

# Package ‘longitudinalData’

February 18, 2010

**Type** Package

**Title** Longitudinal Data

**Version** 0.6.3

**Date** 2010-02-18

**Author** Christophe Genolini

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

**Description** Tools for Longitudinal Data

**License** GPL (>= 2)

**Lazyload** yes

**Depends** methods,clv

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

**Collate** global.r function.r longData.r partition.r imputation.r criterion.r distanceFrechet.r  
plotLongData.r clean.r

**Encoding** latin1

**Repository** CRAN

**Date/Publication** 2010-02-18 13:04:37

## R topics documented:

longitudinalData-package . . . . .	2
as.longData . . . . .	3
criterion . . . . .	5
criterion3 . . . . .	6
distFrechet . . . . .	8
distTraj . . . . .	9
generateArtificialLongData . . . . .	11

imputation . . . . .	13
longData . . . . .	16
LongData-class . . . . .	17
ordered . . . . .	19
partition . . . . .	20
Partition-class . . . . .	21
pathFrechet . . . . .	22
plot,LongData . . . . .	24
plotSubGroups . . . . .	26
selectSupTrajMinSize . . . . .	28

## Index 30

---

longitudinalData-package

~ Package overview: longitudinalData ~

---

## Description

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

## Details

Package:	longitudinalData
Type:	Package
Version:	0.6.3
Date:	2010-02-18
License:	GPL (>= 2)
Lazyload:	yes
Depends:	methods,clv
URL:	<a href="http://www.r-project.org">http://www.r-project.org</a>

## Overview

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

1. [plot](#)
2. [plotSubGroups](#)
3. [imputation](#)
4. quality [criterion](#)

**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>

**See Also**

Classes: [LongData](#), [Partition](#)  
Methods: [longData](#), [as.longData](#), [selectSupTrajMinSize](#), [partition](#), [ordered](#)  
Plot: [plot\(LongData\)](#), [plotSubGroups\(LongData\)](#)  
Imputation: [imputation](#)  
Criterion: [criterion](#)

**Examples**

```
### Generation of artificial longData
data <- gald(percentOfMissing=0.3)
part <- partition(rep(1:4, each=50), 4)
plot(data, part)

### Two methods of imputation
par(mfrow=c(1, 2))
data1 <- imputation(data, method="linearInterpolation", partition=part)
plot(data1, part, legend=FALSE)

data2 <- imputation(data, method="LOCF")
plot(data2, part, legend=FALSE)
```

---

as.longData                      ~ *Function: as.longData* ~

---

**Description**

as.longData turns a data.frame or a matrix into an object of class [LongData](#).

**Usage**

```
as.longData(data, id, timeCol, timeReal, varName, other)
```

**Arguments**

data	[data.frame] or [matrix]: structure containing the longitudinal data.
id	[character]: single identifier for each trajectory (ie each individual). <ul style="list-style-type: none"> <li>• If data is a data.frame, the default id is the first column of the data.frame.</li> <li>• If data is a matrix, the default id is the vector 1:nrow(matrix).</li> </ul>
timeCol	[numeric]: column number in which longitudinal data can be found. <ul style="list-style-type: none"> <li>• If data is a data.frame, the default timeCol is all the column of the data.frame but the first.</li> <li>• If data is a matrix, the default timeCol is all the column of the matrix.</li> </ul>
timeReal	[numeric]: time at which measures were made in "real" life. By default, timeReal is 0:(length(timeCol)-1).
varName	[character]: Name of the variable being measured ("V" by default).
other	[list]: list of additionnal information.

**Details**

as.longData turns a data.frame or a matrix into an object of class [LongData](#).

**Value**

An object of class [LongData](#).

**See Also**

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

**Examples**

```
#####
### Simple data.frame
df <- data.frame(id=11:12, Va1=c(1,4), Va2=c(2,5))
as.longData(df)

#####
### Complexe data.frame
df <- data.frame(id=11:12, Va1=c(1,4), To1=c(0.1,0.4), Va2=c(2,5), To2=c(0.1,0.3))

### Transformation in a LongData, selecting variable "Va"
as.longData(df, timeCol=c(2,4))

### Transformation in a LongData, selecting variable "To"
as.longData(df, timeCol=c(3,5))

### Changing the id's name
as.longData(df, id=c("Ind-1", "Ind-2"), timeCol=c(2,4))

### Changing the real time and the variable name.
```

```

as.longData(df,timeCol=c(3,5,2,4),timeReal=c(1,2,4,8),varName="Size")

#####
### matrix
mat <- matrix(1:8,2,dimnames=list(c("I1","I2"),c("Size1.2","Size1.5","Size2.5","Size4")))
as.longData(mat)
as.longData(mat,timeReal=c(1.2,1.5,2.5,4))
as.longData(mat,timeReal=c(1.2,2.5,4),timeCol=c(1,3,4),varName="Size")

```

---

criterion                      ~ Function: criterion ~

---

## Description

Given a [LongData](#) and a [Partition](#), the fonction `criterion` calculate Calinski & Harabatz criterion.

