surrounding the missing are join by a line.
 - Monotone: imputed by 'locf' or 'nocb'.
- 'linearInterpol.global' (linear interpolation, global slope)
 - Intermitant: values immediatly surrounding the missing are join by a line.
 - Monotone: the line joining the first and last non-missing value is considered (this line is the everage progression of the actual individual trajectoire). Missing-value at start and at end are chosen on this line.
- 'linearInterpol.local' (linear interpolation, global slope)
 - Intermitant: values immediatly surrounding the missing are join by a line.
 - Monotone at start: the line joining the first and second non-missing value is considered. Missing-value at start are chose on this line.
 - Monotone at end: the line joining the last and penultimate non-missing value is considered. Missing-value at end are chosen on this line.
- 'linearInterpol.bisector' (linear interpolation, bisector)
 - Intermitant: values immediatly surrounding the missing are join by a line.
 - Monotone: `linearInterpol.global` is not sensitive to local variation, `linearInterpol.local` might be too much sensitive to abnormal value. `linearInterpol.bisector` offer a medium solution by considering the bissectrice of Global and Local solution. Point are chosen on the bissectrices.
- 'copyMean.locf' (copy mean, locf) this method impute in two stages. First, it use 'linearInterpol.locf'. Then it add to each imputed value a variation that make the imputed value follow the shape of the average trajectory. For more details, see [gen12] and examples' section.
- 'copyMean.global' (copy mean, global slope) this method impute in two stages. First, it use 'linearInterpol.global'. Then it add to each imputed value a variation that make the imputed value follow the shape of the average trajectory. For more details, see [gen12] and examples' section.

- 'copyMean.local' (copy mean, local slope) this method impute in two stages. First, it use 'linearInterpol.local'. Then it add to each imputed value a variation that make the imputed value follow the shape of the average trajectory. For more details, see [gen12] and examples' section.
- 'copyMean.bisector' (copy mean, bisector) this method impute in two stages. First, it use 'linearInterpol.bisector'. Then it add to each imputed value a variation that make the imputed value follow the shape of the average trajectory. For more details, see [gen12] and examples' section.
- lof (Last Occurence Carried Forward)
 - Intermitant and monotone at end: the previous non-missing value is dipplicated forward.
 - Monotone at start: the first non-missing value is duplicated backward (nocb).

THIS METHOD HAS BEEN PROUVEN TO NOT BE EFFICIENT SEVERAL TIME BY VARIOUS AUTHOR, we strongly recommand to not use it !
- nocb (Next Occurence Carried Backward)
 - Intermitant and monotone at start: the next non-missing value is dipplicated backward.
 - Monotone at end: the last non-missing value is duplicated forward (lof).

THIS METHOD HAS BEEN PROUVEN TO NOT BE EFFICIENT SEVERAL TIME BY VARIOUS AUTHOR, we strongly recommand to not use it !
- trajMean missing are imputed by the mean of the trajectory.
- trajMedian missing are imputed by the median of the trajectory.
- trajHotDeck each missing is imputed by one non-missing (randomly choosen) value of the trajectory.
- crossMean missing value at time t are imputed by the mean of all value present at time t.
- crossMedian missing value at time t are imputed by the median of all value present at time t.
- crossHotDeck each missing value at time t is imputed by one non-missing (randomly choosen) value present at time t.

Value

A [LongData](#) or a matrix with no missing values.

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


```

matlines(t(mat2), type="o", col=2, lwd=3, pch=16, lty=1)

### crossHotDeck
matplot(t(imputation(mat2, "crossHotDeck")), type="l", ylim=c(0, 10), lty=1, col=1, main="crossHotDeck")
matlines(t(mat2), type="o", col=2, lwd=3, pch=16, lty=1)

#####
### Methods using trajectory information (traj-methods)

par(mfrow=c(2, 3))
mat1 <- matrix(c(NA, NA, 3, 8, NA, NA, 2, 2, 1, NA, NA), 1, 11)

### locf
matplot(t(imputation(mat1, "locf")), type="l", ylim=c(0, 10), main="locf\n DO NOT USE, BAD METHOD !!!")
matlines(t(mat1), type="o", col=2, lwd=3, pch=16)

### nocb
matplot(t(imputation(mat1, "nocb")), type="l", ylim=c(0, 10), main="nocb\n DO NOT USE, BAD METHOD !!!")
matlines(t(mat1), type="o", col=2, lwd=3, pch=16)

### trajMean
matplot(t(imputation(mat1, "trajMean")), type="l", ylim=c(0, 10), main="trajMean")
matlines(t(mat1), type="o", col=2, lwd=3, pch=16)

### trajMedian
matplot(t(imputation(mat1, "trajMedian")), type="l", ylim=c(0, 10), main="trajMedian")
matlines(t(mat1), type="o", col=2, lwd=3, pch=16)

### trajHotDeck
matplot(t(imputation(mat1, "trajHotDeck")), type="l", ylim=c(0, 10), main="trajHotDeck 1")
matlines(t(mat1), type="o", col=2, lwd=3, pch=16)

### spline
matplot(t(imputation(mat1, "spline", lowerBound=NA, upperBound=NA)), type="l", ylim=c(-10, 10), main="spline")
matlines(t(mat1), type="o", col=2, lwd=3, pch=16)

#####
### Different linear interpolation

par(mfrow=c(2, 2))

### linearInterpol.locf
matplot(t(imputation(mat1, "linearInterpol.locf", NA, NA)), type="l", ylim=c(-5, 10), lty=1, col=1, main="linearInterpo")
matlines(t(mat1), type="o", col=2, lwd=3, pch=16, lty=1)

### linearInterpol.global
matplot(t(imputation(mat1, "linearInterpol.global", NA, NA)), type="l", ylim=c(-5, 10), lty=1, col=1, main="linearInterpo")

```

```

matlines(t(mat1),type="o",col=2,lwd=3,pch=16,lty=1)

### linearInterpol.local
matplot(t(imputation(mat1,"linearInterpol.local",NA,NA)),type="l",ylim=c(-5,10),lty=1,col=1,main="linearInterp
matlines(t(mat1),type="o",col=2,lwd=3,pch=16,lty=1)

### linearInterpol.bisector
matplot(t(imputation(mat1,"linearInterpol.bisector",NA,NA)),type="l",ylim=c(-5,10),lty=1,col=1,main="linearInt
matlines(t(mat1),type="o",col=2,lwd=3,pch=16,lty=1)

#####
### Copy mean

mat3 <- matrix(c(
  NA, 9, 8, 8, 7, 6,NA,
  7, 6,NA,NA,NA, 4,5,
  3, 4, 3,NA,NA, 2,3,
  NA,NA, 1,NA,NA, 1,1),4,7,byrow=TRUE)

par(mfrow=c(2,2))

### copyMean.locf
matplot(t(imputation(mat2,"copyMean.locf",NA,NA)),type="l",ylim=c(-5,10),lty=1,col=1,main="copyMean.locf")
matlines(t(mat2),type="o",col=2,lwd=3,pch=16,lty=1)

### copyMean.global
matplot(t(imputation(mat2,"copyMean.global",NA,NA)),type="l",ylim=c(-5,10),lty=1,col=1,main="copyMean.global")
matlines(t(mat2),type="o",col=2,lwd=3,pch=16,lty=1)

### copyMean.local
matplot(t(imputation(mat2,"copyMean.local",NA,NA)),type="l",ylim=c(-5,10),lty=1,col=1,main="copyMean.local")
matlines(t(mat2),type="o",col=2,lwd=3,pch=16,lty=1)

### copyMean.bisector
matplot(t(imputation(mat2,"copyMean.bisector",NA,NA)),type="l",ylim=c(-5,10),lty=1,col=1,main="copyMean.bisect
matlines(t(mat2),type="o",col=2,lwd=3,pch=16,lty=1)

### crossMean
matImp <- imputation(matMissing,method="crossMean")
matplot(t(matImp),col=c(2,1,1,1),lty=c(2,1,1,1),type="l",lwd=c(2,1,1,1),pch=16, xlab="Dotted red=imputed trajec
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

### crossMedian
matImp <- imputation(matMissing,method="crossMedian")
matplot(t(matImp),col=c(2,1,1,1),lty=c(2,1,1,1),type="l",lwd=c(2,1,1,1),pch=16, xlab="Dotted red=imputed trajec
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

```

```

### crossHotDeck
matImp <- imputation(matMissing,method="crossHotDeck")
matplot(t(matImp),col=c(2,1,1,1),lty=c(2,1,1,1),type="l",lwd=c(2,1,1,1),pch=16, xlab="Dotted red=imputed trajec
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

#####
### Method using trajectory

par(mfrow=c(2,3))
### trajMean
matImp <- imputation(matMissing,method="trajMean")
plot(timeV,matImp[1,],type="l",lwd=2,ylim=c(10,30),ylab="",xlab="nocb")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

### trajMedian
matImp <- imputation(matMissing,method="trajMedian")
plot(timeV,matImp[1,],type="l",lwd=2,ylim=c(10,30),ylab="",xlab="nocb")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

### trajHotDeck
matImp <- imputation(matMissing,method="trajHotDeck")
plot(timeV,matImp[1,],type="l",lwd=2,ylim=c(10,30),ylab="",xlab="nocb")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

### locf
matImp <- imputation(matMissing,method="locf")
plot(timeV,matImp[1,],type="l",lwd=2,ylim=c(10,30),ylab="",xlab="locf")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

### nocb
matImp <- imputation(matMissing,method="nocb")
plot(timeV,matImp[1,],type="l",lwd=2,ylim=c(10,30),ylab="",xlab="nocb")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

par(mfrow=c(2,2))

### linearInterpol.locf
matImp <- imputation(matMissing,method="linearInterpol.locf")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

### linearInterpol.local
matImp <- imputation(matMissing,method="linearInterpol.local")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

### linearInterpol.global
matImp <- imputation(matMissing,method="linearInterpol.global")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

```

```

### linearInterpol.bisector
matImp <- imputation(matMissing,method="linearInterpol.bisector")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)

par(mfrow=c(2,2))

### copyMean.locf
matImp <- imputation(matMissing,method="copyMean.locf")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)
lines(timeV,moy,col=3,type="o",lwd=3)

### copyMean.local
matImp <- imputation(matMissing,method="copyMean.local")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)
lines(timeV,moy,col=3,type="o",lwd=3)

### copyMean.global
matImp <- imputation(matMissing,method="copyMean.global")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)
lines(timeV,moy,col=3,type="o",lwd=3)

### copyMean.bisector
matImp <- imputation(matMissing,method="copyMean.bisector")
plot(timeV,matImp[1,],type="o",ylim=c(0,30),ylab="",xlab="LI-Global")
lines(timeV,matMissing[1,],col=2,type="o",lwd=3)
lines(timeV,moy,col=3,type="o",lwd=3)

par(ask=FALSE)

```

initializePartition ~ *Function: initializePartition* ~

Description

This function provide different way of setting the initial partition for an EM algoritim.

