Package ‘fdasrvf’

September 21, 2017

Type Package
Title Elastic Functional Data Analysis
Version 1.8.2
Date 2017-9-17
Author J. Derek Tucker <jdtuck@sandia.gov>
Maintainer J. Derek Tucker <jdtuck@sandia.gov>
Description Performs alignment, PCA, and modeling of multidimensional and
unidimensional functions using the square-root velocity framework
(Srivastava et al., 2011 <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
SystemRequirements C++11
Imports Rcpp (>= 0.12.1), coda, foreach, mvtnorm, matrixcalc, splines,
parallel, fields, doParallel, viridisLite
Suggests akima, plot3D, plot3Drgl, rgl
LinkingTo Rcpp, RcppArmadillo
Depends R (>= 3.1.0),
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2017-09-21 19:43:32 UTC

R topics documented:

   align_fPCA ................................................................. 3
   AmplitudeBoxplot ...................................................... 4
   beta ................................................................. 5
R topics documented:
calc_shape_dist ................................................. 5
curve_geodesic ....................................................... 6
curve_karcher_cov ..................................................... 7
curve_karcher_mean ................................................... 7
curve_pair_align ...................................................... 8
curve_principal_directions ......................................... 9
curve_srf_align ....................................................... 10
curve_to_q ........................................................... 11
elastic.distance ..................................................... 11
elastic.logistic ...................................................... 12
elastic.mlogistic .................................................... 13
elastic.prediction ................................................... 15
elastic.regression .................................................. 16
deasrf ............................................................... 17
function_group_warp_bayes ....................................... 18
function_mean_bayes ............................................... 19
f_to_srf ............................................................ 20
gauss_model ......................................................... 20
gradient ............................................................. 21
growth_vel .......................................................... 22
horizFPCA .......................................................... 22
im ................................................................. 23
inverGamma ........................................................ 23
kmeans_align ........................................................ 24
optimum.reparam .................................................. 25
outlier.detection ................................................... 26
pair_align_functions ............................................... 27
pair_align_functions_bayes ......................................... 28
pair_align_image ................................................... 29
PhaseBoxplot ....................................................... 30
q_to_curve ........................................................ 31
reparam_curve ....................................................... 32
reparam_image ....................................................... 33
resamplecurve ....................................................... 34
rgam ............................................................... 34
sample_shapes ...................................................... 35
simu_data ........................................................ 36
simu_warp ........................................................ 36
simu_warp_median .................................................. 37
smooth.data ......................................................... 37
SqrtMean ........................................................... 38
SqrtMedian ......................................................... 39
srn_to_f ........................................................... 40
time_warping ......................................................... 40
toy_data ........................................................... 42
toy_warp ........................................................... 42
vertFPCA ........................................................... 43
warp_f_gamma ....................................................... 44
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 = 3, showplot = T, smooth_data = FALSE,
sparm = 25, parallel = FALSE, cores = 8, MaxItr = 51)

Arguments
- f: matrix (N x M) of M functions with N samples
- time: vector of size N describing the sample points
- num_comp: number of principal components to extract (default = 3)
- showplot: shows plots of functions (default = T)
- smooth_data: smooth data using box filter (default = F)
- sparm: 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)
- MaxItr: maximum number of iterations

Value
Returns a list containing
- f0: original functions
- fn: aligned functions - matrix (N x M) of M functions with N samples
- qn: aligned srvfs - similar structure to fn
- q0: original srvfs - similar structure to fn
- mqn: srvf mean - vector of length N
- gam: warping functions - vector of length N
- Dx: cost function
- vfpca: list containing
- q_pca: srvf principal directions
- f_pca: f principal directions
- latent: latent values
- coef: coefficients
- U: eigenvectors
References


Examples

data("simu_data")
out = align_fPCA(simu_data$f,simu_data$time,MaxItr = 1) # use more iterations for accuracy

Description

This function constructs the amplitude boxplot

Usage

AmplitudeBoxplot(warp_median, alpha = 0.05, ka = 1, showplot = TRUE)

Arguments

warp_median fdawarp object from time_warping of aligned data using the median
alpha quantile value (default=.05, i.e., 95%)
ka scalar for outlier cutoff (default=1)
showplot shows plots of functions (default = T)

