Package ‘fdasrvf’

October 5, 2024

Type Package

Title Elastic Functional Data Analysis

Version 2.3.3

Date 2024-10-04

Description Performs alignment, PCA, and modeling of multidimensional and unidimensional functions using the square-root velocity framework (Srivastava et al., 2011 <doi:10.48550/arXiv.1103.3817> and Tucker et al., 2014 <DOI:10.1016/j.csda.2012.12.001>). This framework allows for elastic analysis of functional data through phase and amplitude separation.

License GPL-3

LazyData TRUE

Imports cli, coda, doParallel, fields, foreach, lpSolve, Matrix, mvtnorm, Rcpp, rlang, tolerance, viridisLite

Suggests covr, interp, plot3D, plot3Drgl, rgl, testthat (>= 3.0.0), withr

Depends R (>= 4.3.0),

RoxygenNote 7.3.2

Encoding UTF-8

Config/testthat/edition 3

LinkingTo Rcpp, RcppArmadillo

URL `https://github.com/jdtuck/fdasrvf_R`

BugReports `https://github.com/jdtuck/fdasrvf_R/issues`

NeedsCompilation yes

Author J. Derek Tucker [aut, cre] (<`https://orcid.org/0000-0001-8844-2169`>), Aymeric Stamm [ctb] (<`https://orcid.org/0000-0002-8725-3654`>)

Maintainer J. Derek Tucker <`jdtuck@sandia.gov`>

Repository CRAN

Date/Publication 2024-10-04 22:20:02 UTC

Contents

`align_fPCA` *Group-wise function alignment and PCA Extractions*

Description

This function aligns a collection of functions while extracting principal components.

Usage

```
align_fPCA(
  f,
  time,
  num_comp = 3L,
  showplot = TRUE,
  smooth_data = FALSE,
  sparam = 25L,
  parallel = FALSE,
  cores = NULL,
  max_iter = 51L,
  lambda = 0
)
```

Arguments

`f`	A numeric matrix of shape $M \times N$ specifying a sample of N 1-dimensional curves observed on a grid of size M.
`time`	A numeric vector of length M specifying the grid on which functions `f` have been evaluated.
`num_comp`	An integer value specifying the number of principal components to extract. Defaults to `3L`.
`showplot`	A boolean specifying whether to display plots along the way. Defaults to `TRUE`.
`smooth_data`	A boolean specifying whether to smooth data using box filter. Defaults to `FALSE`.
`sparam`	An integer value specifying the number of times to apply box filter. Defaults to `25L`. This argument is only used if `smooth_data == TRUE`.
`parallel`	A boolean specifying whether computations should run in parallel. Defaults to `FALSE`.

cores	An integer value specifying the number of cores to use for parallel computations. Defaults to NULL in which case it uses all available cores but one. This argument is only used when parallel == TRUE.
max_iter	An integer value specifying the maximum number of iterations. Defaults to 51L.
lambda	A numeric value specifying the elasticity. Defaults to 0.0.

Value

A list with the following components:

- f0: A numeric matrix of shape $M \times N$ storing the original functions;
- fn: A numeric matrix of the same shape as f0 storing the aligned functions;
- qn: A numeric matrix of the same shape as f0 storing the aligned SRSFs;
- q0: A numeric matrix of the same shape as f0 storing the SRSFs of the original functions;
- mqn: A numeric vector of length M storing the mean SRSF;
- gam: A numeric matrix of the same shape as f0 storing the estimated warping functions;
- vfpca: A list storing information about the vertical PCA with the following components:
  - q_pca: A numeric matrix of shape $(M+1) \times 5 \times \mathrm{num\_comp}$ storing the first 3 principal directions in SRSF space; the first dimension is $M+1$ because, in SRSF space, the original functions are represented by the SRSF and the initial value of the functions.
  - f_pca: A numeric matrix of shape $M \times 5 \times \mathrm{num\_comp}$ storing the first 3 principal directions in original space;
  - latent: A numeric vector of length $M+1$ storing the singular values of the SVD decomposition in SRSF space;
  - coef: A numeric matrix of shape $N \times \mathrm{num\_comp}$ storing the scores of the N original functions on the first num_comp principal components;
  - U: A numeric matrix of shape $(M+1) \times (M+1)$ storing the eigenvectors associated with the SVD decomposition in SRSF space.
- Dx: A numeric vector of length max_iter storing the value of the cost function at each iteration.

References

Tucker, J. D., Wu, W., Srivastava, A., Generative models for functional data using phase and amplitude separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
## Not run:
  out <- align_fPCA(simu_data$f, simu_data$time)

## End(Not run)
```

`beta` *MPEG7 Curve Dataset*

Description

Contains the MPEG7 curve data set.

Usage

```
beta
```

Format

`beta`:

An array of shape $2 \times 100 \times 65 \times 20$ storing a sample of 20 curves from R to R^2 distributed in 65 different classes, evaluated on a grid of size 100.

`bootTB` *Tolerance Bound Calculation using Bootstrap Sampling*

Description

This function computes tolerance bounds for functional data containing phase and amplitude variation using bootstrap sampling

Usage

```
bootTB(
  f,
  time,
  a = 0.05,
  p = 0.99,
  B = 500,
  no = 5,
  Nsamp = 100,
  parallel = TRUE
)
```

Arguments

`f`	matrix of functions
`time`	vector describing time sampling
`a`	confidence level of tolerance bound (default = 0.05)
`p`	coverage level of tolerance bound (default = 0.99)

B	number of bootstrap samples (default = 500)
no	number of principal components (default = 5)
Nsamp	number of functions per bootstrap (default = 100)
parallel	enable parallel processing (default = TRUE)

Value

Returns a list containing

amp	amplitude tolerance bounds
ph	phase tolerance bounds

References

J. D. Tucker, J. R. Lewis, C. King, and S. Kurtek, “A Geometric Approach for Computing Tolerance Bounds for Elastic Functional Data,” Journal of Applied Statistics, 10.1080/02664763.2019.1645818, 2019.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Jung, S. L. a. S. (2016). "Combined Analysis of Amplitude and Phase Variations in Functional Data." arXiv:1603.01775.

Examples

```
## Not run:
  out1 <- bootTB(simu_data$f, simu_data$time)

## End(Not run)
```

---

boxplot.fdawarp *Functional Boxplot*

Description

This function computes the required statistics for building up a boxplot of the aligned functional data. Since the process of alignment provides separation of phase and amplitude variability, the computed boxplot can focus either on amplitude variability or phase variability.

Usage

```
## S3 method for class 'fdawarp'
boxplot(
  x,
  variability_type = c("amplitude", "phase"),
  alpha = 0.05,
  range = 1,
```

```
  what = c("stats", "plot", "plot+stats"),
  ...
)

## S3 method for class 'ampbox'
boxplot(x, ...)

## S3 method for class 'phbox'
boxplot(x, ...)

## S3 method for class 'curvebox'
boxplot(x, ...)
```

Arguments

x	An object of class fdawarp typically produced by time_warping() or of class ampbox or phbox typically produced by boxplot.fdawarp().
variability_type	A string specifying which kind of variability should be displayed in the boxplot. Choices are "amplitude" or "phase". Defaults to "amplitude".
alpha	A numeric value specifying the quantile value. Defaults to 0.05 which uses the 95% quantile.
range	A positive numeric value specifying how far the plot whiskers extend out from the box. The whiskers extend to the most extreme data point which is no more than range times the interquartile range from the box. Defaults to 1.0.
what	A string specifying what the function should return. Choices are "plot", "stats" or "plot+stats". Defaults to "stats".
...	Unused here.

Details

The function boxplot.fdawarp() returns optionally an object of class either ampbox if variability_type = "amplitude" or phbox if variability_type = "phase". S3 methods specialized for objects of these classes are provided as well to avoid re-computation of the boxplot statistics.

Value

If what contains stats, a list containing the computed statistics necessary for drawing the boxplot. Otherwise, the function simply draws the boxplot and no object is returned.

Examples

```
## Not run:
out <- time_warping(simu_data$f, simu_data$time)
boxplot(out, what = "stats")

## End(Not run)
```

calc_shape_dist *Elastic Shape Distance*

Description

Calculates the elastic shape distance between two curves beta1 and beta2.

Usage

```
calc_shape_dist(
  beta1,
  beta2,
  mode = "O",
  alignment = TRUE,
  rotation = TRUE,
  scale = TRUE,
  include.length = FALSE,
  lambda = 0
)
```

Arguments

beta1	A numeric matrix of shape $L \times M$ specifying an L-dimensional curve evaluated on M sample points.
beta2	A numeric matrix of shape $L \times M$ specifying an L-dimensional curve evaluated on M sample points. This curve will be aligned to beta1.
mode	A character string specifying whether the input curves should be considered open (mode == "O") or closed (mode == "C"). Defaults to "O".
alignment	A boolean value specifying whether the curves should be aligned before computing the distance matrix. Defaults to TRUE.
rotation	A boolean specifying whether the distance should be made invariant by rotation. Defaults to TRUE.
scale	A boolean specifying whether the distance should be made invariant by scaling. This is effectively achieved by making SRVFs having unit length and switching to an appropriate metric on the sphere between normalized SRVFs. Defaults to TRUE.
include.length	A boolean specifying whether to include information about the actual length of the SRVFs in the metric when using normalized SRVFs to achieve scale invariance. This only applies if scale == TRUE. Defaults to FALSE.
lambda	A numeric value specifying the weight of a penalty term that constraints the warping function to be not too far from the identity. Defaults to 0.0.

Details

Distances are computed between the SRVFs of the original curves. Hence, they are intrinsically invariant to position. The user can then choose to make distances invariant to rotation and scaling by setting `rotation` and `scale` accordingly. Distances can also be made invariant to reparametrization by setting `alignment = TRUE`, in which case curves are aligned using an appropriate action of the diffeomorphism group on the space of SRVFs.

Value

A list with the following components:

- `d`: the amplitude (geodesic) distance;
- `dx`: the phase distance;
- `q1`: the SRVF of `beta1`;
- `q2n`: the SRVF of `beta2` after alignment and possible optimal rotation and scaling;
- `beta1`: the input curve `beta1`;
- `beta2n`: the input curve `beta2` after alignment and possible optimal rotation and scaling.
- `R`: the optimal rotation matrix that has been applied to the second curve;
- `gam`: the optimal warping function that has been applied to the second curve;
- `betascale`: the optimal scaling factor that has been applied to the second curve.

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Kurtek, S., Srivastava, A., Klassen, E., and Ding, Z. (2012), "Statistical Modeling of Curves Using Shapes and Related Features," Journal of the American Statistical Association, 107, 1152–1165.

Srivastava, A., Klassen, E. P. (2016). Functional and shape data analysis, 1. New York: Springer.

Examples

```
out <- calc_shape_dist(beta[, , 1, 1], beta[, , 1, 4])
```

`curve2srvf` *Converts a curve to its SRVF representation*

Description

Converts a curve to its SRVF representation

Usage

```
curve2srvf(beta, is_derivative = FALSE)
```

Arguments

- `beta` — A numeric matrix of size $L \times M$ specifying a curve on an L-dimensional space observed on an evenly spaced grid of $[0, 1]$ of length M.
- `is_derivative` — A boolean value specifying whether the input $beta$ is the derivative of the original curve. Defaults to `FALSE`.

Value

A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the SRVF of the original curve at s.

Examples

```
curve2srvf(beta[, , 1, 1])
```

`curve_boxplot` *Curve Boxplot*

Description

This function computes the required statistics for building up a boxplot of the aligned curve data. The computed boxplot focuses on the aligned curves.

Usage

```
curve_boxplot(
  x,
  alpha = 0.05,
  range = 1,
  what = c("stats", "plot", "plot+stats"),
  ...
)
```

Arguments

- `x` — An object of class `fdacurve` typically produced by `curve_srvf_align()`
- `alpha` — A numeric value specifying the quantile value. Defaults to 0.05 which uses the 95% quantile.
- `range` — A positive numeric value specifying how far the plot whiskers extend out from the box. The whiskers extend to the most extreme data point which is no more than `range` times the interquartile range from the box. Defaults to `1.0`.
- `what` — A string specifying what the function should return. Choices are `"plot"`, `"stats"` or `"plot+stats"`. Defaults to `"plot"`.
- `...` — Unused here.

Details

The function returns optionally an object of class either curvebox

Value

If what contains stats, a list containing the computed statistics necessary for drawing the boxplot. Otherwise, the function simply draws the boxplot and no object is returned.

Examples

```
## Not run:
out <- curve_srvf_align(beta[, , 1, ], ms="median")
curve_boxplot(out, what = "stats")

## End(Not run)
```

---

curve_depth *Calculates elastic depth for a set of curves*

Description

This functions calculates the elastic depth between set of curves. If the curves are describing multidimensional functional data, then rotated == FALSE and mode == 'O'

Usage

```
curve_depth(beta, mode = "O", rotated = TRUE, scale = TRUE, parallel = FALSE)
```

Arguments

beta	Array of sizes $n \times T \times N$ for N curves of dimension n evaluated on a grid of T points
mode	Open ("O") or Closed ("C") curves
rotated	Include rotation (default = TRUE)
scale	scale curves to unit length (default = TRUE)
parallel	run computation in parallel (default = TRUE)

Value

Returns a list containing

amp	amplitude depth
phase	phase depth

References

T. Harris, J. D. Tucker, B. Li, and L. Shand, "Elastic depths for detecting shape anomalies in functional data," Technometrics, 10.1080/00401706.2020.1811156, 2020.

Examples