Usage

```
initializePartition(nbClusters, lengthPart, method = "kmeans++", data)
```

Arguments

nbClusters	[numeric]: number of clusters of that the initial partition should have.
lengthPart	[numeric]: number of individual in the partition.

method	[character]: one off "randomAll", "randomK", "maxDist", "kmeans++", "kmeans+", "kmeans-" or "kmeans-".
data	[matrix]: data is the matrix of the individuals (usefull for the methods that need to compute distance between individual). If data is an array, the distance is computed using "maxDist" is used, the function needs to know the matrix of the distance between each individual.

Details

Before alternating the phase Esperance and Maximisation, the EM algorithm needs to initialize a starting configuration. This initial partition has been proven to have an important impact on the final result and the convergence time.

This function provides different ways of setting the initial partition.

- randomAll: all the individual are randomly assigned to a cluster with at least one individual in each clusters.
- randomK: K individuals are randomly assigned to a cluster, all the other are not assigned (each cluster has only one individual).
- maxDist: K individuals are chosen. The two formers are the individual separated by the highest distance. The latter are added one by one, they are the "farthest" individual among those that are already been selected. "farthest" is the individual with the highest distance (min) to the selected individuals (if "t" are the individual already selected, the next selected individual is "i" such that $\max_i(\min_t(\text{dist}(\text{IND}_i, \text{IND}_t)))$). This method is efficient but time consuming.
- kmeans++: see [3]
- kmeans+, kmeans-, kmeans-: experimental methods derived from [3].

Value

vecteur of numeric.

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

[3] D. Arthur and S. Vassilvitskii
 "k-means++: the advantages of careful seeding"
 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. pp. 1027-1035, 2007.

Examples

```

par(ask=TRUE)
#####
### Construction of some longitudinal data
dn <- gald()
plot(dn)

#####
### partition using randomAll
pa1a <- initializePartition(3,lengthPart=200,method="randomAll")
plot(dn,partition(pa1a),parMean=parMEAN(type="n"),parTraj=parTRAJ(col="clusters"))
pa1b <- initializePartition(3,lengthPart=200,method="randomAll")
plot(dn,partition(pa1b),parMean=parMEAN(type="n"),parTraj=parTRAJ(col="clusters"))

#####
### partition using randomK
pa2a <- initializePartition(3,lengthPart=200,method="randomK")
plot(dn,partition(pa2a),parMean=parMEAN(type="n"),parTraj=parTRAJ(col="clusters"))
pa2b <- initializePartition(3,lengthPart=200,method="randomK")
plot(dn,partition(pa2b),parMean=parMEAN(type="n"),parTraj=parTRAJ(col="clusters"))

#####
### partition using maxDist
pa3 <- initializePartition(3,lengthPart=200,method="maxDist",data=dn["traj"])
plot(dn,partition(pa3),parMean=parMEAN(type="n"),parTraj=parTRAJ(col="clusters"))
### maxDist is deterministic, so no need for a second example

#####
### Example to illustrate "maxDist" method on classical clusters
point <- matrix(c(0,0, 0,1, -1,0, 0,-1, 1,0),5,byrow=TRUE)
points <- rbind(point,t(t(point)+c(10,0)),t(t(point)+c(5,6)))
points <- rbind(points,t(t(points)+c(30,0)),t(t(points)+c(15,20)),t(-t(point)+c(20,10)))
plot(points,main="Some points")

paInit <- initializePartition(2,nrow(points),method="maxDist",points)
plot(points,main="Two farest points")
lines(points[!is.na(paInit),],col=2,type="p",pch=16)

paInit <- initializePartition(3,nrow(points),method="maxDist",points)
plot(points,main="Three farest points")
lines(points[!is.na(paInit),],col=2,type="p",pch=16)

paInit <- initializePartition(4,nrow(points),method="maxDist",points)
plot(points, main="Four farest points")
lines(points[!is.na(paInit),],col=2,type="p",pch=16)

```

```
par(ask=FALSE)
```

ListPartition-class ~ Class: ListPartition ~

Description

An object of class ListPartition contain several liste of Partition sorted by cluster numbers.

Objects from the Class

Objects are mainly design to store the numerous Partition found by kml or kml3d.

Slots

`criterionActif` [character]: Store the criterion name that will be used by fonctions that need a single criterion (like [plotCriterion](#) or [ordered](#)).

`initializationMethod` [vector(character)]: list all the initialization method that has allready been used to find some Partition (usefull to not run several time a deterministic method).

`sorted` [logical]: are the Partition curently hold in the object sorted in decreasing (or increasing, according to `criterionActif`) order ?

`c1` [list(Partition)]: list of Partition with 1 clusters.

`c2` [list(Partition)]: list of Partition with 2 clusters.

`c3` [list(Partition)]: list of Partition with 3 clusters.

`c4` [list(Partition)]: list of Partition with 4 clusters.

`c5` [list(Partition)]: list of Partition with 5 clusters.

`c6` [list(Partition)]: list of Partition with 6 clusters.

`c7` [list(Partition)]: list of Partition with 7 clusters.

`c8` [list(Partition)]: list of Partition with 8 clusters.

`c9` [list(Partition)]: list of Partition with 9 clusters.

`c10` [list(Partition)]: list of Partition with 10 clusters.

`c11` [list(Partition)]: list of Partition with 11 clusters.

`c12` [list(Partition)]: list of Partition with 12 clusters.

`c13` [list(Partition)]: list of Partition with 13 clusters.

`c14` [list(Partition)]: list of Partition with 14 clusters.

`c15` [list(Partition)]: list of Partition with 15 clusters.

`c16` [list(Partition)]: list of Partition with 16 clusters.

`c17` [list(Partition)]: list of Partition with 17 clusters.

`c18` [list(Partition)]: list of Partition with 18 clusters.

`c19` [list(Partition)]: list of Partition with 19 clusters.

- c20 [list(Partition)]: list of Partition with 20 clusters.
- c21 [list(Partition)]: list of Partition with 21 clusters.
- c22 [list(Partition)]: list of Partition with 22 clusters.
- c23 [list(Partition)]: list of Partition with 23 clusters.
- c24 [list(Partition)]: list of Partition with 24 clusters.
- c25 [list(Partition)]: list of Partition with 25 clusters.
- c26 [list(Partition)]: list of Partition with 26 clusters.

Construction

Class ListPartition objects are mainly constructed by `km1`. Nevertheless, it is also possible to construct them from scratch using the fonction `listPartition` that does create an empty object.

Methods

- `object['xxx']` If 'xxx' is 'cX', 'initializationMethod', 'sorted' or 'criterionActif', get the value of the field xxx.
- `object['criterionValues', j]` Give the values of the criterion 'j' for all the Partitions. The result is return as a list. If 'j' is missing, the criterion actif is used.
- `object['criterionValuesAsMatrix', j]` Give the values of the criterion 'j' for all the Partitions. The result is return as a matrix. If 'j' is missing, the criterion actif is used.
- `object['xxx']` If 'xxx' is a criterion, this is equivalent to `object['criterionValuesAsMatrix', 'xxx']`
- `object['initializationMethod']<-value` Set the field to value
- `object['criterionActif']<-value` If 'value' is one of CRITERION_NAMES, it sets the field to the criterion 'value'.
- `object['add']<-value` If 'value' is an object of class 'Partition', then value is added to the Partition already hold in the field 'cX'. Note that a Partition with 'X' clusters is automatically added to the correct list 'cX' according to its number of clusters.
- `object['clear']<- 'cX'` Clear the list 'cX'.
- `listPartition` Constructor. Build an empty object.
- `ordered` Order the Partition according to the criterion actif.
- `regroup` Order then merge identical Partition (usefull to reduce the size of the ListPartition)

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] Christophe M. Genolini and Bruno Falissard
"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010

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

See Also

Classes: [LongData](#)
Methods: [Partition](#)

Examples

```
#####  
### Preparing data  
data <- gald(30)["traj"]  
  
### Some clustering  
part2 <- partition(rep(c("A", "B", "A"), time=40), data)  
part3 <- partition(rep(c("A", "B", "C"), time=40), data)  
part3b <- partition(rep(c("A", "B", "C", "B", "C"), time=24), data)  
part4 <- partition(rep(c("A", "B", "A", "C", "D"), time=24), data)  
  
#####  
### ListPartition  
listPart <- listPartition()  
listPart[ 'criterionActif' ] <- "Davies.Bouldin"  
plotCriterion(listPart)  
  
listPart["add"] <- part2  
listPart["add"] <- part3  
listPart["add"] <- part3b  
listPart["add"] <- part4  
listPart["add"] <- part4  
listPart["add"] <- part3  
listPart["add"] <- part3b  
  
plotCriterion(listPart)  
ordered(listPart)  
plotCriterion(listPart)  
regroup(listPart)  
plotCriterion(listPart)
```

longData ~ Function: longData ~

Description

longData is a constructor for the class [LongData](#). It create object [LongData](#) containing a single variable-trajectory. For creating joint variable-trajectories, see [longData3d](#).

Usage

```
longData(traj, idAll, time, timeInData,varNames,maxNA)
```

Arguments

traj	[matrix(numeric)], [array(numeric)] or [data.frame]: structure containing the trajectories.
idAll	[vector(character)]: single identifier for each trajectory (ie each individual).
time	[vector(numeric)]: time at which measures were made.
timeInData	[list(vector(numeric))]: precise the column containing the trajectories.
varNames	[character]: name of the variable-trajectory 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

longData construct a object of class [LongData](#). 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 [LongData](#).

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

[LongData](#)

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")))
longData(mat)
(ld1 <- longData(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 <- longData(traj=mat,idAll=paste("I-",1:1051,sep=""),time=(1:325)+0.5,varNames="Random"))

#####
### 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
longData(dn)

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

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

Description

LongData is an objet containing the longitudinal data (the individual trajectories) and some associate value (like time, individual identifiant,...). It can be used either for a single variable-trajectory or for joint variable-trajectories.

Objects from the Class

Object LongData for single variable-trajectory can be created using the fonction `longData` on a `data.frame` or on a `matrix`.

LongData for joint trajectories can be created by calling the fonction `longData3d` on a `data.frame` or on an array.

Slots

`idAll` [vector(character)]: Single identifier for each of the longData (each individual). Usefull to export clusters.

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

`time` [numeric]: Time at which measures are made.

`varNames` [character]: Name of the variable measured.

