Package 'fdakma'

May 29, 2015

Type Package

Title Functional Data Analysis: K-Mean Alignment

Version 1.2.1

Date 2015-05-15

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

Maintainer Alice Parodi <alicecarla.parodi@polimi.it>

Description It performs simultaneously clustering and alignment of a multidimensional or unidimensional functional dataset by means of k-mean alignment.

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-05-29 10:53:37

R topics documented:

fdakma-package *Functional Data Analysis: K-Mean Alignment*

Description

fdakma jointly performs clustering and alignment of a functional dataset (multidimensional or unidimensional functions).

Details

Package:	fdakma
Type:	Package
Version:	1.2
Date:	2015-03-12
License:	GPL-3

Author(s)

Alice Parodi, Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

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

kma.compare, kma.similarity, kma.data, kma, kma.show.results

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

## Not run:
# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')
```

```
# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')


# Example: result of kma function with 2 clusters,
# allowing affine transformation for the abscissas
# and considering 'd1.pearson' as similarity.method.
fdakma_example <- kma (
  x=x, y0=y0, y1=y1, n.clust = 2,
  warping.method = 'affine',
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21)
)

kma.show.results(fdakma_example)

names(fdakma_example)

# Labels assigned to each function
fdakma_example$labels

# Total shifts and dilations applied to the original
# abscissa to obtain the aligned abscissa
fdakma_example$shift
fdakma_example$dilation

## End(Not run)
```

---

kma *Clustering and alignment of functional data*

Description

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

Usage