```
data("mpeg7")
# note: use more shapes and iterations, small for speed
out = curve_depth(beta[,,1,1:2])
```

`curve_dist` *Distance Matrix Computation*

Description

Computes the pairwise distance matrix between a set of curves using the elastic shape distance as computed by `calc_shape_dist()`.

Usage

```
curve_dist(
  beta,
  mode = "O",
  alignment = TRUE,
  rotation = FALSE,
  scale = FALSE,
  include.length = FALSE,
  lambda = 0,
  ncores = 1L
)
```

Arguments

beta	A numeric array of shape $L \times M \times N$ specifying the set of N curves of length M in L-dimensional space.
mode	A character string specifying whether the input curves should be considered open (`mode == "O"`) or closed (`mode == "C"`). Defaults to `"O"`.
alignment	A boolean value specifying whether the curves should be aligned before computing the distance matrix. Defaults to `TRUE`.
rotation	A boolean specifying whether the distance should be made invariant by rotation. Defaults to `TRUE`.
scale	A boolean specifying whether the distance should be made invariant by scaling. This is effectively achieved by making SRVFs having unit length and switching to an appropriate metric on the sphere between normalized SRVFs. Defaults to `TRUE`.
include.length	A boolean specifying whether to include information about the actual length of the SRVFs in the metric when using normalized SRVFs to achieve scale invariance. This only applies if `scale == TRUE`. Defaults to `FALSE`.
lambda	A numeric value specifying the weight of a penalty term that constraints the warping function to be not too far from the identity. Defaults to `0.0`.

ncores	An integer value specifying the number of cores to use for parallel computation. If ncores is greater than the number of available cores, a warning is issued and the maximum number of available cores is used. Defaults to 1L.

Details

Distances are computed between the SRVFs of the original curves. Hence, they are intrinsically invariant to position. The user can then choose to make distances invariant to rotation and scaling by setting rotation and scale accordingly. Distances can also be made invariant to reparametrization by setting alignment = TRUE, in which case curves are aligned using an appropriate action of the diffeomorphism group on the space of SRVFs.

Value

A list of two objects, Da and Dp, each of class dist containing the amplitude and phase distances, respectively.

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Kurtek, S., Srivastava, A., Klassen, E., and Ding, Z. (2012), "Statistical Modeling of Curves Using Shapes and Related Features," Journal of the American Statistical Association, 107, 1152–1165.

Srivastava, A., Klassen, E. P. (2016). Functional and shape data analysis, 1. New York: Springer.

Examples

```
out <- curve_dist(beta[, , 1, 1:4])
```

---

curve_geodesic *Form geodesic between two curves*

Description

Form geodesic between two curves using Elastic Method

Usage

```
curve_geodesic(beta1, beta2, k = 5)
```

Arguments

beta1	curve 1, provided as a matrix of dimensions $n \times T$ for n-dimensional curve evaluated on T sample points
beta2	curve 2, provided as a matrix of dimensions $n \times T$ for n-dimensional curve evaluated on T sample points
k	number of curves along geodesic (default 5)

Value

a list containing

geod curves along geodesic (n,T,k)

geod_q srvf's along geodesic

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
out <- curve_geodesic(beta[, , 1, 1], beta[, , 1, 5])
```

---

curve_karcher_cov *Curve Karcher Covariance*

Description

Calculate Karcher Covariance of a set of curves

Usage

```
curve_karcher_cov(v, len = NA)
```

Arguments

v array of sizes $n \times T \times N$ for N shooting vectors of dimension n evaluated on a grid of T points

len lengths of curves (default = NA)

Value

K covariance matrix

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
out <- curve_karcher_mean(beta[, , 1, 1:2], maxit = 2)
# note: use more shapes, small for speed
K <- curve_karcher_cov(out$v)
```

`curve_karcher_mean` *Karcher Mean of Curves*

Description

Calculates Karcher mean or median of a collection of curves using the elastic square-root velocity (SRVF) framework.

Usage

```
curve_karcher_mean(
  beta,
  mode = "O",
  rotated = TRUE,
  scale = TRUE,
  lambda = 0,
  maxit = 20,
  ms = c("mean", "median"),
  ncores = 1L
)
```

Arguments

beta	A numeric array of shape $L \times M \times N$ specifying an N-sample of L-dimensional curves evaluated on M points.
mode	A character string specifying whether the input curves should be considered open (`mode == "O"`) or closed (`mode == "C"`). Defaults to `"O"`.
rotated	A boolean specifying whether to make the metric rotation-invariant. Defaults to `FALSE`.
scale	A boolean specifying whether the distance should be made invariant by scaling. This is effectively achieved by making SRVFs having unit length and switching to an appropriate metric on the sphere between normalized SRVFs. Defaults to `TRUE`.
lambda	A numeric value specifying the elasticity. Defaults to `0.0`.
maxit	An integer value specifying the maximum number of iterations. Defaults to `20L`.
ms	A character string specifying whether the Karcher mean ("mean") or Karcher median ("median") is returned. Defaults to `"mean"`.
ncores	An integer value specifying the number of cores to use for parallel computation. Defaults to `1L`. The maximum number of available cores is determined by the **parallel** package. One core is always left out to avoid overloading the system.

Value

An object of class `fdacurve` which is a list with the following components:

- beta: A numeric array of shape $L \times M \times N$ specifying the N-sample of L-dimensional curves evaluated on M points. The curves might be slightly different from the input curves as they have been centered.
- `mu`: A numeric array of shape $L \times M$ specifying the Karcher mean or median of the SRVFs of the input curves.
- `type`: A character string specifying whether the Karcher mean or median is returned.
- `betamean`: A numeric array of shape $L \times M$ specifying the Karcher mean or median of the input curves.
- `v`: A numeric array of shape $L \times M \times N$ specifying the shooting vectors.
- `q`: A numeric array of shape $L \times M \times N$ specifying the SRVFs of the input curves.
- `E`: A numeric vector of shape N specifying XXX (TO DO)
- `cent`: A numeric array of shape $L \times M$ specifying the centers of the input curves.
- `len`: A numeric vector of shape N specifying the length of the input curves.
- `len_q`: A numeric vector of shape N specifying the length of the SRVFs of the input curves.
- `qun`: A numeric vector of shape $maxit$ specifying the consecutive values of the cost function.
- `mean_scale`: A numeric value specifying the mean length of the input curves.
- `mean_scale_q`: A numeric value specifying the mean length of the SRVFs of the input curves.
- `mode`: A character string specifying the mode of the input curves.
- `rotated`: A boolean specifying whether the metric is rotation-invariant.
- `scale`: A boolean specifying whether the metric is scale-invariant.
- `ms`: A character string specifying whether the Karcher mean or median is returned.
- `lambda`: A numeric value specifying the elasticity ??? (TO DO).
- `rsamps`: ??? (TO DO).

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in Euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
out <- curve_karcher_mean(beta[, , 1, 1:2], maxit = 2)
# note: use more shapes, small for speed
```

curve_pair_align *Pairwise align two curves*

Description

This function aligns to curves using Elastic Framework

Usage

```
curve_pair_align(beta1, beta2, mode = "O", rotation = TRUE, scale = TRUE)
```

Arguments

beta1	curve 1, provided as a matrix of dimensions $n \times T$ for n-dimensional curve evaluated on T sample points
beta2	curve 2, provided as a matrix of dimensions $n \times T$ for n-dimensional curve evaluated on T sample points
mode	Open ("O") or Closed ("C") curves
rotation	Include rotation (default = TRUE)
scale	scale curves to unit length (default = TRUE)

Value

a list containing

beta2n	aligned curve 2 to 1
q2n	aligned srvf 2 to 1
gam	warping function
q1	srvf of curve 1
beta1	centered curve 1
beta2	centered curve 2
R	rotation matrix
tau	seed

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
out <- curve_pair_align(beta[, , 1, 1], beta[, , 1, 5])
```

curve_principal_directions
Curve PCA

Description

Calculate principal directions of a set of curves

Usage

```
curve_principal_directions(v, K, mu, len = NA, no = 3, N = 5, mode = "O")
```

Arguments

v	array of sizes $n \times T \times N1$ for $N1$ shooting vectors of dimension n evaluated on a grid of T points
K	matrix of sizes $nT \times nT$ of covariance matrix
mu	matrix of sizes $n \times T$ of mean srvf
len	length of original curves (default = NA)
no	number of components
N	number of samples on each side of mean
mode	Open ("O") or Closed ("C") curves

Value

Returns a list containing

s	singular values
U	singular vectors
coef	principal coefficients
pd	principal directions

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
out <- curve_karcher_mean(beta[, , 1, 1:2], maxit = 2)
# note: use more shapes, small for speed
K <- curve_karcher_cov(out$v)
out <- curve_principal_directions(out$v, K, out$mu)
```

`curve_srvf_align` *Align Curves*

Description

Aligns a collection of curves using the elastic square-root velocity (srvf) framework. If the curves are describing multidimensional functional data, then `rotated == FALSE` and `mode == 'O'`

Usage

```
curve_srvf_align(
  beta,
  mode = "O",
  rotated = TRUE,
  scale = TRUE,
  lambda = 0,
  maxit = 20,
  ms = "mean",
  parallel = TRUE
)
```

Arguments

`beta`	Array of sizes $n \times T \times N$ for N curves of dimension n evaluated on a grid of T points
`mode`	Open (`"O"`) or Closed (`"C"`) curves
`rotated`	Optimize over rotation (default = `TRUE`)
`scale`	scale curves to unit length (default = `TRUE`)
`lambda`	A numeric value specifying the elasticity. Defaults to `0.0`.
`maxit`	maximum number of iterations
`ms`	string defining whether the Karcher mean ("mean") or Karcher median ("median") is returned (default = "mean")
`parallel`	A boolean specifying whether to run calculations in parallel. Defaults to `TRUE`.

Value

An object of class `fdacurve` which is a list with the following components:

- `mu`: mean srvf
- `beta`: centered curves
- `betamean`: mean or median curve
- `betan`: aligned curves
- `qn`: aligned srvfs
- `type`: string indicating whether mean or median is returned

- v: shooting vectors
- q: array of srvfs
- gam: array of warping functions
- cent: centers of original curves
- len: length of curves
- len_q: length of srvfs
- mean_scale: mean length
- mean_scale_q: mean length srvf
- E: energy
- qun: cost function

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
data("mpeg7")
# note: use more shapes and iterations, small for speed
out = curve_srvf_align(beta[,,1,1:2],maxit=2,parallel=FALSE)
```

---

curve_to_q   *Curve to SRVF Space*

Description

This function computes the SRVF of a given curve. The function also outputs the length of the original curve and the L^2 norm of the SRVF.

Usage

```
curve_to_q(beta, scale = TRUE)
```

Arguments

beta	A numeric matrix of shape $L \times M$ specifying a curve on an L-dimensional space observed on M points.
scale	A boolean value specifying whether the output SRVF function should be scaled to the hypersphere of L^2. When scale is TRUE, the length of the original curve becomes irrelevant. Defaults to TRUE.

Value

A list with the following components:

- `q`: A numeric matrix of the same shape as the input matrix `beta` storing the (possibly scaled) SRVF of the original curve `beta`;
- `len`: A numeric value specifying the length of the original curve;
- `lenq`: A numeric value specifying the L^2 norm of the SRVF of the original curve.

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in Euclidean spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence, **33**(7), 1415-1428.

Examples

```
q <- curve_to_q(beta[, , 1, 1])$q
```

`discrete2curve` *Converts a curve from matrix to functional data object*

Description

Converts a curve from matrix to functional data object

Usage

```
discrete2curve(beta)
```

Arguments

`beta`	A numeric matrix of size $L \times M$ specifying a curve on an L-dimensional space observed on an evenly spaced grid of $[0, 1]$ of length M.

Value

A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the original curve at s.

Examples

```
discrete2curve(beta[, , 1, 1])
```

discrete2warping *Converts a warping function from vector to functional data object*

Description

Converts a warping function from vector to functional data object

Usage

```
discrete2warping(gam)
```

Arguments

gam	A numeric vector of length M specifying a warping function on an evenly spaced grid of $[0, 1]$ of length M.

Value

A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the original warping function at s.

Examples

```
discrete2warping(toy_warp$gam[, 1])
```

elastic.depth *Calculates elastic depth*

Description

This functions calculates the elastic depth between set of functions in R^1

Usage

```
elastic.depth(f, time, lambda = 0, pen = "roughness", parallel = FALSE)
```

Arguments

f	matrix of sizes $M \times N$, specifying values of N function of M time points
time	vector of length M, specifying the sample points of functions
lambda	controls amount of warping (default = `0`)
pen	alignment penalty (default = `"roughness"`) options are second derivative (`"roughness"`), geodesic distance from id (`"geodesic"`), and norm from id (`"norm"`)
parallel	run computation in parallel (default = `TRUE`)

Value

Returns a list containing

amp	amplitude depth
phase	phase depth
amp_dist	amplitude distance matrix
phs_dist	phase distance matrix

References

T. Harris, J. D. Tucker, B. Li, and L. Shand, "Elastic depths for detecting shape anomalies in functional data," Technometrics, 10.1080/00401706.2020.1811156, 2020.

Examples

```
depths <- elastic.depth(simu_data$f[, 1:4], simu_data$time)
```

---

elastic.distance *Calculates two elastic distance*

Description

This functions calculates the distances between functions in R^1 D_y and D_x, where function 1 is aligned to function 2

Usage

