Package 'fdakmapp'

October 17, 2017

Title Functional Data Analysis : K-mean and K-medoid with Alignemnt

Version 2.0.1

Type Package

Author Alessandro Zito, Alice Parodi, Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli

Maintainer Alessandro Zito <zito.ales@gmail.com>

Description
The fdakmapp package provides the kmap function that jointly performs clustering and alignment of a functional dataset (multidimensional or unidimensional). The centers can be computed by mean and medoid center methods. Many options are available as parallal version.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Import Rcpp (>= 0.12.11)

LinkingTo Rcpp , RcppArmadillo

SystemRequirements C++11

RoxygenNote 6.0.1

Suggests testthat

R topics documented:

aneurisk65 *Data for Examples.*

Description

A dataset containing 65 multidimensional curves.

Usage

```
aneurisk65
```

Format

A list with abscissas x and values y:

x matrix 65 x 1380

y array 65 x1380 x 3

fdakmapp *Functional Data Analysis Plus: K-Mean/Medoid Alignment:*

Description

Fdakmapp is the package that allows to jointly perform clustering and alignment of a functional dataset (multidimensional or unidimensional functions).

References

- Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*. Computational Statistics and Data Analysis, 54, 1219-1233.
- Sangalli, L.M., Secchi, P., Vantini, S., 2014. *"Analysis of AneuRisk65 data: K-mean Alignment"*. Electronic Journal of Statistics, Special Section on "Statistics of Time Warpings and Phase Variations", Vol. 8, No. 2, 1891-1904.

See Also

kmap.

Examples

```
############# EXE ##########################
res<-kmap(x=aneurisk65$x, y=aneurisk65$y, n_clust=2)

############# OUTPUT################
kmap_show_results(res,FALSE)
```

kmap	*Clustering and alignment of functional data*

Description

kmap jointly performs clustering and alignment of a functional dataset (multidimensional or unidimensional functions). To run kmap function with different numbers of clusters and/or different alignment methods see

Usage

```
res<-kmap( x, y,  y1, n_clust,warping_method, center_method,similarity_method,
optim_method, seeds, span, delta, d_max, s_max, n_out, toll, fence,
iter_max,show_iter, check_total_similarity)
```

Arguments

`x`	A matrix *n.func* X *grid.size* or vector *grid.size*: the abscissa values where each function is evaluated. *n.func*: number of functions in the dataset. *grid.size*: maximal number of abscissa values where each function is evaluated. The abscissa points may be unevenly spaced and they may differ from function to function. `x` can also be a vector of length *grid.size*. In this case, `x` will be used as abscissa grid for all functions.Furthermore if the grid's size differs from one function to another the matrix must be completed with NA values.The parameter `y` must be provided.
`y`	A matrix *n.func* X *grid.size* or array *n.func* X *grid.size* X *d*: evaluations of the set of original functions on the abscissa grid `x`. *n.func*: number of functions in the dataset. *grid.size*: maximal number of abscissa values where each function is evaluated. *d*: (only if the sample is multidimensional) number of function components, i.e. each function is a *d*-dimensional curve. The parameter `yx` must be provided.
`seeds`	vector max(n.clust) or matrix nstart X n.clust: indexes of the functions to be used as initial centers. If it is a matrix, each row contains the indexes of the initial centers of one of the nstart initializations. In the case where not all the values of seeds are provided, those not provided are randomly chosen among the n.func original functions. If seeds=NULL all the centers are randomly chosen. Default value of seeds is NULL.
`n_clust`	scalar: required number of clusters. Default value is 1. Note that if n.clust=1 kma performs only alignment without clustering.
`warping_method`	character: type of alignment required. If warping.method='noalign' kma performs only clustering (without alignment). If warping.method='affine' kma performs alignment (and possibly clustering) of functions using linear affine transformation as warping functions, i.e., x.final = dilation*x + shift. If warping.method='shift' kma allows only shift, i.e., x.final = x + shift. If warping.method='dilation' kma allows only dilation, i.e., x.final = dilation*x. Default value is 'affine'.