```
kma(x, y0 = NULL, y1 = NULL, n.clust = 1, warping.method = "affine",
similarity.method = "d1.pearson", center.method = "k-means", seeds = NULL,
optim.method = "L-BFGS-B", span = 0.15, t.max = 0.1, m.max = 0.1, n.out = NULL,
tol = 0.01, fence = TRUE, iter.max = 100, show.iter = 0, nstart=2, return.all=FALSE,
check.total.similarity=FALSE)
```

Arguments

x	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.
y0	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. Default value of y0 is NULL. The parameter y0 must be provided if the chosen similarity.method concerns original functions.
y1	matrix *n.func* X *grid.size* or array *n.func* X *grid.size* X *d*: evaluations of the set of original functions first derivatives on the abscissa grid x. Default value of y1 is NULL. The parameter y1 must be provided if the chosen similarity.method concerns original function first derivatives.
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='NOalignment' kma performs only k-mean 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'.
similarity.method	character: required similarity measure. Possible choices are: 'd0.pearson', 'd1.pearson', 'd0.L2', 'd1.L2', 'd0.L2.centered', 'd1.L2.centered'. Default value is 'd1.pearson'. See kma.similarity for details.
center.method	character: type of clustering method to be used. Possible choices are: 'k-means' and 'k-medoids'. Default value is 'k-means'.
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.
optim.method	character: optimization method chosen to find the best warping functions at each iteration. Possible choices are: 'L-BFGS-B' and 'SANN'. See optim function for details. Default method is 'L-BFGS-B'.
span	scalar: the span to be used for the loess procedure in the center estimation step when center.method='k-means'. Default value is 0.15. If center.method='k-medoids' value of span is ignored.

`t.max` scalar: `t.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<t.max<1* (e.g., `t.max=0.1` means that shift is bounded, at each iteration, between *-0.1*range(x)* and *+0.1*range(x)*). Default value is `0.1`. If `warping.method='dilation'` value of `t.max` is ignored.

`m.max` scalar: `m.max` controls the maximal allowed dilation, at each iteration, in the alignment procedure. `m.max` must be such that *0<m.max<1* (e.g., `m.max=0.1` means that dilation is bounded, at each iteration, between *1-0.1* and *1+0.1*). Default value is `0.1`. If `warping.method='shift'` value of `m.max` is ignored.

`n.out` scalar: the desired length of the abscissa for computation of the similarity indexes and the centers. Default value is `round(1.1*grid.size)`.

`tol` scalar: the algorithm stops when the increment of similarity of each function with respect to the corrispondent center is lower than `tol`. Default value is `0.01`.

`fence` boolean: if `fence=TRUE` a control is activated at the end of each iteration. The aim of the control is to avoid shift/dilation outlighers with respect to their computed distributions. If `fence=TRUE` the running time can increase considerably. Default value of `fence` is TRUE.

`iter.max` scalar: maximum number of iterations in the k-mean alignment cycle. Default value is `100`.

`show.iter` boolean: if `show.iter=TRUE` kma shows the current iteration of the algorithm. Default value is `FALSE`.

`nstart` scalar: number of initializations with different seeds. Default value is `2`. This parameter is used only if `center.method` is `'k-medoids'`. When `center.method = 'k-means'` one initialization is performed.

`return.all` boolean: if `return.all=TRUE` the results of all the `nstart` initializations are returned; the output is a list of length `nstart`. If `return.all=FALSE` only the best result is provided (the one with higher mean similarity if `similarity.method` is `'d0.pearson'` or `'d1.pearson'`, or the one with lower distance if `similarity.method` is `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'` or `'d1.L2.centered'`). Default value 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. Default value is `FALSE`

Value

The function output is a list containing the following elements:

`iterations` scalar: total number of iterations performed by kma function.

`x` as input.

`y0` as input.

`y1` as input.

`n.clust` as input.

`warping.method` as input.

`similarity.method` as input.

`center.method` as input.

`x.center.orig` vector *n.out*: abscissa of the original center.

`y0.center.orig` matrix *1* X *n.out*: the unique row contains the evaluations of the original function center. If `warping.method='k-means'` there are two scenarios: if `similarity.method='d0.pearson'` or `'d0.L2'` or `d0.L2.centered` the original function center is computed via loess procedure applied to original data; if `similarity.method='d1.pearson'` or `'d1.L2'` or `d1.L2.centered` it is computed by integration of first derivatives center `y1.center.orig` (the integration constant is computed minimizing the sum of the weighed L2 distances between the center and the original functions). If `warping.method='k-medoids'` the original function center is the medoid of original functions.

`y1.center.orig` matrix *1* X *n.out*: the unique row contains the evaluations of the original function first derivatives center. If `warping.method='k-means'` the original center is computed via loess procedure applied to original function first derivatives. If `warping.method='k-medoids'` the original center is the medoid of original functions.

`similarity.orig` vector: original similarities between the original functions and the original center.

`x.final` matrix *n.func* X *grid.size*: aligned abscissas.

`n.clust.final` scalar: final number of clusters. Note that, when `center.method='k.means'`, the parameter `n.clust.final` may differ from initial number of clusters (i.e., from `n.clust`) if some clusters are found to be empty. In this case a warning message is issued.

`x.centers.final` vector *n.out*: abscissas of the final function centers and/or of the final function first derivatives centers.

`y0.centers.final` matrix *n.clust.final* X *n.out*: rows contain the evaluations of the final functions centers. `y0.centers.final` is `NULL` if `y0` is not given as input.

`y1.centers.final` matrix *n.clust.final* X *n.out*: rows contains the evaluations of the final derivatives centers. `y1.centers.final` is `NULL` if the chosen similarity measure does not concern function first derivatives.

`labels` vector: cluster assignments.

`similarity.final` vector: similarities between each function and the center of the cluster the function is assigned to.

`dilation.list` list: dilations obtained at each iteration of kma function.

`shift.list` list: shifts obtained at each iteration of kma function.

`dilation`	vector: dilation applied to the original abscissas `x` to obtain the aligned abscissas `x.final`.
`shift`	vector: shift applied to the original abscissas `x` to obtain the aligned abscissas `x.final`.

Author(s)

Alice Parodi, Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

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

`kma.compare`, `kma.similarity`, `fdakma`, `kma.data`, `kma.show.results`

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

## Not run:
# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')

# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')


# Example: result of kma function with 2 clusters,
# allowing affine transformation for the abscissas
# and considering 'd1.pearson' as similarity.method.
kma_example <- kma (
  x=x, y0=y0, y1=y1, n.clust = 2,
  warping.method = 'affine',
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21)
  )

kma.show.results(kma_example)
```

```
names(kma_example)

# Labels assigned to each function
kma_example$labels

# Total shifts and dilations applied to the original
# abscissa to obtain the aligned abscissa
kma_example$shift
kma_example$dilation

## End(Not run)
```