```
elastic.distance(f1, f2, time, lambda = 0, pen = "roughness")
```

Arguments

f1	sample function 1, provided as a vector of length M
f2	sample function 2, provided as a vector of length M
time	sample points of functions, provided as a vector of length M
lambda	controls amount of warping (default = `0`)
pen	alignment penalty (default = `"roughness"`) options are second derivative (`"roughness"`), geodesic distance from id (`"geodesic"`), and norm from id (`"norm"`)

Value

Returns a list containing

Dy	amplitude distance
Dx	phase distance

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
distances <- elastic.distance(
  f1 = simu_data$f[, 1],
  f2 = simu_data$f[, 2],
  time = simu_data$time
)
```

---

elastic.logistic *Elastic Logistic Regression*

Description

This function identifies a logistic regression model with phase-variability using elastic methods

Usage

```
elastic.logistic(
  f,
  y,
  time,
  B = NULL,
  df = 20,
  max_itr = 20,
  smooth_data = FALSE,
  sparam = 25,
  parallel = FALSE,
  cores = 2
)
```

Arguments

f	matrix (N x M) of M functions with N samples
y	vector of size M labels (1/-1)
time	vector of size N describing the sample points
B	matrix defining basis functions (default = NULL)
df	scalar controlling degrees of freedom if B=NULL (default=20)
max_itr	scalar number of iterations (default=20)
smooth_data	smooth data using box filter (default = F)

`sparam`	number of times to apply box filter (default = 25)
`parallel`	enable parallel mode using `foreach` and `doParallel` package
`cores`	set number of cores to use with `doParallel` (default = 2)

Value

Returns a list containing

`alpha`	model intercept
`beta`	regressor function
`fn`	aligned functions - matrix (N x M) of M functions with N samples
`qn`	aligned srvfs - similar structure to fn
`gamma`	warping functions - similar structure to fn
`q`	original srvf - similar structure to fn
`B`	basis matrix
`b`	basis coefficients
`Loss`	logistic loss
`type`	model type ('logistic')

References

Tucker, J. D., Wu, W., Srivastava, A., Elastic Functional Logistic Regression with Application to Physiological Signal Classification, Electronic Journal of Statistics (2014), submitted.

`elastic.lpcr.regression`

Elastic logistic Principal Component Regression

Description

This function identifies a logistic regression model with phase-variability using elastic pca

Usage

```
elastic.lpcr.regression(
  f,
  y,
  time,
  pca.method = "combined",
  no = 5,
  smooth_data = FALSE,
  sparam = 25
)
```

Arguments

f	matrix (N x M) of M functions with N samples
y	vector of size M labels
time	vector of size N describing the sample points
pca.method	string specifying pca method (options = "combined", "vert", or "horiz", default = "combined")
no	scalar specify number of principal components (default=5)
smooth_data	smooth data using box filter (default = F)
sparam	number of times to apply box filter (default = 25)

Value

Returns a lpcr object containing

alpha	model intercept
b	regressor vector
y	label vector
warp_data	fdawarp object of aligned data
pca	pca object of principal components
Loss	logistic loss
pca.method	string specifying pca method used

References

J. D. Tucker, J. R. Lewis, and A. Srivastava, "Elastic Functional Principal Component Regression," Statistical Analysis and Data Mining, 10.1002/sam.11399, 2018.

---

elastic.mlogistic *Elastic Multinomial Logistic Regression*

Description

This function identifies a multinomial logistic regression model with phase-variability using elastic methods

Usage

```
elastic.mlogistic(
  f,
  y,
  time,
  B = NULL,
  df = 20,
```

```
  max_itr = 20,
  smooth_data = FALSE,
  sparam = 25,
  parallel = FALSE,
  cores = 2
)
```

Arguments

f	matrix (N x M) of M functions with N samples
y	vector of size M labels (1,2,...,m) for m classes
time	vector of size N describing the sample points
B	matrix defining basis functions (default = NULL)
df	scalar controlling degrees of freedom if B=NULL (default=20)
max_itr	scalar number of iterations (default=20)
smooth_data	smooth data using box filter (default = F)
sparam	number of times to apply box filter (default = 25)
parallel	enable parallel mode using `foreach` and `doParallel` package
cores	set number of cores to use with `doParallel` (default = 2)

Value

Returns a list containing

alpha	model intercept
beta	regressor function
fn	aligned functions - matrix (N x M) of M functions with N samples
qn	aligned srvfs - similar structure to fn
gamma	warping functions - similar structure to fn
q	original srvf - similar structure to fn
B	basis matrix
b	basis coefficients
Loss	logistic loss
type	model type ('mlogistic')

References

Tucker, J. D., Wu, W., Srivastava, A., Elastic Functional Logistic Regression with Application to Physiological Signal Classification, Electronic Journal of Statistics (2014), submitted.

`elastic.mlpcr.regression`
Elastic Multinomial logistic Principal Component Regression

Description

This function identifies a multinomial logistic regression model with phase-variability using elastic pca

Usage

```
elastic.mlpcr.regression(
  f,
  y,
  time,
  pca.method = "combined",
  no = 5,
  smooth_data = FALSE,
  sparam = 25
)
```

Arguments

`f`	matrix (N x M) of M functions with N samples
`y`	vector of size M labels
`time`	vector of size N describing the sample points
`pca.method`	string specifying pca method (options = "combined", "vert", or "horiz", default = "combined")
`no`	scalar specify number of principal components (default=5)
`smooth_data`	smooth data using box filter (default = F)
`sparam`	number of times to apply box filter (default = 25)

Value

Returns a mlpcr object containing

`alpha`	model intercept
`b`	regressor vector
`y`	label vector
`Y`	Coded labels
`warp_data`	fdawarp object of aligned data
`pca`	pca object of principal components
`Loss`	logistic loss
`pca.method`	string specifying pca method used

References

J. D. Tucker, J. R. Lewis, and A. Srivastava, "Elastic Functional Principal Component Regression," Statistical Analysis and Data Mining, 10.1002/sam.11399, 2018.

`elastic.pcr.regression` *Elastic Linear Principal Component Regression*

Description

This function identifies a regression model with phase-variability using elastic pca

Usage

```
elastic.pcr.regression(
  f,
  y,
  time,
  pca.method = "combined",
  no = 5,
  smooth_data = FALSE,
  sparam = 25,
  parallel = F,
  C = NULL
)
```

Arguments

`f`	matrix (N x M) of M functions with N samples
`y`	vector of size M responses
`time`	vector of size N describing the sample points
`pca.method`	string specifying pca method (options = "combined", "vert", or "horiz", default = "combined")
`no`	scalar specify number of principal components (default = 5)
`smooth_data`	smooth data using box filter (default = F)
`sparam`	number of times to apply box filter (default = 25)
`parallel`	run in parallel (default = F)
`C`	scale balance parameter for combined method (default = NULL)

Value

Returns a pcr object containing

`alpha`	model intercept
`b`	regressor vector
`y`	response vector
`warp_data`	fdawarp object of aligned data
`pca`	pca object of principal components
`SSE`	sum of squared errors
`pca.method`	string specifying pca method used

References

J. D. Tucker, J. R. Lewis, and A. Srivastava, "Elastic Functional Principal Component Regression," Statistical Analysis and Data Mining, 10.1002/sam.11399, 2018.

`elastic.prediction` *Elastic Prediction from Regression Models*

Description

This function performs prediction from an elastic regression model with phase-variability

Usage

```
elastic.prediction(f, time, model, y = NULL, smooth_data = FALSE, sparam = 25)
```

Arguments

`f`	matrix (N x M) of M functions with N samples
`time`	vector of size N describing the sample points
`model`	list describing model from elastic regression methods
`y`	responses of test matrix f (default=NULL)
`smooth_data`	smooth data using box filter (default = F)
`sparam`	number of times to apply box filter (default = 25)

Value

Returns a list containing

`y_pred`	predicted values of f or probabilities depending on model
`SSE`	sum of squared errors if linear
`y_labels`	labels if logistic model
`PC`	probability of classification if logistic

References

Tucker, J. D., Wu, W., Srivastava, A., Elastic Functional Logistic Regression with Application to Physiological Signal Classification, Electronic Journal of Statistics (2014), submitted.

---

elastic.regression *Elastic Linear Regression*

Description

This function identifies a regression model with phase-variability using elastic methods

Usage

```
elastic.regression(
  f,
  y,
  time,
  B = NULL,
  lam = 0,
  df = 20,
  max_itr = 20,
  smooth_data = FALSE,
  sparam = 25,
  parallel = FALSE,
  cores = 2
)
```

Arguments

f	matrix (N x M) of M functions with N samples
y	vector of size M responses
time	vector of size N describing the sample points
B	matrix defining basis functions (default = NULL)
lam	scalar regularization parameter (default=0)
df	scalar controlling degrees of freedom if B=NULL (default=20)
max_itr	scalar number of iterations (default=20)
smooth_data	smooth data using box filter (default = F)
sparam	number of times to apply box filter (default = 25)
parallel	enable parallel mode using `foreach` and `doParallel` package
cores	set number of cores to use with `doParallel` (default = 2)

Value

Returns a list containing

alpha	model intercept
beta	regressor function
fn	aligned functions - matrix (N x M) of M functions with N samples
qn	aligned srvfs - similar structure to fn
gamma	warping functions - similar structure to fn
q	original srvf - similar structure to fn
B	basis matrix
b	basis coefficients
SSE	sum of squared errors
type	model type ('linear')

References

Tucker, J. D., Wu, W., Srivastava, A., Elastic Functional Logistic Regression with Application to Physiological Signal Classification, Electronic Journal of Statistics (2014), submitted.

---

elastic_amp_change_ff *Elastic Amplitude Changepoint Detection*

Description

This function identifies a amplitude changepoint using a fully functional approach

Usage

```
elastic_amp_change_ff(
  f,
  time,
  d = 1000,
  h = 0,
  smooth_data = FALSE,
  sparam = 25,
  showplot = TRUE
)
```

Arguments

f	matrix (N x M) of M functions with N samples
time	vector of size N describing the sample points
d	number of monte carlo iterations of Brownian Bridge (default = 1000)
h	window selection of long range covariance function (default = 0)
smooth_data	smooth data using box filter (default = F)
sparam	number of times to apply box filter (default = 25)
showplot	show results plots (default = T)

Value

Returns a list object containing

pvalue	p value
change	indice of changepoint
DataBefore	functions before changepoint
DataAfter	functions after changepoint
MeanBefore	mean function before changepoint
MeanAfter	mean function after changepoint
WarpingBefore	warping functions before changepoint
WarpingAfter	warping functions after changepoint
WarpingMeanBefore	mean warping function before changepoint
WarpingMeanAfter	mean warping function after changepoint
change_fun	amplitude change function
Sn	test statistic values
mu	mean srsfs
mu_f	mean functions

References

J. D. Tucker and D. Yarger, "Elastic Functional Changepoint Detection of Climate Impacts from Localized Sources", Envirometrics, 10.1002/env.2826, 2023.

elastic_change_fpca *Elastic Changepoint Detection*

Description

This function identifies changepoints using a functional PCA

Usage

```
elastic_change_fpca(
  f,
  time,
  pca.method = "combined",
  pc = 0.95,
  d = 1000,
  n_pcs = 5,
  smooth_data = FALSE,
  sparam = 25,
  showplot = TRUE
)
```

Arguments

f	matrix (N x M) of M functions with N samples
time	vector of size N describing the sample points
pca.method	string specifying pca method (options = "combined", "vert", or "horiz", default = "combined")
pc	percentage of cummulation explained variance (default = 0.95)
d	number of monte carlo iterations of Brownian Bridge (default = 1000)
n_pcs	scalar specify number of principal components (default = 5)
smooth_data	smooth data using box filter (default = F)
sparam	number of times to apply box filter (default = 25)
showplot	show results plots (default = T)

Value

Returns a list object containing

pvalue	p value
change	indice of changepoint
DataBefore	functions before changepoint
DataAfter	functions after changepoint
MeanBefore	mean function before changepoint

MeanAfter	mean function after changepoint
WarpingBefore	warping functions before changepoint
WarpingAfter	warping functions after changepoint
WarpingMeanBefore	mean warping function before changepoint
WarpingMeanAfter	mean warping function after changepoint
change_fun	amplitude change function
Sn	test statistic values

References

J. D. Tucker and D. Yarger, "Elastic Functional Changepoint Detection of Climate Impacts from Localized Sources", Envirometrics, 10.1002/env.2826, 2023.

---

elastic_ph_change_ff *Elastic Phase Changepoint Detection*

Description

This function identifies a phase changepoint using a fully functional approach

Usage

```
elastic_ph_change_ff(
  f,
  time,
  d = 1000,
  h = 0,
  smooth_data = FALSE,
  sparam = 25,
  showplot = TRUE
)
```

Arguments

f	matrix (N x M) of M functions with N samples
time	vector of size N describing the sample points
d	number of monte carlo iterations of Brownian Bridge (default = 1000)
h	window selection of long range covariance function (default = 0)
smooth_data	smooth data using box filter (default = F)
sparam	number of times to apply box filter (default = 25)
showplot	show results plots (default = T)

Value

Returns a list object containing

pvalue	p value
change	indice of changepoint
DataBefore	functions before changepoint
DataAfter	functions after changepoint
MeanBefore	mean function before changepoint
MeanAfter	mean function after changepoint
WarpingBefore	warping functions before changepoint
WarpingAfter	warping functions after changepoint
WarpingMeanBefore	mean warping function before changepoint
WarpingMeanAfter	mean warping function after changepoint
change_fun	amplitude change function
Sn	test statistic values
mu	mean shooting vectors

References

J. D. Tucker and D. Yarger, "Elastic Functional Changepoint Detection of Climate Impacts from Localized Sources", Envirometrics, 10.1002/env.2826, 2023.