`traj` [matrix(numeric)]: Contains the longitudianl data. Each lines is the trajectories of an individual. Each column is the time at which measures are made.

`dimTraj` [vector3(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 value are exclude from `traj` and will no be use in the analysis. Their identifier is preserved in `idAll` but not in `idFewNA`. 'too many' is define by `maxNA`: a trajectory with more missing than `maxNA` is exclude.

`reverse` [matrix(numeric)]: if the trajectories are scale using the fonction `scale`, the 'scaling parameters' (probably mean and standard deviation) are saved in reverse. This is usefull to restaure the original data after a scaling operation.

Construction

Object LongData for single variable-trajectory can be created by calling the fonction `longData` on a `data.frame` or on a `matrix`.

LongData for joint trajectories can be created by calling the fonction `longData3d` on a `data.frame` or on an array.

Get [

Object["idAll"] [vecteur(character)]: Gets the full list of individual identifiant (the value of the slot `idAll`)

Object["idFewNA"] [vecteur(character)]: Gets the list of individual identifiant with not too many missing values (the value of the slot `idFewNA`)

Object["varNames"] [character]: Gets the name(s) of the variable (the value of the slot `varNames`)

Object["time"] [vecteur(numeric)]: Gets the times (the value of the slot time)

Object["traj"] [array(numeric)]: Gets all the longData' values (the value of the slot traj)

Object["dimTraj"] [vector3(numeric)]: Gets the dimension of traj.

Object["nbIdFewNA"] [numeric]: Gets the first dimension of traj (ie the number of individual include in the analysis).

Object["nbTime"] [numeric]: Gets the second dimension of traj (ie the number of time measurement).

Object["nbVar"] [numeric]: Gets the third dimension of traj (ie the number of variables).

Object["maxNA"] [vecteur(numeric)]: Gets maxNA.

Object["reverse"] [matrix(numeric)]: Gets the matrix of the scaling parameters.

Methods

scale scale the trajectories. Usefull to normalize variable trajectories measured with different units.

restaureRealData restaure original data that have been modified after a scaling operation.

generateArtificialLongData (or gald) Generate an artifiial dataset of a single variable-trajectory.

generateArtificialLongData3d (or gald3d) Generate a artifiial dataset of some joint variable-trajectory.

longDataFrom3d Extract a variable trajectory form a dataset of joint trajectories.

plot plot all the variable of the LongData, optionnaly according to a **Partition**.

plot3d plot tow variable of the LongData in 3 dimensions, optionnaly according to a **Partition**.

plot3dPdf create 'Triangle objects' representing in 3D the cluster's center according to a **Partition**.
'Triangle object' can latter be include in a LaTeX file to get a dynamique (rotationg) pdf figure.

imputation Impute the missing values of the trajectories.

qualityCriterion Compute some quality criterion that can be use to compare the quality of dif-ferents **Partition**.

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: [longitudinalData-package](#)
 Methods: [longData](#), [longData3d](#), [imputation](#), [qualityCriterion](#)
 Plot: [plot](#), [plot3d](#), [plot3dPdf](#)

Examples

```
#####
### building trajectory (longData)
mat <- matrix(c(NA,2,3,4,1,6,2,5,1,3,8,10),4)
ld <- longData(mat,idAll=c("I1","I2","I3","I4"),time=c(2,4,8),varNames="Age")

### '[' and '[<-'
ld["idAll"]
ld["idFewNA"]
ld["varNames"]
ld["traj"]
(ld)

### Plot
plot(ld)

#####
### building join trajectories (longData3d)
dn <- data.frame(id=1:3,v1=c(11,14,16),t1=c(1,5,7),v2=c(12,10,13),t2=c(2,5,0),t3=c(3,6,8))
(ld <- longData3d(dn,timeInData=list(Vir=c(2,4,NA),Tes=c(3,5,6))))

### Scaling
scale(ld)
(ld)

### Plotting
plot(ld)
plot3d(ld)
restaureRealData(ld)
```

longData3d

~ Function: longData3d ~

Description

longData3d is a constructor of the class [LongData](#). It create object [LongData](#) containing several joint trajectory (two or more variable-trajectories). For creating a single variable-trajectory, see [longData](#).

Usage

```
longData3d(traj, idAll, time, timeInData,varNames,maxNA)
```

Arguments

traj	[array(numeric)] or data.frame: structure containing the variable-trajectories.
idAll	[vector(character)]: single identifier for each trajectory (ie each individual).
time	[vector(numeric)]: time at which measures were made.
timeInData	[list(vector(numeric))]: Precise the column containing the trajectories. If traj is a data.frame, it could be a list.
varNames	[character]: name of the variable-trajectories being measured.
maxNA	[vector(numeric)]: maximum number of NA that are tolerates on a trajectory (one for each variable). If a trajectory has more missing than maxNA, then it is remove from the analysis.

Details

longData3d construct a object of class [LongData](#). Two cases can be distinguished:

traj is an array: the first dimension (line) are individual. The second dimension (column) are time at which the measurement are made. The third dimension are the differents variable-trajectories. For example, `traj[, , 2]` is the second variable-trajectory.

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

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

If traj is a data.frame: lines are individual. Time of measurement and variables should be provide through `timeInData`. `timeInData` is a list. The label of the list are the variable-trajectories names. Elements of the list are the column containning the trajectories. For example, if `timeInData=list(V=c(2,3,4),W=c(6,8,12))`, then the first variable-trajectory is 'V', its measurement are in column 2,3 and 4. The second variable-trajectory is 'W', its measurement are in column 6,8 and 12.

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

Value

An object of class [LongData](#).

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

[LongData](#)

Examples

```
#####
### From array

### Small data
mat <- array(c(1,NA,3,2,3,6,1,8,10,1,NA,1,2,NA,3,2,3,2),dim=c(3,3,2))
longData3d(mat)
(ld1 <- longData3d(mat,varNames=c("Hyp","Col"),idAll=c("i101","i104","i105")))
plot3d(ld1)

### Big data
mat <- array(c(runif(1051*325),rnorm(1051*325),rexp(1051*325)),c(1051,325,3))
(ld2 <- longData3d(traj=mat,time=(1:325)+0.5,varNames=c("unif","norm","rexp")))
plot3d(ld2,nbSample=200)

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

dn <- data.frame(id=1:3,v1=c(2,2,1),t1=c(20,21,22),v1=c(3,2,2),t2=c(23,20,28),t3=c(25,24,29))
longData3d(dn,timeInData=list(c(2,4),c(3,5)),varNames=c("V","T"))
(ld3 <- longData3d(dn,timeInData=list(V=c(2,4,NA),T=c(3,5,6))))
plot3d(ld3)
```

LongData3d-class ~ Class: LongData3d ~

Description

LongData3d is an objet containing joint longitudinal data and some associate value (like time, individual identifiant,...).

Objects from the Class

Object LongData3d can be created using the fonction [longData3d](#) on a data.frame or on an array.

Slots

idAll [vector(character)]: Single identifier for each of the longData3d (each individual). Useful to export clusters.

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

time [numeric]: Time at which measures are made.
varNames [vector(character)]: Names of the variable measured.
traj [array(numeric)]: Contains the joint variable-trajectories. Each horizontal plan (first dimension) corresponds to the joint-trajectories of an individual. Vertical plans (second dimension) refer to the time at which measures are made. Transversal plans (the third dimension) are for variables.
dimTraj [vector3(numeric)]: size of the array traj (ie $\text{dimTraj} = \text{c}(\text{length}(\text{idFewNA}), \text{length}(\text{time}), \text{length}(\text{varNames}))$).
maxNA [numeric] or [vector(numeric)]: Individual whose trajectories contain 'too many' missing value are exclude from traj and will no be use in the analysis. Their identifier is preserved in `idAll` but not in `idFewNA`. 'too many' is define by `maxNA`: a trajectory with more missing than `maxNA` is exclude. When `maxNA` is a single number, it is recycled for all the variables.
reverse [matrix(numeric)]: if the trajectories are scale using the function [scale](#), the 'scaling parameters' (probably mean and standard deviation) are saved in `reverse`. This is usefull to restaure the original data after a scaling operation.

Construction

LongData3d can be created by calling the fonction [longData3d](#) on a data.frame or on an array.

Get [

Object["idAll"] [vecteur(character)]: Gets the full list of individual identifiant (the value of the slot `idAll`)
Object["idFewNA"] [vecteur(character)]: Gets the list of individual identifiant with not too many missing values (the value of the slot `idFewNA`)
Object["varNames"] [character]: Gets the name(s) of the variable (the value of the slot `varNames`)
Object["time"] [vecteur(numeric)]: Gets the times (the value of the slot `time`)
Object["traj"] [array(numeric)]: Gets all the joint trajectories (the value of the slot `traj`)
Object["dimTraj"] [vector3(numeric)]: Gets the dimension of traj.
Object["nbIdFewNA"] [numeric]: Gets the first dimension of traj (ie the number of individual include in the analysis).
Object["nbTime"] [numeric]: Gets the second dimension of traj (ie the number of time measurement).
Object["nbVar"] [numeric]: Gets the third dimension of traj (ie the number of variables).
Object["maxNA"] [vecteur(numeric)]: Gets `maxNA`.
Object["reverse"] [matrix(numeric)]: Gets the matrix of the scaling parameters.

Methods

[scale](#) scale the trajectories. Usefull to normalize variable trajectories measured with different units.
[restaureRealData](#) restaure original data that have been modified after a scaling operation.
[generateArtificialLongData3d \(or `galD3d`\)](#) Generate a artifiial dataset of some joint variable-trajectory.

[longDataFrom3d](#) Create a [LongData](#) by extracting a single variable trajectory form a dataset of joint variable-trajectories.

[plot](#) plot all the variable of the LongData3d, optionnaly according to a [Partition](#).

[plot3d](#) plot tow variables of the LongData3d in a 3 dimensions graph, optionnaly according to a [Partition](#).

[plot3dPdf](#) create 'Triangle objects' representing in 3D the cluster's center according to a [Partition](#).
'Triangle object' can latter be include in a LaTeX file to get a dynamique (rotationg) pdf figure.

[imputation](#) Impute the missing values of the trajectories.

[qualityCriterion](#) Compute some quality criterion that can be use to compare the quality of dif-ferents [Partition](#).

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: [longitudinalData-package](#)

Methods: [LongData](#), [longData3d](#), [imputation](#), [qualityCriterion](#)

Plot: [plot](#), [plot3d](#), [plot3dPdf](#)

Examples

```
#####
### building joint trajectories

dn <- data.frame(id=1:3,v1=c(11,14,16),t1=c(1,5,7),v2=c(12,10,13),t2=c(2,5,0),t3=c(3,6,8))
(ld <- longData3d(dn,timeInData=list(Vir=c(2,4,NA),Tes=c(3,5,6))))

### Scaling
scale(ld)
(ld)

### Plotting
plot(ld)
```

```
plot3d(ld)  
restaureRealData(ld)
```

longDataFrom3d ~ Function: longDataFrom3d ~

Description

Extract a single variable-trajectory from an object [LongData](#) that contain some joint-trajectories.