Value

Returns a ampbox object containing

median_y median function
Q1 First quartile
Q3 Second quartile
Q1a First quantile based on alpha
Q3a Second quantile based on alpha
minn minimum extreme function
maxx maximum extreme function
outlier_index indexes of outlier functions
fmedian median function

References

Examples

data("simu_warp_median")
out <- AmplitudeBoxplot(simu_warp_median, showplot=FALSE)

**MPEG7 Curve Dataset**

**Description**

Contains the MPEG7 curve data set which is 20 curves in 65 classes. The array is structured with dimension (2,100,65,20)

**Usage**

data("mpeg7")

**Format**

an array of shape (2,100,65,20)

**calc_shape_dist**

**Elastic Shape Distance**

**Description**

Calculate elastic shape distance between two curves beta1 and beta2

**Usage**

calc_shape_dist(beta1, beta2, mode = "O")

**Arguments**

- beta1: array describing curve1 (n,T)
- beta2: array describing curve
- mode: Open ("O") or Closed ("C") curves

**Value**

d geodesic distance

**References**

Examples

data("mpeg7")
d = calc_shape_dist(beta[,1,1],beta[,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    array describing curve 1 (n,T)
beta2    array describing curve 2 (n,T)
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


Examples

data("mpeg7")
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(betamean, beta, mode = "O")

Arguments
betamean array (n,T) of mean curve
beta array (n,T,N) for N number of curves
mode Open ("O") or Closed ("C") curves

Value
K covariance matrix

References

Examples
data("mpeg7")
out = curve_srvf_align(beta[,1,1:2],maxit=2) # note: use more shapes, small for speed
K = curve_karcher_cov(out$betamean, beta[,1,1:2])

curve_karcher_mean  Karcher Mean of Curves

Description
Calculates Karcher mean of a collection of curves using the elastic square-root velocity (srvf) framework.

Usage
curve_karcher_mean(beta, mode = "O", maxit = 20)
curve_pair_align

Arguments

- **beta**: array (n,T,N) for N number of curves
- **mode**: Open ("O") or Closed ("C") curves
- **maxit**: maximum number of iterations

Value

Returns a list containing

- **mu**: mean srvf
- **betamean**: mean curve
- **v**: shooting vectors
- **q**: array of srvfs

References


Examples

data("mpeg7")
out = curve_karcher_mean(beta[,1,1:2],maxit=2) # note: use more shapes, small for speed
Value

a list containing

- beta2n: aligned curve 2 to 1
- q2n: aligned srvf 2 to 1
- gam: warping function
- q1: srvf of curve 1

References


Examples

data("mpeg7")
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(betamean, mu, K, mode = "O", no = 3, N = 5)

Arguments

- betamean: array (n,T) of mean curve
- mu: array (n,T) of mean srvf
- K: array (2*T,2*T) covariance matrix
- mode: Open ("O") or Closed ("C") curves
- no: number of components
- N: number of samples on each side of mean

Value

pd list describing principal directions
References


Examples

data("mpeg7")
out = curve_srvf_align(beta[,1,1:2], maxit=20) # note: use more shapes, small for speed
K = curve_karcher_cov(out$betamean, beta[,1,1:2])
pd = curve_principal_directions(out$betamean, out$q_mu, K)

curve_srvf_align  Align Curves

Description

Aligns a collection of curves using the elastic square-root velocity (srvf) framework.

Usage

curve_srvf_align(beta, mode = "O", maxit = 20)

Arguments

beta array (n,T,N) for N number of curves
mode Open ("O") or Closed ("C") curves
maxit maximum number of iterations

Value

Returns a list containing

betan aligned curves
qn aligned srvfs
betamean mean curve
q_mu mean SRVFs

References


Examples

data("mpeg7")
out = curve_srvf_align(beta[,1,1:2], maxit=20) # note: use more shapes, small for speed
K = curve_karcher_cov(out$betamean, beta[,1,1:2])
**curve_to_q**

*Convert to SRVF space*

**Description**

This function converts curves to SRVF

**Usage**

\[
\text{curve_to_q}(\beta)
\]

**Arguments**

\[
\beta \quad \text{array describing curve (n,T)}
\]

**Value**

\[
q \quad \text{array describing srvf}
\]

**References**


**Examples**

\[
\text{data("mpeg7")}
\]
\[
q = \text{curve_to_q}(\beta[,1,1])
\]

