Package ‘longitudinalData’

February 9, 2023

Type Package
Title Longitudinal Data
Version 2.4.5
Date 2023-02-09
Description Tools for longitudinal data and joint longitudinal data (used by packages kml and kml3d).
License GPL (>= 2)
LazyData yes
Depends methods,clv.class,rgl,utils,misc3d
URL http://www.r-project.org
Collate global.r function.r constants.r myMisc3d.r longData.r
   longData3d.r distanceFrechet.R imputCross.R imputTraj.R
   imputLinearInterpol.R imputCopyMean.R imputation.r partition.r
   listPartition.r parLongData.r parWindows.r newPlot.r
Encoding latin1
NeedsCompilation no
Author Christophe Genolini [cre, aut],
   Bruno Falissard [ctb],
   Dai Fang [ctb],
   Patrice Kiener [ctb],
   Luke Tierney [ctb]
Maintainer Christophe Genolini <christophe.genolini@u-paris10.fr>
Repository CRAN
Date/Publication 2023-02-09 20:40:02 UTC

R topics documented:

   longitudinalData-package .............................................. 2
   artificialJointLongData ............................................. 4
   artificialLongData .................................................. 5
   Constants ............................................................ 6
   distFrechet .......................................................... 7
### longitudinalData-package

#### Description

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

Package: longitudinalData
Type: Package
Version: 2.4.1
Date: 2016-02-02
License: GPL (>= 2)
LazyData: yes
Depends: methods, clv, rgl, misc3d
URL: http://www.r-project.org

Overview

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

1. plotTrajMeans
2. imputation
3. qualityCriterion

Author

Christophe Genolini
1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France
2. CeRSM, 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: plotTrajMeans, plotTrajMeans3d
Imputation: imputation
criterion: qualityCriterion
Examples

```r
### Generation of artificial longData
data(artificialJointLongData)
myData <- longData3d(artificialJointLongData, timeInData=list(var1=2:12, var2=13:23, var3=24:34))

part <- partition(rep(1:3, each=50))
plotTrajMeans3d(myData, part)

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

---

**artificialJointLongData**

~ *Data: artificialJointLongData ~

**Description**

Some artificial joint longitudinal data.

**Usage**

data(artificialJointLongData)

**Format**

Some joint longitudinal data in wide format. It includes 90 trajectories divided in 3 groups.

- **id** unique identifier for each patient.
- **v0** Measurement of variable 'V' at time t0
- **v1** Measurement of variable 'V' at time t1
- **...**
- **v10** Measurement of variable 'V' at time t0
- **w0** Measurement of variable 'W' at time t0
- **w1** Measurement of variable 'W' at time t1
- **...**
- **w10** Measurement of variable 'W' at time t0
- **x0** Measurement of variable 'X' at time t0
- **x1** Measurement of variable 'X' at time t1
- **...**
- **x10** Measurement of variable 'X' at time t0

**Details**

Some joint longitudinal data in wide format. It includes 90 trajectories divided in 3 groups.
artificialLongData

Author(s)
Christophe Genolini

Examples
data(artificialLongData)
str(artificialLongData)

---

Description
Some artificial longitudinal data.

Usage
data(artificialLongData)

Format
Some longitudinal data in wide format. It includes 120 trajectories divided in 4 groups.

id unique identifier for each patient.
t0 Measurement at time t0
t1 Measurement at time t1
... ...
t10 Measurement at time t10

Details
Some artificial longitudinal data in wide format. It includes 120 trajectories divided in 4 groups.

Author(s)
Christophe Genolini

Examples
data(artificialLongData)
str(artificialLongData)
**Constants**

### Description

Constants define in the package ~

### Usage

```r
MAX_CLUSTERS

CRITERION_NAMES
DISTANCE_METHODS
CHOICE_STYLE
```

### Value

```r
MAX_CLUSTERS = 26
CLUSTER_NAMES = paste("c",2:MAX_CLUSTERS,sep="")

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","b","b","l","l","l","n"),
  colMean=c("clusters","black","clusters","black","clusters","black","black"),
  pchMean=c("letters","letters","symbols","symbols","letters","letters","letters")
)
```

### Examples

```r
### 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(Px, Py, Qx, Qy, timeScale=0.1, FrechetSumOrMax = "max")

**Arguments**

- **Px** [vector(numeric)] Times (abscisse) of the first trajectories.
- **Py** [vector(numeric)] Values of the first trajectories.
- **Qx** [vector(numeric)] Times of the second trajectories.
- **Qy** [vector(numeric)] Values of the second trajectories.
- **timeScale** [numeric]: allow to modify the time scale, increasing or decreasing the cost of the horizontal shift. If timeScale is very big, then the Frechet's distance is equal to the euclidienne distance. If timeScale is very small, then it is equal to the Dynamic Time Warping.
- **FrechetSumOrMax** [character]: The Frechet's distance can be define using the 'sum' function or the 'max' function. This option let the user to chose one or the other.

**Details**

Given two curve P and Q, 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.

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 distFrechet is C compiled, the function distFrechetR is in R, the function distFrechetRec is in recursive (the slowest) in R.