Usage

```
longDataFrom3d(xLongData3d, variable)
```

Arguments

xLongData3d	[LongData3d]: structure containing some joint-trajectories.
variable	[character]: either the name of one of the variable of xLongData3d, or its number.

Details

Extract a single variable-trajectory from an object [LongData3d](#) that contain some joint-trajectories.

Value

An object of class [LongData](#).

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[LongData](#)**Examples**

```
### Creation of joint-trajectories
mat <- array(c(1,NA,3,2,3,6,1,8,10,1,NA,1,2,NA,3,2,3,2),dim=c(3,3,2))
(ldJoint <- longData3d(mat,varNames=c("Hyp","Som")))

### Extraction of the first variable-trajectory
(ldHyp <- longDataFrom3d(ldJoint,variable="Hyp"))

### Extraction of the second variable-trajectory
(ldSom <- longDataFrom3d(ldJoint,variable="Som"))

### Extraction of the second variable-trajectory, using number
(ldSom <- longDataFrom3d(ldJoint,variable=2))
```

longDataTo3d ~ Function: longDataTo3d ~

Description

Build a object [LongData3d](#) from an object [LongData](#). The resulting object has a single variable-trajectory stored in a array.

Usage

```
longDataTo3d(xLongData)
```

Arguments

xLongData [[LongData](#)]: structure containing a variable-trajectory.

Details

Build a object [LongData3d](#) from an object [LongData](#). The resulting object has a single variable-trajectory stored in a array.

Value

An object of class [LongData3d](#).

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

[LongData](#)

Examples

```
### Creation of single variable-trajectory
mat <- matrix(c(1,NA,3,2,3,6,1,8,10,1,NA,1,2,NA,3,2,3,2),6,3)
(ldSingle <- longData(mat))

### Extension to joint trajectories
(ldHyp <- longDataTo3d(ldSingle))
```

makeLatexFile ~ *Function: makeLatexFile* ~

Description

Create a LaTeX document that include 3D objects into PDF documents.

Usage

```
makeLatexFile(filename = "main.tex", asyToInclude = "scene+0.prc")
```

Arguments

filename	Name of the LaTeX file
asyToInclude	Name of the file holding the 3D graph to include.

Details

Create a LaTeX document that include 3D objects into PDF documents with PDF-1.5/1.6 compatibility.

Value

A LaTeX file, in the current directory.

Author(s)

Christophe Genolini
PSIGIAM: Paris Sud Innovation Group in Adolescent Mental Health
INSERM U669 / Maison de Solenn / Paris

Contact author : <genolini@u-paris10.fr>

English translation

Raphaël Ricaud
Laboratoire "Sport & Culture" / "Sports & Culture" Laboratory
University of Paris 10 / Nanterre

See Also

[makeTriangles](#), [plot3dPdf](#), [saveTrianglesAsASY](#).

Examples

```
### Generating the data
myLd <- gal3d()
part <- partition(rep(1:3,each=50))
plot3d(myLd,part)

### Creation of the scene
scene <- plot3dPdf(myLd,part)
drawScene.rgl(scene)

### Export in '.azy' file
saveTrianglesAsASY(scene)

### Creation of a '.prc' file
# Open a console, then run:
# 'asy -inlineimage -tex pdflatex scene.azy'

### Creation of the LaTeX main document
makeLatexFile()

### Creation of the '.pdf'
# Open a console window, then run
# pdfLatex main.tex
```

ordered(ListPartition)

~ Function: ordered(ListPartition) ~

Description

Sort the [Partition](#) of a [ListPartition](#) according to a quality criterion.

Usage

```
ordered(x,...)
```

Arguments

x [ListPartition]: Object whose Partition should be sort.
... Note used, for S4 compatibility only.

Details

Sort the Partition of a ListPartition for each list (sort the 'c2' list, the 'c3' list,...) according to a quality criterion. The criterion used to sort is the one in the field `criterionActif`.

Value

This function change internally the order of the fields `c2`, `c3`, ... `c26` of an object. In addition, it return the permutation matrix (the matrix use to re-ordered the `ci`).

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] Christophe M. Genolini and Bruno Falissard
"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010
- [2] Christophe M. Genolini and Bruno Falissard
"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

Examples

```
#####  
### Preparing data  
  
data <- gald(30)["traj"]  
  
### Some clustering  
part2 <- partition(rep(c("A","B","A"),time=40),data)  
part3 <- partition(rep(c("A","B","C"),time=40),data)  
part3b <- partition(rep(c("A","B","C","B","C"),time=24),data)
```

```

part4 <- partition(rep(c("A","B","A","C","D"),time=24),data)

#####
### ListPartition
listPart <- listPartition()
listPart['criterionActif'] <- "Davies.Bouldin"
plotCriterion(listPart)

listPart["add"] <- part2
listPart["add"] <- part3
listPart["add"] <- part3b
listPart["add"] <- part4
listPart["add"] <- part4
listPart["add"] <- part3
listPart["add"] <- part3b

plotCriterion(listPart)
ordered(listPart)
plotCriterion(listPart)

listPart['criterionActif'] <- "Calinski.Harabatz"
plotCriterion(listPart)
ordered(listPart)
plotCriterion(listPart)

```

parLongData

~ Functions: *parLongData*, *parTraj* and *parMean*~

Description

parLongData, parTraj and parMean are constructors for the class [ParLongData](#).

Usage

```

parLongData(type, col, pch, pchPeriod, cex, xlab, ylab)
parTRAJ(type = "l", col = "black", pch = "1", pchPeriod = 0, cex = 1, xlab = "Time", ylab = "")
parMEAN(type = "b", col = "clusters", pch = "letters", pchPeriod = 1, cex = 1.2, xlab = "Time", ylab = "")

```

Arguments

type	[character]: Set type of the plot should be drawn ('p' for point, 'l' for line, 'b' for both, 'c' line appart, 'o' for overplot, 'h' for histogram, 's' and 'S' for steps, 'n' for no plotting)
col	[character]: Set the plotting color. Vector of values are accepted. The special value 'clusters' can be use to color each trajectories according to its clusters (see details).

pch	[numeric] or [character]: Either an integer specifying a symbol or special values 'letters' or 'symbol' (see details).
pchPeriod	[numeric]: Fix the number of point that should be plot. Usefull to plot points on trajectories with a lot of mesurement (see examples in plot for LongData for details).
cex	[numeric]: Set the amount by which plotting text and symbols should be magnified relative to the default.
xlab	[character]: Title for the x axis.
ylab	[character]: Title for the y axis.

Details

parLongData is the basic constructor of the class [ParLongData](#).

parTRAJ create an object with default values for plotting individual trajectories ;

parMEAN create an object with default values for plotting mean trajectories.

If col='clusters', pch='letters' or pch='symbol', the object can not be use directly, it should first be prepared using the function [expandParLongData](#).

Value

An object of class [ParLongData](#)

Author(s)

Christophe Genolini
 PSIGIAM: Paris Sud Innovation Group in Adolescent Mental Health
 INSERM U669 / Maison de Solenn / Paris

Contact author : <genolini@u-paris10.fr>

English translation

Raphaël Ricaud
 Laboratoire "Sport & Culture" / "Sports & Culture" Laboratory
 University of Paris 10 / Nanterre

Examples

```
#####
### Construction of LongData

time=c(1,2,3,4,8,12,16,20)
id2=1:120
f <- function(id,t)((id-1)%3-1) * t
g <- function(id,t)(id%2+1)*t
ld2 <- longData3d(array(cbind(outer(id2,time,f),outer(id2,time,g))+rnorm(120*8*2,0,3),dim=c(120,8,2)))
```

```

### Example with default value
plot(ld2)
plot(ld2,parTraj=parTRAJ())

### Example with default values for mean trajectories
plot(ld2,parTraj=parMEAN())

### Example with default value except for the color
plot(ld2,parTraj=parTRAJ(col="blue"))

```

ParLongData-class ~ Class: ParLongData ~

Description

ParLongData is an objet containing some graphical parameter used to plot [LongData](#) object and / or mean trajectories. They work as define in par.

Slots

type [character]: Type of the plot that should be drawn ('p' for point, 'l' for line, 'b' for both, 'c' line appart, 'o' for overplot, 'h' for histogram, 's' and 'S' for steps, 'n' for no plotting)

col [character]: A specification for the default plotting color. Can be either a single value or a vector.

pch [numeric] or [character]: Either an integer specifying a symbol or a single character to be used as the default in plotting points. See example in [points](#) for possible values and their interpretation.

pchPeriod [numeric]: Fix the number of point that should be plot. Usefull to plot points on trajectories with a lot of mesurement (see examples in [plot](#) for LongData for details).

cex [numeric]: A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default.

xlab [character]: A title for the x axis.

ylab [character]: A title for the y axis.

Construction

Object ParLongData can be created by three functions:

1. parLongData create an object from scratch ;
2. parTraj create an object containing default value to plot individutal trajectories;
3. parMean create an object containing default value to plot mean trajectories.

Methods

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

object['xxx']<-value Set the field xxx to value.

Author(s)

Christophe Genolini
 PSIGIAM: Paris Sud Innovation Group in Adolescent Mental Health
 INSERM U669 / Maison de Solenn / Paris

Contact author : <genolini@u-paris10.fr>

English translation

Raphaël Ricaud
 Laboratoire "Sport & Culture" / "Sports & Culture" Laboratory
 University of Paris 10 / Nanterre

Examples

```
### Building ParLongData
parMyData <- parLongData(type="n", col=3, pch="1", pchPeriod=20, cex=1, xlab="Time", ylab="Size")

### Get
parMyData['col']

### Set
parMyData['cex'] <- 3
(parMyData)
```

partition ~ *Function: partition* ~

Description

partition is the constructor of the class [Partition](#). It can be build either alone or relatively to a object LongData.

Usage

```
partition(clusters, traj, details=character())
```

Arguments

clusters	[vector(factor)]: cluters to which each individual belongs. Each clusters is represented by an upper letters.
traj	[matrix] or [array]: if an object LongData is provide, it will be used to compute the quality criterion of the clustering. array are simply turn into matrix by "sticking" all the variables one behind the other.
details	[vector(character)]: the slot details is used to store various informations. If the Partition has been find using an algorithm, it can store the name of the algorithm, the time before convergence, the number of iteration and any other informations. The syntaxe is details=c(algorithm="kmeans", convergenceTime="6", otherInfo="Wha

Details

partition construct a object of class `Partition`. It does not provide any default values. `yLongData` and `details` are optional.