---

**elastic.distance**

*Calculates two elastic distance*

**Description**

This functions calculates the distances between functions, \(D_y\) and \(D_x\), where function 1 is aligned to function 2

**Usage**

\[
\text{elastic.distance}(f1, f2, time, lambda = 0)
\]
Arguments

- f1: sample function 1
- f2: sample function 2
- time: sample points of functions
- lambda: controls amount of warping (default = 0)

Value

Returns a list containing

- Dy: amplitude distance
- Dx: phase distance

References


Examples

data("simu_data")
distances <- elastic.distance(simu_data$f[,1], simu_data$f[,2], 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)
elastic.mlogistic

**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 ($N \times M$) 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**


---

**elastic.mlogistic**  *Elastic Multinomial Logistic Regression*

**Description**

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

**Usage**

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

- **f**: matrix \((N \times M)\) 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 \(B=\text{NULL}\) (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 \((N \times M)\) 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

elasticsearch

elasticsearch

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

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 \((N \times M)\) 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 \((N \times M)\) 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

**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.

**References**


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 = 500000, 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**: Dirchelet 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 (default 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


Examples

```r
## Not run:
data("simu_data")
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 ($N \times M$) of $M$ functions with $N$ samples
- `time`: sample points of functions
- `times`: factor of length of subsample points to look at (default = 5)
- `group`: (default 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**

**Examples**
```
## Not run:
data("simu_data")
out = function_mean_bayes(simu_data$f, simu_data$time)
## End(Not run)
```
### f_to_srvf

**Convert to SRSF**

**Description**
This function converts functions to srsf

**Usage**

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

**Arguments**

- `f` : matrix of functions
- `time` : time

**Value**

- `q` : matrix of SRSFs

**References**


**Examples**

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

### 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**

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

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


Examples

dataHBsimu_warpBI
outQ] gauss_modelHsimu_warpLn] QPI

---

**gradient**

*Gradient using finite differences*

Description

This function takes the gradient of f using finite differences

Usage

`gradient(f, binsize)`

Arguments

- f: vector with N samples
- binsize: scalar of time samples

Value

- g: vector with N samples which is the gradient of f

Examples

dataHBsimu_dataBI
out = gradient(simu_data$[,1],mean(diff(simu_data$time)))
growth_vel

Berkley Growth Velocity Dataset

Description

Combination of both boys and girls growth velocity from the Berkley Dataset

Usage

data("growth_vel")

Format

A list which contains f and time

horizFPCA

Horizontal Functional Principal Component Analysis

Description

This function calculates vertical functional principal component analysis on aligned data

Usage

horizFPCA(warp_data, no, showplot = TRUE)

Arguments

warp_data fdawarp object from time_warping of aligned data
no number of principal components to extract
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


Examples

data("simu_warp")
hfpc = horizFPCA(simu_warp,no = 3)

---

Example Image Data set

Description

Contains two simulated images for registration

Usage

data("image")

Format

a list containing two images of dimension (64,64)

---

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


Examples

data("simu_warp")
out = invertGamma(simu_warp$gam[,1])

kmeans_align  K-Means Clustering and Alignment

Description

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

Usage

kmeans_align(f, time, K, seeds = NULL, lambda = 0, showplot = TRUE,
smooth_data = FALSE, sparam = 25, parallel = FALSE, alignment = TRUE,
method = "DP", MaxItr = 50, thresh = 0.01)

Arguments

f  matrix (N x M) of M functions with N samples

time  vector of size N describing the sample points

K  number of clusters

seeds  indexes of cluster center functions (default = NULL)

lambda  controls the elasticity (default = 0)

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)

alignment  whether to perform alignment (default = T)

method  optimization method (DP,DP2,RBFGS)

MaxItr  maximum number of iterations

thresh  cost function threshold
Value

Returns a fdakma object containing

- `f0`: original functions
- `fn`: aligned functions - matrix \((N \times M)\) of \(M\) functions with \(N\) samples which is a list for each cluster
- `qn`: aligned SRSFs - similar structure to `fn`
- `q0`: original SRSFs
- `labels`: cluster labels
- `templates`: cluster center functions
- `templates.q`: cluster center SRSFs
- `gam`: warping functions - similar structure to `fn`
- `qun`: Cost Function Value

References


Examples