`fdasrvf` *Elastic Functional Data Analysis*

Description

A library for functional data analysis using the square root velocity framework which performs pair-wise and group-wise alignment as well as modeling using functional component analysis.

Author(s)

Maintainer: J. Derek Tucker <jdtuck@sandia.gov> (ORCID)

Other contributors:

- Aymeric Stamm <aymeric.stamm@math.cnrs.fr> (ORCID) [contributor]

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using Fisher-Rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative models for functional data using phase and amplitude separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

J. D. Tucker, W. Wu, and A. Srivastava, Phase-amplitude separation of proteomics data using extended Fisher-Rao metric, Electronic Journal of Statistics, Vol 8, no. 2. pp 1724-1733, 2014.

J. D. Tucker, W. Wu, and A. Srivastava, "Analysis of signals under compositional noise with applications to SONAR data," IEEE Journal of Oceanic Engineering, Vol 29, no. 2. pp 318-330, Apr 2014.

Tucker, J. D. 2014, Functional Component Analysis and Regression using Elastic Methods. Ph.D. Thesis, Florida State University.

Robinson, D. T. 2012, Function Data Analysis and Partial Shape Matching in the Square Root Velocity Framework. Ph.D. Thesis, Florida State University.

Kurtek, S., Srivastava, A., Klassen, E., and Ding, Z. (2012), "Statistical Modeling of Curves Using Shapes and Related Features," Journal of the American Statistical Association, 107, 1152–1165.

Huang, W. 2014, Optimization Algorithms on Riemannian Manifolds with Applications. Ph.D. Thesis, Florida State University.

Cheng, W., Dryden, I. L., and Huang, X. (2016). Bayesian registration of functions and curves. Bayesian Analysis, 11(2), 447-475.

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Cheng, W., Dryden, I. L., and Huang, X. (2016). Bayesian registration of functions and curves. Bayesian Analysis, 11(2), 447-475.

W. Xie, S. Kurtek, K. Bharath, and Y. Sun, A geometric approach to visualization of variability in functional data, Journal of American Statistical Association 112 (2017), pp. 979-993.

Lu, Y., R. Herbei, and S. Kurtek, 2017: Bayesian registration of functions with a Gaussian process prior. Journal of Computational and Graphical Statistics, 26, no. 4, 894–904.

Lee, S. and S. Jung, 2017: Combined analysis of amplitude and phase variations in functional data. arXiv:1603.01775, 1–21.

J. D. Tucker, J. R. Lewis, and A. Srivastava, "Elastic Functional Principal Component Regression," Statistical Analysis and Data Mining, vol. 12, no. 2, pp. 101-115, 2019.

J. D. Tucker, J. R. Lewis, C. King, and S. Kurtek, "A Geometric Approach for Computing Tolerance Bounds for Elastic Functional Data," Journal of Applied Statistics, 10.1080/02664763.2019.1645818, 2019.

T. Harris, J. D. Tucker, B. Li, and L. Shand, "Elastic depths for detecting shape anomalies in functional data," Technometrics, 10.1080/00401706.2020.1811156, 2020.

J. D. Tucker and D. Yarger, "Elastic Functional Changepoint Detection of Climate Impacts from Localized Sources", Envirometrics, 10.1002/env.2826, 2023.

See Also

Useful links:

- https://github.com/jdtuck/fdasrvf_R
- Report bugs at https://github.com/jdtuck/fdasrvf_R/issues

---

function_group_warp_bayes
Bayesian Group Warping

Description

This function aligns a set of functions using Bayesian SRSF framework

Usage

```
function_group_warp_bayes(
  f,
  time,
  iter = 50000,
  powera = 1,
  times = 5,
  tau = ceiling(times * 0.04),
  gp = seq(dim(f)[2]),
  showplot = TRUE
)
```

Arguments

f	matrix (N x M) of M functions with N samples
time	sample points of functions
iter	number of iterations (default = 150000)
powera	Dirichlet prior parameter (default 1)
times	factor of length of subsample points to look at (default = 5)
tau	standard deviation of Normal prior for increment (default ceil(times*.4))
gp	number of colors in plots (defaults seq(dim(f)[2]))
showplot	shows plots of functions (default = T)

Value

Returns a list containing

f0	original functions
f_q	f aligned quotient space

gam_q	warping functions quotient space
f_a	f aligned ambient space
gam_a	warping ambient space
qmn	mean srsf

References

Cheng, W., Dryden, I. L., and Huang, X. (2016). Bayesian registration of functions and curves. Bayesian Analysis, 11(2), 447-475.

Examples

```
## Not run:
  out <- function_group_warp_bayes(simu_data$f, simu_data$time)

## End(Not run)
```

---

function_mean_bayes *Bayesian Karcher Mean Calculation*

Description

This function calculates karcher mean of functions using Bayesian method

Usage

```
function_mean_bayes(f, time, times = 5, group = 1:dim(f)[2], showplot = TRUE)
```

Arguments

f	matrix (N x M) of M functions with N samples
time	sample points of functions
times	factor of length of subsample points to look at (default = 5)
group	(defaults 1:dim(f)[2])
showplot	shows plots of functions (default = T)

Value

Returns a list containing

distfamily	dist matrix
match.matrix	matrix of warping functions
position	position
mu_5	function mean
rtmatrix	rtmatrix

sumdist	sumdist
qt.fitted	aligned srsf functions
estimator	estimator
estimator2	estimator2
regfuncs	registered functions

References

Cheng, W., Dryden, I. L., and Huang, X. (2016). Bayesian registration of functions and curves. Bayesian Analysis, 11(2), 447-475.

Examples

```
## Not run:
  out <- function_mean_bayes(simu_data$f, simu_data$time)

## End(Not run)
```

---

f_plot *Plot functional data*

Description

This function plots functional data on a time grid

Usage

```
f_plot(t, f)
```

Arguments

t	A numeric vector of length M specifying the common grid on which all curves f have been observed.
f	A numeric matrix of shape $M \times N$ specifying a sample of N curves observed on a grid of size M.

`f_to_srvf` *Transformation to SRVF Space*

Description

This function transforms functions in R^1 from their original functional space to the SRVF space.

Usage

```
f_to_srvf(f, time)
```

Arguments

`f` Either a numeric vector of a numeric matrix or a numeric array specifying the functions that need to be transformed.

- If a vector, it must be of shape M and it is interpreted as a single 1-dimensional curve observed on a grid of size M.
- If a matrix, it must be of shape $M \times N$. In this case, it is interpreted as a sample of N curves observed on a grid of size M, unless $M = 1$ in which case it is interpreted as a single 1-dimensional curve observed on a grid of size M.

`time` A numeric vector of length M specifying the grid on which the functions are evaluated.

Value

A numeric array of the same shape as the input array `f` storing the SRVFs of the original curves.

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using Fisher-Rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative models for functional data using phase and amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
q <- f_to_srvf(simu_data$f, simu_data$time)
```

gam_to_h *map warping function to tangent space at identity*

Description

map warping function to tangent space at identity

Usage

```
gam_to_h(gam, smooth = FALSE)
```

Arguments

gam	Either a numeric vector of a numeric matrix or a numeric array specifying the warping functions
smooth	Apply smoothing before gradient

Value

A numeric array of the same shape as the input array gamma storing the shooting vectors of gamma obtained via finite differences.

gam_to_v *map warping function to tangent space at identity*

Description

map warping function to tangent space at identity

Usage

```
gam_to_v(gam, smooth = FALSE)
```

Arguments

gam	Either a numeric vector of a numeric matrix or a numeric array specifying the warping functions
smooth	Apply smoothing before gradient

Value

A numeric array of the same shape as the input array gamma storing the shooting vectors of gamma obtained via finite differences.

`gauss_model` *Gaussian model of functional data*

Description

This function models the functional data using a Gaussian model extracted from the principal components of the srvfs

Usage

```
gauss_model(warp_data, n = 1, sort_samples = FALSE)
```

Arguments

`warp_data`	`fdawarp` object from time_warping of aligned data
`n`	number of random samples (n = 1)
`sort_samples`	sort samples (default = F)

Value

Returns a `fdawarp` object containing

`fs`	random aligned samples
`gams`	random warping function samples
`ft`	random function samples

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
out1 <- gauss_model(simu_warp, n = 10)
```

`get_distance_matrix` *Computes the distance matrix between a set of shapes.*

Description

Computes the distance matrix between a set of shapes.

Usage

```
get_distance_matrix(
  qfuns,
  alignment = FALSE,
  rotation = FALSE,
  scale = FALSE,
  lambda = 0,
  M = 200L,
  parallel_setup = 1L
)
```

Arguments

qfuns	A list of functions representing the shapes. Each function should be a SRVF. See `curve2srvf()` for more details on how to obtain the SRVF of a shape.
alignment	A boolean value specifying whether the two SRVFs should be optimally aligned before computing the distance. Defaults to `FALSE`.
rotation	A boolean value specifying whether the two SRVFs should be optimally rotated before computing the distance. Defaults to `FALSE`.
scale	A boolean value specifying whether the two SRVFs should be projected onto the Hilbert sphere before computing the distance. Defaults to `FALSE`.
lambda	A numeric value specifying the regularization parameter for the optimal alignment. Defaults to `0`.
M	An integer value specifying the number of points to use when searching for the optimal rotation and alignment. Defaults to `200L`.
parallel_setup	An integer value specifying the number of cores to use for parallel computing or an object of class `cluster`. In the latter case, it is used as is for parallel computing. If `parallel_setup` is `1L`, then no parallel computing will be used. The maximum number of cores to use is the number of available cores minus 1. Defaults to `1L`.

Value

An object of class `dist` containing the distance matrix between the shapes.

Examples

```
N <- 4L
srvfs <- lapply(1:N, \(n) curve2srvf(beta[, , 1, n]))
get_distance_matrix(srvfs, parallel_setup = 1L)
```

---

get_hilbert_sphere_distance

Computes the geodesic distance between two SRVFs on the Hilbert sphere

---

Description

Computes the geodesic distance between two SRVFs on the Hilbert sphere

Usage

```
get_hilbert_sphere_distance(q1fun, q2fun)
```

Arguments

q1fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the first SRVF at s.
q2fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the second SRVF at s.

Value

A numeric value storing the geodesic distance between the two SRVFs.

Examples

```
q1 <- curve2srvf(beta[, , 1, 1])
q2 <- curve2srvf(beta[, , 1, 2])
q1p <- to_hilbert_sphere(q1)
q2p <- to_hilbert_sphere(q2)
get_hilbert_sphere_distance(q1p, q2p)
```

get_identity_warping *Computes the identity warping function*

Description

Computes the identity warping function

Usage

```
get_identity_warping()
```

Value

A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the identity warping function at s.

Examples

```
get_identity_warping()
```

get_l2_distance *Computes the $L\hat{}2$ distance between two SRVFs*

Description

Computes the L^2 distance between two SRVFs

Usage

```
get_l2_distance(q1fun, q2fun, method = "quadrature")
```

Arguments

q1fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the first SRVF at s.
q2fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the second SRVF at s.
method	A character string specifying the method to use for computing the L^2 distance. Options are `"quadrature"` and `"trapz"`. Defaults to `"quadrature"`.

Value

A numeric value storing the L^2 distance between the two SRVFs.

Examples

```
q1 <- curve2srvf(beta[, , 1, 1])
q2 <- curve2srvf(beta[, , 1, 2])
get_l2_distance(q1, q2)
```

---

get_l2_inner_product *Computes the $L\hat{}2$ inner product between two SRVFs*

Description

Computes the L^2 inner product between two SRVFs

Usage

```
get_l2_inner_product(q1fun, q2fun)
```

Arguments

q1fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the first SRVF at s.
q2fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the second SRVF at s.

Value

A numeric value storing the L^2 inner product between the two SRVFs.

Examples

```
q1 <- curve2srvf(beta[, , 1, 1])
q2 <- curve2srvf(beta[, , 1, 2])
get_l2_inner_product(q1, q2)
```

---

get_shape_distance *Computes the distance between two shapes*

Description

Computes the distance between two shapes

Usage

```
get_shape_distance(
  q1fun,
  q2fun,
  alignment = FALSE,
  rotation = FALSE,
  scale = FALSE,
  lambda = 0,
  M = 200L
)
```

Arguments

q1fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the first SRVF at s.
q2fun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the second SRVF at s.
alignment	A boolean value specifying whether the two SRVFs should be optimally aligned before computing the distance. Defaults to `FALSE`.
rotation	A boolean value specifying whether the two SRVFs should be optimally rotated before computing the distance. Defaults to `FALSE`.
scale	A boolean value specifying whether the two SRVFs should be projected onto the Hilbert sphere before computing the distance. Defaults to `FALSE`.
lambda	A numeric value specifying the regularization parameter for the optimal alignment. Defaults to `0`.
M	An integer value specifying the number of points to use when searching for the optimal rotation and alignment. Defaults to `200L`.

Value

A numeric value storing the distance between the two shapes.

Examples

```
q1 <- curve2srvf(beta[, , 1, 1])
q2 <- curve2srvf(beta[, , 1, 2])
get_shape_distance(q1, q2)
```

---

get_warping_distance *Computes the distance between two warping functions*

Description

Computes the distance between two warping functions

Usage

```
get_warping_distance(gam1fun, gam2fun)
```

Arguments

`gam1fun`	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the first warping function at s.
`gam2fun`	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the second warping function at s.