Value

An object of class `Partition`.

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] Christophe M. Genolini and Bruno Falissard
"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010

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

See Also

[Partition,ordered](#)

Examples

```
### Empty partition
partition()

### Small partition
partition(clusters=c("A","B","A","C","C"))

### Random partition
partition(clusters=LETTERS[floor(runif(100,1,5))])

### Partition that clusters correctly some data
### Quality criterion are high
data <- gald()["traj"]
partition(clusters=rep(1:4,each=50),data)

### Partition that does not cluster correctly the data
### Quality criterion are low
partition(clusters=rep(1:4,50),data)
```

Partition-class ~ Class: Partition ~

Description

An object of class `Partition` is a partition of a population into subgroups. The object also contains some information like the percentage of trajectories in each group or some qualities criterion.

Objects from the Class

Objects are mainly intend to be created by some clustering methods (like k-means, fuzzy k-means, mixture modeling, latent class analysis,...)

Slots

`nbClusters` [numeric]: number of groups, between 1 and 26

`clusters` [vector(factor)]: vector containing the groups of each individual. Groups are in upper-case letters.

`percentEachCluster` [vector(numeric)]: percentage of trajectories contained in each group.

`postProba` [matrix(numeric)]: assuming that in each clusters *C* and for each time *T*, variable follow a normal law (mean and standard deviation of the variable at time *T* restricted to clusters *C*), then it is possible to compute the postterior probabilities of each individual (that is the probabilities that an individual has to belong to each clusters). These probabilities are hold in `postProba`.

`postProbaEachCluster` [vector(numeric)]: for each clusters *C*, mean of the post probabilities to belong to *C* of the individual that effectively belong to *C*. A high percent means that the individual that are in this cluter really meant to be here.

`criterionValues` [vector(numeric)]: Value of the quality criterions used to evaluate the quality of the Clustering. See [qualityCriterion](#) for details.

`details` [vector(character)]: hold different optionnal informations like the algorithm (if any) used to find the partition, the convergence time, the imputation methods, the starting condition. Examples: `details=c(algorithm="kmeans",convergenceTime="3")`.

validation rules

A class `Partition` object must follow some rules to be valid:

- Slots should be either all empty, or all non empty.
- `nbClusters` has to be lower or equal to 26.
- `clusters` is a factor in `LETTERS[1:nbCluster]`.

Construction

Class `Partition` objects are mainly constructed by some clustering methods (like k-means, fuzzy k-means, mixture modeling, latent class analysis,...). Nevertheless, it is also possible to construct them from scratch using the fonction [partition](#).

Get [

- Object["nbClusters"]** [numeric]: Gets the number of clusters (the value of the slot nbClusters)
- Object["clusters"]** [vector(factor)]: Gets the cluster of each individual (the value of the slot clusters)
- Object["clustersAsInteger"]** [vector(integer)]: Gets the cluster of each individual and turn them into integer
- Object["percentEachClusters"]** [vector(numeric)]: Get the percent of individual in each clusters (the value of the slot nbClusters)
- Object["postProbaEachClusters"]** [vector(numeric)]: Get the post probabilities for each clusters.
- Object["postProba"]** [matrix(numeric)]: Get the post probabilities for each individual and each clusters.
- Object["criterionValues"]** [vector(numeric)]: gives the values of all the criterion values (the value of the slot criterionValues)
- Object["details"]** [vector(character)]: Get the values of the slot details.
- Object["XcriterionX"]** [numeric]: Get the value of the criterion XcriterionX. It can be one of Calinski.Harabatz, Krzysztof.Calinski, Genolini.Calinski, Ray.Turi, Davies.Bouldin, BIC, AIC, AICc or random.
- Object["XspecialX"]** [character]: Get the value named XspecialX in the slot details (probably one of multiplicity, convergenceTime, imputationMethod or algorithm.)

Setteur [<-

- Object["multiplicity" <-value]** [numeric]: In the slot details, sets the values names multiplicity to value.
- Object["convergenceTime" <-value]** [numeric]: In the slot details, sets the values names convergenceTime to value.

The others slot can not be change after the object creation.

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: [longitudinalData-package](#)

Classes: [LongData](#)

Methods: [partition](#)

Examples

```
#####
### Building Partition

### number
part <- partition(rep(c(1,2,1,3),time=3))

### LETTERS
part <- partition(rep(c("A","B","D"),time=4),details=c(convergenceTime="3",multiplicity="1"))

### Others don't work
try(partition(rep(c("A","Bb","C"),time=3)))

#####
### Setteur and Getteur

### '['
part["clusters"]
part["clustersAsInteger"]
part["nbClusters"]

### '[<-'
part["multiplicity"] <- 2
(part)
```

parWindows

~ Function: *parWindows* ~

Description

parWindows is the constructor of object [ParWindows](#).

Usage

```
parWindows(nbRow, nbCol, addLegend,closeScreen)
```

Arguments

nbRow	[numeric]: Number of row of the screen matrix.
nbCol	[numeric]: Number of column of the screen matrix.
addLegend	[logical]: Shall a legend be added on the graph?

`closeScreen` [logical]: Some function need to add details on a graph. This option let them call a plot function that will not call a `close.screen` on exit, so the graph will be modifiable.

Details

`parWindows` is the constructor of object `ParWindows`. Given a number of rows and colonnes, it computes the `screenMatrix` that is use by `split.screen` for plot object `LongData`. If `addLegend` is true, an extra space is added on the top of the graphes to print the legend.

Value

An object of class `ParWindows`.

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

Examples

```
### Building ParWindows
(paramWin <- parWindows(3,2,FALSE,TRUE))

### Get
figsScreen <- paramWin['screenMatrix']

### Usage
listScreen <- split.screen(figsScreen)
screen(listScreen[1])
plot(-5:5/10,2.5-(-5:5)^2/20,ylim=c(0,6),axes=FALSE,xlab="",ylab="",type="l",lwd=3)
lines(-5:5/10,(-5:5)^2/20,ylim=c(0,6),type="l",lwd=3)

screen(listScreen[3])
plot(-5:5/10,2.5-(-5:5)^2/20,ylim=c(0,6),axes=FALSE,xlab="",ylab="",type="l",lwd=3)
lines(-5:5/10,(-5:5)^2/20,ylim=c(0,6),type="l",lwd=3)

screen(listScreen[5])
```

```
plot(-5:5/10, (-5:5)^2/10, ylim=c(0,6), axes=FALSE, xlab="", ylab="", type="l", lwd=3)
lines(-5:5/10, (-5:5)^2/20+1.25, ylim=c(0,6), type="l", lwd=3)
close.screen(all.screens=TRUE)

### :-)
```

ParWindows-class ~ Class: ParWindows ~

Description

ParWindows is an objet containing graphical parameter used to set the screen display.

Slots

nbCol [numeric]: Number of column of the screen matrix.

nbRow [numeric]: Number of row of the screen matrix.

addLegend [logical]: Shall a legend be added on the graph?

closeScreen [logical]: On exit, high level plot function can either close the screen that they open and return nothing ; or not close it and return the list of the screen number.

screenMatrix [matrix(numeric)]: Matrix with 4 column defining the screen region, like the figs argument of the function [screen](#). The screenMatrix can be specified by the user (bad idea) or can be compute automatically according to nbCol, nbRow and addLegend. For that, use [windowsCut](#).

Construction

Object ParWindows can be created by the constructor [parWindows](#) or by the function [windowsCut](#).

Methods

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

object['xxx']<-value Set the field xxx to value.

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

Examples

```
### Building ParWindows
(paramWin <- parWindows(3,2,FALSE,TRUE))

### Get
figsScreen <- paramWin['screenMatrix']

### Usage
listScreen <- split.screen(figsScreen)
screen(listScreen[1])
plot(-5:5/10,2.5-(-5:5)^2/20,ylim=c(0,6),axes=FALSE,xlab="",ylab="",type="l",lwd=3)
lines(-5:5/10,(-5:5)^2/20,ylim=c(0,6),type="l",lwd=3)

screen(listScreen[3])
plot(-5:5/10,2.5-(-5:5)^2/20,ylim=c(0,6),axes=FALSE,xlab="",ylab="",type="l",lwd=3)
lines(-5:5/10,(-5:5)^2/20,ylim=c(0,6),type="l",lwd=3)

screen(listScreen[5])
plot(-5:5/10,(-5:5)^2/10,ylim=c(0,6),axes=FALSE,xlab="",ylab="",type="l",lwd=3)
lines(-5:5/10,(-5:5)^2/20+1.25,ylim=c(0,6),type="l",lwd=3)
close.screen(all.screens=TRUE)

### Sorry for that...
```

pathFrechet

~ Function: Frechet distance ~

Description

Compute Frechet distance and Frechet path between two trajectories.

Usage

```
pathFrechet(P, Q, method = "max", Fdist = dist)
```

Arguments

P	[vector(numeric)] First trajectories.
Q	[vector(numeric)] First trajectories.
method	[character] Method used. Can be either 'max' or 'sum'
Fdist	[numeric <- function(numeric,numeric)] Frechet distance between two trajectories use a distance that is use the compute the distance between points of the trajectories. Fdist can be used to define a specific distance. The special value "2D" is used for "euclidean" distance. The special value "1D" can be use for considering that the trajectories are in 1D.

Details

Given two curve P and Q, given a distance d, Frechet distance between P and Q is define as $\inf_{a,b} \max_{t} d(P(a(t)), Q(b(t)))$. It's computation is a NP-complex problem. When P and Q are trajectories (discrete curve), the problem is polynomial (and quite simple).

The Frechet distance can also be define using a sum instead of a max: $\inf_{a,b} \sum_{t} d(P(a(t)), Q(b(t)))$

The Frechet path [...]

The function pathFrechetR is code in R. The function pathFrechet is coded in C (and is thus much faster than the two other). Note that pathFrechet (the fastest) can only use the "2D" and "1D" distance.

Value

A numeric value and the Frechet path.

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

Thomas Eiter & Heikki Mannila: "Computing Discrete Fréchet Distance"

[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

distFrechet

Examples

```
P <- rnorm(7)
Q <- rnorm(6)

### Function compiled in C
pathFrechet(P,Q)

### Frechet using sum instead of max.
pathFrechet(P,Q,method="sum")

### Frechet using "manhattan" distance
pathFrechet(P,Q,Fdist=function(x)dist(x,method="manhattan"))
```

plot,LongData ~ function: plot for LongData or LongData3d~

Description

plot the [LongData](#) or [LongData3d](#) optionnaly relatively to a [Partition](#). For joint trajectories, one graphe for each variable trajectory is displayed.