**Value**

A numeric value.
Author

Christophe Genolini
1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France
2. CeRSM, EA 2931, UFR STAPS, Université de Paris Ouest-Nanterre-La Défense / Nanterre / France

References

[1] Thomas Eiter & Heikki Mannila:
"Computing Discrete Fréchet Distance"

"KmL: k-means for longitudinal data"
Computational Statistics, vol 25(2), pp 317-328, 2010

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also
distTraj

Examples

```r
Px <- 1:20
Py <- dnorm(1:20,12,2)
Qx <- 1:20
Qy <- dnorm(1:20,8,2)
distFrechet(Px,Py,Qx,Qy)
```

```r
### Frechet using sum instead of max.
distFrechet(Px,Py,Qx,Qy,FrechetSumOrMax="sum")
```

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 cannot be directly used by graphical functions plotTrajMeans and plotTrajMeans3d. This function modifies these values to make them fit with plotTrajMeans and plotTrajMeans3d expectations.

The field col and pch are the ones concerned 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 cluster. Then the field col receive a vector of color such that each individual (if y is a Partition) or each cluster (if y is a number of clusters) get its corresponding color.

If pch='letters', a letter is affected to each cluster. Then the field pch receive a vector of letters such that each individual (if y is a Partition) or each cluster (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. CeRSEM, 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

"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)

---

**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="globalMin",upperBound="globalMax")

**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 [3]. 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)
  – Intermittant: values immediately surrounding the missing are join by a line.
  – Monotone: imputed by 'locf' or 'nocb'.

• 'linearInterpol.global' (linear interpolation, global slope)
  – Intermittant: values immediately 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 average progression of the actual individual trajectory). Missing-value at start and at end are chosen on this line.

• 'linearInterpol.local' (linear interpolation, global slope)
  – Intermittant: values immediately 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)
  – Intermittant: values immediately 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 [3] 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 [3] 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 [3] 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 [3] and examples’ section.
• locf (Last Occurrence Carried Forward) THIS METHOD HAS BEEN PROVEN TO NOT BE EFFICIENT SEVERAL TIMES BY VARIOUS AUTHOR, we strongly recommend not to use it!
  – Intermittent and monotone at end: the previous non-missing value is duplicated forward.
  – Monotone at start: the first non-missing value is duplicated backward (nocb).

• nocb (Next Occurrence Carried Backward) THIS METHOD HAS BEEN PROVEN TO NOT BE EFFICIENT SEVERAL TIMES BY VARIOUS AUTHOR, we strongly recommend not to use it!
  – Intermittent and monotone at start: the next non-missing value is duplicated backward.
  – Monotone at end: the last non-missing value is duplicated forward (locf).

• 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 chosen) 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 chosen) 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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

[3] Christophe Genolini, René Écochard and Hélène Jacqmin-Gadda
"Copy Mean: A New Method to Impute Intermittent Missing Values in Longitudinal Studies"
Open Journal of Statistics, vol 3(26), 2013
See Also

LongData, Partition, qualityCriterion

Examples

# # # # # # # # # # # # # # # # # # # # # # # # # #
### Preparation of the data
par(ask=TRUE)
timeV <- 1:14

matMissing <- matrix(
  c(NA, NA, NA, NA, NA, NA, 18, 22, NA, 24, 22, NA, NA, NA, NA, 18, 22, NA, NA, NA, 24, 21, 24, 26, 27, 32, 30, 22, 26, 26, 28, 24, 23, 21, 14, 13, 10, 8, 7, 18, 16, 8, 12, 6, 10, 10, 9, 7, 3, 1, 1, 1, 3, 9, 7, -1, 3, 2, 4, 1, 0, -2), 4, byrow=TRUE)

matplot(t(matMissing), col=c(2,1,1,1), lty=1, type="l", lwd=c(3,1,1,1), pch=16, xlab="Black=trajectories; Green=mean trajectory\nRed=trajectory to impute", ylab="", main="Four trajectories")
moy <- apply(matMissing, 2, mean, na.rm=TRUE)
lines(moy, col=3, lwd=3)

# # # # # # # # # # # # # # # # # # # # # # # # # #
### Methods using cross sectionnal information (cross-methods)
par(mfrow=c(1,3))

mat2 <- 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)

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

### crossMedian
matplot(t(imputation(mat2,"crossMedian")), type="l", ylim=c(0,10), lty=1, col=1, main="crossMedian")
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
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\nDO 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="linearInterpol.locf")
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="linearInterpol.global")
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="linearInterpol.local")
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="linearInterpol.bisector")
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.bisector")
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 trajectory\nFull red=trajectory to impute",ylab="",main="Method 'crossMean'")
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 trajectory\nFull red=trajectory to impute",ylab="",main="Method 'crossMedian'")
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 trajectory\nFull red=trajectory to impute",ylab="",main="Method 'crossHotDeck'")
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")
imputation

```r
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 provides different ways of setting the initial partition for an EM algorithm.

**Usage**

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

**Arguments**

- `nbClusters` [numeric]: number of clusters of that the initial partition should have.
- `lengthPart` [numeric]: number of individuals in the partition.
- `method` [character]: one of "randomAll", "randomK", "maxDist", "kmeans++", "kmeans+", "kmeans-" or "kmeans-".
- `data` [matrix]: data is the matrix of the individuals (useful for the methods that need to compute distance between individuals). 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 Expectation 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 individuals are randomly assigned to a cluster with at least one individual in each cluster.
- **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 former 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(dist(IND_i,IND_t))))). This method is efficient but time consuming.
- **kmeans++**: see [3]
- **kmeans+, kmeans-, kmeans-**: experimental methods derived from [3].

**Value**

`vecteur` of numeric.
# initializePartition

**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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

"k-means++: the advantages of careful seeding"

**Examples**

