Package 'funLBM'

March 17, 2021

Type Package

Title Model-Based Co-Clustering of Functional Data

Version 2.2

Date 2021-03-17

Author Charles Bouveyron, Julien Jacques and Amandine Schmutz

Maintainer Charles Bouveyron <charles.bouveyron@gmail.com>

Depends fda, parallel, funFEM, abind, ggplot2, R (>= 3.4.0)

Description
The funLBM algorithm allows to simultaneously cluster the rows and the columns of a data matrix where each entry of the matrix is a function or a time series.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2021-03-17 10:50:09 UTC

R topics documented:

ari *Adjusted Rand index*

Description

The adjusted Rand index (ARI) allows to compare two clustering partitions.

Usage

```
ari(x, y)
```

Arguments

x	The first partition to compare (as vector).
y	The second partition to compare (as vector).

Value

ari	The value of the ARI.

See Also

funLBM

Examples

```
x = sample(1:3,20,replace = TRUE)
y = sample(1:3,20,replace = TRUE)
ari(x,y)
```

funLBM *The functional latent block model*

Description

The funLBM algorithm, proposed by Bouveyron et al. (2018) <doi:10.1111/rssc.12260>, allows to simultaneously cluster the rows and the columns of one or more data matrix where each entry of the matrix is a (univariate or multivariate) function or a time series.

Usage

```
funLBM(X, K, L, maxit = 50, burn = 25, basis.name = "fourier", nbasis = 15,
      nbinit = 1, gibbs.it = 3, display = FALSE, init = "funFEM", mc.cores = 1, ...)
```

Arguments

`X`	Univariate case: The data array (n x p x T) where each entry corresponds to the measure of one individual i, i=1,...,n, for a functional variable j, j=1,...,p, at point t, t=1,...,T. Multivariate case: a list of data array as described hereinabove with one data array by variable.
`K`	The number of row clusters,
`L`	The number of column clusters,
`maxit`	The maximum number of iterations of the SEM-Gibbs algorithm (default is 100),
`burn`	The number of of iterations of the burn-in period (default is 50),
`basis.name`	The name ('fourier' or 'spline') of the basis functions used for the decomposition of the functions (default is 'fourier'),
`nbasis`	Number of the basis functions used for the decomposition of the functions (default is 15),
`nbinit`	Number of initializations (default is 3),
`gibbs.it`	Number of Gibbs iterations (default is 3),
`display`	Binary value. If TRUE, information about the iterations is displayed,
`init`	The type of initialization: 'random', 'kmeans' or 'funFEM'. Default is 'kmeans',
`mc.cores`	The number of cores for parallel computing (default is 1),
`...`	Additional parameters to provide to sub-functions.

Value

The resulting object contains, in addition to call information:

`prms`	A list containing all fited parameters for the best model (according to ICL)
`Z`	The dummy matrix of row clustering
`W`	The dummy matrix of column clustering
`row_clust`	The group memberships of rows
`col_clust`	The group memberships of columns
`allPrms`	A list containing the fited parameters for all tested models
`loglik`	The log-likelihood of the best model
`icl`	The value of ICL for the best model

References

C. Bouveyron, L. Bozzi, J. Jacques and F.-X. Jollois, The Functional Latent Block Model for the Co-Clustering of Electricity Consumption Curves, Journal of the Royal Statistical Society, Series C, 2018 (https://doi.org/10.1111/rssc.12260).

Examples

```
## Univariate example: Co-clustering on simulated data
set.seed(12345)
X = simulateData(n = 30, p = 30, t = 15)
out = funLBM(X$data,K=4,L=3)

# Visualization of results
plot(out,type='blocks')
plot(out,type='proportions')
plot(out,type='means')

# Evaluating clustering results
ari(out$col_clust,X$col_clust)
ari(out$row_clust,X$row_clust)

## Multivariate example:
X = simulateData2(n = 50, p = 50, t = 15)
out = funLBM(list(X$data1,X$data2),K=4,L=3)

# Visualization of results
plot(out,type='blocks')
plot(out,type='proportions')
plot(out,type='means')

# Evaluating clustering results
ari(out$col_clust,X$col_clust)
ari(out$row_clust,X$row_clust)

## The following examples could take a few minutes to run!

## Co-clustering on simulated data with parallel model selection
#X = simulateData(n = 30, p = 30, t = 15)
#out = funLBM(X$data,K=2:4,L=2:4,mc.cores = 4)

## Evaluating clustering results
#ari(out$col_clust,X$col_clust)
#ari(out$row_clust,X$row_clust)

## Co-clustering of Velib data
#data(Velib)
#out = funLBM(Velib$data,K=4,L=2)

## Visualization of results
#plot(out,type='blocks')
#plot(out,type='proportions')
#plot(out,type='means')
```

`plot.fd` *Plot a Functional Data Object*

Description

Functional data observations, or a derivative of them, are plotted. These may be either plotted simultaneously, as `matplot` does for multivariate data, or one by one with a mouse click to move from one plot to another. The function also accepts the other plot specification arguments that the regular `plot` does. Calling `plot` with an `fdSmooth` or an `fdPar` object plots its `fd` component.