```r
data("growth_vel")
# use more iterations for accuracy
out <- kmeans_align(growth_vel$f, growth_vel$time, K=2, MaxIt=1, showplot=FALSE)
```

optimum.reparam: Align two functions

Description

This function aligns two SRSF functions using Dynamic Programming

Usage

```r
optimum.reparam(Q1, T1, Q2, T2, lambda = 0, method = "DP", w = 0.01,
    f1o = 0, f2o = 0)
```
outlier.detection

Arguments

- **Q1**: srsf of function 1
- **T1**: sample points of function 1
- **Q2**: srsf of function 2
- **T2**: sample points of function 2
- **lambda**: controls amount of warping (default = 0)
- **method**: controls which optimization method (default="DP") options are Dynamic Programming ("DP"), Coordinate Descent ("DP2"), and Riemannian BFGS ("RBFGS")
- **w**: controls LRBFGS (default = 0.01)
- **f1o**: initial value of f1, vector or scalar depending on q1, defaults to zero
- **f2o**: initial value of f2, vector or scalar depending on q1, defaults to zero

Value

- **gam**: warping function

References


Examples

```r
data("simu_data")
q = f_to_srvf(simu_data$f, simu_data$time)
gam = optimum.reparam(q[,1], simu_data$time, q[,2], simu_data$time)
```

Description

This function calculates outlier’s using geodesic distances of the SRVF from the median.

Usage

```r
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)
pair_align_functions

Value

q_outlier   outlier functions

References


Examples

data(“toy_data”)data(“toy_warp”)q_outlier = outlier.detection(toy_warp$q0,toy_data$time,toy_warp$qn,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, method = “DP”, w = 0.01, f1o = 0, f2o = 0)

Arguments

f1 function 1
f2 function 2
time sample points of functions
lambda controls amount of warping (default = 0)
method controls which optimization method (default="DP") options are Dynamic Programming (“DP”), Coordinate Descent (“DP2”), Riemannian BFGS (“RBFGS”) and Simultaneous Alignment (“SIMUL”)
w controls LRBFGS (default = 0.01)
f1o initial value of f1, vector or scalar depending on q1, defaults to zero
f2o initial value of f2, vector or scalar depending on q1, defaults to zero

Value

Returns a list containing

f2tilde aligned f2
gam warping function
References


Examples

data("simu_data")
out = pair_align_functions(simu_data$f[,1],simu_data$f[,2],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(timem * 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 Dirchelet 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
Pairwise align two images. This function aligns images using the q-map framework.

**Usage**

```r
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)

**References**

PhaseBoxplot

Value

Returns a list containing

\[ \text{Inew} \quad \text{aligned I2} \]

\[ \text{gam} \quad \text{warping function} \]

References


Examples

```r
## Not run:
# This is a gradient descent algorithm and takes a long time to run
data("image")
out <- pair_align_image(im$11, im$12)
## End(Not run)
```

Description

This function constructs the amplitude boxplot

Usage

```r
PhaseBoxplot(warp_median, alpha = 0.05, kp = 1, showplot = TRUE)
```

Arguments

- `warp_median`: fdawarp object from `time_warping` of aligned data using the median
- `alpha`: quantile value (default=.05, i.e., 95%)
- `kp`: scalar for outlier cutoff (default=1)
- `showplot`: shows plots of functions (default = T)

Value

Returns a phbox object containing

\[ \text{median}_x \quad \text{median warping function} \]

\[ \text{Q1} \quad \text{First quartile} \]

\[ \text{Q3} \quad \text{Second quartile} \]

\[ \text{Q1a} \quad \text{First quantile based on alpha} \]
q_to_curve

<table>
<thead>
<tr>
<th>Q3a</th>
<th>Second quantile based on alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>minn</td>
<td>minimum extreme function</td>
</tr>
<tr>
<td>maxx</td>
<td>maximum extreme function</td>
</tr>
<tr>
<td>outlier_index</td>
<td>indexes of outlier functions</td>
</tr>
</tbody>
</table>

References


Examples

data("simu_warp_median")
out <- PhaseBoxplot(simu_warp_median, showplot=FALSE)

Description

This function converts SRVFs to curves

Usage

q_to_curve(q)

Arguments

q array describing SRVF (n,T)

Value

beta array describing curve

References


Examples

data("mpeg7")
q = curve_to_q(beta[,1,1])
beta1 = q_to_curve(q)
reparam_curve  

**Align two curves**

**Description**

This function aligns two SRVF functions using Dynamic Programming.

**Usage**