Value

A numeric value storing the distance between the two warping functions.

Examples

```
gam1 <- discrete2warping(toy_warp$gam[, 1])
gam2 <- discrete2warping(toy_warp$gam[, 2])
get_warping_distance(gam1, gam2)
get_warping_distance(gam1, get_identity_warping())
```

`gradient` *Gradient using finite differences*

Description

This function computes the gradient of `f` using finite differences.

Usage

```
gradient(f, binsize, multidimensional = FALSE)
```

Arguments

`f` Either a numeric vector of a numeric matrix or a numeric array specifying the curve(s) that need to be differentiated.

- If a vector, it must be of shape M and it is interpreted as a single 1-dimensional curve observed on a grid of size M.
- If a matrix and `multidimensional == FALSE`, it must be of shape $M \times N$. In this case, it is interpreted as a sample of N curves observed on a grid of size M, unless $M = 1$ in which case it is interpreted as a single 1-dimensional curve observed on a grid of size M.
- If a matrix and `multidimensional == TRUE`,it must be of shape $L \times M$ and it is interpreted as a single L-dimensional curve observed on a grid of size M.
- If a 3D array, it must be of shape $L \times M \times N$ and it is interpreted as a sample of N L-dimensional curves observed on a grid of size M.

binsize A numeric value specifying the size of the bins for computing finite differences.

multidimensional
A boolean specifying if the curves are multi-dimensional. This is useful when f is provided as a matrix to determine whether it is a single multi-dimensional curve or a collection of uni-dimensional curves. Defaults to FALSE.

Value

A numeric array of the same shape as the input array f storing the gradient of f obtained via finite differences.

Examples

```
out <- gradient(simu_data$f[, 1], mean(diff(simu_data$time)))
```

---

growth_vel *Berkeley Growth Velocity Dataset*

Description

Combination of both boys and girls growth velocity from the Berkley dataset.

Usage

```
growth_vel
```

Format

growth_vel:

A list with two components:

- f: A numeric matrix of shape 69×93 storing a sample of size $N = 93$ of curves evaluated on a grid of size $M = 69$.
- time: A numeric vector of size $M = 69$ storing the grid on which the curves f have been evaluated.

horizFPCA *Horizontal Functional Principal Component Analysis*

Description

This function calculates vertical functional principal component analysis on aligned data

Usage

```
horizFPCA(warp_data, no = 3, var_exp = NULL, ci = c(-1, 0, 1), showplot = TRUE)
```

Arguments

`warp_data`	fdawarp object from time_warping of aligned data
`no`	number of principal components to extract
`var_exp`	compute no based on value percent variance explained (example: 0.95) will override `no`
`ci`	geodesic standard deviations (default = c(-1,0,1))
`showplot`	show plots of principal directions (default = T)

Value

Returns a hfpca object containing

`gam_pca`	warping functions principal directions
`psi_pca`	srvf principal directions
`latent`	latent values
`U`	eigenvectors
`vec`	shooting vectors
`mu`	Karcher Mean

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
hfpca <- horizFPCA(simu_warp, no = 3)
```

h_to_gam *map shooting vector to warping function at identity*

Description

map shooting vector to warping function at identity

Usage

```
h_to_gam(h)
```

Arguments

h	Either a numeric vector of a numeric matrix or a numeric array specifying the shooting vectors

Value

A numeric array of the same shape as the input array h storing the warping functions of h.

im *Example Image Data set*

Description

Contains two simulated images for registration.

Usage

```
im
```

Format

im:

A list with two components:

- I1: A numeric matrix of shape 64×64 storing the 1st image;
- I2: A numeric matrix of shape 64×64 storing the 2nd image.

invertGamma *Invert Warping Function*

Description

This function calculates the inverse of gamma

Usage

```
invertGamma(gam)
```

Arguments

gam	vector of N samples

Value

Returns gamI inverted vector

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
out <- invertGamma(simu_warp$warping_functions[, 1])
```

inv_exp_map *map square root of warping function to tangent space*

Description

map square root of warping function to tangent space

Usage

```
inv_exp_map(Psi, psi)
```

Arguments

Psi	vector describing psi function at center of tangent space
psi	vector describing psi function to map to tangent space

Value

A numeric array of the same length as the input array psi storing the shooting vector of psi

---

jointFPCA *Joint Vertical and Horizontal Functional Principal Component Analysis*

Description

This function calculates amplitude and phase joint functional principal component analysis on aligned data

Usage

```
jointFPCA(
  warp_data,
  no = 3,
  var_exp = NULL,
  id = round(length(warp_data$time)/2),
  C = NULL,
  ci = c(-1, 0, 1),
  showplot = T
)
```

Arguments

warp_data	fdawarp object from time_warping of aligned data
no	number of principal components to extract (default = 3)
var_exp	compute no based on value percent variance explained (example: 0.95) will override no
id	integration point for f0 (default = midpoint)
C	balance value (default = NULL)
ci	geodesic standard deviations (default = c(-1,0,1))
showplot	show plots of principal directions (default = T)

Value

Returns a list containing

q_pca	srvf principal directions
f_pca	f principal directions
latent	latent values
coef	coefficients
U	eigenvectors

mu_psi	mean psi function
mu_g	mean g function
id	point use for f(0)
C	optimized phase amplitude ratio

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Jung, S. L. a. S. (2016). "Combined Analysis of Amplitude and Phase Variations in Functional Data." arXiv:1603.01775.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
jfpca <- jointFPCA(simu_warp, no = 3)
```

`jointFPCAh` *Joint Vertical and Horizontal Functional Principal Component Analysis*

Description

This function calculates amplitude and phase joint functional principal component analysis on aligned data using the SRVF framework using MFPCA and h representation

Usage

```
jointFPCAh(
  warp_data,
  var_exp = 0.99,
  id = round(length(warp_data$time)/2),
  C = NULL,
  ci = c(-1, 0, 1),
  srvf = TRUE,
  showplot = TRUE
)
```

Arguments

warp_data	fdawarp object from time_warping of aligned data
var_exp	compute no based on value percent variance explained (default: 0.99) will override no
id	integration point for f0 (default = midpoint)

C	balance value (default = NULL)
ci	geodesic standard deviations (default = c(-1,0,1))
srvf	use srvf (default = TRUE)
showplot	show plots of principal directions (default = T)

Value

Returns a list containing

q_pca	srvf principal directions
f_pca	f principal directions
latent	latent values
coef	coefficients
U	eigenvectors
mu_psi	mean psi function
mu_g	mean g function
id	point use for f(0)
C	optimized phase amplitude ratio

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Jung, S. L. a. S. (2016). "Combined Analysis of Amplitude and Phase Variations in Functional Data." arXiv:1603.01775.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
jfpcah <- jointFPCAh(simu_warp)
```

---

joint_gauss_model *Gaussian model of functional data using joint Model*

Description

This function models the functional data using a Gaussian model extracted from the principal components of the srvfs using the joint model

Usage

```
joint_gauss_model(warp_data, n = 1, no = 5)
```

Arguments

`warp_data`	fdawarp object from time_warping of aligned data
`n`	number of random samples (n = 1)
`no`	number of principal components (n=4)

Value

Returns a fdawarp object containing

`fs`	random aligned samples
`gams`	random warping function samples
`ft`	random function samples
`qs`	random srvf samples

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Jung, S. L. a. S. (2016). "Combined Analysis of Amplitude and Phase Variations in Functional Data." arXiv:1603.01775.

Examples

```
out1 <- joint_gauss_model(simu_warp, n = 10)
```

`kmeans_align` *K-Means Clustering and Alignment*

Description

This function clusters functions and aligns using the elastic square-root velocity function (SRVF) framework.

Usage

```
kmeans_align(
  f,
  time,
  K = 1L,
  seeds = NULL,
  centroid_type = c("mean", "medoid"),
  nonempty = 0L,
  lambda = 0,
  showplot = FALSE,
  smooth_data = FALSE,
  sparam = 25L,
```

```
  parallel = FALSE,
  alignment = TRUE,
  rotation = FALSE,
  scale = TRUE,
  omethod = c("DP", "RBFGS"),
  max_iter = 50L,
  thresh = 0.01,
  use_verbose = FALSE
)
```

Arguments

f
: Either a numeric matrix or a numeric 3D array specifying the functions that need to be jointly clustered and aligned.
  - If a matrix, it must be of shape $M \times N$. In this case, it is interpreted as a sample of N curves observed on a grid of size M.
  - If a 3D array, it must be of shape $L \times M \times N$ and it is interpreted as a sample of N L-dimensional curves observed on a grid of size M.

  If this is multidimensional functional data, it is advised that `rotation==FALSE`

time
: A numeric vector of length M specifying the grid on which the curves are evaluated.

K
: An integer value specifying the number of clusters. Defaults to `1L`.

seeds
: An integer vector of length `K` specifying the indices of the curves in `f` which will be chosen as initial centroids. Defaults to `NULL` in which case such indices are randomly chosen.

centroid_type
: A string specifying the type of centroid to compute. Choices are `"mean"` or `"medoid"`. Defaults to `"mean"`.

nonempty
: An integer value specifying the minimum number of curves per cluster during the assignment step. Set it to a positive value to avoid the problem of empty clusters. Defaults to `0L`.

lambda
: A numeric value specifying the elasticity. Defaults to `0.0`.

showplot
: A Boolean specifying whether to show plots. Defaults to `FALSE`.

smooth_data
: A Boolean specifying whether to smooth data using a box filter. Defaults to `FALSE`.

sparam
: An integer value specifying the number of box filters applied. Defaults to `25L`.

parallel
: A Boolean specifying whether parallel mode (using `foreach::foreach()` and the **doParallel** package) should be activated. Defaults to `FALSE`.

alignment
: A Boolean specifying whether to perform alignment. Defaults to `TRUE`.

rotation
: A Boolean specifying whether to perform rotation. Defaults to to `FALSE`.

scale
: A Boolean specifying whether to scale curves to unit length. Defaults to `TRUE`.

omethod
: A string specifying which method should be used to solve the optimization problem that provides estimated warping functions. Choices are `"DP"` or `"RBFGS"`. Defaults to `"DP"`.

max_iter	An integer value specifying the maximum number of iterations. Defaults to 50L.
thresh	A numeric value specifying a threshold on the cost function below which convergence is assumed. Defaults to 0.01.
use_verbose	A Boolean specifying whether to display information about the calculations in the console. Defaults to FALSE.

Value

An object of class fdakma which is a list containing:

- f0: the original functions;
- q0: the original SRSFs;
- fn: the aligned functions as matrices or a 3D arrays of the same shape than f0 by clusters in a list;
- qn: the aligned SRSFs as matrices or a 3D arrays of the same shape than f0 separated in clusters in a list;
- labels: the cluster memberships as an integer vector;
- templates: the centroids in the original functional space;
- templates.q: the centroids in SRSF space;
- distances_to_center: A numeric vector storing the distances of each observed curve to its center;
- gam: the warping functions as matrices or a 3D arrays of the same shape than f0 by clusters in a list;
- qun: cost function value.

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using Fisher-Rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative models for functional data using phase and amplitude separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Sangalli, L. M., et al. (2010). "k-mean alignment for curve clustering." Computational Statistics & Data Analysis 54(5): 1219-1233.

Examples

```
## Not run:
  out <- kmeans_align(growth_vel$f, growth_vel$time, K = 2)

## End(Not run)
```

LongRunCovMatrix — *Long Run Covariance Matrix Estimation for Multivariate Time Series*

Description

This function estimates the long run covariance matrix of a given multivariate data sample.

Usage

```
LongRunCovMatrix(mdobj, h = 0, kern_type = "bartlett")
```

Arguments

`mdobj`	A multivariate data object
`h`	The bandwidth parameter. It is strictly non-zero. Choosing the bandwidth parameter to be zero is identical to estimating covariance matrix assuming iid data.
`kern_type`	Kernel function to be used for the estimation of the long run covariance matrix. The choices are `c("BT", "PR", "SP", "FT")` which are respectively, bartlett, parzen, simple and flat-top kernels. By default the function uses a `"barlett"` kernel.

Value

Returns long run covariance matrix

multiple_align_functions — *Group-wise function alignment to specified mean*

Description

This function aligns a collection of functions using the elastic square-root slope (srsf) framework.

Usage

```
multiple_align_functions(
  f,
  time,
  mu,
  lambda = 0,
  pen = "roughness",
  showplot = TRUE,
  smooth_data = FALSE,
  sparam = 25,
  parallel = FALSE,
```

```
  omethod = "DP",
  MaxItr = 20,
  iter = 2000,
  verbose = TRUE
)
```

Arguments

f	matrix (N x M) of M functions with N samples
time	vector of size N describing the sample points
mu	vector of size N that f is aligned to
lambda	controls the elasticity (default = 0)
pen	alignment penalty (default="roughness") options are second derivative ("roughness"), geodesic distance from id ("geodesic"), and norm from id ("norm")
showplot	shows plots of functions (default = T)
smooth_data	smooth data using box filter (default = F)
sparam	number of times to apply box filter (default = 25)
parallel	enable parallel mode using `foreach` and `doParallel` package (default=F)
omethod	optimization method (DP,DP2,RBFGS,dBayes,expBayes)
MaxItr	maximum number of iterations
iter	bayesian number of mcmc samples (default 2000)
verbose	verbose printing (default TRUE)

Value

Returns a fdawarp object containing

f0	original functions
fn	aligned functions - matrix (N x M) of M functions with N samples
qn	aligned SRSFs - similar structure to fn
q0	original SRSF - similar structure to fn
fmean	function mean or median - vector of length N
mqn	SRSF mean or median - vector of length N
gam	warping functions - similar structure to fn
orig.var	Original Variance of Functions
amp.var	Amplitude Variance
phase.var	Phase Variance
qun	Cost Function Value

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

multivariate_karcher_mean
Karcher Mean of Multivariate Functional Data

Description

Calculates the Karcher mean or median of a collection of multivariate functional data using the elastic square-root velocity (SRVF) framework. While most of the time, the setting does not require a metric that is invariant to rotation and scale, this can be achieved through the optional arguments `rotation` and `scale`.