Usage

```
## S3 method for class 'fd'
  plot(x, y, Lfdobj=0, href=TRUE, titles=NULL,
                    xlim=NULL, ylim=NULL, xlab=NULL,
                    ylab=NULL, ask=FALSE, nx=NULL, axes=NULL, col=1, ...)
```

Arguments

`x`	functional data object(s) to be plotted.
`y`	sequence of points at which to evaluate the functions 'x' and plot on the horizontal axis. Defaults to seq(rangex[1], rangex[2], length = nx). NOTE: This will be the values on the horizontal axis, NOT the vertical axis.
`Lfdobj`	either a nonnegative integer or a linear differential operator object. If present, the derivative or the value of applying the operator is plotted rather than the functions themselves.
`href`	a logical variable: If `TRUE`, add a horizontal reference line at 0.
`titles`	a vector of strings for identifying curves
`xlab`	a label for the horizontal axis.
`ylab`	a label for the vertical axis.
`xlim`	a vector of length 2 containing axis limits for the horizontal axis.
`ylim`	a vector of length 2 containing axis limits for the vertical axis.
`ask`	a logical value: If `TRUE`, each curve is shown separately, and the plot advances with a mouse click
`nx`	the number of points to use to define the plot. The default is usually enough, but for a highly variable function more may be required.
`axes`	Either a logical or a list or `NULL`. • logical whether axes should be drawn on the plot • list a list used to create custom `axes` used to create axes via `x$axes[[1]]` and `x$axes[-1]`. The primary example of this uses `list("axesIntervals",...)`, e.g., with `Fourier` bases to create `CanadianWeather` plots
`col`	line colors
`...`	additional plotting arguments that can be used with function `plot`

Details

Note that for multivariate data, a suitable array must first be defined using the `par` function.

Value

'done'

Side Effects

a plot of the functional observations

See Also

lines.fd, plotfit.fd

Examples

```
##
## plot.fd
##

daybasis65 <- create.fourier.basis(c(0, 365), 65,
                    axes=list("axesIntervals"))
harmaccelLfd <- vec2Lfd(c(0,(2*pi/365)^2,0), c(0, 365))

harmfdPar     <- fdPar(daybasis65, harmaccelLfd, lambda=1e5)

daytempfd <- with(CanadianWeather, smooth.basis(day.5,
        dailyAv[,,"Temperature.C"], daybasis65)$fd)

#  plot all the temperature functions for the monthly weather data
plot(daytempfd, main="Temperature Functions")

## Not run:
# To plot one at a time:
# The following pauses to request page changes.
\dontshow{
# (Without 'dontrun', the package build process
# might encounter problems with the par(ask=TRUE)
# feature.)
}
plot(daytempfd, ask=TRUE)

## End(Not run)

##
## plot.fdSmooth
##
b3.4 <- create.bspline.basis(norder=3, breaks=c(0, .5, 1))
# 4 bases, order 3 = degree 2 =
# continuous, bounded, locally quadratic
fdPar3 <- fdPar(b3.4, lambda=1)



# Penalize excessive slope Lfdobj=1;
```

```
# (Can not smooth on second derivative Lfdobj=2 at it is discontinuous.)
fd3.4s0 <- smooth.basis(0:1, 0:1, fdPar3)

# using plot.fd directly
plot(fd3.4s0$fd)




##
## with Date and POSIXct argvals
##
# Date
invasion1 <- as.Date('1775-09-04')
invasion2 <- as.Date('1812-07-12')
earlyUS.Canada <- as.numeric(c(invasion1, invasion2))
BspInvasion    <- create.bspline.basis(earlyUS.Canada)

earlyUSyears <- seq(invasion1, invasion2, length.out=7)
earlyUScubic <- (as.numeric(earlyUSyears-invasion1)/365.24)^3
earlyUSyears <- as.numeric(earlyUSyears)
fitCubic     <- smooth.basis(earlyUSyears, earlyUScubic, BspInvasion)$fd
plot(fitCubic)

# POSIXct
AmRev.ct    <- as.POSIXct1970(c('1776-07-04', '1789-04-30'))
AmRevYrs.ct <- seq(AmRev.ct[1], AmRev.ct[2], length.out=14)
AmRevLin.ct <- as.numeric(AmRevYrs.ct-AmRev.ct[2])
AmRevYrs.ct <- as.numeric(AmRevYrs.ct)
BspRev.ct   <- create.bspline.basis(AmRev.ct)
fitLin.ct   <- smooth.basis(AmRevYrs.ct, AmRevLin.ct, BspRev.ct)$fd
plot(fitLin.ct)
```

`plot.funLBM` *Plotting co-clustering results of funLBM*

Description

Plotting of funLBM co-clustering results: functional means, block matrix, parameters, ...