## Usage

```
criterion(object, partition, method="linearInterpolation")
```

## Arguments

<code>object</code>	[ <a href="#">LongData</a> ]: object on which the criterion is calculate
<code>partition</code>	[ <a href="#">Parition</a> ]: clusterization of the <a href="#">LongData</a>
<code>method</code>	[ <a href="#">character</a> ]: if some value are missing in the <a href="#">LongData</a> , it is necessary to impute them. The fonction <code>criterion</code> call the fonction <a href="#">imputation</a> using the method <code>method</code> .

## Details

Given a [LongData](#) and a [Partition](#), the fonction `criterion` calculate Calinski and Harabatz criterion.

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 [longData](#), most of the criterions are unavailable.

## Value

A list:

- `calinski` [[numeric](#)]: Calinski and Harabatz criterion  $c(k)=\text{Trace}(B)/\text{Trace}(W)*(n-k)/(k-1)$

## See Also

[LongData](#), [Partition](#), [imputation](#), [criterion3](#)

**Examples**

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

### Correct partition
part1 <- partition(rep(1:4,each=50),4)
(cr1 <- criterion(traj,part1))
plot(traj,part1,main=paste("Calinski =",formatC(cr1[["calinski"]]))))

### Random partition
part2 <- partition(floor(runif(200,1,5)),4)
(cr2 <- criterion(traj,part2))
plot(traj,part2,main=paste("Calinski =",formatC(cr2[["calinski"]]))))

### Partition with 3 clusters instead of 4
part3 <- partition(rep(c(1,2,3,3),each=50),3)
(cr3 <- criterion(traj,part3))
plot(traj,part3,main=paste("Calinski =",formatC(cr3[["calinski"]]))))

### Comparisons of the Partition
cr1["calinski"]
cr2["calinski"]
cr3["calinski"]
par(ask=FALSE)
```

---

criterion3

~ Function: criterion3 ~

---

**Description**

Given a [LongData](#) and a [Partition](#), the fonction `criterion3` calculate some criterions that are classically use to estimate the quality of a clusterization.

**Usage**

```
criterion3(object, partition, method="linearInterpolation")
```

**Arguments**

<code>object</code>	[LongData]: object on which the criterion are calculate
<code>partition</code>	[Paritition]: clusterization of the LongData
<code>method</code>	[character]: if some value are missing in the LongData, it is necessary to impute them. The function <code>criterion</code> call the fonction <code>imputation</code> using the method <code>method</code> .

## Details

Given a [LongData](#) and a [Partition](#), the fonction `criterion` calculate three criterions that are classically use to estimate the quality of a clusterization : Calinski and Harabatz, Ray and Turi, Davies and Bouldin

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 `longData`, most of the criterions are unavailable.

## Value

A list:

- `calinski[numeric]`: Calinski and Harabatz criterion  $c(k)=\text{Trace}(B)/\text{Trace}(W)*(n-k)/(k-1)$
- `ray[numeric]`: Ray and Turi  $R(k) = 1/N \sum_i \text{in } 1:k \sum_x \text{in } C_i |x-z_i|^2 / \min(|z_i - z_j|^2)$
- `davies[numeric]`: Davies and Bouldin  $D(k) = \text{Avg}(\max(\text{pairwise comparison}))$

## See Also

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

## Examples

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

### Correct partition
part1 <- partition(rep(1:4,each=50),4)
(cr1 <- criterion3(traj,part1))
plot(traj,part1,main=paste("Ray =",formatC(cr1[["ray"]]))

### Random partition
part2 <- partition(floor(runif(200,1,5)),4)
(cr2 <- criterion3(traj,part2))
plot(traj,part2,main=paste("Ray =",formatC(cr2[["ray"]]))

### Partition with 3 clusters instead of 4
part3 <- partition(rep(c(1,2,3,3),each=50),3)
(cr3 <- criterion(traj,part3))
plot(traj,part3,main=paste("Ray =",formatC(cr3[["ray"]]))

### Comparisons of the Partition
cr1[["ray"]]
cr2[["ray"]]
cr3[["ray"]]
par(ask=FALSE)
```

---

distFrechet                    ~ Function: Frechet distance ~

---

## Description

Compute Frechet distance between two trajectories.

