

# Package ‘FunCC’

January 31, 2020

**Title** Functional Cheng and Church Bi-clustering

**Version** 0.0.0.9000

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

**Description** The FunCC package 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

## R topics documented:

|  |   |
|--|---|
| find_best_delta . . . . .                | 1 |
| funCCdata . . . . .                      | 3 |
| funcc_biclust . . . . .                  | 3 |
| funcc_show_bicluster_coverage . . . . .  | 4 |
| funcc_show_bicluster_dimension . . . . . | 5 |
| funcc_show_block_matrix . . . . .        | 5 |
| funcc_show_results . . . . .             | 6 |

|              |          |
|--------------|----------|
| <b>Index</b> | <b>7</b> |
|--------------|----------|

---

|                 |  |
|-----------------|--|
| find_best_delta | <i>Functional Cheng and Church Algorithm varying the delta value</i> |
|-----------------|--|

---

## 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 = F,
  const_beta = F,
  shift.alignement = F,
  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: 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      | logical: indicates if row shift is constrained as constant   |
| shift.alignement | logical: If shift.alignement=True the shift alignment 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  |
| const_b          | logical: indicates if col shift is constrained as constant   |

**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)

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 = F,
  const_beta = F,
  shift.alignement = F,
  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      | logical: Indicates if row shift is constrained as constant.  |
| shift.alignement | logical: If shift.alignement=True the shift alignment 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  |
| const_b          | logical: Indicates if col shift is constrained as constant.  |

**Value**

a list of two elements containing respectively the Biclustresults and a dataframe containing the parameters setting of the algorithm

---

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 = T,
  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 | logical: 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

---

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)

---

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

---

|                    |  |
|--------------------|--|
| funcc_show_results | <i>Plotting co-clustering results of funCC</i> |
|--------------------|--|

---

**Description**

funcc\_show\_results graphically shows the results of the bi-clustering

**Usage**

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

**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 | logical: if True only the template functions for each bi-cluster is displayed   |
| aligned   | logical: if True the alignemd functions are displayed   |
| warping   | logical: 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

# Index

## \*Topic **datasets**

funCCdata, [3](#)

find\_best\_delta, [1](#)

funcc\_biclust, [3](#)

funcc\_show\_bicluster\_coverage, [4](#)

funcc\_show\_bicluster\_dimension, [5](#)

funcc\_show\_block\_matrix, [5](#)

funcc\_show\_results, [6](#)

funCCdata, [3](#)