`kma.compare` *kma.compare runs kma with different numbers of clusters and different warping methods.*

Description

In kma.compare the user can specify multiple values for n.clust and warping.method. kma.compare runs the K-Mean Alignment algorithm (kma function) for all couples of specified values of `n.clust` and `warping.method`.

Usage

```
kma.compare(x, y0 = NULL, y1 = NULL, n.clust = c(1, 2),
warping.method = c("NOalignment", "shift", "dilation", "affine"),
similarity.method = "d1.pearson", center.method = "k-means", seeds = NULL,
optim.method = "L-BFGS-B", span = 0.15, t.max = 0.1, m.max = 0.1, n.out = NULL,
tol = 0.01, fence = TRUE, iter.max = 100, show.iter = 0, plot.graph = 0,
nstart = 2, return.all = FALSE)
```

Arguments

`x`	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.
`y0`	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. Default value of `y0` is `NULL`. The parameter `y0` must be provided if the chosen `similarity.method` concerns original functions.

`y1` matrix *n.func* X *grid.size* or array *n.func* X *grid.size* X *d*: evaluations of the set of original functions first derivatives on the abscissa grid `x`. Default value of `y1` is `NULL`. The parameter `y1` must be provided if the chosen `similarity.method` concerns original function first derivatives.

`n.clust` vector: `n.clust` contains the numbers of clusters with which kma.compare runs kma function. Default value is `c(1,2)`. See details.

`warping.method` vector: `warping.method` contains the types of alignment with which kma.compare runs kma function. See details.

`similarity.method` character: required similarity measure. Possible choices are: `'d0.pearson'`, `'d1.pearson'`, `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'`, `'d1.L2.centered'`. Default value is `'d1.pearson'`. See kma.similarity for details.

`center.method` character: type of clustering method to be used. Possible choices are: `'k-means'` and `'k-medoids'`. Default value is `'k-means'`.

`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; if not all the values of `seeds` are provided, the ones not introduced are randomly chosen among the `n.func` original functions. If `seeds=NULL` all the centers are randomly chosen. Default value of `seeds` is `NULL`.

`optim.method` character: optimization method chosen to find the best warping functions at each iteration. Possible choices are: `'L-BFGS-B'` and `'SANN'`. See optim function for details. Default method is `'L-BFGS-B'`.

`span` scalar: the span to be used for the loess procedure in the center estimation step when `center.method='k-means'`. Default value is 0.15. If `center.method='k-medoids'` value of `span` is ignored.

`t.max` scalar: `t.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<t.max<1* (e.g., `t.max=0.1` means that shift is bounded, at each iteration, between *-0.1*range(x)* and *+0.1*range(x)*). Default value is `0.1`. If `warping.method='dilation'` value of `t.max` is ignored.

`m.max` scalar: `m.max` controls the maximal allowed dilation, at each iteration, in the alignment procedure. `m.max` must be such that *0<m.max<1* (e.g., `m.max=0.1` means that dilation is bounded, at each iteration, between *1-0.1* and *1+0.1*). Default value is `0.1`. If `warping.method='shift'` value of `m.max` is ignored.

`n.out` scalar: the desired length of the abscissa for computation of the similarity indexes and the centers. Default value is `round(1.1*grid.size)`.