```r
reparam_curve(beta1, beta2, lambda = 0, method = "DP", w = 0.01, rotated = T, isclosed = F)
```

**Arguments**

- `beta1`: array defining curve 1
- `beta2`: array defining curve 1
- `lambda`: controls amount of warping (default = 0)
- `method`: controls which optimization method (default="DP") options are Dynamic Programming ("DP"), Coordinate Descent ("DP2"), Riemannian BFGS ("RBFGS")
- `w`: controls LRBFGS (default = 0.01)
- `rotated`: boolean if rotation is desired
- `isclosed`: boolean if curve is closed

**Value**

return a List containing

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

**References**


**Examples**

```r
data("mpeg7")
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

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


Examples

data("mpeg7")

xn = resamplecurve(beta[,1],200)

rgam Random Warping

Description

Generates random warping functions.

Usage

rgam(N, sigma, num)
sample_shapes

Arguments

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

Value

gam warping functions

References


Examples

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

sample_shapes  Sample shapes from model

Description

Sample shapes from model

Usage

sample_shapes(mu, K, mode = "O", no = 3, numSamp = 10)

Arguments

mu  array (n,T) of mean srvf
K  array (2*T,2*T) covariance matrix
mode  Open ("O") or Closed ("C") curves
no  number of principal components
numSamp  number of samples

Value

samples list of sample curves
References

Examples
data("mpeg7")
out = curve_srvf_align(bet[1,1:2],maxit=2) # note: use more shapes, small for speed
K = curve_karcher_cov(out$betamean, bet[1:2])
samples = sample_shapes(out$q.mu, K)

simu_data
Simulated two Gaussian Dataset

Description
A functional dataset where the individual functions are given by: 
\[ y_i(t) = z_{i,1} e^{-(t-1.5)^2/2} + z_{i,2} e^{-(t+1.5)^2/2} \]
\[ t \in [-3,3], \ i = 1,2,...,21, \] 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:
\[ \gamma_i(t) = 6(e^{a_i(t+3)/(a_i-1)} - 1) - 3 \] 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
data("simu_data")

Format
A list which contains \( f \) and time

simu_warp
Aligned Simulated two Gaussian Dataset

Description
A functional dataset where the individual functions are given by: 
\[ y_i(t) = z_{i,1} e^{-(t-1.5)^2/2} + z_{i,2} e^{-(t+1.5)^2/2} \]
\[ t \in [-3,3], \ i = 1,2,...,21, \] 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:
\[ \gamma_i(t) = 6(e^{a_i(t+3)/(a_i-1)} - 1) - 3 \] 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
**simu_warp_median**

**Usage**

```r
data("simu_warp")
```

**Format**

A list which contains the outputs of the time_warping function

---

**Description**

A functional dataset where the individual functions are given by: 
\[ y_i(t) = z_{i,1}e^{-(t-1.5)^2/2} + z_{i,2}e^{-(t+1.5)^2/2}, \quad t \in [-3,3], \quad i = 1, 2, \ldots, 21 \]
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: 
\[ \gamma_i(t) = 6(e^{a_i(t+3)/6} - e^{a_i-1} - 3) \text{ if } a_i \neq 0, \text{ otherwise } \gamma_i = \gamma_{id}(\gamma_{id}(t) = t) \text{ is the identity warping}. \]
The variables are as follows: 
- \( f \) containing the 21 functions of 101 samples and 
- \( \text{time} \) which describes the sampling which has been aligned

**Usage**

```r
data("simu_warp_median")
```

**Format**

A list which contains the outputs of the time_warping function finding the median

---

**smooth.data**

**Smooth Functions**

**Description**

This function smooths functions using standard box filter

**Usage**

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

**Arguments**

- \( f \) matrix \((N \times M)\) of \( M \) functions with \( N \) samples
- \( \text{sparam} \) number of times to run box filter

**Value**

- \( f \) smoothed functions
References


Examples

data("simu_data")
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


Examples

data("simu_warp")
out = SqrtMean(simu_warp$gam)
**SqrtMedian**

_SRFV 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 \((N \times M)\) of \(M\) warping functions with \(N\) samples