## Usage

```
distFrechet(P, Q, method = "max", Fdist = dist)
distFrechetR(P, Q, method = "max", Fdist = dist)
distFrechetRec(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 efficiant algorithm in R. The function `distFrechet` use the efficiant 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(s)

Christophe Genolini

## References

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

**See Also**

distTraj

**Examples**

```

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

### Function from Eiter and Mannila
distFrechetRec(P,Q)

### Optimized function
distFrechetR(P,Q)

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

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

### Frechet using "manhattan" distance
distFrechetR(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 trajectories.

**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  $p$ th root of the sum of the  $p$ th 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(s)

Christophe Genolini

## References

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

## 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"))

```

---

```
generateArtificialLongData
```

```
~ Function: generateArtificialLongData (or gald) ~
```

---

## Description

This function build up an artificial longitudinal data set an turn it into an object of class `LongData`.

## Usage

```

gald(nbEachClusters=50,time=0:10,decimal=2,percentOfMissing=0,
     functionClusters=list(function(t){0},function(t){t},function(t){10-t},function(t){t}),
     functionNoise=function(t){rnorm(1,0,3)})

```

```

generateArtificialLongData(nbEachClusters=50,time=0:10,decimal=2,percentOfMissing=0,
                           functionClusters=list(function(t){0},function(t){t},function(t){10-t},function(t){t}),
                           functionNoise=function(t){rnorm(1,0,3)})

```

## 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 (see detail).

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

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

`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 (see detail).

**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`.

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

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

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

Note that the number of cluster is define as the biggest length of variables `nbEachClusters`, `functionCluters`, `functionNoise` and `percentOfMissing`. So at least one of these four variables should be define for each clusters.

**Value**

An object of class `LongData`. Note that the field `other` of the object `LongData` contains the informations that were used to generate the set of data: `functionClusters`, `functionNoise`, `percentOfMissing` and `trueClusters`.

**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>

**See Also**

[LongData](#), [longData](#), [as.longData](#), [plot](#)

**Examples**

```
par(ask=TRUE)

### Default example
ex1 <- generateArtificialLongData()
ex1
plot(ex1, col=1, type.mean="n")
part1 <- partition(rep(1:4, each=50), 4)
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), 3)
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)),3)
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*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)),4)
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)

```

---

imputation

~ *Function: imputation* ~

---

## Description

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

## Usage

```
imputation(object, method, partition)
```

## Arguments

object	[LongData] or [matrix] : longitudinal data to impute
method	[character] : Name of the imputation method (see detail)
partition	[Partition] : for some imputation technique (like "copyMean"), a specific partition is needed. See detail.

## Details

`imputation` is a function that impute missing value of a `LongData`. Several imputation methods are available. For each method, the imputation has to deal with three kind of missing value : at start of the trajectory (first values are missing), at the end (last values are missing) or in the middle (the missing value have surround by non-missing value). Here is a description of each methods (for all of them, an example is provided in the `Examples` section):

- `LOCF` (Last Occurrence Carried Forward)
  - Missing in the middle / at the end: the previous non-missing value is duplicated forward.
  - Missing at start: the first non-missing occurrence is duplicated backward (`LOCB`).
- `LOCB` (Last Occurrence Carried Backward)
  - Missing at start / in the middle: the next non-missing value is duplicated backward.
  - Missing at the end: the previous non-missing occurrence is duplicated forward (`LOCF`).
- `linearInterpolation`
  - Missing in the middle: the value immediately surrounding the missing are join by a line.
  - Missing at start / at the end: the line joining the first and last non-missing value is considered (this line is the average progression of the actual individual trajectory). Missing-value at start and at the end are chose on this line.
- `linearInterpolation2`
  - Missing in the middle: like `linearInterpolation`, the value immediately surrounding the missing are join by a line;
  - Missing at start: they are chosen to prolongate the line going through the first and second non missing value.
  - Missing at the end: They are chosen to prolongate the line going through the last and the penultimate value.
- `linearInterpolation3`
  - Missing in the middle: like `linearInterpolation`, the value immediately surrounding the missing are join by a line.
  - Missing at start: They are imputed by `LOCB`.
  - Missing at the end: They are imputed by `LOCF`.
- `copyMean`: this method impute `LongData` relatively to a `Partition`. More precisely, each individual trajectory is imputed relatively to its clusters center. For all kind of missing value, the shape of the clusters center is copied and is level up (or down) to fit with the non missing value of the imputed trajectory.

## Value

A `matrix` with no missing values.

## See Also

[LongData](#), [Partition](#), [criterion](#)

**Examples**

```
#####
### Preparation of the data
timeV <- 1:19
trajMissing <- longData(
  matrix(c(NA,NA ,2 ,3 ,NA,5 ,5.5,5.8,6 ,NA ,NA,6.5 ,7.5 ,NA ,NA ,NA ,4 ,NA,NA,
          2 ,0.5,-2,-2.6,2 ,1 ,1.5,0 , -2,1.2,1 , -3.5,-4.9,0.7,1.2,2.5,-1,-1, 1),2,byrow=T
        id=1:2,time=timeV,varName="V"
  )

plot(timeV,trajMissing["traj"][1,],col=2,type="o",lwd=3,ylim=c(-2,10),
      xlab="Trajectorie to impute",ylab="")
par(ask=TRUE)