`tol` scalar: the algorithm stops when the increment of similarity of each function with respect to the corrispondent center is lower than `tol`. Default value is `0.01`.

`fence` boolean: if `fence=TRUE` a control is activated at the end of each iteration. The aim of the control is to avoid shift/dilation outlighers with respect to their computed distributions. If `fence=TRUE` the running time can increase considerably. Default value of `fence` is TRUE.

`iter.max` scalar: maximum number of iterations in the k-mean alignment cycle. Default value is `100`.

`show.iter` boolean: if `show.iter=TRUE` kma shows the current iteration of the algorithm. Default value is `FALSE`.

`plot.graph` boolean: if `plot.graph=TRUE`, kma.compare plots a graphic with the means of similarity indexes as ordinate and the number of clusters as abscissa. Default value is `FALSE`.

`nstart` scalar: number of initializations with different seeds. Default value is `1`.

`return.all` boolean: if `return.all=TRUE` the results of all the `nstart` initializations are return; the output is a list of length `nstart`. If `return.all=FALSE` only the best result is provided (the one with higher mean similarity if `similarity.method` is `'d0.pearson'` or `'d1.pearson'`, or the one with lower similarity if `similarity.method` is `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'` or `'d1.L2.centered'`), Default value is `FALSE`.

Details

Example of use: if `n.clust=c(1,2,3)` and `warping.method=c('shift','affine')`, kma.compare runs kma function with number of clusters equal to `1`, `2` and `3` using `warping.method='shift'` and `warping.method='affine'`.

Value

The function output is a list containing the following elements:

`Result.NOalignment`
list of outputs of kma function with `warping.type='NOalignment'`. The sublist `Result.NOalignment[[k]]` corresponds to the results when number of clusters is `n.clust[k]`. Note that if `'NOalignment'` is not chosen as `warping.type`, then `Result.NOalignment` will be `NULL`.

`Result.shift` list of outputs of kma function with `warping.type='shift'`. The sublist `Result.shift[[k]]` corresponds to the results when number of clusters is `n.clust[k]`. Note that if `'shift'` is not chosen as `warping.type`, then `Result.shift` will be `NULL`.

`Result.dilation`
list of outputs of kma function with `warping.type='dilation'`. The sublist `Result.dilation[[k]]` corresponds to the results when number of clusters is `n.clust[k]`. Note that if `'dilation'` is not chosen as `warping.type`, then `Result.dilation` will be `NULL`.

`Result.affine` list of outputs of kma function with `warping.type='affine'`. The sublist `Result.affine[[k]]` corresponds to the results when number of clusters is `n.clust[k]`. Note that if `'affine'` is not chosen as `warping.type`, then `Result.affine` will be `NULL`.

`n.clust` as input.

`mean.similarity.NOalignment`
vector: mean similarity indexes of functions after running kma function with all elements of `n.clust` and `warping.type='NOalignment'`. `mean.similarity.NOalignment` contains the ordinates of the black curve ("without alignment" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

`mean.similarity.shift`
: vector: mean similarity indexes of curves after running kma function with all elements of `n.clust` and `warping.type='shift'`. `mean.similarity.shift` contains the ordinates of the blue curve ("shift" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

`mean.similarity.dilation`
: vector: mean similarity indexes of curves after running kma function with all elements of `n.clust` and `warping.type='dilation'`. `mean.similarity.dilation` contains the ordinates of the green curve ("dilation" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

`mean.similarity.affine`
: vector: mean similarity indexes of curves after running kma function with all elements of `n.clust` and `warping.type='affine'`. `mean.similarity.affine` contains the ordinates of the orange curve ("affine" in the legend) of the output graphic of the kma.compare function (if `plot.graph=1`).