```r
par(ask=TRUE)
########################
### Constrution of some longitudinal data
data(artificialLongData)
dn <- longData(artificialLongData)
plotTrajMeans(dn)

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

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

########################
### partition using maxDist
pa3 <- initializePartition(3,lengthPart=200,method="maxDist",data=dn["traj"])
plotTrajMeans(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

```r
point <- matrix(c(0, 0, 0, 1, -1, 0, 0, -1, 1, 0), 5, byrow=TRUE)
points <- rbind(point, t(point)+c(10, 0), t(point)+c(5, 6))
points <- rbind(points, t(points)+c(30, 0), t(points)+c(15, 20), t(-t(point)+c(20, 10))

plot(points, main="Some points")

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

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)
lines(points[!is.na(paInit),], col=2, type="p", pch=16)

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)
lines(points[!is.na(paInit),], col=2, type="p", pch=16)

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 list 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.
**Construction**

Class ListPartition objects are mainly constructed by km1. Nevertheless, it is also possible to construct them from scratch using the function `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.
ListPartition-class

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^1,2
1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France
2. CeRSM, 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(artificialLongData)
traj <- as.matrix(artificialLongData[,-1])

### Some clustering
part2 <- partition(rep(c("A","B"),time=100),traj)
part3 <- partition(rep(c("A","B","C","A"),time=50),traj)
part3b <- partition(rep(c("A","B","C","B","C"),time=40),traj)
part4 <- partition(rep(c("A","B","A","C","D"),time=40),traj)

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

listPart["add"] <- part2
listPart["add"] <- part3
listPart["add"] <- part3b
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 distinguised:

- **traj is an array**: lines are individual. Column are time of measurement.
  
  If idAll is missing, the individuels 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

"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"))`
`plotTrajMeans(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
```r
(lid3 <- longData(dn,maxNA=1))
```

---

**LongData-class** ~ **Class: LongData** ~

**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 function `scale`, the 'scaling parameters' (probably mean and standard deviation) are saved in reverse. This is usefull to restore 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.

restoreRealData restore original data that have been modified after a scaling operation.

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

plotTrajMeans plot all the variables of the LongData, optionnaly according to a Partition.

plotTrajMeans3d plot two variables 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 differeents 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
longData3d

References

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

"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: plotTrajMeans, plotTrajMeans3d, 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
plotTrajMeans(ld,parMean=parMEAN(type="n"))

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 should 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 constructs an 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 different 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 containing 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

"KmL: A package to cluster longitudinal data"
LongData3d-class

Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

LongData

Examples

### From array

```r
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")))
plotTrajMeans3d(ld1)
```

### From data.frame

```r
dn <- data.frame(id=1:3,v1=c(2,2,1),t1=c(20,21,22),v2=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))))
plotTrajMeans3d(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). 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 [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.
LongData3d-class

**dimTraj** [vector3(numeric)]: size of the array traj (ie dimTraj=[length(idFewNA),length(time),length(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 restore 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 identifier (the value of the slot idAll)

`Object["idFewNA"]` [vecteur(character)]: Gets the list of individual identifier 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.

`restoreRealData` restore original data that have been modified after a scaling operation.

`longDataFrom3d` Create a LongData by extracting a single variable trajectory form a dataset of joint variable-trajectories.

`plotTrajMeans` plot all the variable of the LongData3d, optionnaly according to a Partition.

`plotTrajMeans3d` plot two 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 differents 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

"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: plotTrajMeans, plotTrajMeans3d, 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))
(l1d <- longData3d(dn,timeInData=list(Vir=c(2,4,NA),Tes=c(3,5,6))))

### Scaling
scale(l1d)
(l1d)

### Plotting
plotTrajMeans3d(l1d)
restoreRealData(l1d)

longDataFrom3d ~ Function: longDataFrom3d ~

Description
Extract a single variable-trajectory from an object LongData that contain some joint-trajectories.

Usage
longDataFrom3d(xLongData3d, variable)
Arguments

- xLongData3d: Structure containing some joint-trajectories.
- variable: Either the name of one of the variables of xLongData3d, or its number.

Details

Extract a single variable-trajectory from an object LongData3d that contains 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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

- LongData

Examples

```r
### 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))
```
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 containning 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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

LongData
Examples

```r
### 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 includes 3D objects into PDF documents.

**Usage**

```r
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 includes 3D objects into PDF documents with PDF-1.5/1.6 compatibility.

**Value**

A LaTeX file, in the current directory.

**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

"KmL: A package to cluster longitudinal data"
ordered(ListPartition)

Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

makeTriangles, plot3dPdf, saveTrianglesAsASY.

Examples

### Move to tempdir

```r
wd <- getwd()
setwd(tempdir()); getwd()
```

### Generating the data

```r
data(artificialJointLongData)
myLd <- longData3d(artificialJointLongData, timeInData=list(var1=2:12, var2=13:23))
part <- partition(rep(1:3, each=50))
plotTrajMeans3d(myLd, part)
```

### Creation of the scene

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

### Export in `.asy' file

```r
saveTrianglesAsASY(scene)
```

### Creation of a `.prc' file

```r
# Open a console, then run:
# 'asy -inlineimage -tex pdflatex scene.asy'
```

### Creation of the LaTeX main document

```r
makeLatexFile()
```

### Creation of the `.pdf'

```r
# Open a console window, then run
# pdfLatex main.tex
```

### Go back to current dir

```r
setwd(wd)
```

---

ordered(ListPartition)

~ Function: ordered(ListPartition) ~

Description

Sort the Partition of a ListPartition according to a quality criterion.

