Package 'FunCC'

June 8, 2020

Title Functional Cheng and Church Bi-Clustering

Version 1.0

Author Agostino Torti [aut, cre], Marta Galvani [aut, cre], Alessandra Menafoglio [aut], Simone Vantini[aut]

Maintainer Agostino Torti <agostino.torti@polimi.it>

Description The FunCC algorithm allows to apply the FunCC algorithm to simultaneously cluster the rows and the columns of a data matrix whose inputs are functions.

Depends R (>= 3.5.1)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Imports narray, biclust, reshape, RColorBrewer, ggplot2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-08 10:10:02 UTC

R topics documented:

find_best_delta *Functional Cheng and Church Algorithm varying the delta value*

Description

The find_best_delta function evaluate the results of FunCC algorithm in terms of total H-score value, the number of obtained bi-clusters and the number of not assigned elements when varying the delta value

Usage

```
find_best_delta(
  fun_mat,
  delta_min,
  delta_max,
  num_delta = 10,
  template.type = "mean",
  theta = 1.5,
  number = 100,
  alpha = 0,
  beta = 0,
  const_alpha = FALSE,
  const_beta = FALSE,
  shift.alignement = FALSE,
  shift.max = 0.1,
  max.iter.align = 100
)
```

Arguments

fun_mat	The data array (n x m x T) where each entry corresponds to the measure of one observation i, i=1,...,n, for a functional variable m, m=1,...,p, at point t, t=1,...,T
delta_min	scalar: Manimum value of the maximum of accepted score, should be a real value > 0
delta_max	scalar: Maximum value of the maximum of accepted score, should be a real value > 0
num_delta	integer: number of delta to be evaluated between delta_min and delta_max
template.type	character: type of template required. If template.type='mean' the template is evaluated as the average function, if template.type='medoid' the template is evaluated as the medoid function.
theta	scalar: Scaling factor should be a real value > 1
number	integer: Maximum number of iterations
alpha	binary: if alpha=1 row shift is allowed, if alpha=0 row shift is avoided
beta	binary: if beta=1 row shift is allowed, if beta=0 row shift is avoided

const_alpha	logicol: indicates if row shift is contrained as constant
const_beta	logicol: indicates if col shift is contrained as constant
shift.alignement	logicol: If shift.alignement=True the shift aligment is performed, if shift.alignement=False no alignment is performed
shift.max	scalar: shift.max controls the maximal allowed shift, at each iteration, in the alignment procedure with respect to the range of curve domains. t.max must be such that 0<shift.max<1
max.iter.align	integer: maximum number of iteration in the alignment procedure

Value

a dataframe containing for each evaluated delta: Htot_sum (the sum of totale H-score), num_clust (the number of found Bi-clusters), not_assigned (the number of not assigned elements)

Examples

```
## Not run:
data("funCCdata")
find_best_delta(funCCdata,delta_min=0.1,delta_max=20,num_delta=20,alpha=1,beta=0,const_alpha=TRUE)

## End(Not run)
```

---

funCCdata *Simulated data*

Description

funCC.data is a functional dataset displaying block structure

Usage

```
data(funCCdata)
```

Format

An object of class `array` of dimension 30 x 7 x 240.

Examples

```
data(funCCdata)
```

funcc_biclust *Functional Cheng and Church algorithm*

Description

The funCC 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

Usage

```
funcc_biclust(
  fun_mat,
  delta,
  theta = 1,
  template.type = "mean",
  number = 100,
  alpha = 0,
  beta = 0,
  const_alpha = FALSE,
  const_beta = FALSE,
  shift.alignement = FALSE,
  shift.max = 0.1,
  max.iter.align = 100
)
```

Arguments

fun_mat	The data array (n x m x T) where each entry corresponds to the measure of one observation i, i=1,...,n, for a functional variable m, m=1,...,p, at point t, t=1,...,T
delta	scalar: Maximum of accepted score, should be a real value > 0
theta	scalar: Scaling factor should be a real value > 1
template.type	character: type of template required. If template.type='mean' the template is evaluated as the average function, if template.type='medoid' the template is evaluated as the medoid function.
number	integer: Maximum number of iteration
alpha	binary: if alpha=1 row shift is allowed, if alpha=0 row shift is avoided
beta	binary: if beta=1 row shift is allowed, if beta=0 row shift is avoided
const_alpha	logicol: Indicates if row shift is contrained as constant.
const_beta	logicol: Indicates if col shift is contrained as constant.
shift.alignement	logicol: If shift.alignement=True the shift aligment is performed, if shift.alignement=False no alignment is performed
shift.max	scalar: shift.max controls the maximal allowed shift, at each iteration, in the alignment procedure with respect to the range of curve domains. t.max must be such that 0<shift.max<1
max.iter.align	integer: maximum number of iteration in the alignment procedure