Author(s)

Alice Parodi, Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

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

`kma`, `kma.similarity`, `fdakma`, `kma.data`, `kma.show.results`

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

## Not run:
# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')

# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')
```

```
# Example: results of kma function with 3 different
# numbers of clusters (1,2,3) combined with four alignment
# methods ('NOalignment' by default, 'shift', 'dilation',
# 'affine') and considering 'd1.pearson' as similarity.method.
kma.compare_example <- kma.compare (
  x=x, y0=y0, y1=y1, n.clust = 1:3,
  warping.method = c('affine'),
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21,30),
  plot.graph=1)

names (kma.compare_example)

# To see results for kma function with n.clust=2
# and warping.method='affine'.
kma.show.results (kma.compare_example$Result.affine[[2]])


# Labels assigned to each function for the
# kma function with n.clust=2 and warping.method='affine'.
kma.compare_example$Result.affine[[2]]$labels

## End(Not run)
```

`kma.data` *Simulated Data*

Description

kma.data is a functional dataset displaying both amplitude and phase variability.

Usage

```
data(kma.data)
```

Format

List of 3 elements:

$x : abscissa values where each function is evaluated

$y0: evaluations of the original functions on the abscissa grid `kma.data$x`

$y1: evaluations of the original function first derivatives on the abscissa grid `kma.data$x`.

Author(s)

Alice Parodi, Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

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

kma.compare, kma.similarity, fdakma, kma, kma.show.results

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

## Not run:
# Plot of original functions
matplot(t(x),t(y0), type='l', xlab='x', ylab='orig.func')
title ('Original functions')

# Plot of original function first derivatives
matplot(t(x),t(y1), type='l', xlab='x', ylab='orig.deriv')
title ('Original function first derivatives')

## End(Not run)
```

---

kma.show.results *Auxiliary function plotting results of kma function.*

Description

kma.show.results graphically shows the output results of kma function or one of the outuput results of kma.compare.

The function generates the following plots:

- d plots of original functions (one for each of the d dimensions of the input functions -for unidimensional functions d is 1-) with center (if `y0` is given as input in kma/kma.compare function).

- d plots of aligned functions with centers (if `y0` is given as input in kma/kma.compare function).

- d plots of original function first derivatives with center (if `y1` is given as input and the chosen `similarity.method` concerns function first derivatives).

- d plots of aligned function first derivatives with centers (if `y1` is given as input and the chosen `similarity.method` concerns function first derivatives).

- Plot of warping functions.

- Boxplot of similarity/dissimilarity indexes of original and aligned functions.

Usage

```
kma.show.results(Result, lwd.functions = 1, lwd.centers = 3)
```

Arguments

`Result`	list: output of kma function or one of the outputs of kma.compare functions.
`lwd.functions`	integer: the desired line width of the curves. Default value is 1.
`lwd.centers`	integer: the desired line width of the centers. Default value is 3.

Author(s)

Alice Parodi, Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

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

kma, kma.compare, kma.similarity, fdakma, kma.data

Examples

```
data(kma.data)

x <- kma.data$x # abscissas
y0 <- kma.data$y0 # evaluations of original functions
y1 <- kma.data$y1 # evaluations of original function first derivatives

## Not run:
# kma function with 2 clusters, allowing affine
# transformation for the abscissas and considering
# 'd1.pearson' as similarity.method.
kma.show.results_example1 <- kma (
  x=x, y0=y0, y1=y1, n.clust = 2,
  warping.method = 'affine',
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21)
)

# Example: kma.show.results shows the results of kma function
kma.show.results(kma.show.results_example1)


# Example using outputs of kma.compare function
```

```
# Results of kma function with 3 different
# numbers of clusters (1,2,3) combined with four alignment
# methods ('NOalignment' by default, 'shift', 'dilation',
# 'affine') and considering 'd1.pearson' as similarity.method.
kma.show.results_example2 <- kma.compare (
  x=x, y0=y0, y1=y1, n.clust = 1:3,
  warping.method = c('affine'),
  similarity.method = 'd1.pearson',
  center.method = 'k-means',
  seeds = c(1,21,30),
  plot.graph=1)

names (kma.show.results_example2)

# To see results for kma function with n.clust=2
# and warping.method='affine'.
kma.show.results (kma.show.results_example2$Result.affine[[2]])

# Labels assigned to each function for the
# kma function with n.clust=2 and warping.method='affine'.
kma.show.results_example2$Result.affine[[2]]$labels

## End(Not run)
```