Usage

```r
ordered(x, ...)
```
ordered(ListPartition)

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 used 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(artificialLongData)
traj <- as.matrix(artificialLongData[,-1])

####### Some clustering
part2 <- partition(rep(c("A","B"),time=100),traj)
part3 <- partition(rep(c("A","B","C","A"),time=50),traj)
part3b <- partition(rep(c("A","B","C","B"),time=50),traj)
part4 <- partition(rep(c("A","B","C","D"),time=50),traj)

####### ListPartition
listPart <- listPartition()
listPart['criterionActif'] <-"Davies.Bouldin"
plotCriterion(listPart)
```
```r
listPart["add"] <- part2
drop(listPart["add"]) <- part2
listPart["add"] <- part3
drop(listPart["add"]) <- part3
listPart["add"] <- part3b
drop(listPart["add"]) <- part3b
listPart["add"] <- part4
drop(listPart["add"]) <- part4
listPart["add"] <- part3
listPart["add"] <- part3b
drop(listPart["add"]) <- part3b

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

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

---

**parLongData**

~ Function: `parLongData`, `parTraj` and `parMean`~

**Description**

`parLongData`, `parTraj` and `parMean` are constructors for the class `ParLongData`.

**Usage**

```r
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 apart, '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 `plotTrajMeans` for details).
parLongData

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
plotTrajMeans3d(ld2)
plotTrajMeans3d(ld2,parTraj=parTRAJ())
### Example with default value except for the color

```r
plotTrajMeans3d(lld2,parTray=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 apart, 'o' for overplot, 'h' for histogram, 's' and 'S' for steps, 'n' for no ploting)
- **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 *plotTrajMeans* 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* creates an object from scratch;
2. *parTray* creates an object containing default value to plot individual trajectories;
3. *parMean* creates 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 Soenn / Paris

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

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

Examples

```r
### 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

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

Arguments

- `clusters` [vector(factor)]: clusters 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(algoritm="kmeans",convergenceTime="6",otherInfo="WhateverIWant")`.

Details

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

Author

Christophe Genolini
1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France
2. CeRSM, 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

```r
### 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(artificialLongData)
traj <- as.matrix(artificialLongData[,1])
partition(clusters=rep(1:4,each=50),traj)

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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011
### Building Partition

#### Number

```r
part <- partition(rep(c(1,2,1,3),time=3))
```

#### LETTERS

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

#### Others don't work

```r
try(partition(rep(c("A","Bb","C"),time=3)))
```

### Setteur and Getteur

#### '[

```r
part["clusters"]
part["clustersAsInteger"]
part["nbClusters"]
```

#### '[<-'

```r
part["multiplicity"] <- 2
(part)
```

---

**parWindows**

~ Function: parWindows ~

---

**Description**

`parWindows` is the constructor of object `ParWindows`.

**Usage**

```r
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 graphs 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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

Examples

### Building ParWindows

```r
(paramWin <- parWindows(3,2,FALSE,TRUE))
```

### Get

```r
figsScreen <- paramWin['screenMatrix']
```

### Usage

```r
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])
```
ParWindows-class

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 automaticaly 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
plot3dPdf

References

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

"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...

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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011
### Function: plotAllCriterion

This function graphically displays the quality criterion of all the **Partition** of a **ListPartition** object.

#### Usage

```r
plotAllCriterion(x, criterion=CRITERION_NAMES[1:5], standardized = TRUE)
```
plotCriterion

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.
standardized [logical]: If standardized=TRUE, all the criterion will be mapped into [0,1]. This makes them more easily comparable.

Details

This function displays graphically several quality criterion, probably to decide the best clusters’ number.

Value

No value are return. A graph is printed.

Examples

###########
### Data generation
data(artificialLongData)
traj <- as.matrix(artificialLongData[, -1])

### Some clustering
listPart <- listPartition()
listPart["add"] <- partition(rep(c("A", "B"), time=100), traj)
listPart["add"] <- partition(rep(c("A", "B", "B", "B"), time=50), traj)
listPart["add"] <- partition(rep(c("A", "B", "C", "A"), time=50), traj)
listPart["add"] <- partition(rep(c("A", "B", "C", "D"), time=50), traj)
ordered(listPart)

################
### graphical display
plotAllCriterion(listPart)
plotAllCriterion(listPart, criterion=CRITERION_NAMES[1:5], TRUE)

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)
**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.

**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**

```r
### Data generation
data(artificialLongData)
traj <- as.matrix(artificialLongData[, -1])

### Some clustering
listPart <- listPartition()
listPart["add"] <- partition(rep(c("A", "B"), time=100), traj)
listPart["add"] <- partition(rep(c("A", "B", "B", "B"), time=50), traj)
listPart["add"] <- partition(rep(c("A", "B", "C", "A"), time=50), traj)
listPart["add"] <- partition(rep(c("A", "B", "C", "D"), time=50), traj)
ordered(listPart)