Value

a list of two elements containing respectively the Biclustresults and a dataframe containing the parameters setting of the algorithm @examples data("funCCdata") res <- funcc_biclust(funCCdata,delta=10,theta=1,alpha=1,be
res

`funcc_show_bicluster_coverage` *plotting coverage of each bi-cluster*

Description

funcc_show_bicluster_coverage graphically shows the coverage of each bi-cluster in terms of percentage of included functions

Usage

```
funcc_show_bicluster_coverage(
  fun_mat,
  res_input,
  not_assigned = TRUE,
  max_coverage = 1
)
```

Arguments

`fun_mat`	The data array (n x m x T) where each entry corresponds to the measure of one observation i, i=1,...,n, for a functional variable m, m=1,...,p, at point t, t=1,...,T
`res_input`	An object produced by the funcc_biclust function
`not_assigned`	logicol: if true also the cluster of not assigned elements is included
`max_coverage`	scalar: percentage of maximum cumulative coverage to be shown

Value

a figure representing for each bi-cluster the coverage in terms of percentage of included functions

Examples

```
data("funCCdata")
res <- funcc_biclust(funCCdata,delta=10,theta=1,alpha=1,beta=0,const_alpha=TRUE)
funcc_show_bicluster_coverage(funCCdata,res)
```

funcc_show_bicluster_dimension

plotting dimensions of each bi-cluster

Description

funcc_show_bicluster_dimension graphically shows the dimensions of each bi-cluster (i.e. number of rows and columns)

Usage

```
funcc_show_bicluster_dimension(fun_mat, res_input)
```

Arguments

fun_mat	The data array (n x m x T) where each entry corresponds to the measure of one observation i, i=1,...,n, for a functional variable m, m=1,...,p, at point t, t=1,...,T
res_input	An object produced by the funcc_biclust function

Value

a figure representing the dimensions of each bi-cluster (i.e. number of rows and columns)

Examples

```
data("funCCdata")
res <- funcc_biclust(funCCdata,delta=10,theta=1,alpha=1,beta=0,const_alpha=TRUE)
funcc_show_bicluster_dimension(funCCdata,res)
```

funcc_show_bicluster_hscore

plotting hscore of each bi-cluster on bicluster dimension

Description

funcc_show_bicluster_hscore graphically shows the hscore vs the dimension (i.e. number of rows and columns) of each bi-cluster

Usage

```
funcc_show_bicluster_hscore(fun_mat, res_input)
```

Arguments

fun_mat	The data array (n x m x T) where each entry corresponds to the measure of one observation i, i=1,...,n, for a functional variable m, m=1,...,p, at point t, t=1,...,T
res_input	An object produced by the funcc_biclust function

Value

a figure representing the dimensions of each bi-cluster (i.e. number of rows and columns)

Examples

```
data("funCCdata")
res <- funcc_biclust(funCCdata,delta=10,theta=1,alpha=1,beta=0,const_alpha=TRUE)
funcc_show_bicluster_hscore(funCCdata,res)
```

---

funcc_show_block_matrix

Plotting co-clustering results of funCC on the data matrix

Description

funcc_show_block_matrix graphically shows the bi-clusters positions in the original data matrix

Usage

```
funcc_show_block_matrix(fun_mat, res_input)
```

Arguments

fun_mat	The data array (n x m x T) where each entry corresponds to the measure of one observation i, i=1,...,n, for a functional variable m, m=1,...,p, at point t, t=1,...,T
res_input	An object produced by the funcc_biclust function

Value

a figure representing the bi-clusters positions in the original data matrix

Examples

```
data("funCCdata")
res <- funcc_biclust(funCCdata,delta=10,theta=1,alpha=1,beta=0,const_alpha=TRUE)
funcc_show_block_matrix(funCCdata,res)
```

funcc_show_results *Plotting co-clustering results of funCC*

Description

funcc_show_results graphically shows the results of the bi-clustering

Usage

```
funcc_show_results(
  fun_mat,
  res_input,
  only.mean = FALSE,
  aligned = FALSE,
  warping = FALSE
)
```

Arguments

fun_mat	The data array (n x m x T) where each entry corresponds to the measure of one observation i, i=1,...,n, for a functional variable m, m=1,...,p, at point t, t=1,...,T
res_input	An object produced by the funcc_biclust function
only.mean	logicol: if True only the template functions for each bi-cluster is displayed
aligned	logicol: if True the alignemd functions are displayed
warping	logicol: if True also a figure representing the warping functions are displayed

Value

a figure representing each bi-cluster in terms of functions contained in it or templates

Examples

```
data("funCCdata")
res <- funcc_biclust(funCCdata,delta=10,theta=1,alpha=1,beta=0,const_alpha=TRUE)
funcc_show_results(funCCdata,res)
```

Index