`kma.similarity` *Similarity/dissimilarity index between two functions*

Description

kma.similarity computes a similarity/dissimilarity measure between two functions f and g. Users can choose among different types of measures.

Usage

```
kma.similarity(x.f = NULL, y0.f = NULL, y1.f = NULL,
x.g = NULL, y0.g = NULL, y1.g = NULL, similarity.method, unif.grid = TRUE)
```

Arguments

`x.f`	vector *length.f*: abscissa grid where function f and his first derivatives f' is evaluated. *length.f*: numbrt of abscissa values where f is evaluated. `x.f` must always be provided.
`y0.f`	vector *length.f* or matrix *length.f* X d: evaluations of function f on the abscissa grid `x.f`. *length.f*: number of abscissa values where f is evaluated. d (only if f and g are multidimensional) number of function's components, i.e. f is d-dimensional curve. Default value of `y0.f` is `NULL`. The vectory`0.f` must be provided if the chosen `similarity.method` concerns original functions.

`y1.f` vector *length.f* or matrix *length.f* X d: evaluations of f first derivative, i.e., f', on the abscissa grid `x.f`. Default value of `y1.f` is `NULL`. The vector `y1.f` must be provided if the chosen `similarity.method` concerns function first derivatives.

`x.g` vector *length.g*: abscissa grid where function g and his first derivatives g' is evaluated. *length.g*: numbrt of abscissa values where g is evaluated. `x.g` must always be provided.

`y0.g` vector *length.g* or matrix *length.g* X d: evaluations of function g on the abscissa grid `x.g`. *length.g*: number of abscissa values where g is evaluated. d (only if f and g are multidimensional) number of function's components, i.e. g is d-dimensional curve. Default value of `y0.g` is `NULL`. The vector `y0.g` must be provided if the chosen `similarity.method` concerns original functions.

`y1.g` vector *length.g* or matrix *length.g* X d: evaluations of g first derivative, i.e., g', on the abscissa grid `x.g`. Default value is of `y1.g` `NULL`. The vector `y1.g` must be provided if the chosen `similarity.method` concerns function first derivatives.

`similarity.method`
character: similarity/dissimilarity between f and g. Possible choices are: `'d0.pearson'`, `'d1.pearson'`, `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'`, `'d1.L2.centered'`. Default value is `'d1.pearson'`. See details.

`unif.grid` boolean: if equal to `TRUE` the similarity measure is computed over an uniform grid built in the intersection domain of the two functions, that is an additional discretization is performed. If equal to `FALSE` the additional discretization is not performed, so the functions are supposed to be already defined on the same abscissa grid and the grid is supposed to be fine enough to well compute similarity.

Details

We report the list of the currently available similarities/dissimilarities. Note that all norms and inner products are computed over D, that is the intersection of the domains of f and g. $\overline{f}$ and $\overline{g}$ denote the mean value, respectively, of functions f and g.