#####
### LOCF
trajImp <- imputation(trajMissing,method="LOCF")
plot(timeV,trajImp["traj"][1,],type="o",ylim=c(-2,10),ylab="",xlab="LOCF")
lines(timeV,trajMissing["traj"][1,],col=2,type="o",lwd=3)

### LOCB
trajImp <- imputation(trajMissing,method="LOCB")
plot(timeV,trajImp["traj"][1,],type="o",ylim=c(-2,10),ylab="",xlab="LOCB")
lines(timeV,trajMissing["traj"][1,],col=2,type="o",lwd=3)

### linearInterpolation
trajImp <- imputation(trajMissing,method="linearInterpolation")
plot(timeV,trajImp["traj"][1,],type="o",ylim=c(-2,10),ylab="",xlab="linearInterpolation")
lines(timeV,trajMissing["traj"][1,],col=2,type="o",lwd=3)

### linearInterpolation2
trajImp <- imputation(trajMissing,method="linearInterpolation2")
plot(timeV,trajImp["traj"][1,],type="o",ylim=c(-2,10),ylab="",xlab="linearInterpolation2")
lines(timeV,trajMissing["traj"][1,],col=2,type="o",lwd=3)

### linearInterpolation3
trajImp <- imputation(trajMissing,method="linearInterpolation3")
plot(timeV,trajImp["traj"][1,],type="o",ylim=c(-2,10),ylab="",xlab="linearInterpolation3")
lines(timeV,trajMissing["traj"][1,],col=2,type="o",lwd=3)

### copyMean
meanTraj <- apply(trajMissing["traj"],2,meanNA)
plot(timeV,trajMissing["traj"][1,],type="o",ylim=c(-2,10),col=2,lwd=3,ylab="",
      xlab="LongData to impute and its model (in green)")
lines(timeV,meanTraj,col=3,type="b",lwd=3)

trajImp <- imputation(trajMissing,method="copyMean",partition=partition(nbCluster=1,clusters=
plot(timeV,trajImp["traj"][1,],type="o",ylim=c(-2,10),ylab="",xlab="copyMean")
lines(timeV,trajMissing["traj"][1,],col=2,type="o",lwd=3)
lines(timeV,meanTraj,col=3,type="o",lwd=3)
```

```
### copyMean with several clusters
trajMiss <- longData(
  matrix(c(2,3,NA,0, NA,4,0,NA, 4,NA,-1,0),4),
  id=1:4,time=1:3
)
part <- partition(nbCluster=2,clusters=c(1,1,2,2))
imputation(trajMiss,method="copyMean",partition=part)
par(ask=FALSE)
```

---

longData                      ~ *Function: longData* ~

---

## Description

longData is the constructor of the class [LongData](#).

## Usage

```
longData(traj, id, time, varName, other)
```

## Arguments

traj	[matrix(numeric)]: structure containing the trajectories.
id	[vector(character)]: single identifier for each trajectory (ie each individual).
time	[vector(numeric)]: time at which measures were made.
varName	[character]: name of the variable being measured.
other	[list]: list of additionnal information.

## Details

longData construct a object of class [LongData](#). It does not provide any default values.

## Value

An object of class [LongData](#).

## See Also

[as.longData](#), [LongData](#), [plot](#)

**Examples**

```
### 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")))
(ld1 <- longData(traj=mat, id=as.character(c(101, 102, 104)), time=c(2, 4, 8), varName="T", other=list()))
plot(ld1, col=1, type="mean")