Usage

```
## S4 method for signature 'LongData,Partition'
plot(x,y,parTraj=parTRAJ(),parMean=parMEAN(),parWin=windowsCut(x['nbVar'],addLegend=TRUE),nbSample=
## S4 method for signature 'LongData3d,Partition'
plot(x,y,parTraj=parTRAJ(),parMean=parMEAN(),parWin=windowsCut(x['nbVar'],addLegend=TRUE),nbSample=
```

Arguments

x	[LongData] or [LongData3d]: Object containing the trajectories to plot.
y	[numeric]: Partition that will be use to plot the object. If y is missing, a Partition with a single cluster is considered.
parTraj	[ParLongData]: Set the graphical parameters used to plot the trajectories. See ParLongData and examples for details.
parMean	[ParLongData]: Set the graphical parameters used to plot the mean trajectories of each clusters (only when y is non missing). See ParLongData and examples for details.
parWin	[ParWindows]: Set the graphical display of the windows. See ParWindows 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.

Details

plot either a [LongData](#), or each variable of a [LongData3d](#) optionnaly according 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](#).

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

[LongData](#), [LongData3d](#), [plot3d](#).

Examples

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

ld <- gald3d()
part <- partition(rep(1:3,each=50))

### Basic plotting
plot(ld)
plot(ld,part)

### Change the windows orientation
plot(ld,parWin=windowsCut(c(1,2),addLegend=FALSE))

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

### No letters on the mean trajectories
```

```

plot(ld,part,parMean=parMEAN(type="l"))

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

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

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

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

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

#####
### single variable trajectory

ld2 <- gald()
part2 <- partition(rep(1:4,each=50))
plot(ld2)
plot(ld2,part2)

```

plot3d,LongData ~ Function: plot3d for LongData3d ~

Description

Plot two variables of a [LongData3d](#) object in 3D, optionnaly relatively to a [Partition](#).

Usage

```

## S4 method for signature 'LongData3d,missing'
plot3d(x,y,varY=1,varZ=2,parTraj=parTRAJ(),parMean=parMEAN(),nbSample=1000)
## S4 method for signature 'LongData3d,Partition'
plot3d(x,y,varY=1,varZ=2,parTraj=parTRAJ(type="n"),parMean=parMEAN(),nbSample=1000)

```

Arguments

x	[LongData3d]: Object containing the trajectories to plot.
y	[Partition]: Partition that will be use to plot the object. If y is missing, a Partition with a single clusters is considered.
varY	[numeric] or [character]: either the number or the name of the first variable to display. 1 by default.

varZ	[numeric] or [character]: either the number or the name of the second variable to display. 2 by default.
parTraj	[parLongData]: Set the graphical parameters used to plot the trajectories of the LongData3d. See ParLongData and examples for details.
parMean	[parLongData]: Set the graphical parameters used to plot the mean trajectories of each clusters LongData3d (only when y is non missing). See ParLongData and examples for details.
nbSample	[numeric]: Graphical display of huge sample can be time consuming. This parameters fixe the maximum nombre of trajectories (randomly chosen) that will be drawn.

Details

Plot two variables of a [LongData3d](#) object in 3D. It use the [rgl](#) library. The user can make the graphical representation turn using the mouse.

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

[LongData3d](#)

Examples

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

time=c(1,2,3,4,8,12,16,20)
id2=1:120
f <- function(id,t)((id-1)%3-1) * t
g <- function(id,t)(id%2+1)*t
h <- function(id,t)(id%4-0.5)*(20-t)
```

```

ld <- longData3d(array(cbind(outer(id2,time,f),outer(id2,time,g),outer(id2,time,h))+rnorm(120*8*3,0,3),dim=c(120,8,3)),dim=c(120,8,3))
part <- partition(rep(1:6,20))

### Basic plotting
plot3d(ld)
plot3d(ld,part)

### Variable 1 and 3, then 2 and 3
plot3d(ld,part)
plot3d(ld,part,varY=3,varZ=2)
plot3d(ld,part,varY=1,varZ=3)

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

### Color individual according to its clusters (col="clusters")
plot3d(ld,part,parTraj=parTRAJ(col="clusters"))
plot3d(ld,part,parTraj=parTRAJ(col="clusters"),varY=1,varZ=3)

### No mean trajectories (type="n"), only few trajectories
### Color individual according to its clusters (col="clusters")
plot3d(ld,part,parTraj=parTRAJ(col="clusters"),parMean=parMEAN(type="n"),nbSample=10)

```

plot3dPdf

~ Function: plot3dPdf for LongData ~

Description

Given a [LongData](#) and a [Partition](#), this function create 'Triangle objects' representing the 3D plot the clusters centers. Triangle object can latter be used to include dynamic rotating graph in a pdf file.

Usage

```

## S4 method for signature 'LongData3d,missing'
plot3dPdf(x,y,varY=1,varZ=2)
## S4 method for signature 'LongData3d,numeric'
plot3dPdf(x,y,varY=1,varZ=2)

```

Arguments

x	[LongData]: Object containing the trajectories to plot.
y	[numeric]: Partition that will be use to plot the object.
varY	[numeric] or [character]: either the number or the name of the first variable to display. 1 by default.
varZ	[numeric] or [character]: either the number or the name of the second variable to display. 2 by default.

Details

Create Triangle objects representing the 3D plot of the main trajectories of a [LongData](#).

The three functions [plot3dPdf](#), [saveTrianglesAsASY](#) and [makeLatexFile](#) are design to export a 3D graph to a Pdf file. The process is the following:

1. [plot3dPdf](#): Create a scene, that is a collection of Triangle object that represent a 3D images.
2. [saveTrianglesAsASY](#): Export the scene in an '.asy' file.
3. '.asy' can not be include in LaTeX file. LaTeX can read only '.pre' file. So the next step is to use the software asymptote to convert '.asy' to '.pre'. This is done by the command `asy -inlineimage -tex pdflatex scene.asy` (not in R, in a console).
4. The previous step did produce a file `scene+0.prc` that can be include in a LaTeX file. [makeLatexFile](#) create a LaTeX file that is directly compilable (using pdfLatex). It produce a pdf file that contain the 3D object.

Value

A Triangle object.

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

[saveTrianglesAsASY](#), [makeLatexFile](#), [makeTriangles](#)

Examples

```
### Generating the data
myLd <- gal3d()
part <- partition(rep(1:3,each=50))
plot3d(myLd,part)

### Creation of the scene
```

```

scene <- plot3dPdf(myLd,part)
drawScene.rgl(scene)

### Export in '.asy' file
saveTrianglesAsASY(scene)

### Creation of a '.prc' file
# Open a console, then run:
# 'asy -inlineimage -tex pdflatex scene.asy'

### Creation of the LaTeX main document
makeLatexFile()

### Creation of the '.pdf'
# Open a console window, then run
# pdfLatex main.tex

```

plotCriterion ~ Function: plotCriterion ~

Description

This function graphically displays the quality criterion of all the [Partition](#) of a [ListPartition](#) object.

Usage

```
plotCriterion(x, criterion=x["criterionActif"],nbCriterion=100,standardized = FALSE)
```

Arguments

x	[ClusterLongData]: object whose quality criterion should be displayed.
criterion	[character]: name of the criterion(s) to plot. It can either display all the value for a single specific criterion or display several criterion, only the best value for each clusters number and for each criterion.
nbCriterion	[numeric]: if there is a big number of Partition , the graphical display of all of them can be slow. nbCriterion lets the user limit the number of criteria that will be taken in account.
standardized	[logical]: some criterion should be maximized (like Calinski & Harabatz), some other should be minimized (like Davies & Bouldin). Some take big value, some small value, some négatives values. If standardized=TRUE, all the criterion will be transform such that they should all be maximized and they all take value into [0,1]. This makes them more easily comparable.

Details

This function display graphically the quality criterion (probably to decide the best clusters' number). It can either display all the criterion ; this is useful to see the consistency of the result : is the best clusterization obtain several time or only one ? It can also display only the best result for each clusters number : this helps to find the local maximum, which is classically used to chose the "correct" clusters' number.

Value

No value are return. A graph is printed.

Examples

```
#####
### Data génération

data <- gald(30)["traj"]

### Some clustering
listPart <- listPartition()
listPart["add"] <- partition(rep(c("A","B","A"),time=40),data)
listPart["add"] <- partition(rep(c("A","B","B"),time=40),data)
listPart["add"] <- partition(rep(c("A","B","C"),time=40),data)
listPart["add"] <- partition(rep(c("A","B","C","B","C"),time=24),data)
listPart["add"] <- partition(rep(c("A","B","A","C","D"),time=24),data)
ordered(listPart)

#####
### graphical display
plotCriterion(listPart)
plotCriterion(listPart,criterion=CRITERION_NAMES)
```

qualityCriterion ~ Function: qualityCriterion ~

Description

Given a [LongData](#) and a [Partition](#), the fonction qualityCriterion calculate some qualities criterion.

Usage

```
qualityCriterion(traj,clusters,imputationMethod="copyMean")
```

Arguments

traj	[LongData] or [matrix]: object containing the trajectories on which the criterion is calculate.
clusters	[Partition] or [vector(integer)]: clusters to which individual belongs.
imputationMethod	[character]: if some value are missing in the LongData, it is necessary to impute them. Then the function qualityCriterion call the function <code>imputation</code> using the method method.

Details

Given a `LongData` and a `Partition` (or a matrix and a vector of integer), the fonction `qualityCriterion` calculate several quality criterion and return then as a list (see 'value' below).

If some individual have no clusters (ie if `Partition` has some missing values), the corresponding trajectories are exclude from the calculation.

Note that if there is an empty cluster or an empty trajectory, most of the criterions are unavailable.

Basicly, 6 non-parametrics criterions are computed. In addition, ASSUMING THAT in each clusters C and for each time T , the variable follow a NORMAL LAW (mean and standard deviation of the variable at time T restricted to clusters C), it is possible to compute the the posterior probabilities of the individual trajectories and the likelihood. From there, we can also compute the BIC, the AIC and the global posterior probability. The function `qualityCriterion` also compute there criterion. But the user should always keep in mind that these criterion are valid ONLY under the hypothesis of normality. If this hypothesis is not respected, algorithm like k-means will converge but the BIC and AIC will have no meaning.

IMPORTANT NOTE: Some criterion should be maximized, some other should be minimized. This might be confusing for the non expert. In order to simplify the comparison of the criterion, `qualityCriterion` compute the OPPOSITE of the criterion that should be minimized (the opposite off Ray & Bouldin, Davies & Turi, BIC and AIC). Thus, all the criterion computed by this function should be maximized.