1. `'d0.pearson'`: this similarity measure is the cosine of the angle between the two functions f and g.

$$\frac{< f, g >_{L^2}}{\|f\|_{L^2}\|g\|_{L^2}}$$

2. `'d1.pearson'`: this similarity measure is the cosine of the angle between the two function derivatives f' and g'.

$$\frac{< f', g' >_{L^2}}{\|f'\|_{L^2}\|g'\|_{L^2}}$$

3. `'d0.L2'`: this dissimilarity measure is the L2 distance of the two functions f and g normalized by the length of the common domain D.

$$\frac{\|f - g\|_{L^2}}{|D|}$$

4. `'d1.L2'`: this dissimilarity measure is the L2 distance of the two function first derivatives f' and g' normalized by the length of the common domain D.

$$\frac{\|f' - g'\|_{L^2}}{|D|}$$

5. `'d0.L2.centered'`: this dissimilarity measure is the L2 distance of $f - \overline{f}$ and $g - \overline{g}$ normalized by the length of the common domain D.

$$\frac{\|(f - \overline{f}) - (g - \overline{g})\|_{L^2}}{|D|}$$

6. `'d1.L2.centered'`: this dissimilarity measure is the L2 distance of $f' - \overline{f'}$ and $g' - \overline{g'}$ normalized by the length of the common domain D.

$$\frac{\|(f' - \overline{f'}) - (g' - \overline{g'})\|_{L^2}}{|D|}$$

For multidimensional functions, if `similarity.method='d0.pearson'` or `'d1.pearson'` the similarity/dissimilarity measure is computed via the average of the indexes in all directions.

The coherence properties specified in Sangalli et al. (2010) implies that if `similarity.method` is set to `'d0.L2'`, `'d1.L2'`, `'d0.L2.centered'` or `'d1.L2.centered'`, value of `warping.method` must be `'shift'` or `'NOalignment'`. If `similarity.method` is set to `'d0.pearson'` or `'d1.pearson'` all values for `warping.method` are allowed.

Value

scalar: similarity/dissimilarity measure between the two functions f and g computed via the similarity/dissimilarity measure specified.

Author(s)

Alice Parodi, Mirco Patriarca, Laura Sangalli, Piercesare Secchi, Simone Vantini, Valeria Vitelli.

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

`kma`, `kma.compare`, `kma.show.results`, `fdakma`, `kma.data`

Examples

```
data(kma.data)

x.f <- kma.data$x # abscissas of f and f'
x.g <- kma.data$x # abscissas of g and g'

y0.f <- kma.data$y0[1,] # evaluations of f on the abscissa grid x.f
y1.f <- kma.data$y1[1,] # evaluations of f' on the abscissa grid x.f
y0.g <- kma.data$y0[3,] # evaluations of g on the abscissa grid x.g
y1.g <- kma.data$y1[3,] # evaluations of g' on the abscissa grid x.g


## Not run:
  # Plot of the two functions f and g
  plot(t(x.g),t(y0.g), type='l', xlab='x', ylab='y',, col='red')
  points(t(x.f),t(y0.f), type='l')
  title ('f and g')
  legend('bottomleft', legend=c('f','g'),
         col=c('black','red'), lty=c(1,1), cex = 0.5)

## End(Not run)

# Example: 'd0.pearson'
kma.similarity (x.f=x.f, y0.f=y0.f, x.g=x.g,  y0.g=y0.g, similarity.method='d0.pearson')

# Example: 'd0.L2'
kma.similarity (x.f=x.f, y0.f=y0.f, x.g=x.g,  y0.g=y0.g, similarity.method='d0.L2')

## Not run:
  # Plot of the two function first derivatives f' and g'
  plot(t(x.g),t(y1.g), type='l', xlab='x', ylab='y', col='red')
  points(t(x.f),t(y1.f), type='l')
  title ("f' and  g'")
  legend('bottomleft', legend=c("f'","g'"),
         col=c('black','red'), lty=c(1,1), cex = 0.5)

## End(Not run)

# Example: 'd1.pearson'
kma.similarity (x.f=x.f, y1.f=y1.f, x.g=x.g,  y1.g=y1.g, similarity.method='d1.pearson')

# Example: 'd1.L2'
kma.similarity (x.f=x.f, y1.f=y1.f, x.g=x.g,  y1.g=y1.g, similarity.method='d1.L2')
```

Index