Usage

```
multivariate_karcher_mean(
  beta,
  mode = "O",
  alignment = TRUE,
  rotation = FALSE,
  scale = FALSE,
  lambda = 0,
  maxit = 20L,
  ms = c("mean", "median"),
  exact_medoid = FALSE,
  ncores = 1L,
  verbose = FALSE
)
```

Arguments

`beta`	A numeric array of shape $L \times M \times N$ specifying an N-sample of L-dimensional functional data evaluated on a same grid of size M.
`mode`	A character string specifying whether the input curves should be considered open (`mode == "O"`) or closed (`mode == "C"`). Defaults to `"O"`.
`alignment`	A boolean value specifying whether the curves should be aligned before computing the distance matrix. Defaults to `TRUE`.
`rotation`	A boolean specifying whether the distance should be made invariant by rotation. Defaults to `TRUE`.
`scale`	A boolean specifying whether the distance should be made invariant by scaling. This is effectively achieved by making SRVFs having unit length and switching to an appropriate metric on the sphere between normalized SRVFs. Defaults to `TRUE`.
`lambda`	A numeric value specifying the weight of a penalty term that constraints the warping function to be not too far from the identity. Defaults to `0.0`.
`maxit`	An integer value specifying the maximum number of iterations. Defaults to `20L`.

ms	A character string specifying whether the Karcher mean ("mean") or Karcher median ("median") is returned. Defaults to "mean".
exact_medoid	A boolean specifying whether to compute the exact medoid from the distance matrix or as the input curve closest to the pointwise mean. Defaults to FALSE for saving computational time.
ncores	An integer value specifying the number of cores to use for parallel computation. Defaults to 1L. The maximum number of available cores is determined by the **parallel** package. One core is always left out to avoid overloading the system.
verbose	A boolean specifying whether to print the progress of the algorithm. Defaults to FALSE.

Value

A list with the following components:

- beta: A numeric array of shape $L \times M \times N$ storing the original input data.
- q: A numeric array of shape $L \times M \times N$ storing the SRVFs of the input data.
- betan: A numeric array of shape $L \times M \times N$ storing the aligned, possibly optimally rotated and optimally scaled, input data.
- qn: A numeric array of shape $L \times M \times N$ storing the SRVFs of the aligned, possibly optimally rotated and optimally scaled, input data.
- betamean: A numeric array of shape $L \times M$ storing the Karcher mean or median of the input data.
- qmean: A numeric array of shape $L \times M$ storing the Karcher mean or median of the SRVFs of the input data.
- type: A character string indicating whether the Karcher mean or median has been returned.
- E: A numeric vector storing the energy of the Karcher mean or median at each iteration.
- qun: A numeric vector storing the cost function of the Karcher mean or median at each iteration.

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence, **33** (7), 1415-1428.

Examples

```
out <- multivariate_karcher_mean(beta[, , 1, 1:2], maxit = 2)
# note: use more functions, small for speed
```

`optimum.reparam` *Align two functions*

Description

This function aligns the SRVFs of two functions in R^1 defined on an interval $[t_{\min}, t_{\max}]$ using dynamic programming or RBFGS

Usage

```
optimum.reparam(
  Q1,
  T1,
  Q2,
  T2,
  lambda = 0,
  pen = "roughness",
  method = c("DP", "DPo", "SIMUL", "RBFGS"),
  f1o = 0,
  f2o = 0,
  nbhd_dim = 7
)
```

Arguments

Q1	A numeric matrix of shape `n_points x n_dimensions` specifying the SRSF of the 1st `n_dimensions`-dimensional function evaluated on a grid of size `n_points` of its univariate domain.
T1	A numeric vector of size `n_points` specifying the grid on which the 1st SRSF is evaluated.
Q2	A numeric matrix of shape `n_points x n_dimensions` specifying the SRSF of the 2nd `n_dimensions`-dimensional function evaluated on a grid of size `n_points` of its univariate domain.
T2	A numeric vector of size `n_points` specifying the grid on which the 1st SRSF is evaluated.
lambda	A numeric value specifying the amount of warping. Defaults to `0.0`.
pen	alignment penalty (default="roughness") options are second derivative ("roughness"), geodesic distance from id ("geodesic"), and norm from id ("l2gam"), srvf norm from id ("l2psi")
method	A string specifying the optimization method. Choices are `"DP"`, `"DPo"`, `"SIMUL"`, or `"RBFGS"`. Defaults to `"DP"`.
f1o	A numeric vector of size `n_dimensions` specifying the value of the 1st function at $t = t_{\min}$. Defaults to `rep(0, n_dimensions)`.
f2o	A numeric vector of size `n_dimensions` specifying the value of the 2nd function at $t = t_{\min}$. Defaults to `rep(0, n_dimensions)`.
nbhd_dim	size of the grid (default = 7)

Value

A numeric vector of size n_points storing discrete evaluations of the estimated boundary-preserving warping diffeomorphism on the initial grid.

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using Fisher-Rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative models for functional data using phase and amplitude separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
q <- f_to_srvf(simu_data$f, simu_data$time)
gam <- optimum.reparam(q[, 1], simu_data$time, q[, 2], simu_data$time)
```

---

outlier.detection *Outlier Detection*

Description

This function calculates outlier's using geodesic distances of the SRVFs from the median

Usage

```
outlier.detection(q, time, mq, k = 1.5)
```

Arguments

q	matrix (N x M) of M SRVF functions with N samples
time	vector of size N describing the sample points
mq	median calculated using time_warping()
k	cutoff threshold (default = 1.5)

Value

q_outlier	outlier functions

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
q_outlier <- outlier.detection(
  q = toy_warp$q0,
  time = toy_data$time,
  mq = toy_warp$mqn,
  k = .1
)
```

`pair_align_functions` *Align two functions*

Description

This function aligns two functions using SRSF framework. It will align f2 to f1

Usage

```
pair_align_functions(
  f1,
  f2,
  time,
  lambda = 0,
  pen = "roughness",
  method = "DP",
  w = 0.01,
  iter = 2000
)
```

Arguments

f1	function 1
f2	function 2
time	sample points of functions
lambda	controls amount of warping (default = 0)
pen	alignment penalty (default="roughness") options are second derivative ("roughness"), geodesic distance from id ("geodesic"), and norm from id ("norm")
method	controls which optimization method (default="DP") options are Dynamic Programming ("DP"), Coordinate Descent ("DP2"), Riemannian BFGS ("RBFGS"), Simultaneous Alignment ("SIMUL"), Dirichlet Bayesian ("dBayes"), and Expo-Map Bayesian ("expBayes")
w	controls LRBFGS (default = 0.01)
iter	number of mcmc iterations for mcmc method (default 2000)

Value

Returns a list containing

f2tilde	aligned f2
gam	warping function

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Cheng, W., Dryden, I. L., and Huang, X. (2016). Bayesian registration of functions and curves. Bayesian Analysis, 11(2), 447-475.

Lu, Y., Herbei, R., and Kurtek, S. (2017). Bayesian registration of functions with a Gaussian process prior. Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2017.1336444.

Examples

```
out <- pair_align_functions(
  f1 = simu_data$f[, 1],
  f2 = simu_data$f[, 2],
  time = simu_data$time
)
```

---

pair_align_functions_bayes

Align two functions

Description

This function aligns two functions using Bayesian SRSF framework. It will align f2 to f1

Usage

```
pair_align_functions_bayes(
  f1,
  f2,
  timet,
  iter = 15000,
  times = 5,
  tau = ceiling(times * 0.4),
  powera = 1,
  showplot = TRUE,
  extrainfo = FALSE
)
```

Arguments

f1	function 1
f2	function 2
timet	sample points of functions
iter	number of iterations (default = 15000)
times	factor of length of subsample points to look at (default = 5)
tau	standard deviation of Normal prior for increment (default ceil(times*.4))
powera	Dirichlet prior parameter (default 1)
showplot	shows plots of functions (default = T)
extrainfo	T/F whether additional information is returned

Value

Returns a list containing

f1	function 1
f2_q	registered function using quotient space
gam_q	warping function quotient space
f2_a	registered function using ambient space
q2_a	warping function ambient space
match_collect	posterior samples from warping function (returned if extrainfo=TRUE)
dist_collect	posterior samples from the distances (returned if extrainfo=TRUE)
kappa_collect	posterior samples from kappa (returned if extrainfo=TRUE)
log_collect	log-likelihood of each sample (returned if extrainfo=TRUE)
pct_accept	vector of acceptance ratios for the warping function (returned if extrainfo=TRUE)

References

Cheng, W., Dryden, I. L., and Huang, X. (2016). Bayesian registration of functions and curves. Bayesian Analysis, 11(2), 447-475.

Examples

```
out <- pair_align_functions_bayes(
  f1 = simu_data$f[, 1],
  f2 = simu_data$f[, 2],
  timet = simu_data$time
)
```

pair_align_functions_expomap

Align two functions using geometric properties of warping functions

Description

This function aligns two functions using Bayesian framework. It will align f2 to f1. It is based on mapping warping functions to a hypersphere, and a subsequent exponential mapping to a tangent space. In the tangent space, the Z-mixture pCN algorithm is used to explore both local and global structure in the posterior distribution.

Usage

```
pair_align_functions_expomap(
  f1,
  f2,
  timet,
  iter = 20000,
  burnin = min(5000, iter/2),
  alpha0 = 0.1,
  beta0 = 0.1,
  zpcn = list(betas = c(0.5, 0.05, 0.005, 1e-04), probs = c(0.1, 0.1, 0.7, 0.1)),
  propvar = 1,
  init.coef = rep(0, 2 * 10),
  npoints = 200,
  extrainfo = FALSE
)
```

Arguments

`f1`	observed data, numeric vector
`f2`	observed data, numeric vector
`timet`	sample points of functions
`iter`	length of the chain
`burnin`	number of burnin MCMC iterations
`alpha0, beta0`	IG parameters for the prior of sigma1
`zpcn`	list of mixture coefficients and prior probabilities for Z-mixture pCN algorithm of the form list(betas, probs), where betas and probs are numeric vectors of equal length
`propvar`	variance of proposal distribution
`init.coef`	initial coefficients of warping function in exponential map; length must be even
`npoints`	number of sample points to use during alignment
`extrainfo`	T/F whether additional information is returned

Details

The Z-mixture pCN algorithm uses a mixture distribution for the proposal distribution, controlled by input parameter zpcn. The zpcn$betas must be between 0 and 1, and are the coefficients of the mixture components, with larger coefficients corresponding to larger shifts in parameter space. The zpcn$probs give the probability of each shift size.

Value

Returns a list containing

`f2_warped`	f2 aligned to f1
`gamma`	Posterior mean gamma function
`g.coef`	matrix with iter columns, posterior draws of g.coef
`psi`	Posterior mean psi function
`sigma1`	numeric vector of length iter, posterior draws of sigma1
`accept`	Boolean acceptance for each sample (if extrainfo=TRUE)
`betas.ind`	Index of zpcn mixture component for each sample (if extrainfo=TRUE)
`logl`	numeric vector of length iter, posterior loglikelihood (if extrainfo=TRUE)
`gamma_mat`	Matrix of all posterior draws of gamma (if extrainfo=TRUE)
`gamma_q025`	Lower 0.025 quantile of gamma (if extrainfo=TRUE)
`gamma_q975`	Upper 0.975 quantile of gamma (if extrainfo=TRUE)
`sigma_eff_size`	Effective sample size of sigma (if extrainfo=TRUE)
`psi_eff_size`	Vector of effective sample sizes of psi (if extrainfo=TRUE)
`xdist`	Vector of posterior draws from xdist between registered functions (if extrainfo=TRUE)
`ydist`	Vector of posterior draws from ydist between registered functions (if extrainfo=TRUE)

References

Lu, Y., Herbei, R., and Kurtek, S. (2017). Bayesian registration of functions with a Gaussian process prior. Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2017.1336444.