**Value**

Returns a list containing

- \(\text{median}\) Karcher median psi function
- \(\text{gam\_median}\) Karcher mean warping function
- \(\text{psi}\) srvf of warping functions
- \(\text{vec}\) shooting vectors

**References**


**Examples**

```r
data("simu_warp_median")
out = SqrtMedian(simu_warp_median$gam)
```
srsf_to_f  

Convert SRSF to f

Description

This function converts SRSFs to functions

Usage

srsf_to_f(q, time, f0 = 0)

Arguments

q  
matrix of srsf

time  
time

f0  
initial value of f

Value

f matrix of functions

References


Examples

data("simu_data")
q = f_to_srvf(simu_data$f, simu_data$time)
f = srsf_to_f(q, simu_data$time, simu_data$f[1,])

time_warping  

Group-wise function alignment

Description

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

Usage

time_warping(f, time, lambda = 0, method = "mean", showplot = TRUE,
smooth_data = FALSE, sparam = 25, parallel = FALSE, omethod = "DP",
MaxIt = 20)
Arguments

- **f**: matrix \((N \times M)\) of \(M\) functions with \(N\) samples
- **time**: vector of size \(N\) describing the sample points
- **lambda**: controls the elasticity (default = 0)
- **method**: warp and calculate to Karcher Mean or Median (options = "mean" or "median", default = "mean")
- **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)
- **MaxItr**: maximum number of iterations

Value

Returns a fdawarp object containing

- **f0**: original functions
- **fn**: aligned functions - matrix \((N \times M)\) 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


Examples

```r
data("simu_data")
# use more iterations for accuracy
out = time_warping(simu_data$f,simu_data$time, MaxItr=1)
```
### 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**

```r
data("toy_data")
```

**Format**

A list which contains $f$ and $time$.

### 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**

```r
data("toy_warp")
```

**Format**

A list which contains the outputs of the time_warping function.
vertFPCA

Vertical Functional Principal Component Analysis

Description

This function calculates vertical functional principal component analysis on aligned data.

Usage

vertFPCA(warp_data, no, showplot = TRUE)

Arguments

- `warp_data`: fdawarp object from `time_warping` of aligned data.
- `no`: number of principal components to extract.
- `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.

References


Examples

data("simu_warp")

vfpca = vertFPCA(simu_warp, no = 3)
warp_f_gamma  

**Warp Function**

**Description**

This function warps function \( f \) by \( \gamma \)

**Usage**

\[
\text{warp\_f\_gamma}(f, \text{time}, \gamma)
\]

**Arguments**

- \( f \) : vector function
- \( \text{time} \) : time
- \( \gamma \) : vector warping function

**Value**

\( f_{\text{new}} \) warped function

**References**


**Examples**

```r
\text{data("simu\_data")}
\text{fnew = warp\_f\_gamma(simu\_data\$f[,1],simu\_data\$time,seq(0,1,length.out=101))}
```

---

warp_q_gamma  

**Warp SRSF**

**Description**

This function warps srsf \( q \) by \( \gamma \)