### graphical display
plotCriterion(listPart)
plotAllCriterion(listPart, criterion=CRITERION_NAMES[1:5], TRUE)
```

**Description**

Plot the LongData or LongData3d optionnaly relatively to a Partition. For joint trajectories, one graphe for each variable trajectory is displayed.
plotTrajMeans(x, y, parTraj=parTRAJ(), parMean=parMEAN(),...)

Arguments

x [LongData] or [LongData3d]: Object containing the trajectories to plot.
y [numeric]: Partition that will be used 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 cluster (only when y is non-missing). See ParLongData and examples for details.

Details

Plot either a LongData, or each variable of a LongData3d optionally 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. CeRSM, 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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

LongData, LongData3d, plotTrajMeans3d.
Examples

# Construction of the data
data(artificialLongData)
ld <- longData(artificialJointLongData)
part <- partition(rep(1:3,each=50))

# Basic plotting
plotTrajMeans(ld)
plotTrajMeans(ld,part,xlab="Time")

# Changing graphical parameters 'par'

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

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

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

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

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

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

# single variable trajectory

data(artificialLongData)
ld2 <- longData(artificialLongData)
part2 <- partition(rep(1:4,each=50))
plotTrajMeans(ld2)
plotTrajMeans(ld2,part2)
Description

Plot two variables of a LongData3d object in 3D, optionally relatively to a Partition.

Usage

plotTrajMeans3d(x, y, varY = 1, varZ = 2, 
                 parTraj = parTraj(), parMean = parMean(type = "n"), ...)

Arguments

x [LongData3d]: Object containing the trajectories to plot.
y [Partition]: Partition that will be used 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.
...

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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011
### 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)))
part <- partition(rep(1:6,20))

### Basic plotting
plotTrajMeans3d(ld)
plotTrajMeans3d(ld,part)

### Variable 1 and 3, then 2 and 3
plotTrajMeans3d(ld,part)
plotTrajMeans3d(ld,part,varY=3,varZ=2)
plotTrajMeans3d(ld,part,varY=1,varZ=3)

### Changing graphical parameters 'par'

### Color individual according to its clusters (col="clusters")
plotTrajMeans3d(ld,part,parTraj=parTRAJ(col="clusters"))
plotTrajMeans3d(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")
plotTrajMeans3d(ld,part,parTraj=parTRAJ(col="clusters"),parMean=parMEAN(type="n"),nbSample=10)

### Description

Given a LongData and a Partition, the fonction qualityCriterion calculate some qualities criterion.
Usage

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

Arguments

- `traj` [LongData] or [matrix]: object containing the trajectories on which the criterion is calculate.
- `clusters` [Paritition] 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 `imputationMethod` using the method method.

Details

Given a `LongData` and a `Partition` (or a matrix and a vector of integer), the function `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 unavaiable.

Basicaly, 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 these criterion.

But the user should alway keep in mind that these criterion are valid ONLY under the hypothesis of normality. If this hypothèsis 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 (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: \( c(k) = \frac{\text{Trace}(B)}{\text{Trace}(W)} \times \frac{n-k}{k-1} \).
- `Calinski.Harabatz2[numeric]`: Calinski and Harabatz criterion modified by Krysczuk: \( c(k) = \frac{\text{Trace}(B)}{\text{Trace}(W)} \times \frac{n-1}{n-k} \).
- `Calinski.Harabatz3[numeric]`: Calinski and Harabatz criterion modified by Genolini: \( g(k) = \frac{\text{Trace}(B)}{\text{Trace}(W)} \times \frac{n-1}{\sqrt{k-1}} \).
• Ray.Turi\[\text{numeric}\]: Ray and Turi criterion: \( r(k) = \frac{-\text{Vintra}}{\text{Vinter}} \) with \( \text{Vintra} = \text{Sum} (\text{dist}(x, \text{center}(x))) \) and \( \text{Vinter} = \text{min} (\text{dist}(\text{center}_i, \text{center}_j)^2) \). (The "true" index of Ray and Turi is \( \frac{\text{Vintra}}{\text{Vinter}} \) and should be minimized. See IMPORTANT NOTE above.)

• Davies.Bouldin\[\text{numeric}\]: Davies and Bouldin criterion: \( d(k) = -\text{mean}(\text{Proximite(cluster}_i,\text{cluster}_j)) \) with \( \text{Proximite}(i,j) = \frac{(\text{DistInterne}(i)+\text{DistInterne}(j))}{(\text{DistExterne}(i,j))} \). (The "true" index of Davies and Bouldin is \( \text{mean}(\text{Proximite}()) \) and should be minimized. See IMPORTANT NOTE above.)

• random\[\text{numeric}\]: random value following the normal law \( N(0,1) \).

**Parametric criterion**

All the parametric indices should be minimized. So the function qualityCriterion compute their opposite (see IMPORTANT NOTE above.)

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

SECOND IMPORTANT NOTE: the formula of parametrics criterion often include the size of the population. In the specific case on longitudinal data, the definition of the "size of the population" is not obvious. It can be either the number of individual \( n \), or the number of measurement \( N = n.t \). So, the function qualityCriterion gives two version of all the non parametrics criterion, the first using \( n \), the second using \( N \).

• BIC\[\text{numeric}\]: Bayesian Information Criterion: \( \text{BIC} = 2*\text{log}(L) - h*\text{log}(n) \). See IMPORTANT NOTE above.

• BIC2\[\text{numeric}\]: Bayesian Information Criterion: \( \text{BIC} = 2*\text{log}(L) - h*\text{log}(N) \). See IMPORTANT NOTE above.

• AIC\[\text{numeric}\]: Akaike Information Criterion, bis: \( \text{AIC} = 2*\text{log}(L) - 2*h \). See IMPORTANT NOTE above.

• AICc\[\text{numeric}\]: Akaike Information Criterion with correction: \( \text{AICc} = \frac{\text{AIC} + (2h(h+1))/(n-h-1)}{n-h} \). See IMPORTANT NOTE above.

• AICc2\[\text{numeric}\]: Akaike Information Criterion with correction, bis: \( \text{AICc2} = \frac{\text{AIC} + (2h(h+1))/(n-h-1)}{n-h-1} \). See IMPORTANT NOTE above.

**Author**

Christophe Genolini
1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France
2. CeRSM, 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

"KmL: A package to cluster longitudinal data"
See Also

LongData, Partition, imputation.

Examples

# Preparation of some artificial data
par(ask=TRUE)
data(artificialLongData)
ld <- longData(artificialLongData)

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

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

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

# Comparisons of the Partition
plot(c(cr1[[1]],cr2[[1]],cr3[[1]]),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 clustering algorithm can find a Partition several times. It is stored several times in object ListPartition (or in ClusterLongData or in ClusterLongData3d), encombering the memory. regroup removes the duplicate Partition. Note that if the ListPartition is not ordered, then regroup sorts it unless toOrder=FALSE.

Value

None (this function changes 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(artificialLongData)
myLd <- as.matrix(artificialLongData[,-1])
### Some clustering
part2 <- partition(rep(c("A","B","A","C"),time=50),myLd)
part3 <- partition(rep(c("A","B","C","D"),time=50),myLd)

################
### ListPartition
listPart <- listPartition()
listPart["add"] <- part2
listPart["add"] <- part3
listPart["add"] <- part2
listPart["add"] <- part3

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

---

**reshapeLongToWide**

~ Function: longToWide (or reshapeWide) ~

**Description**

This function reshapes a data frame in 'long' format (repeated measurements in the same column) into a data frame in 'wide' format (repeated measurements in separate columns). It also corrects a bug of reshape.

**Usage**