Examples

```
## Not run:
  # This is an MCMC algorithm and takes a long time to run
  myzpcn <- list(
    betas = c(0.1, 0.01, 0.005, 0.0001),
    probs = c(0.2, 0.2, 0.4, 0.2)
  )
  out <- pair_align_functions_expomap(
    f1 = simu_data$f[, 1],
    f2 = simu_data$f[, 2],
    timet = simu_data$time,
    zpcn = myzpcn,
    extrainfo = TRUE
  )
```

```
  # overall acceptance ratio
  mean(out$accept)
  # acceptance ratio by zpcn coefficient
  with(out, tapply(accept, myzpcn$betas[betas.ind], mean))

## End(Not run)
```

`pair_align_image` *Pairwise align two images This function aligns to images using the q-map framework*

Description

Pairwise align two images This function aligns to images using the q-map framework

Usage

```
pair_align_image(
  I1,
  I2,
  M = 5,
  ortho = TRUE,
  basis_type = "t",
  resizei = FALSE,
  N = 64,
  stepsize = 1e-05,
  itermax = 1000
)
```

Arguments

`I1`	reference image
`I2`	image to warp
`M`	number of basis elements (default=5)
`ortho`	orthonormalize basis (default=TRUE)
`basis_type`	("t","s","i","o"; default="t")
`resizei`	resize image (default=TRUE)
`N`	size of resized image (default=64)
`stepsize`	gradient stepsize (default=1e-5)
`itermax`	maximum number of iterations (default=1000)

Value

Returns a list containing

`Inew`	aligned I2
`gam`	warping function

References

Q. Xie, S. Kurtek, E. Klassen, G. E. Christensen and A. Srivastava. Metric-based pairwise and multiple image registration. IEEE European Conference on Computer Vision (ECCV), September, 2014

Examples

```
## Not run:
  # This is a gradient descent algorithm and takes a long time to run
  out <- pair_align_image(im$I1, im$I2)

## End(Not run)
```

---

pcaTB *Tolerance Bound Calculation using Elastic Functional PCA*

Description

This function computes tolerance bounds for functional data containing phase and amplitude variation using principal component analysis

Usage

```
pcaTB(f, time, m = 4, B = 1e+05, a = 0.05, p = 0.99)
```

Arguments

`f`	matrix of functions
`time`	vector describing time sampling
`m`	number of principal components (default = 4)
`B`	number of monte carlo iterations
`a`	confidence level of tolerance bound (default = 0.05)
`p`	coverage level of tolerance bound (default = 0.99)

Value

Returns a list containing

`pca`	pca output
`tol`	tolerance factor

References

J. D. Tucker, J. R. Lewis, C. King, and S. Kurtek, “A Geometric Approach for Computing Tolerance Bounds for Elastic Functional Data,” Journal of Applied Statistics, 10.1080/02664763.2019.1645818, 2019.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Jung, S. L. a. S. (2016). "Combined Analysis of Amplitude and Phase Variations in Functional Data." arXiv:1603.01775.

Examples

```
## Not run:
  out1 <- pcaTB(simu_data$f, simu_data$time)

## End(Not run)
```

`plot_curve` *Plot Curve*

Description

This function plots open or closed curves

Usage

```
plot_curve(beta, add = FALSE, ...)
```

Arguments

`beta`	Array of sizes $n \times T$ describing a curve of dimension n evaluated on T points
`add`	add to current plot (default = `TRUE`)
`...`	additional plotting parameters

Value

Return shape confidence intervals

predict.hfpca *Elastic Prediction for functional PCA*

Description

This function performs projection of new functions on fPCA basis

Usage

```
## S3 method for class 'hfpca'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	Object of class inheriting from "horizFPCA"
`newdata`	An optional matrix in which to look for functions with which to predict. If omitted, the original functions are used.
`...`	additional arguments affecting the predictions produced

Value

Returns a matrix

`a`	principle coefficients

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

predict.jfpca *Elastic Prediction for functional PCA*

Description

This function performs projection of new functions on fPCA basis

Usage

```
## S3 method for class 'jfpca'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	Object of class inheriting from "jointFPCA"
`newdata`	An optional matrix in which to look for functions with which to predict. If omitted, the original functions are used.
`...`	additional arguments affecting the predictions produced

Value

Returns a matrix

`a`	principle coefficients

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

`predict.jfpcah` *Elastic Prediction for functional PCA*

Description

This function performs projection of new functions on fPCA basis

Usage

```
## S3 method for class 'jfpcah'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	Object of class inheriting from "jointFPCA"
`newdata`	An optional matrix in which to look for functions with which to predict. If omitted, the original functions are used.
`...`	additional arguments affecting the predictions produced

Value

Returns a matrix

`a`	principle coefficients

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

predict.lpcr	*Elastic Prediction for functional logistic PCR Model*

Description

This function performs prediction from an elastic logistic fPCR regression model with phase-variability

Usage

```
## S3 method for class 'lpcr'
predict(object, newdata = NULL, y = NULL, ...)
```

Arguments

object	Object of class inheriting from "elastic.pcr.regression"
newdata	An optional matrix in which to look for variables with which to predict. If omitted, the fitted values are used.
y	An optional vector of labels to calculate PC. If omitted, PC is NULL
...	additional arguments affecting the predictions produced

Value

Returns a list containing

y_pred	predicted probabilities of the class of newdata
y_labels	class labels of newdata
PC	probability of classification

References

J. D. Tucker, J. R. Lewis, and A. Srivastava, "Elastic Functional Principal Component Regression," Statistical Analysis and Data Mining, 10.1002/sam.11399, 2018.

predict.mlpcr	*Elastic Prediction for functional multinomial logistic PCR Model*

Description

This function performs prediction from an elastic multinomial logistic fPCR regression model with phase-variability

Usage

```
## S3 method for class 'mlpcr'
predict(object, newdata = NULL, y = NULL, ...)
```

Arguments

`object`	Object of class inheriting from "elastic.pcr.regression"
`newdata`	An optional matrix in which to look for variables with which to predict. If omitted, the fitted values are used.
`y`	An optional vector of labels to calculate PC. If omitted, PC is NULL
`...`	additional arguments affecting the predictions produced

Value

Returns a list containing

`y_pred`	predicted probabilities of the class of newdata
`y_labels`	class labels of newdata
`PC`	probability of classification per class
`PC.comb`	total probability of classification

References

J. D. Tucker, J. R. Lewis, and A. Srivastava, "Elastic Functional Principal Component Regression," Statistical Analysis and Data Mining, 10.1002/sam.11399, 2018.

`predict.pcr` *Elastic Prediction for functional PCR Model*

Description

This function performs prediction from an elastic pcr regression model with phase-variability

Usage

```
## S3 method for class 'pcr'
predict(object, newdata = NULL, y = NULL, ...)
```

Arguments

`object`	Object of class inheriting from "elastic.pcr.regression"
`newdata`	An optional matrix in which to look for variables with which to predict. If omitted, the fitted values are used.
`y`	An optional vector of responses to calculate SSE. If omitted, SSE is NULL
`...`	additional arguments affecting the predictions produced

Value

Returns a list containing

`y_pred`	predicted values of newdata
`SSE`	sum of squared errors

References

J. D. Tucker, J. R. Lewis, and A. Srivastava, "Elastic Functional Principal Component Regression," Statistical Analysis and Data Mining, 10.1002/sam.11399, 2018.

`predict.vfpca` *Elastic Prediction for functional PCA*

Description

This function performs projection of new functions on fPCA basis

Usage

```
## S3 method for class 'vfpca'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	Object of class inheriting from "vertFPCA"
`newdata`	An optional matrix in which to look for functions with which to predict. If omitted, the original functions are used.
`...`	additional arguments affecting the predictions produced

Value

Returns a matrix

`a`	principle coefficients

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

`q_to_curve` *Convert to curve space*

Description

This function converts SRVFs to curves

Usage

```
q_to_curve(q, scale = 1)
```

Arguments

`q`	either a matrix of shape $n \times T$ describing SRVF or SRVF of multidimensional functional data in R^n, where n is the dimension and T is the number of time points
`scale`	scale of from curve_to_q (default = 1)

Value

beta array describing curve

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
q <- curve_to_q(beta[, , 1, 1])$q
beta1 <- q_to_curve(q)
```

`reparam_curve` *Align two curves*

Description

This function aligns two SRVF functions using Dynamic Programming. If the curves beta1 and beta2 are describing multidimensional functional data, then `rotation` == `FALSE` and `mode == 'O'`

Usage

```
reparam_curve(
  beta1,
  beta2,
  lambda = 0,
  method = "DP",
  w = 0.01,
  rotated = TRUE,
  isclosed = FALSE,
  mode = "O"
)
```

Arguments

`beta1`	curve 1, provided as a matrix of dimensions $n \times M$ for n-dimensional curve evaluated on M sample points
`beta2`	curve 1, provided as a matrix of dimensions $n \times M$ for n-dimensional curve evaluated on M sample points
`lambda`	controls amount of warping (default = `0`)
`method`	controls which optimization method. Options are Dynamic Programming (`"DP"`). (default = `"DP"`)
`w`	controls LRBFGS (default = `0.01`)
`rotated`	boolean if rotation is desired
`isclosed`	boolean if curve is closed
`mode`	Open (`"O"`) or Closed (`"C"`) curves

Value

return a List containing

`gam`	warping function
`R`	rotation matrix
`tau`	seed point

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
gam <- reparam_curve(beta[, , 1, 1], beta[, , 1, 5])$gam
```

reparam_image *Find optimum reparameterization between two images*

Description

Finds the optimal warping function between two images using the elastic framework

Usage

```
reparam_image(It, Im, gam, b, stepsize = 1e-05, itermax = 1000, lmark = FALSE)
```

Arguments

It	template image matrix
Im	test image matrix
gam	initial warping array
b	basis matrix
stepsize	gradient stepsize (default=1e-5)
itermax	maximum number of iterations (default=1000)
lmark	use landmarks (default=FALSE)

Value

Returns a list containing

gamnew	final warping
Inew	aligned image
H	energy
stepsize	final stepsize

References

Q. Xie, S. Kurtek, E. Klassen, G. E. Christensen and A. Srivastava. Metric-based pairwise and multiple image registration. IEEE European Conference on Computer Vision (ECCV), September, 2014

resamplecurve *Resample Curve*

Description

This function resamples a curve to a number of points

Usage

```
resamplecurve(x, N = 100, mode = "O")
```

Arguments

x	matrix defining curve (n,T)
N	Number of samples to re-sample curve, N usually is > T
mode	Open ("O") or Closed ("C") curves

Value

xn matrix defining resampled curve

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
xn <- resamplecurve(beta[, , 1, 1], 200)
```

rgam *Random Warping*

Description

Generates random warping functions

Usage

```
rgam(N, sigma, num)
```

Arguments

N	length of warping function
sigma	variance of warping functions
num	number of warping functions

Value

gam warping functions

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
gam = rgam(N=101, sigma=.01, num=35)
```

---

sample_shapes *Sample shapes from model*

Description

Sample shapes from model

Usage

```
sample_shapes(x, no = 3, numSamp = 10)
```

Arguments

x	An object of class fdacurve typically produced by curve_srvf_align()
no	number of principal components
numSamp	number of samples

Value

Returns a fdacurve object containing

betans	random aligned curves
qns	random aligned srvfs
betans	random curves
qs	random srvfs
gams	random reparametrization functions
R	random rotation matrices

References

Srivastava, A., Klassen, E., Joshi, S., Jermyn, I., (2011). Shape analysis of elastic curves in euclidean spaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (7), 1415-1428.

Examples

```
out <- curve_srvf_align(beta[, , 1, 1:5], maxit = 2, parallel=FALSE)
# note: use more shapes, small for speed
out.samples <- sample_shapes(out)
```

---

shape_CI *Shape Confidence Interval Calculation using Bootstrap Sampling*

Description

This function computes Confidence bounds for shapes using elastic metric

Usage

```
shape_CI(
  beta,
  a = 0.95,
  no = 5,
  Nsamp = 100,
  mode = "O",
  rotated = TRUE,
  scale = TRUE,
  lambda = 0,
  parallel = TRUE
)
```

Arguments

beta	Array of sizes $n \times T \times N$ describing N curves of dimension n evaluated on T points
a	confidence level (default = 0.95)
no	number of principal components (default = 5)
Nsamp	number of functions to generate (default = 100)
mode	Open ("O") or Closed ("C") curves
rotated	Optimize over rotation (default = TRUE)
scale	scale curves to unit length (default = TRUE)
lambda	A numeric value specifying the elasticity. Defaults to 0.0.
parallel	enable parallel processing (default = T)

Value

Return shape confidence intervals

References

J. D. Tucker, J. R. Lewis, C. King, and S. Kurtek, "A Geometric Approach for Computing Tolerance Bounds for Elastic Functional Data," Journal of Applied Statistics, 10.1080/02664763.2019.1645818, 2019.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

`simu_data` *Simulated two Gaussian Dataset*

Description

A functional dataset where the individual functions are given by: $y_i(t) = z_{i,1}e^{-(t-1.5)^2/2} + z_{i,2}e^{-(t+1.5)^2/2}$, $t \in [-3,3]$, $i = 1,2,\ldots,21$, where $z_{i,1}$ and $z_{i,2}$ are *i.i.d.* normal with mean one and standard deviation 0.25. Each of these functions is then warped according to: $\gamma_i(t) = 6(\frac{e^{a_i(t+3)/6}-1}{e^{a_i}-1}) - 3$ if $a_i \neq 0$, otherwise $\gamma_i = \gamma_{id}$ ($gamma_{id}(t) = t$) is the identity warping). The variables are as follows: f containing the 21 functions of 101 samples and time which describes the sampling.

Usage

```
simu_data
```

Format

`simu_data`:

A list with 2 components:

- `f`: A numeric matrix of shape 101×21 storing a sample of size $N = 21$ of curves evaluated on a grid of size $M = 101$.
- `time`: A numeric vector of size $M = 101$ storing the grid on which the curves `f` have been evaluated.