**Usage**

\[
\text{warp\_q\_gamma}(q, \text{time}, \gamma)
\]
warp_q_gamma

Arguments

- **q**: vector
- **time**: time
- **gamma**: vector warping function

Value

- **qnew warped function**

References


Examples

data("simu_data")
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))
Index

*Topic alignment,
   align_FPCA, 3
   elastic.distance, 11
   function_group_warp_bayes, 18
   function_mean_bayes, 19
   pair_align_functions_bayes, 28
*Topic alignment
   AmplitudeBoxplot, 4
   calc_shape_dist, 5
   curve_geodesic, 6
   curve_karcher_cov, 7
   curve_karcher_mean, 7
   curve_pair_align, 8
   curve_principal_directions, 9
   curve_srvf_align, 10
   curve_to_q, 11
   elastic.logistic, 12
   elastic.mlogistic, 13
   elastic.prediction, 15
   elastic.regression, 16
   f_to_srvf, 20
   gradient, 21
   horizFPCA, 22
   invertGamma, 23
   kmeans_align, 24
   optimum.reparam, 25
   pair_align_functions, 27
   pair_align_image, 29
   PhaseBoxplot, 30
   q_to_curve, 31
   reparam_curve, 32
   reparam_image, 33
   resamplecurve, 34
   sample_shapes, 35
   smooth.data, 37
   SqrtMean, 38
   SqrtMedian, 39
   srsf_to_f, 40
   time_warping, 40
   vertFPCA, 43
   warp_f_gamma, 44
   warp_q_gamma, 44
*Topic bayesian
   function_group_warp_bayes, 18
   function_mean_bayes, 19
   pair_align_functions_bayes, 28
*Topic boxplot
   AmplitudeBoxplot, 4
   PhaseBoxplot, 30
*Topic clustering
   kmeans_align, 24
*Topic datasets
   beta, 5
   growth_vel, 22
   im, 23
   simu_data, 36
   simu_warp, 36
   simu_warp_median, 37
   toy_data, 42
   toy_warp, 42
*Topic detection
   outlier.detection, 26
*Topic diffeomorphism,
   rgam, 34
*Topic distances
   elastic.distance, 11
*Topic function
   rgam, 34
*Topic image
   pair_align_image, 29
   reparam_image, 33
*Topic outlier
   outlier.detection, 26
*Topic pca
   align_FPCA, 3
   gauss_model, 20
*Topic regression
   elastic.logistic, 12
elastic.m.logistic, 13
elastic.prediction, 15
elastic.regression, 16
+Topic srsv
  f_to_srsv, 20
  function_group_warp_bayes, 18
  function_mean_bayes, 19
  kmeans_align, 24
  optimum.reparam, 25
  pair_align_functions, 27
  pair_align_functions_bayes, 28
  srsv_to_f, 40
  time_warping, 40
+Topic srvf
  align_fPCA, 3
  AmplitudeBoxplot, 4
  calc_shape_dist, 5
  curve_geodesic, 6
  curve_karcher_cov, 7
  curve_karcher_mean, 7
  curve_pair_align, 8
  curve_principal_directions, 9
  curve_srvf_align, 10
  curve_to_q, 11
  elastic.distance, 11
  elastic.logistic, 12
  elastic.m.logistic, 13
  elastic.prediction, 15
  elastic.regression, 16
  gradient, 21
  horizFPCA, 22
  invertGamma, 23
  outlier_detection, 26
  PhaseBoxplot, 30
  q_to_curve, 31
  reparam_curve, 32
  resamplecurve, 34
  sample_shapes, 35
  smooth.data, 37
  SqrtMean, 38
  SqrtMedian, 39
  vertFPCA, 43
  warp_f_gamma, 44
  warp_q_gamma, 44
+Topic warping
  rgam, 34
  align_fPCA, 3
  AmplitudeBoxplot, 4
  beta, 5
  calc_shape_dist, 5
  curve_geodesic, 6
  curve_karcher_cov, 7
  curve_karcher_mean, 7
  curve_pair_align, 8
  curve_principal_directions, 9
  curve_srvf_align, 10
  curve_to_q, 11
  elastic.distance, 11
  elastic.logistic, 12
  elastic.m.logistic, 13
  elastic.prediction, 15
  elastic.regression, 16
  f_to_srsv, 20
  fdasrvf, 17
  fdasrvf-package (fdasrvf), 17
  foreach, 3, 13, 14, 16, 24, 41
  function_group_warp_bayes, 18
  function_mean_bayes, 19
  gauss_model, 20
  gradient, 21
  growth_vel, 22
  horizFPCA, 22
  im, 23
  invertGamma, 23
  kmeans_align, 24
  optimum.reparam, 25
  outlier_detection, 26
  pair_align_functions, 27
  pair_align_functions_bayes, 28
  pair_align_image, 29
  PhaseBoxplot, 30
  q_to_curve, 31
  reparam_curve, 32
  reparam_image, 33
  resamplecurve, 34
  rgam, 34
  sample_shapes, 35
simu_data, 36
simu_warp, 36
simu_warp_median, 37
smooth.data, 37
SqrtMean, 38
SqrtMedian, 39
srsf_to_f, 40
time_warping, 4, 21, 22, 26, 30, 40, 43
toy_data, 42
toy_warp, 42
vertFPCA, 43
warp_f_gamma, 44
warp_q_gamma, 44