```r
longToWide(trajLong)
reshapeLongToWide(trajLong)
```

**Arguments**

- `trajLong` ([data.frame]): data.frame that hold the trajectories in long format. The data.frame has to be (no choice!) in the following format: the first column should be the individual indentifiant. The second should be the times at which the measurement are made and should be numeric. The third one should be the measurement.

**Details**

This function reshapes a data frame in 'long' format (repeated measurements in the same column) into a data frame in 'wide' format (repeated measurements in separate columns).

**Value**

A data frame in 'wide' format (repeated measurements in separate columns).

**Note**

`longToWide` is just a 'friendly overlay' of the function `reshape`. It also corrects a reshape bug (modification of the order of some trajectories value when some times are missing).

**Author(s)**

Christophe Genolini

**See Also**

`wideToLong`, `reshape`. 
Examples

summary(Indometh)
longToWide(Indometh)

df2 <- data.frame(id = rep(LETTERS[1:4], rep(2,4)),
  visit = I(rep(c("3","6"), 4)),
  x = rnorm(4), y = runif(4),
  sex=rep(c("H","F","H"),time=c(4,2,2)))[1:7,]
longToWide(df2[,1:3])
longToWide(df2[,c(1,2,4)])

Description

This function reshapes a data frame in 'wide' format (repeated measurements in separate column) into a data frame in 'long' format (repeated measurements in the same columns).

Usage

wideToLong(trajWide,times=1:(ncol(trajWide)-1))
reshapeWideToLong(trajWide,times=1:(ncol(trajWide)-1))

Arguments

  trajWide  [data.frame]: a data frame in 'wide' format (repeated measurements in separate column). The first column has to be the individual identifier. All the other column should be the trajectories.
  times     [vector(numeric)]: specification of the times at which the longitudinal data have been measured (like ages, year, month). If times is missing, it takes the value 1:(ncol(trajWide)-1).

Details

This function reshapes a data frame in 'wide' format (repeated measurements in separate column) into a data frame in 'long' format (repeated measurements in the same columns). The first column has to be the individual identifier. All the other column should be the trajectories. The missing values are removed in long format.

Value

A data frame in 'long' format.

Author(s)

Christophe Genolini
See Also

longToWide, reshape.

Examples

df3 <- data.frame(id = LETTERS[rep(1:4)], sex=c("H","F","H","F"),
                  v1=rnorm(4), v2=rnorm(4), w1=rnorm(4), w2=rnorm(4))

wideToLong(df3[,c(1,3,4)])
wideToLong(df3[,c(1,5,6)])
wideToLong(df3[,c(1,3:6)])
wideToLong(df3[,c(1,3:6)], times=c(1,2,4,8))

Description

This function revert the effect of scale by restauring the initial values of trajectories.

Usage

restoreRealData(object)

Arguments

object [LongData]: Object containning trajectories to restore.

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
saveTrianglesAsASY

References

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

"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)))
  plotTrajMeans3d(ld1)

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

  scale(ld1,scale=c(-1,-1))
  plotTrajMeans3d(ld1)

#------------------------------------------------------------------------------
### Back to the first version of the data

  restoreRealData(ld1)
  plotTrajMeans3d(ld1)

saveTrianglesAsASY  ~ Function: saveTrianglesAsASY ~

Description

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

Usage

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

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


See Also

plot3dPdf, makeLatexFile, makeTriangles

Examples

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

### Generating the data
data(artificialJointLongData)
myLd <- longData3d(artificialJointLongData,timeInData=list(var1=2:12,var2=13:23))
part <- partition(rep(1:3,each=50))
plotTrajMeans3d(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

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

---

**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 containing trajectories to be scaled.
- `center`: [logical] or [vector(numeric)]: Value that will be subtracted from each measurement of a variable. If `center=TRUE`, the mean of each variable-trajectory is used. Otherwise, `center` should have a value for each variable.
- `scale`: [logical] or [vector(numeric)]: Value that will be divided, after the subtraction, each measurement of a variable. If `scale=TRUE`, the standard deviation of each variable-trajectory is used. Otherwise, `scale` should have a value for each variable.

**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 scaled 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=mean(x[,k],na.rm=TRUE)`. Similarly, `scale=TRUE` is a special value that set `s_k=sd(x[,k],na.rm=TRUE)`.

**Value**

`scale` directly modify the internal value of the LongData. No value is returned.
Author

Christophe Genolini

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

"KmL: A package to cluster longitudinal data"
Computer Methods and Programs in Biomedicine, 104, pp e112-121, 2011

See Also

restoreRealData

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)))
plotTrajMeans3d(ld1)

############################
### Scaling by 'mean' and 'standard deviation'
plotTrajMeans3d(ld1)
scale(ld1)
plotTrajMeans3d(ld1)

############################
### Scaling by some parameters
scale(ld1,center=c(10,100),scale=c(3,-1))
plotTrajMeans3d(ld1)

############################
### To restore the data
restoreRealData(ld1)
Description

windowsCut prepare an object `ParWindows` according to its arguments.

Usage