### Big data
mat <- matrix(runif(1051*325), 1051, 325)
(ld2 <- longData(traj=mat, id=paste("I-", 1:1051, sep=""), time=(1:325)+0.5, varName="Random"))
```

---

LongData-class      ~ Class: LongData ~

---

**Description**

LongData is an objet containing the longitudinal data (the individual trajectories) and some associate value (like time, individual identifiant,...)

**Objects from the Class**

Object LongData can be created either directly by calling the fonction `longData` or form an existing structure (`data.frame` or `matrix`) by using `as.longData`.

**Slots**

`id` [`character`]: Single identifier for each of the longData (each individual).  
`time` [`numeric`]: Time during which measures were made.  
`varName` [`character`]: Name of the variable measured.  
`traj` [`array(numeric)`]: Contains the longitudinal data. Each line corresponds to the trajectory of an individual. The columns refer to the time during which measures were made.  
`other` [`list`]: list of additionnal information (see section Value in `generateArtificialLongData` for an example).

**Construction**

Object LongData can be created either directly by calling the fonction `longData` (build from scratch) or via `as.longData` (turning a `data.frame` into a LongData).

**Get [**

**Object["id"]** [`vecteur(character)`]: Gets each individual identifiant (the value of the slot `id`)  
**Object["time"]** [`vecteur(numeric)`]: Gets the times (the value of the slot `time`)  
**Object["varName"]** [`character`]: Gets the name of the variable (the value of the slot `varName`)  
**Object["traj"]** [`matrix(numeric)`]: Gets all the longData' values (the value of the slot `traj`)

**Set [<-**

- Object["id" <-value]** [vecteur(character)]: Sets each individual indentifiant to value
- Object["time" <-value]** [vecteur(numeric)]: Sets the times to value.
- Object["varName" <-value]** [character]: Sets the name of the variable to value.
- Object["traj" ]** [matrix(numeric)]: Sets all the longData' to values.
- Object["traj" [3,<-value]** [vecteur(numeric)]: Sets the trajectorie of the individual 3 to value.

**Methods**

- `plot(LongData)` display the LongData.
- `plot(LongData,Partition)` display the LongData according to a specific `Partition`.
- `plotSubGroups(LongData,Partition)` display one graph for each clusters, sub-groups by sub-groups according to a specific `Partition`.
- `selecSupTrajMinSize(LongData,minSize)` select the individual trajectories that are made by at least `minSize` values (usefull to exclude from the LongData some individual trajectories with to many missing values.)

**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>

**References**

Article submitted

**See Also**

Overview: [longitudinalData-package](#)  
 Methods: [longData](#), [as.longData](#), [selectSupTrajMinSize](#), [imputation](#), [criterion](#)  
 Plot: [plot\(LongData\)](#), [plotSubGroups\(LongData\)](#)

**Examples**

```
### building longData
mat <- matrix(c(NA,2,3,4,1,6,2,5,1,3,8,10),4)
ld <- new("LongData",id=c("I1","I2","I3","I4"),time=c(2,4,8),varName="Age",traj=mat)

### '[' and '[<-'
ld["id"]
ld["time"]<- c(1,3,9)
ld["varName"]
```

```
ld["traj"]
ld["traj"][3,]<-c(2,7,9)
(ld)

### Plot
plot(ld,type.mean="n",legend=FALSE)

### Only trajectories with at least 3 values
selectSupTrajMinSize(ld,3)
```

---

ordered                      *~ Function: ordered ~*

---

### Description

Given a `Partition`, `ordered` order its clusters letters according to the effectif of each clusters. The is meanly a "label changing".

### Usage

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

### Arguments

`x`                      [`Partition` ]: `Partition` to be ordered.  
`...`                    For compatibility only, no other arguments are accepted.

### Details

Given a `Partition`, `ordered` order its clusters letters according to the effectif of each clusters. So A will be the cluster with be biggest frequency, then B, then C...

### Value

An object of class `Partition`.

### See Also

[Partition](#)

### Examples

```
### Small partition
smallPart <- partition(clusters=c("A", "B", "A", "C", "C", "C"),nbClusters=3)

### Same partition but ordered
ordered(smallPart)
```

---

partition                      *~ Function: partition ~*

---

## Description

partition is the constructor of the class [Partition](#).

## Usage

```
partition(clusters, nbClusters)
```

## Arguments

clusters            `[vector (factor)]`: cluters to which each individual belongs. Each clusters is represented by a letters (upper and lower represent different clusters).

nbClusters        `[numeric]`: number of clusters.

## Details

partition construct a object of class [Partition](#). It does not provide any default values.

## Value

An object of class [Partition](#).

## See Also

[Partition](#), [ordered](#)

## Examples

```
### Empty partition
partition()

### Small partition
partition(clusters=c("A", "B", "A", "C", "C"), nbClusters=3)

### Big random partition
partition(clusters=LETTERS[floor(runif(1000, 1, 5))], nbClusters=5)
```

---

 Partition-class     ~ Class: Partition ~
 

---

### Description

An object of class `Partition` is a clusterization of `LongData` in subgroups.

### Objects from the Class

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

### Slots

`nbClusters` [numeric]: number of groups, between 1 and 52

`clusters` [vector(factor)]: vector containing the affectation groups of each individual. The 26 first groups are in upper-case letters, the 26 next are in lower case letters.

### 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 52 (fifty two clusters maximum).
- `clusters` is a factor in `LETTERS[1:nbCluster]` if `nbClusters` <= 26, in `c(LETTERS, letters[1:(nbClusters-26)])` otherwise).

### Construction

Class `Partition` objects are mainly constructed by some clusterization 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

### Setteur [<-

**Object["nbClusters" <-value]** [numeric]: Sets the number of clusters to value.

**Object["clusters" <-value]** [vector(factor)]: Sets the cluster of each individual.

**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>

**References**

Article submitted

**See Also**

Overview: [longitudinalData-package](#)  
Classes: [LongData](#)  
Methods: [partition](#)

**Examples**