Usage

```
## S3 method for class 'funLBM'
plot(x,type='blocks',...)
```

Arguments

`x`	An object produced by the funLBM function,

type	The type of plot to display. Possible plots are 'blocks' (default), 'means', 'evolution', 'likelihood', 'proportions',
...	Additional arguments to provide.

See Also

funLBM

Examples

```
## Co-clustering of the Velib data
X = simulateData(n = 30, p = 30, t = 15)
out = funLBM(X$data,K=4,L=3)

# Visualization of results
plot(out,type='blocks')
plot(out,type='proportions')
plot(out,type='means')
```

---

print.funLBM *Printing co-clustering results of funLBM*

Description

Printing a summary of the funLBM co-clustering results

Usage

```
## S3 method for class 'funLBM'
print(x,...)
```

Arguments

x	An object produced by the funLBM function,
...	Additional arguments to provide.

See Also

funLBM

Examples

```
## Co-clustering of the Velib data
X = simulateData(n = 30, p = 30, t = 15)
out = funLBM(X$data,K=4,L=3)
out
```

`simulateData` *Simulate data for funLBM*

Description

Simulate data according to the funLBM model with K=4 groups for rows and L=3 groups for columns.

Usage

```
simulateData(n = 100, p = 100, t = 30)
```

Arguments

`n`	The number of rows (individuals) of the simulated data array,
`p`	The number of columns (functional variables) of the simulated data array,
`t`	The number of measures for the functions of the simulated data array.

Value

The resulting object contains:

`data`	data array of size n x p x t
`row_clust`	Group memberships of rows
`col_clust`	Group memberships of columns

References

C. Bouveyron, L. Bozzi, J. Jacques and F.-X. Jollois, The Functional Latent Block Model for the Co-Clustering of Electricity Consumption Curves, Journal of the Royal Statistical Society, Series C, 2018 (https://doi.org/10.1111/rssc.12260).

See Also

`funLBM`

Examples

```
# Simulate data and co-clustering
X = simulateData(n = 30, p = 30, t = 15)


# Co-clustering with funLBM
out = funLBM(X$data,K=4,L=3)

# Visualization of results
plot(out,type='blocks')
plot(out,type='proportions')
```

```
plot(out,type='means')

# Evaluating clustering results
ari(out$col_clust,X$col_clust)
ari(out$row_clust,X$row_clust)
```

`simulateData2` *Simulate bivariate data for funLBM*

Description

Simulate bivariate data according to the funLBM model with K=4 groups for rows and L=3 groups for columns.

Usage

```
simulateData2(n = 100, p = 100, t = 30)
```

Arguments

`n`	The number of rows (individuals) of the simulated data array,
`p`	The number of columns (functional variables) of the simulated data array,
`t`	The number of measures for the functions of the simulated data array.

Value

The resulting object contains:

`data1`	data array of size n x p x t for first variable
`data2`	data array of size n x p x t for second variable
`row_clust`	Group memberships of rows
`col_clust`	Group memberships of columns

References

C. Bouveyron, L. Bozzi, J. Jacques and F.-X. Jollois, The Functional Latent Block Model for the Co-Clustering of Electricity Consumption Curves, Journal of the Royal Statistical Society, Series C, 2018 (https://doi.org/10.1111/rssc.12260).

See Also

`funLBM`

Examples

```
# Simulate data and co-clustering
set.seed(12345)
X = simulateData2(n = 50, p = 50, t = 15)


# Co-clustering with funLBM
out = funLBM(list(X$data1,X$data2),K=4,L=3)

# Visualization of results
plot(out,type='blocks')
plot(out,type='proportions')
plot(out,type='means')

# Evaluating clustering results
ari(out$col_clust,X$col_clust)
ari(out$row_clust,X$row_clust)
```

Velib *The Velib data set.*

Description

The Velib data set contains data from the bike sharing system of Paris, called Velib. The data are loading profiles of the bike stations over seven days. The data were collected every hour during the period Sunday 1st Sept. - Sunday 7th Sept., 2014.

Usage

```
data("Velib")
```

Format

The format is: - data: the loading profiles (nb of available bikes / nb of bike docks) of the 1189 stations for 7 days every hour. - position: the longitude and latitude of the 1189 bike stations.

Source

The real time data are available at https://developer.jcdecaux.com/ (with an api key).

References

The data were first used in C. Bouveyron, E. Come and J. Jacques, The discriminative functional mixture model for a comparative analysis of bike sharing systems, The Annals of Applied Statistics, vol. 9 (4), pp. 1726-1760, 2015 (http://dx.doi.org/10.1214/15-AOAS861).

Examples

```
data(Velib)


# Co-clustering with funLBM
out = funLBM(Velib$data,K=4,L=2)

# Visualization of results
plot(out,type='blocks')
plot(out,type='proportions')
plot(out,type='means')
```

Index