`center_method`	character: type of clustering method to be used. Possible choices are: 'mean' and 'medoid' and 'pseudomedoid'. Default value is 'mean'.
`similarity_method`	character: required similarity measure. Possible choices are: 'pearson','l2'. Default value is 'pearson'. See (what to define?) for details.
`optim_method`	character: optimization method chosen to find the best warping functions at each iteration. Possible choices are: 'bobyqa'. See optim function for details. Default method is 'bobyqa'
`warping_opt`	numeric vector. The parameters to set depend on the warping_method chosen. If warping_method ='affine' warping_opt <- c(max_dilation , max_shift). If warping_method <- 'dilation' warping_opt <- c(max_dilation). If warping_method <- 'shift' warping_opt <- c(max_shift). If warping_method <- 'noalign' warping_opt <- as.numeric().
`center_opt`	numeric vector. The parameters to set depend on the center_method chosen. If center_method ='mean' center_opt <- c(span, delta). If center_method ='medoid' center_opt <- as.numeric(). If center_method ='pseudomedoid' center_opt <- as.numeric().
`out_opt`	numeric vector. The parameters to set are (n_out , tollerance, max_iteration). n_out is the size of the grid where the centers will be computed. tollerance is a stop condition parameter. max_iterationa is a stop condition parameter.
`fence`	boolean: if fence=TRUE a control is activated at the end of each iteration. The aim of the control is to avoid warping outliers with respect to their computed distributions. If fence=TRUE the running time can increase considerably. Default value of fence is FALSE.
`check_total_similarity`	boolean: if check.total.similarity=TRUE at each iteration the algorithm checks if there is a decrease of the total similarity and stops. In the affermative case the result obtained in the penultimate iteration is returned. Defaultvalue is TRUE.
`show_iter`	boolean: if show.iter=TRUE kmap shows the current iteration of the algorithm. Default value is TRUE.
`comp_original_center`	boolean: if comp_original_center=TRUE the initial center with relative dissimilarities is computed otherwise this step is skipped. Default value is FALSE.
`par_opt`	numeric vector. Parallel options. The parameters to set are (num_threads,parallel_version) parallel_version available are 0 and 1 : 0 is a trivial parallelization in which each thread compute the center of a cluster; 1 is a more efficient parallelization in which all the threads compute the centers sequentially(available only with center_method = 'medoid').

Value

The function output is a list containing the following elements:

`x`	as input.
`y`	as input.
`seeds`	vector with the indeces used in the algorithm.

`warping.method`	as input.
`similarity.method`	as input.
`center.method`	as input.
`iterations`	scalar: total number of iterations performed by kma function.
`n.clust`	as input.
`x.center.orig`	numeric vector *n_out*: abscissa of the center computed if *comp_original_center*=TRUE.
`y.center.orig`	numeric vector *n_out* or matrix *n_out* X *n_dim*: value of the center computed if *comp_original_center*=TRUE.
`similarity.origin`	numeric vector *n_obs* dissimilarity,similarity or distance of the original center respect the obserbations computed if *comp_original_center*=TRUE.
`x.final`	matrix n.func X grid.size: aligned abscissas.
`n.clust.final`	scalar: final number of clusters. Note that n.clust.final may differ from initial number of clusters (i.e.,from n.clust) if some clusters are found to be empty.
`x.centers.final`	matrix n.clust.final X grid.size: abscissas of the final function centers.
`y.centers.final`	matrix n.clust.final X n.out or array n.clust.final X n.out x n_dim , contain the evaluations of the final functions centers.
`templates_vec`	list iteration : each element of the list contain centers of that iteration.
`x_out_vec`	list iteration : each element of the list contain the abscissa of the centers of that iteration.
`labels`	vector n_obs: cluster assignments.
`similarity.final`	vector n_obs: similarities,dissimilarities or distance between each function and the center of the cluster the function is assigned to.
`parameters.list`	list iterations: warping parameters at each iteration.
`parameters`	matrix n_par X n_obs: warping parameters applied to the original abscissas x to obtain the aligned abscissas x.final.
`timer`	vector : time of execution.

`kmap_show_results` *plot results kmap*

Description

Show results of the clustering with alignment of functional data

Usage

```
kma_show_results <-function (Result, bp_sim)
```

Arguments

`Result`	output of kmap.
`bp_sim`	boolean: if TRUE dissimilarity,similarity or distance boxplot are plotted.

`simulated30` *Data for Examples.*

Description

A dataset containing 30 simulated curves.

Usage

```
simulated30
```

Format

A list with abscissas x and values y:

x matrix 30 x 200

y array 30 x 200 x 1

Index