```
### Building Partition
part <- partition(rep(c("A", "B", "C"), time=10), nbClusters=4)

### '[' and '[<-'
part["clusters"]
part["nbClusters"]

part["nbClusters"]<-3
(part)
```

---

pathFrechet

~ *Function: Frechet distance* ~

---

**Description**

Compute Frechet distance and Frechet path between two trajectories.

**Usage**

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

Christophe Genolini

**References**

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

**See Also**

`distFrechet`

**Examples**

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

### Optimized function
pathFrechetR(P,Q)

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

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

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

---

plot, LongData      ~ function: plot for LongData ~

---

### Description

plot the `LongData`, either alone or relatively to a `Partition`. This function can plot the individual trajectories, the mean trajectories or both.

### Usage

```
plot(x,y,...)
```

### Arguments

- |     |   |
|-----|---|
| x   | [LongData]: Object containing the trajectories to plot.   |
| y   | [Partition]: Gives the Partition used to color the trajectories. If y is missing, a Partition with a single cluster is considered.  |
| ... | Graphical parameters to be passed to methods (see plot). For LongData object specifically : <ul style="list-style-type: none"> <li>• subGroups[vector(character)]: which subGroups of the Partition should be plotted ? By default, all are selected.</li> <li>• type[character]: what type of plot should be drawn for the individual trajectories.</li> <li>• type.mean[character]: what type of plot should be drawn for the mean trajectories.</li> <li>• col[character], [numeric] or vector[numeric]: Specification of the plotting color of the individual trajectories. In addition to the standard possibles values, col="clusters" can be use to color the individual trajectories according to their clusters.</li> <li>• col.mean[character], [numeric] or vector[numeric]: Specification of the plotting color of the mean trajectories. In addition to the standard possibles values, col="clusters" can be use to color each mean trajectories according to its clusters.</li> <li>• lty[character]: the line type for the individual trajectories.</li> <li>• lty.mean[character]: the line type for the mean trajectories.</li> <li>• pch[character]: specify the symbol to be used as plotting point on the individual trajectories.</li> </ul> |

- `pch.mean` [character]: specify the symbol to be used as plotting point on the mean trajectories. Option `pch.mean="symbols"` or `pch.mean="letters"` can be used.
- `pch.time` [vector (numeric)]: precise the time at which a point should be plot (usefull if there is a important number of time, see example).
- `cex.mean` [numeric]: The magnification to be used for the symbol used as plotting point for the mean trajectories.
- `legends` [logical]: print the percent of individual in each groups.
- `minSize` [numeric]: minimum number of non missing value that an individual trajectories must contain to be drawn.

### Details

plot the `LongData`, either alone or relatively to a `Partition`. This function can plot the individual trajectories, the mean trajectories or both. Some graphical parameters are available twice, once for the individual trajectories (like `type` or `col`), once for the mean trajectories (like `type.mean` or `col.mean`).

### See Also

[LongData](#), [plotSubGroups \(LongData\)](#)

### Examples

```
#####
### Small example
traj1 <- gald()
part1 <- partition(rep(LETTERS[1:4], each=50), nbClusters=4)

### Several way to print the longData
par(ask=TRUE)
plot(traj1)
plot(traj1, part1)

### Only the longData, in black
plot(traj1, part1, type="n", col=1)

### Only the clusters centers
plot(traj1, part1, type="n")

### only the clusters centers, in black with no letters
plot(traj1, part1, type="n", col.mean="black", type.mean="l")

### Big letter, in black (for publication...)
plot(traj1, part1, type="n", col.mean="black", cex.mean=2)

#####
### Big example
traj2 <- gald(time=0:500, functionClusters=list(
  function(t){5},
```

```

    function(t){-5},
    function(t){5*sin(t/25)}
  ))
part2 <- partition(rep(LETTERS[1:3],each=50),nbClusters=3)
plot(traj2,part2)

### Only the clusters mean
plot(traj2,part2,type="n")

### With not as much letters...
plot(traj2,part2,pch.time=25+0:9*50,cex=2,type="n")
par(ask=FALSE)

```

---

plotSubGroups      ~ Function: plotSubGroups for LongData ~

---

## Description

Plot the trajectories of all the subgroups on several graph (one on each graph) of an object [LongData](#) relatively to a [Partition](#). This function can plot the individual trajectories, the mean trajectories or both.

## Usage