`simu_warp` *Aligned Simulated two Gaussian Dataset*

Description

A functional dataset where the individual functions are given by: $y_i(t) = z_{i,1}e^{-(t-1.5)^2/2} + z_{i,2}e^{-(t+1.5)^2/2}$, $t \in [-3,3]$, $i = 1,2,\ldots,21$, where $z_{i,1}$ and $z_{i,2}$ are *i.i.d.* normal with mean one and standard deviation 0.25. Each of these functions is then warped according to: $\gamma_i(t) = 6(\frac{e^{a_i(t+3)/6}-1}{e^{a_i}-1}) - 3$ if $a_i \neq 0$, otherwise $\gamma_i = \gamma_{id}$ ($gamma_{id}(t) = t$) is the identity warping). The variables are as follows: f containing the 21 functions of 101 samples and time which describes the sampling which has been aligned.

Usage

```
simu_warp
```

Format

`simu_warp`:
A list which contains the output of the `time_warping()` function applied on the data set `simu_data`.

`simu_warp_median` *Aligned Simulated two Gaussian Dataset using Median*

Description

A functional dataset where the individual functions are given by: $y_i(t) = z_{i,1} e^{-(t-1.5)^2/2} + z_{i,2} e^{-(t+1.5)^2/2}$, $t \in [-3, 3]$, $i = 1, 2, \ldots, 21$, where $z_{i,1}$ and $z_{i,2}$ are *i.i.d.* normal with mean one and standard deviation 0.25. Each of these functions is then warped according to: $\gamma_i(t) = 6(\frac{e^{a_i(t+3)/6} - 1}{e^{a_i} - 1}) - 3$ if $a_i \neq 0$, otherwise $\gamma_i = \gamma_{id}$ ($gamma_{id}(t) = t$) is the identity warping). The variables are as follows: f containing the 21 functions of 101 samples and time which describes the sampling which has been aligned.

Usage

```
simu_warp_median
```

Format

`simu_warp_median`:
A list which contains the output of the `time_warping()` function finding the median applied on the data set `simu_data`.

`smooth.data` *Smooth Functions*

Description

This function smooths functions using standard box filter

Usage

```
smooth.data(f, sparam)
```

Arguments

`f`	matrix (N x M) of M functions with N samples
`sparam`	number of times to run box filter

Value

fo smoothed functions

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
fo <- smooth.data(simu_data$f, 25)
```

SqrtMean — *SRVF transform of warping functions*

Description

This function calculates the srvf of warping functions with corresponding shooting vectors and finds the mean

Usage

```
SqrtMean(gam)
```

Arguments

gam	matrix (N x M) of M warping functions with N samples

Value

Returns a list containing

mu	Karcher mean psi function
gam_mu	Karcher mean warping function
psi	srvf of warping functions
vec	shooting vectors

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
out <- SqrtMean(simu_warp$warping_functions)
```

SqrtMeanInverse *SRVF transform of warping functions*

Description

This function calculates the srvf of warping functions with corresponding shooting vectors and finds the inverse of mean

Usage

```
SqrtMeanInverse(gam)
```

Arguments

gam	matrix (N x M) of M warping functions with N samples

Value

gamI inverse of Karcher mean warping function

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
gamI <- SqrtMeanInverse(simu_warp$warping_functions)
```

SqrtMedian *SRVF transform of warping functions*

Description

This function calculates the srvf of warping functions with corresponding shooting vectors and finds the median

Usage

```
SqrtMedian(gam)
```

Arguments

gam	matrix (N x M) of M warping functions with N samples

Value

Returns a list containing

median	Karcher median psi function
gam_median	Karcher mean warping function
psi	srvf of warping functions
vec	shooting vectors

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
out <- SqrtMedian(simu_warp_median$warping_functions)
```

---

srvf2curve *Converts from SRVF to curve representation*

Description

Converts from SRVF to curve representation

Usage

```
srvf2curve(qfun, beta0 = NULL)
```

Arguments

qfun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the SRVF of an underlying curve at s.
beta0	A numeric vector of length L specifying the initial value of the underlying curve at $s = 0$.

Value

A function that takes a numeric vector t of values in $[0, 1]$ as input and returns the values of the underlying curve at t.

Examples

```
srvf2curve(curve2srvf(beta[, , 1, 1]))
```

srvf_to_f *Transformation from SRSF Space*

Description

This function transforms SRVFs back to the original functional space for functions in R^1.

Usage

```
srvf_to_f(q, time, f0 = 0)
```

Arguments

q Either a numeric vector of a numeric matrix or a numeric array specifying the SRSFs that need to be transformed.

- If a vector, it must be of shape M and it is interpreted as a single 1-dimensional curve observed on a grid of size M.
- If a matrix, it must be of shape $M \times N$. In this case, it is interpreted as a sample of N curves observed on a grid of size M, unless $M = 1$ in which case it is interpreted as a single 1-dimensional curve observed on a grid of size M.

time A numeric vector of length M specifying the grid on which SRSFs are evaluated.

f0 Either a numeric value or a numeric vector of or a numeric matrix specifying the initial value of the curves in the original functional space. It must be:

- a value if q represents a single SRSF.
- a vector of length N if q represents a sample of N SRVFs

Value

A numeric array of the same shape as the input q storing the transformation of the SRVFs q back to the original functional space.

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative models for functional data using amplitude and phase separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
q <- f_to_srvf(simu_data$f, simu_data$time)
f <- srvf_to_f(q, simu_data$time, simu_data$f[1, ])
```

time_warping *Alignment of univariate functional data*

Description

This function aligns a collection of 1-dimensional curves that are observed on the same grid.

Usage

```
time_warping(
  f,
  time,
  lambda = 0,
  penalty_method = c("roughness", "geodesic", "norm"),
  centroid_type = c("mean", "median"),
  center_warpings = TRUE,
  smooth_data = FALSE,
  sparam = 25L,
  parallel = FALSE,
  optim_method = c("DP", "DPo", "DP2", "RBFGS"),
  max_iter = 20L
)
```

Arguments

`f`	A numeric matrix of shape $M \times N$ specifying a sample of N curves observed on a grid of size M.
`time`	A numeric vector of length M specifying the common grid on which all curves `f` have been observed.
`lambda`	A numeric value specifying the elasticity. Defaults to `0.0`.
`penalty_method`	A string specifying the penalty term used in the formulation of the cost function to minimize for alignment. Choices are `"roughness"` which uses the norm of the second derivative, `"geodesic"` which uses the geodesic distance to the identity and `"norm"` which uses the Euclidean distance to the identity. Defaults to `"roughness"`.
`centroid_type`	A string specifying the type of centroid to align to. Choices are `"mean"` or `"median"`. Defaults to `"mean"`.
`center_warpings`	A boolean specifying whether to center the estimated warping functions. Defaults to `TRUE`.
`smooth_data`	A boolean specifying whether to smooth curves using a box filter. Defaults to `FALSE`.
`sparam`	An integer value specifying the number of times to apply the box filter. Defaults to `25L`. This is used only when `smooth_data = TRUE`.
`parallel`	A boolean specifying whether to run calculations in parallel. Defaults to `FALSE`.

optim_method	A string specifying the algorithm used for optimization. Choices are "DP", "DPo", and "RBFGS". Defaults to "DP".
max_iter	An integer value specifying the maximum number of iterations. Defaults to 20L.

Value

An object of class fdawarp which is a list with the following components:

- time: a numeric vector of length M storing the original grid;
- f0: a numeric matrix of shape $M \times N$ storing the original sample of N functions observed on a grid of size M;
- q0: a numeric matrix of the same shape as f0 storing the original SRSFs;
- fn: a numeric matrix of the same shape as f0 storing the aligned functions;
- qn: a numeric matrix of the same shape as f0 storing the aligned SRSFs;
- fmean: a numeric vector of length M storing the mean or median curve;
- mqn: a numeric vector of length M storing the mean or median SRSF;
- warping_functions: a numeric matrix of the same shape as f0 storing the estimated warping functions;
- original_variance: a numeric value storing the variance of the original sample;
- amplitude_variance: a numeric value storing the variance in amplitude of the aligned sample;
- phase_variance: a numeric value storing the variance in phase of the aligned sample;
- qun: a numeric vector of maximum length max_iter + 2 storing the values of the cost function after each iteration;
- lambda: the input parameter lambda which specifies the elasticity;
- centroid_type: the input centroid type;
- optim_method: the input optimization method;
- inverse_average_warping_function: the inverse of the mean estimated warping function;
- rsamps: TO DO.

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using Fisher-Rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative models for functional data using phase and amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
## Not run:
  out <- time_warping(simu_data$f, simu_data$time)

## End(Not run)
```

toy_data *Distributed Gaussian Peak Dataset*

Description

A functional dataset where the individual functions are given by a Gaussian peak with locations along the x-axis. The variables are as follows: f containing the 29 functions of 101 samples and time which describes the sampling.

Usage

```
toy_data
```

Format

`toy_data`:

A list with two components:

- `f`: A numeric matrix of shape 101×29 storing a sample of size $N = 29$ of curves evaluated on a grid of size $M = 101$.
- `time`: A numeric vector of size $M = 101$ storing the grid on which the curves `f` have been evaluated.

toy_warp *Aligned Distributed Gaussian Peak Dataset*

Description

A functional dataset where the individual functions are given by a Gaussian peak with locations along the x-axis. The variables are as follows: f containing the 29 functions of 101 samples and time which describes the sampling which as been aligned.

Usage

```
toy_warp
```

Format

`toy_warp`:

A list which contains the output of the `time_warping()` function applied on the data set `toy_data`.

to_hilbert_sphere — *Projects an SRVF onto the Hilbert sphere*

Description

Projects an SRVF onto the Hilbert sphere

Usage

```
to_hilbert_sphere(qfun, qnorm = sqrt(get_l2_inner_product(qfun, qfun)))
```

Arguments

qfun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the SRVF at s.
qnorm	A numeric value specifying the L^2 norm of the SRVF. Defaults to $\sqrt{\langle q, q \rangle}$.

Value

A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the SRVF projected onto the Hilbert sphere.

Examples

```
q <- curve2srvf(beta[, , 1, 1])
to_hilbert_sphere(q)
```

vertFPCA — *Vertical Functional Principal Component Analysis*

Description

This function calculates vertical functional principal component analysis on aligned data

Usage

```
vertFPCA(
  warp_data,
  no = 3,
  var_exp = NULL,
  id = round(length(warp_data$time)/2),
  ci = c(-1, 0, 1),
  showplot = TRUE
)
```

Arguments

warp_data	fdawarp object from time_warping of aligned data
no	number of principal components to extract
var_exp	compute no based on value percent variance explained (example: 0.95) will override no
id	point to use for f(0) (default = midpoint)
ci	geodesic standard deviations (default = c(-1,0,1))
showplot	show plots of principal directions (default = T)

Value

Returns a vfpca object containing

q_pca	srvf principal directions
f_pca	f principal directions
latent	latent values
coef	coefficients
U	eigenvectors
id	point used for f(0)

References

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
vfpca <- vertFPCA(simu_warp, no = 3)
```

---

v_to_gam — *map shooting vector to warping function at identity*

Description

map shooting vector to warping function at identity

Usage

```
v_to_gam(v)
```

Arguments

v	Either a numeric vector of a numeric matrix or a numeric array specifying the shooting vectors

Value

A numeric array of the same shape as the input array v storing the warping functions v.

---

warp_curve *Applies a warping function to a given curve*

Description

Applies a warping function to a given curve

Usage

```
warp_curve(betafun, gamfun)
```

Arguments

betafun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the underlying curve at s.
gamfun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of a diffeomorphic warping function at s.

Value

A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the warped curve at s.

Examples

```
curv <- discrete2curve(beta[, , 1, 1])
gamf <- discrete2warping(seq(0, 1, length = 100)^2)
warp_curve(curv, gamf)
```

---

warp_f_gamma *Warp Function*

Description

This function warps function f by γ

Usage

```
warp_f_gamma(f, time, gamma, spl.int = FALSE)
```

Arguments

f	vector function
time	time
gamma	vector warping function
spl.int	use spline interpolation (default F)

Value

fnew warped function

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
fnew <- warp_f_gamma(
  f = simu_data$f[, 1],
  time = simu_data$time,
  gamma = seq(0, 1, length.out = 101)
)
```

---

warp_q_gamma *Warp SRSF*

Description

This function warps srsf q by γ

Usage

```
warp_q_gamma(q, time, gamma, spl.int = FALSE)
```

Arguments

q	vector
time	time
gamma	vector warping function
spl.int	use spline interpolation (default F)

Value

qnew warped function

References

Srivastava, A., Wu, W., Kurtek, S., Klassen, E., Marron, J. S., May 2011. Registration of functional data using fisher-rao metric, arXiv:1103.3817v2.

Tucker, J. D., Wu, W., Srivastava, A., Generative Models for Function Data using Phase and Amplitude Separation, Computational Statistics and Data Analysis (2012), 10.1016/j.csda.2012.12.001.

Examples

```
q <- f_to_srvf(simu_data$f, simu_data$time)
qnew <- warp_q_gamma(q[, 1], simu_data$time, seq(0, 1, length.out = 101))
```

---

warp_srvf — *Applies a warping function to a given SRVF*

Description

Applies a warping function to a given SRVF

Usage

```
warp_srvf(qfun, gamfun, betafun = NULL)
```

Arguments

qfun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the SRVF of an underlying curve at s.
gamfun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of a diffeomorphic warping function at s.
betafun	A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the underlying curve at s. Defaults to NULL.

Value

A function that takes a numeric vector s of values in $[0, 1]$ as input and returns the values of the warped SRVF.

Examples

```
q <- curve2srvf(beta[, , 1, 1])
warp_srvf(q, get_identity_warping())
```

Index