```r
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 `addLegend` is 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 <- ceiling(sqrt(x))` and `nbRow <- ceiling(x/nbCol)`.

Value

An object of class `ParWindows`.

Author

Christophe Genolini
1. UMR U1027, INSERM, Université Paul Sabatier / Toulouse III / France
2. CeRSM, 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

"KmL: A package to cluster longitudinal data"
Examples

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

### Simple cut with legend
windowsCut(5)
Index

* **NA**
  - imputation, 10
  - `aplplot`
    - `plotTrajMeans,LongData`, 51
    - `plotTrajMeans3d,LongData`, 53

* **classes**
  - `ListPartition-class`, 20
  - `LongData-class`, 25
  - `LongData3d-class`, 29
  - `ParLongData-class`, 39
  - `Partition-class`, 42
  - `ParWindows-class`, 46
  - regroup, 58

* **classif**
  - `LongData-class`, 25
  - `LongData3d-class`, 29
  - `longitudinalData-package`, 2
  - `ParWindows-class`, 46

* **cluster**
  - `imputation`, 10
  - `ListPartition-class`, 20
  - `longData`, 23
  - `LongData-class`, 25
  - `longData3d`, 27
  - `LongData3d-class`, 29
  - `longDataFrom3d`, 31
  - `longDataTo3d`, 33
  - `longitudinalData-package`, 2
  - `ParWindows-class`, 46
  - `partition`, 40
  - `ParLongData`, 37
  - regroup, 58

* **datasets**
  - `artificialJointLongData`, 4
  - `artificialLongData`, 5

* **documentation**
  - `artificialJointLongData`, 4
  - `artificialLongData`, 5

* **dplot**
  - `longitudinalData-package`, 2

* **methods**
  - `imputation`, 10
  - `longData`, 23
  - `longData3d`, 27
  - `longDataFrom3d`, 31
  - `longDataTo3d`, 33
  - `ordered(ListPartition)`, 35
  - `parLongData`, 37
  - `partition`, 40
  - `qualityCriterion`, 55

* **method**
  - `scale`, 65

* **package**
  - `imputation`, 10
  - `longData`, 23
  - `longData3d`, 27
  - `longDataFrom3d`, 31
  - `longDataTo3d`, 33
  - `longitudinalData-package`, 2
  - `plotTrajMeans,LongData`, 51
  - `plotTrajMeans3d,LongData`, 53
  - `qualityCriterion`, 55

* **ts**
  - `imputation`, 10
  - `ListPartition-class`, 20
  - `longData-class`, 25
  - `longData3d-class`, 29
  - `ParLongData-class`, 39
  - `partition`, 40
  - `ParWindows-class`, 46
  - `qualityCriterion`, 55
  - regroup, 58

[,] `ListPartition-ANY,ANY,ANY`
  - `(ListPartition-class)`, 20
[,, `ListPartition-ANY,ANY,ANY-method`
  - `(ListPartition-class)`, 20
[,, `ListPartition-ANY,ANY,ANY-methods`

69
(ListPartition-class), 20
[,.ListPartition-method (ListPartition-class), 20
[,.ListPartition-methods (ListPartition-class), 20
[,.LongData-ANY,ANY,ANY (LongData-class), 25
[,.LongData-method (LongData-class), 25
[,.LongData3d-ANY,ANY,ANY (LongData3d-class), 29
[,.LongData3d-ANY,ANY,ANY-method (LongData3d-class), 29
[,.LongData3d-method (LongData3d-class), 29
[,.ParLongData-method (ParLongData-class), 39
[,.ParWindows-method (ParWindows-class), 46
[,.Partition-method (Partition-class), 42
<-,.ListPartition-ANY,ANY,ANY (ListPartition-class), 20
<-,.ListPartition-method (ListPartition-class), 20
<-,.LongData-ANY,ANY,ANY (LongData-class), 25
<-,.LongData-method (LongData-class), 25
<-,.LongData3d-ANY,ANY,ANY-method (LongData3d-class), 29
<-,.ParLongData-method (ParLongData-class), 39
<-,.ParWindows-method (ParWindows-class), 46
<-,.Partition-method (Partition-class), 42