Value

A list with three fields: the first is the list of the criterions. the second is the clusters post probabilities; the third is the matrix of the individual post probabilities.

Non-parametric criterion

Notations: k =number of clusters; n =number of individual; B =Between variance ; W =Within variance The criterion are:

- `Calinski.Harabatz[numeric]`: Calinski and Harabatz criterion: $c(k)=\text{Trace}(B)/\text{Trace}(W)*(n-k)/(k-1)$.
- `Krysczuk.Calinski[numeric]`: Calinski and Harabatz criterion modified by Krysczuk: $c(k)=\text{Trace}(B)/\text{Trace}(W)*(n-1)/(n-k)$.
- `Genolini.Calinski[numeric]`: Calinski and Harabatz criterion modified by Genolini: $g(k)=\text{Trace}(B)/\text{Trace}(W)*(n-k)/\sqrt{k-1}$.

- Ray.Turi[numeric]: Ray and Turi criterion: $r(k) = -\text{Vintra}/\text{Vinter}$ with $\text{Vintra} = \text{Sum}(\text{dist}(x, \text{center}(x)))$ and $\text{Vinter} = \min(\text{dist}(\text{center}_i, \text{center}_j)^2)$. (The "true" index of Ray and Turi is $\text{Vintra}/\text{Vinter}$ and should be minimized. See IMPORTANT NOTE above.)
- Davies.Bouldin[numeric]: Davies and Bouldin criterion: $d(k) = -\text{mean}(\text{Proximate}(\text{cluster}_i, \text{cluster}_j))$ with $\text{Proximate}(i, j) = (\text{DistInterne}(i) + \text{DistInterne}(j)) / (\text{DistExterne}(i, j))$. (The "true" index of Davies and Bouldin is $\text{mean}(\text{Proximate}())$ and should be minimized. See IMPORTANT NOTE above.)
- random[numeric]: random value following the normal law $N(0,1)$.

Parametric criterion

Under their classic form, all these indices should be minimized. So the function `qualityCriterion` compute their opposite (see IMPORTANT NOTE above.)

Notation: L=likelihood; h=number of parameters; n=number of trajectories; t=number of time measurement; N=total number of measurement ($N=t.n$).

- BIC[numeric]: Bayesian Information Criterion: $\text{BIC} = 2 \cdot \log(L) - h \cdot \log(n)$. See IMPORTANT NOTE above.
- BIC2[numeric]: Bayesian Information Criterion: $\text{BIC} = 2 \cdot \log(L) - h \cdot \log(N)$. See IMPORTANT NOTE above.
- AIC[numeric]: Akaike Information Criterion, bis: $\text{AIC} = 2 \cdot \log(L) - 2 \cdot h$. See IMPORTANT NOTE above.
- AICc[numeric]: Akaike Information Criterion with correction: $\text{AIC} = \text{AIC} + (2h(h+1))/(n-h-1)$. See IMPORTANT NOTE above.
- AICc2[numeric]: Akaike Information Criterion with correction, bis: $\text{AIC} = \text{AIC} + (2h(h+1))/(n-h-1)$. See IMPORTANT NOTE above.

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

[LongData](#), [Partition](#), [imputation](#).

Examples

```
#####
### Preparation of some artificial data
par(ask=TRUE)
ld <- gald()

### Correct partition
part1 <- partition(rep(1:4,each=50))
plot(ld,part1)
(cr1 <- qualityCriterion(ld,part1))

### Random partition
part2 <- partition(floor(runif(200,1,5)))
plot(ld,part2)
(cr2 <- qualityCriterion(ld,part2))

### Partition with 3 clusters instead of 4
part3 <- partition(rep(c(1,2,3,3),each=50))
plot(ld,part3)
(cr3 <- qualityCriterion(ld,part3))

### Comparisons of the Partition
plot(c(cr1[[1]],cr2[[2]],cr3[[3]]),main="The highest give the best partition
(according to Calinski & Harabatz criterion)")
par(ask=FALSE)
```

regroup

~ Function: regroup ~

Description

Remove duplicate [Partition](#) present in a [ListPartition](#) (or, by inheritance, in [ClusterLongData](#) and [ClusterLongData3d](#) objects).

Usage

```
regroup(object)
```

Arguments

object [\[ListPartition\]](#): object that should be simplified.

Details

A clusterizing algorithm can find a [Partition](#) several time. It is store several time in object [ListPartition](#)(or in [ClusterLongData](#) or in [ClusterLongData3d](#)), encombering the memory. `regroup` remove the duplicate [Partition](#). Note that if the [ListPartition](#) is not ordered, then `regroup` sort it unless `toOrder=FALSE`.

Value

None (this function change internally the field of an object, it does not return any values.)

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] Christophe M. Genolini and Bruno Falissard
"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010

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

Examples

```
### Some data
data <- gald(30)["traj"]

### Some clustering
part2 <- partition(rep(c("A", "B", "A"), time=40), data)
part3 <- partition(rep(c("A", "B", "C"), time=40), data)

#####
### ListPartition
listPart <- listPartition()

listPart["add"] <- part2
listPart["add"] <- part3
listPart["add"] <- part2
listPart["add"] <- part3

### Some clustering has been found several time
### regroup will suppress the duplicate one
regroup(listPart)
plotCriterion(listPart)
```

restaureRealData ~ *Function: restaureRealData* ~

Description

This function revert the effect of [scale](#) by restauring the initial values of trajectories.

Usage

```
restaureRealData(object)
```

Arguments

object [LongData]: Object containnig trajectories to restaure.

Details

This function revert the effect of [scale](#) by restauring the initial values of trajectories.

Value

None: this function change internaly the field of an object, it does not return any values.)

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

[scale](#)

Examples

```
#####
### Building LongData

time=c(1,2,3,4,8,12,16,20)
id2=1:12
f <- function(id,t)((id-1)%3-1) * t
g <- function(id,t)(id%2+1)*t
ld1 <- longData3d(array(cbind(outer(id2,time,f),outer(id2,time,g))+rnorm(12*8*2,0,1),dim=c(12,8,2)))
plot3d(ld1)

#####
```

```
### Scaling by 'mean' and 'standard deviation'
scale(ld1,scale=c(-1,-1))
plot(ld1)

#####
### Back to the first version of the data
restaureRealData(ld1)
plot(ld1)
```

saveTrianglesAsASY ~ Function: saveTrianglesAsASY ~

Description

Export a Triangle object to an '.azy' file.

Usage

```
saveTrianglesAsASY(scene, filename = "scene.asy")
```

Arguments

scene [Triangle]: Object representing the graph to plot, probably produce by [plot3dPdf](#).
filename [character]: Name of exported file.

Details

Export a Triangle object to an '.asy' file. See [plot3dPdf](#) for a summary of the overall procedure.

Value

An '.asy' file, in the current directory.

Author(s)

Luke Tierney
Chair, Statistics and Actuarial Science
Ralph E. Wareham Professor of Mathematical Sciences
University of Iowa

References

<http://www.stat.uiowa.edu/~luke/R/misc3d/misc3d-pdf/>

See Also

[plot3dPdf](#),[makeLatexFile](#),[makeTriangles](#)

Examples

```

### Generating the data
myLd <- gald3d()
part <- partition(rep(1:3,each=50))
plot3d(myLd,part)

### Creation of the scene
scene <- plot3dPdf(myLd,part)
drawScene.rgl(scene)

### Export in '.azy' file
saveTrianglesAsASY(scene)

### Creation of a '.prc' file
# Open a console, then run:
# 'asy -inlineimage -tex pdflatex scene.azy'

### Creation of the LaTeX main document
makeLatexFile()

### Creation of the '.pdf'
# Open a console window, then run
# pdfLatex main.tex

```

scale

~ *Function: scale for LongData* ~

Description

scale the trajectories of the different variable of a [LongData](#) object.

Usage

```
scale(x, center = TRUE, scale = TRUE)
```

Arguments

x	[LongData]: Object containning trajectories to be scale.
center	[logical] or [vector(numeric)]: Value that will be subtract from each mesurement of a variable. If center=TRUE, the mean of each variable-trajectory is used. Otherwise, center should have a value for each variables.
scale	[logical] or [vector(numeric)]: Value that will divided, after the substraction, each mesurement of a variable. If scale=TRUE, the standard deviation of each variable-trajectory is used. Otherwise, scale should have a value for each variables.

Details

When variable with different unit are used jointly, it might be necessary to change their scale them in order to change their individual influence. This is what scale do.

More precisely, all the value $x[i,j,k]$ of the variable k will be scale according to the classic formula $(x[i,j,k] - m_k) / s_k$ where m_k and s_k are respectively the k -ieme value of the argument center and scale.

Note that `center=TRUE` is a special value that set $m_k = \text{mean}(x[, , k], na.rm=TRUE)$. Similarly, `scale=TRUE` is a special value that set $s_k = \text{sd}(x[, , k], na.rm=TRUE)$.

Value

scale directly modify the internal value of the LongData. No value is return.

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

[restaureRealData](#)

Examples

```
#####
### Building LongData

time=c(1,2,3,4,8,12,16,20)
id2=1:12
f <- function(id,t)((id-1)%3-1) * t
g <- function(id,t)(id%2+1)*t
ld1 <- longData3d(array(cbind(outer(id2,time,f),outer(id2,time,g))+rnorm(12*8*2,0,1),dim=c(12,8,2)))
plot3d(ld1)

#####
### Scaling by 'mean' and 'standard deviation'
```

```

plot(ld1)
scale(ld1)
plot(ld1)

### Scaling by some parameters
scale(ld1,center=c(10,100),scale=c(3,-1))
plot(ld1)

#####
### To restaure the data
restaureRealData(ld1)

```

windowsCut

~ Function: windowsCut ~

Description

windowsCut prepare an object [ParWindows](#) according to its arguments.

Usage

```
windowsCut(x, addLegend = TRUE,closeScreen=TRUE)
```

Arguments

x	[numeric] or [couple(numeric)]: x is used to calculate the fields nbCol and nbRow of the object ParWindows . If x is a couple, then x[1] is nbRow and x[2] is nbCol. If x is a single number (the number of plot that should be display), nbCol and nbRow parameters are calculate consequently (see detail).
addLegend	[logical]: If addLegendis true, an extra space is reserved on the top of the screen to print the legend.
closeScreen	[logical]: Some function need to add details on a graph. This option let them call a plot function that will not call a close.screen on exit, so the graph will be modifiable.

Details

If x is a number of variable, the column and row number are estimate according to the formula $nbCol \leftarrow \text{ceiling}(\sqrt{x})$ and $nbRow \leftarrow \text{ceiling}(x/nbCol)$.

Value

An object of class [ParWindows](#).