```
plotSubGroups(x, y, ...)
```

## Arguments

- |     |  |
|-----|--|
| x   | [LongData]: Object containing the trajectories to plot.  |
| y   | [Partition]: Give the Partition used to color the individual trajectories. If y is missing, a Partition with a single cluster is considered.   |
| ... | Graphical parameters to be passed to methods (see plot). For LongData object specifically : <ul style="list-style-type: none"> <li>• <code>subGroups[vector(character)]</code>: which subGroups of the Partition should be plotted ? By default, all are selected.</li> <li>• <code>type[character]</code>: what type of plot should be drawn for the individual trajectories.</li> <li>• <code>type.mean[character]</code>: what type of plot should be drawn for the mean trajectories.</li> <li>• <code>col[character], [numeric] or vector[numeric]</code>: Specification of the plotting color of the individual trajectories. In addition to the standard possibles values, <code>col="clusters"</code> can be use to color the individual trajectories according to their clusters.</li> <li>• <code>col.mean[character], [numeric] or vector[numeric]</code>: Specification of the plotting color of the mean trajectories. In addition to the standard possibles values, <code>col="clusters"</code> can be use to color each mean trajectories according to its clusters.</li> </ul> |

- `lty[character]`: the line type for the individual trajectories.
- `lty.mean[character]`: the line type for the mean trajectories.
- `pch[character]`: specify the symbol to be used as plotting point on the individual trajectories.
- `pch.mean[character]`: specify the symbol to be used as plotting point on the mean trajectories. Option `pch.mean="symbols"` or `pch.mean="letters"` can be used.
- `pch.time[vector(numeric)]`: precise the time at which a point should be plot (usefull if there is a important number of time).
- `cex.mean[numeric]`: The magnification to be used for the symbol used as plotting point for the mean trajectories.
- `legends[logical]`: print the percent of individual in each groups.
- `minSize[numeric]`: minimum number of non missing value that an individual trajectories must contain to be drawn.

### Details

Plot the trajectories of all the subgroups on several graph (one on each graph) of an object `LongData` relatively to a `Partition`. This function can plot the individual trajectories, the mean trajectories or both. Some graphical parameters are available twice, once for the individual trajectories (like `type` or `col`), once for the mean trajectories (like `type.mean` or `col.mean`).

### 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>

### See Also

`LongData,plot(LongData)`

### Examples

```
#####
### Building data
ld <- gald()
par(ask=TRUE)

### Default plotting
plotSubGroups(ld)

### Only the trajectories in black
plotSubGroups(ld,col=1,type.mean="n")

### Plot according to a partition
```

```

### (regular plot, then plotSubGroups
part <- partition(clusters=rep(LETTERS[1:4],each=50),nbClusters=4)
plot(ld,part,type.mean="n")
plotSubGroups(ld,part,type.mean="n")

### Only the mean trajectories, with letters (for publication ?)
plotSubGroups(ld,part,type="n",col.mean=1,size=2)

### All at once.
plotSubGroups(ld,part)

par(ask=FALSE)

```

---

```
selectSupTrajMinSize
```

*~ Function: selecSupTrajMinSize ~*

---

## Description

When an individual trajectories contain too many missing value, it might be necessary to exclude it. Give an object of class `LongData`, `selectSupTrajMinSize` select the trajectories that does not contain too many missing value.

## Usage

```
selectSupTrajMinSize(object,minSize)
```

## Arguments

<code>object</code>	[ <code>LongData</code> ]: Object from who some individual trajectories may be exclude
<code>minSize</code>	[ <code>numeric</code> ]: minimum number of non missing value that an individual trajectories must contain to not be exclude.

## Details

Individual trajectories whose values are partially missing can either be excluded or included in a computation. `minSize` sets the minimum number of values that a trajectory must contain to not be excluded. For example, if a trajectories have 7 measurements (`time=7`) and `minSize` is set to 3, the trajectory `(5, 3, NA, 4, NA, NA, NA)` will be include while `(2, NA, NA, NA, 4, NA, NA)` will be exclude. Please note that trajectories that are totally missing (i.e. 0 present values) are always excluded.

## Value

A logical vecteur taking values `TRUE` for the trajectories to include, `FALSE` for the trajectories to exclude.

**See Also**

[LongData](#)

**Examples**

```
### Construction of the LongData
trajA <- as.longData(matrix(c(4, 1, 1, NA, NA, 1, NA, NA, 2, 2, 3, NA), 4))