artificialJointLongData, 4
artificialLongData, 5
CHOICE_STYLE (Constants), 6
CLUSTER_NAMES (Constants), 6
Constants, 6
CRITERION_MIN_OR_MAX (Constants), 6
CRITERION_NAMES (Constants), 6
DISTANCE_METHODS (Constants), 6
distFrechet, 7
extendParLongData, 8, 38
INDEX

longData3d, missing, missing, missing, missing, missing, missing-method
(longData3d), 27
LongData3d-class, 29
LongData3d_show (LongData3d-class), 29
LongData_show (LongData-class), 25
longDataFrom3d, 26, 30, 31
longDataTo3d, 33
longitudinalData (longitudinalData-package), 2
longitudinalData-package, 2
longToWide, 62
longToWide (reshapeLongToWide), 60
makeLatexFile, 34, 48, 49, 64
makeTriangles, 35, 49, 64
MAX_CLUSTERs (Constants), 6
ordered, 3, 20, 41
ordered (ordered(ListPartition)), 35
ordered (ListPartition), 35
ordered, ListPartition
(ordered (ListPartition)), 35
ordered, ListPartition-method
(ordered (ListPartition)), 35
ParLongData, 8, 9, 37, 38, 52, 54
ParLongData (ParLongData-class), 39
parLongData, 37
ParLongData-class, 39
parMEAN (parLongData), 37
Partition, 3, 13, 22, 26, 30, 35, 40, 41, 47,
49–51, 54–56, 58, 59
partition, 3, 40, 42, 44
partition, ANY, array, ANY-method
(partition), 40
partition, ANY, LongData, ANY-method
(partition), 40
partition, ANY, LongData3d, ANY-method
(partition), 40
partition, ANY, matrix, ANY-method
(partition), 40
partition, ANY, missing, ANY-method
(partition), 40
partition, missing, missing, missing-method
(partition), 40
Partition-class, 42
parTRAJ (parLongData), 37
ParWindows, 44, 45, 67
ParWindows (ParWindows-class), 46
plot3dPdf, 27, 31, 35, 47, 48, 64
plot3dPdf, LongData3d, missing-method
(plot3dPdf), 47
plot3dPdf, LongData3d, numeric-method
(plot3dPdf), 47
plot3dPdf, LongData3d, Partition-method
(plot3dPdf), 47
plot3dPdf, LongData3d-method
(plot3dPdf), 47
plotAllCriterion, 49
plotAllCriterion, ListPartition
(plotAllCriterion), 49
plotAllCriterion, ListPartition-method
(plotAllCriterion), 49
plotAllCriterion-method
(plotAllCriterion), 49
plotCriterion, 20, 50
plotCriterion, ListPartition
(plotCriterion), 50
plotCriterion, ListPartition-method
(plotCriterion), 50
plotCriterion-method (plotCriterion), 50
plotTrajMeans, 3, 9, 27, 31, 37, 39
plotTrajMeans (plotTrajMeans, LongData),
51
plotTrajMeans, LongData, 51
plotTrajMeans, LongData, missing-method
(plotTrajMeans, LongData), 51
plotTrajMeans, LongData, Partition-method
(plotTrajMeans, LongData), 51
plotTrajMeans, LongData-method
(plotTrajMeans, LongData), 51
plotTrajMeans3d, 3, 9, 27, 31, 52
plotTrajMeans3d
(plotTrajMeans3d, LongData), 53
plotTrajMeans3d, LongData, 53
plotTrajMeans3d, LongData3d
(plotTrajMeans3d, LongData), 53
plotTrajMeans3d, LongData3d, missing-method
(plotTrajMeans3d, LongData), 53
plotTrajMeans3d, LongData3d, Partition-method
(plotTrajMeans3d, LongData), 53
plotTrajMeans3d, LongData3d-method
(plotTrajMeans3d, LongData), 53
points, 39
qualityCriterion, 3, 13, 27, 31, 42, 55
qualityCriterion, array, ANY-method
   (qualityCriterion), 55
qualityCriterion, LongData, Partition-method
   (qualityCriterion), 55
qualityCriterion, LongData3d, Partition-method
   (qualityCriterion), 55
qualityCriterion, matrix, ANY-method
   (qualityCriterion), 55

regroup, 58
reshape, 60, 62
reshapeLongToWide, 60
reshapeWideToLong, 61
restoreRealData, 26, 30, 62, 66
restoreRealData, LongData
   (restoreRealData), 62
restoreRealData, LongData-method
   (restoreRealData), 62
restoreRealData, LongData3d
   (restoreRealData), 62
restoreRealData, LongData3d-method
   (restoreRealData), 62
rgl, 54

saveTrianglesAsASY, 35, 48, 49, 63
scale, 25, 26, 30, 62, 63, 65
scale, LongData (scale), 65
scale, LongData-method (scale), 65
scale, LongData3d (scale), 65
scale, LongData3d-method (scale), 65
screen, 46
show, ListPartition-method
   (ListPartition-class), 20
show, LongData-method (LongData-class), 25
show, LongData3d-method
   (LongData3d-class), 29
show, Partition-method
   (Partition-class), 42
split.screen, 45

wideToLong, 60
wideToLong (reshapeWideToLong), 61
windowsCut, 46, 67