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

Examples

```
### Simple cut with no space for legend  
windowsCut(3,FALSE)  
windowsCut(4,FALSE)  
windowsCut(5,FALSE)  
  
### Simple cut with legend  
windowsCut(5)
```

Index

- *Topic **NA**
 - imputation, 15
- *Topic **aplot**
 - plot, LongData, 54
 - plot3d, LongData, 56
- *Topic **classes**
 - ListPartition-class, 25
 - LongData-class, 29
 - LongData3d-class, 34
 - ParLongData-class, 44
 - Partition-class, 47
 - ParWindows-class, 51
 - regroup, 64
- *Topic **classif**
 - LongData-class, 29
 - LongData3d-class, 34
 - longitudinalData-package, 2
 - ParWindows-class, 51
- *Topic **cluster**
 - generateArtificialLongData, 10
 - generateArtificialLongData3d, 13
 - imputation, 15
 - ListPartition-class, 25
 - longData, 28
 - LongData-class, 29
 - longData3d, 32
 - LongData3d-class, 34
 - longDataFrom3d, 37
 - longDataTo3d, 38
 - longitudinalData-package, 2
 - ParLongData-class, 44
 - partition, 45
 - Partition-class, 47
 - ParWindows-class, 51
 - qualityCriterion, 61
 - regroup, 64
- *Topic **datagen**
 - generateArtificialLongData, 10
 - generateArtificialLongData3d, 13
- *Topic **datasets**
 - Constants, 4
- *Topic **dplot**
 - longitudinalData-package, 2
- *Topic **methods**
 - imputation, 15
 - longData, 28
 - longData3d, 32
 - longDataFrom3d, 37
 - longDataTo3d, 38
 - ordered(ListPartition), 40
 - parLongData, 42
 - partition, 45
 - qualityCriterion, 61
- *Topic **method**
 - scale, 68
- *Topic **package**
 - imputation, 15
 - longData, 28
 - longData3d, 32
 - longDataFrom3d, 37
 - longDataTo3d, 38
 - longitudinalData-package, 2
 - plot, LongData, 54
 - plot3d, LongData, 56
 - qualityCriterion, 61
- *Topic **ts**
 - generateArtificialLongData, 10
 - generateArtificialLongData3d, 13
 - imputation, 15
 - ListPartition-class, 25
 - LongData-class, 29
 - LongData3d-class, 34
 - ParWindows-class, 51
 - plot3d, LongData, 56
 - regroup, 64
 - [, ListPartition-method (ListPartition-class), 25
 - [, LongData-method (LongData-class), 29

- [,LongData3d-method (LongData3d-class), 34
- [,ParLongData-method (ParLongData-class), 44
- [,ParWindows-method (ParWindows-class), 51
- [,Partition-method (Partition-class), 47
- [<-,ListPartition-method (ListPartition-class), 25
- [<-,LongData-method (LongData-class), 29
- [<-,LongData3d-method (LongData3d-class), 34
- [<-,ParLongData-method (ParLongData-class), 44
- [<-,ParWindows-method (ParWindows-class), 51
- [<-,Partition-method (Partition-class), 47

- CHOICE_STYLE (Constants), 4
- CLUSTER_NAMES (Constants), 4
- Constants, 4
- CRITERION_NAMES (Constants), 4

- dist, 7, 8
- DISTANCE_METHODS (Constants), 4
- distFrechet, 5
- distTraj, 6

- expandParLongData, 8, 43
- expandParLongData,ParLongData,numeric (expandParLongData), 8
- expandParLongData,ParLongData,numeric-method (expandParLongData), 8
- expandParLongData,ParLongData,Partition (expandParLongData), 8
- expandParLongData,ParLongData,Partition-method (expandParLongData), 8

- gald, 31
- gald (generateArtificialLongData), 10
- gald3d, 31, 35
- gald3d (generateArtificialLongData3d), 13
- generateArtificialLongData, 10, 14, 31
- generateArtificialLongData3d, 11, 13, 31, 35

- imputation, 3, 15, 32, 36, 62, 63
- imputation,array-method (imputation), 15
- imputation,LongData-method (imputation), 15
- imputation,LongData3d-method (imputation), 15
- imputation,matrix-method (imputation), 15
- initializePartition, 22
- initializePartition,numeric,numeric,character,ANY (initializePartition), 22
- initializePartition,numeric,numeric,character,ANY-method (initializePartition), 22
- initializePartition,numeric,numeric,character,array (initializePartition), 22
- initializePartition,numeric,numeric,character,array-method (initializePartition), 22

- ListPartition, 41, 60, 64
- ListPartition (ListPartition-class), 25
- listPartition, 26
- listPartition (ListPartition-class), 25
- ListPartition-class, 25
- listPartition-method (ListPartition-class), 25

- LongData, 3, 10, 11, 13–18, 27–29, 32–34, 36–39, 44, 49, 50, 54, 55, 58, 59, 61–63, 68
- LongData (LongData-class), 29
- longData, 3, 11, 28, 30, 32
- longData,ANY,ANY,ANY,ANY,ANY,ANY-method (longData), 28
- longData,missing,missing,missing,missing,missing,missing-method (longData), 28
- LongData-class, 29
- LongData3d, 14, 37, 38, 54–57
- LongData3d (LongData3d-class), 34
- longData3d, 14, 28, 30, 32, 34–36
- longData3d,ANY,ANY,ANY,ANY,ANY,ANY-method (longData3d), 32
- longData3d,missing,missing,missing,missing,missing,missing-method (longData3d), 32
- LongData3d-class, 34
- longDataFrom3d, 31, 36, 37
- longDataTo3d, 38
- longitudinalData (longitudinalData-package), 2
- longitudinalData-package, 32, 36, 49
- longitudinalData-package, 2

- makeLatexFile, [39](#), [59](#), [67](#)
- makeTriangles, [40](#), [59](#), [67](#)
- MAX_CLUSTERS (Constants), [4](#)
- ordered, [3](#), [25](#), [46](#)
- ordered (ordered(ListPartition)), [40](#)
- ordered(ListPartition), [40](#)
- ordered,ListPartition
 - (ordered(ListPartition)), [40](#)
- ordered,ListPartition-method
 - (ordered(ListPartition)), [40](#)
- ParLongData, [8](#), [9](#), [42](#), [43](#), [54](#), [55](#), [57](#)
- ParLongData (ParLongData-class), [44](#)
- parLongData, [42](#)
- ParLongData-class, [44](#)
- parMEAN (parLongData), [42](#)
- Partition, [3](#), [18](#), [27](#), [31](#), [36](#), [41](#), [45](#), [46](#), [54](#), [56](#), [58](#), [60–64](#)
- partition, [3](#), [45](#), [47](#), [49](#)
- partition,ANY,array,ANY-method
 - (partition), [45](#)
- partition,ANY,LongData,ANY-method
 - (partition), [45](#)
- partition,ANY,LongData3d,ANY-method
 - (partition), [45](#)
- partition,ANY,matrix,ANY-method
 - (partition), [45](#)
- partition,ANY,missing,ANY-method
 - (partition), [45](#)
- partition,missing,missing,missing-method
 - (partition), [45](#)
- Partition-class, [47](#)
- parTRAJ (parLongData), [42](#)
- ParWindows, [49](#), [50](#), [54](#), [70](#)
- ParWindows (ParWindows-class), [51](#)
- parWindows, [49](#), [51](#)
- ParWindows-class, [51](#)
- pathFrechet, [52](#)
- plot, [3](#), [9](#), [32](#), [36](#), [43](#), [44](#)
- plot (plot,LongData), [54](#)
- plot(LongData), [3](#)
- plot,LongData, [54](#)
- plot,LongData,missing-method
 - (plot,LongData), [54](#)
- plot,LongData,Partition-method
 - (plot,LongData), [54](#)
- plot,LongData-method (plot,LongData), [54](#)
- plot,LongData3d (plot,LongData), [54](#)
- plot,LongData3d,missing-method
 - (plot,LongData), [54](#)
- plot,LongData3d,Partition-method
 - (plot,LongData), [54](#)
- plot,LongData3d-method (plot,LongData), [54](#)
- plot3d, [9](#), [32](#), [36](#), [55](#)
- plot3d (plot3d,LongData), [56](#)
- plot3d(LongData), [3](#)
- plot3d,LongData, [56](#)
- plot3d,LongData3d (plot3d,LongData), [56](#)
- plot3d,LongData3d,missing-method
 - (plot3d,LongData), [56](#)
- plot3d,LongData3d,Partition-method
 - (plot3d,LongData), [56](#)
- plot3d,LongData3d-method
 - (plot3d,LongData), [56](#)
- plot3dPdf, [32](#), [36](#), [40](#), [58](#), [59](#), [67](#)
- plot3dPdf,LongData3d,missing-method
 - (plot3dPdf), [58](#)
- plot3dPdf,LongData3d,numeric-method
 - (plot3dPdf), [58](#)
- plot3dPdf,LongData3d,Partition-method
 - (plot3dPdf), [58](#)
- plot3dPdf,LongData3d-method
 - (plot3dPdf), [58](#)
- plotCriterion, [25](#), [60](#)
- plotCriterion,ListPartition
 - (plotCriterion), [60](#)
- plotCriterion,ListPartition-method
 - (plotCriterion), [60](#)
- plotCriterion-method (plotCriterion), [60](#)
- points, [44](#)
- qualityCriterion, [3](#), [18](#), [32](#), [36](#), [47](#), [61](#)
- qualityCriterion,array,ANY-method
 - (qualityCriterion), [61](#)
- qualityCriterion,LongData,Partition-method
 - (qualityCriterion), [61](#)
- qualityCriterion,LongData3d,Partition-method
 - (qualityCriterion), [61](#)
- qualityCriterion,matrix,ANY-method
 - (qualityCriterion), [61](#)
- regroup, [64](#)
- restaureRealData, [31](#), [35](#), [65](#), [69](#)
- restaureRealData,LongData
 - (restaureRealData), [65](#)

restaureRealData, LongData-method
 (restaureRealData), 65
restaureRealData, LongData3d
 (restaureRealData), 65
restaureRealData, LongData3d-method
 (restaureRealData), 65
rgl, 57

saveTrianglesAsASY, 40, 59, 67
scale, 30, 31, 35, 65, 66, 68
scale, LongData (scale), 68
scale, LongData-method (scale), 68
scale, LongData3d (scale), 68
scale, LongData3d-method (scale), 68
screen, 51
show, ListPartition-method
 (ListPartition-class), 25
show, LongData-method (LongData-class),
 29
show, LongData3d-method
 (LongData3d-class), 34
show, Partition-method
 (Partition-class), 47
split.screen, 50

windowsCut, 51, 70