### Selection of the longData with at least 2 values
selectSupTrajMinSize(trajA, 2)

### Selection of the longData with no missing value
selectSupTrajMinSize(trajA, 3)
```

# Index

- \*Topic **NA**
  - imputation, 13
- \*Topic **aplot**
  - plot, LongData, 23
- \*Topic **classes**
  - LongData-class, 16
  - Partition-class, 20
- \*Topic **classif**
  - LongData-class, 16
  - longitudinalData-package, 2
- \*Topic **cluster**
  - criterion, 4
  - criterion3, 6
  - generateArtificialLongData, 10
  - imputation, 13
  - longData, 15
  - LongData-class, 16
  - longitudinalData-package, 2
  - ordered, 18
  - partition, 19
  - Partition-class, 20
  - plotSubGroups, 25
- \*Topic **datagen**
  - generateArtificialLongData, 10
- \*Topic **dplot**
  - longitudinalData-package, 2
- \*Topic **hplot**
  - plotSubGroups, 25
- \*Topic **methods**
  - as.longData, 3
  - criterion, 4
  - criterion3, 6
  - imputation, 13
  - longData, 15
  - ordered, 18
  - partition, 19
  - plotSubGroups, 25
  - selectSupTrajMinSize, 27
- \*Topic **package**
  - as.longData, 3
  - criterion, 4
  - criterion3, 6
  - imputation, 13
  - longData, 15
  - longitudinalData-package, 2
  - plot, LongData, 23
  - selectSupTrajMinSize, 27
- \*Topic **ts**
  - as.longData, 3
  - criterion, 4
  - criterion3, 6
  - generateArtificialLongData, 10
  - imputation, 13
  - longData, 15
  - LongData-class, 16
  - longitudinalData-package, 2
  - ordered, 18
  - partition, 19
  - Partition-class, 20
  - plot, LongData, 23
  - plotSubGroups, 25
  - selectSupTrajMinSize, 27
  - [, LongData-method
    - (LongData-class), 16
  - [, Partition-method
    - (Partition-class), 20
  - [<-, LongData-method
    - (LongData-class), 16
  - [<-, Partition-method
    - (Partition-class), 20
  - as.longData, 2, 3, 12, 16–18
  - as.longData, data.frame-method
    - (as.longData), 3
  - as.longData, matrix-method
    - (as.longData), 3

- criterion, 2, 3, 4, 7, 14, 18
- criterion, LongData, Partition-method  
(*criterion*), 4
- criterion, matrix, Partition-method  
(*criterion*), 4
- criterion3, 5, 6
- criterion3, LongData, Partition-method  
(*criterion3*), 6
- criterion3, matrix, Partition-method  
(*criterion3*), 6
  
- dist, 9, 10
- distFrechet, 7
- distFrechetR(*distFrechet*), 7
- distFrechetRec(*distFrechet*), 7
- distTraj, 9
  
- gald  
(*generateArtificialLongData*),  
10
- generateArtificialLongData, 10, 17
  
- imputation, 2, 3, 5–7, 13, 18
- imputation, LongData-method  
(*imputation*), 13
- imputation, matrix-method  
(*imputation*), 13
  
- LongData, 2–7, 10, 12–16, 20, 21, 23–28
- LongData (*LongData-class*), 16
- longData, 2, 4, 12, 15, 16–18
- longData, ANY, ANY, ANY, ANY-method  
(*longData*), 15
- longData, missing, missing, missing, missing, missing-method  
(*longData*), 15
- LongData-class, 16
- longitudinalData-package, 18, 21
- longitudinalData-package, 2
  
- ordered, 2, 18, 20
- ordered, Partition (*ordered*), 18
- ordered, Partition-method  
(*ordered*), 18
  
- Partition, 2, 4–7, 14, 17, 19, 20, 23–26
- partition, 2, 19, 21
- partition, ANY, ANY-method  
(*partition*), 19
- partition, missing, missing-method  
(*partition*), 19
  
- Partition-class, 20
- pathFrechet, 22
- pathFrechetR(*pathFrechet*), 22
- plot, 2, 12, 16
- plot (*plot*, LongData), 23
- plot (LongData), 3, 17, 18, 27
- plot (LongData, Partition), 17
- plot, LongData, 23
- plot, LongData, ANY-method  
(*plot*, LongData), 23
- plot, LongData, missing-method  
(*plot*, LongData), 23
- plotSubGroups, 2, 25
- plotSubGroups (LongData), 3, 18, 24
- plotSubGroups (LongData, Partition),  
17
- plotSubGroups, LongData  
(*plotSubGroups*), 25
- plotSubGroups, LongData-method  
(*plotSubGroups*), 25
- plotTraj (*plot*, LongData), 23
- plotTraj, LongData  
(*plot*, LongData), 23
- plotTraj, LongData-method  
(*plot*, LongData), 23
  
- selecSupTrajMinSize (LongData, minSize),  
17
- selectSupTrajMinSize, 2, 18, 27
- selectSupTrajMinSize, LongData-method  
(*selectSupTrajMinSize*), 27
- show, LongData-method  
(*LongData-class*), 16
- show, Partition-method  
(*Partition-class*), 20