Package ‘elastes’

August 21, 2023

Type Package

Title Elastic Full Procrustes Means for Sparse and Irregular Planar Curves

Version 0.1.7

Description Provides functions for the computation of functional elastic shape means over sets of open planar curves. The package is particularly suitable for settings where these curves are only sparsely and irregularly observed. It uses a novel approach for elastic shape mean estimation, where planar curves are treated as complex functions and a full Procrustes mean is estimated from the corresponding smoothed Hermitian covariance surface. This is combined with the methods for elastic mean estimation proposed in Steyer, Stöcker, Greven (2022) <doi:10.1111/biom.13706>. See Stöcker et. al. (2022) <arXiv:2203.10522> for details.

License GPL (>= 3)

Encoding UTF-8

Language en-US

Imports elasdics, utils, graphics, stats, splines, mgcv, sparseFLMM, orthogonalsplinebasis

Suggests knitr, covr, testthat (>= 3.0.0), rmarkdown, shapes

RoxygenNote 7.2.3

Config/testthat/edition 3

VignetteBuilder knitr

URL `https://mpff.github.io/elastes/`, `https://github.com/mpff/elastes`

BugReports `https://github.com/mpff/elastes/issues`

NeedsCompilation no

Author Manuel Pfeuffer [aut, cre],
Lisa Steyer [aut],
Almond Stoecker [aut]

Maintainer Manuel Pfeuffer <`manuel.pfeuffer@hu-berlin.de`>

Repository CRAN

Date/Publication 2023-08-21 15:00:29 UTC

R topics documented:

`compute_elastic_shape_mean`

Compute an elastic full Procrustes mean for a collection of curves

---

Description

Computes an elastic full Procrustes mean for curves stored in `data_curves`. Constructor function for class `elastic_shape_mean`.

Usage

```
compute_elastic_shape_mean(
  data_curves,
  knots = seq(0, 1, len = 13),
  type = c("smooth", "polygon"),
  penalty = 2,
  var_type = c("smooth", "constant", "zero"),
  pfit_method = c("smooth", "polygon"),
  smooth_warp = function(i) 0,
  eps = 0.05,
  max_iter = 50,
  verbose = FALSE,
  cluster = NULL
)
```

Arguments

`data_curves`	list of `data.frames` with observed points in each row. Each variable is one coordinate direction. If there is a variable `t`, it is treated as the time parametrization, not as an additional coordinate.
`knots`	set of knots for the mean spline curve

`type` if "smooth" linear srv-splines are used which results in a differentiable mean curve if "polygon" the mean will be piecewise linear.

`penalty` the penalty to use in the covariance smoothing step. use '-1' for no penalty.

`var_type` (experimental) assume "smooth", "constant" or "zero" measurement-error variance along t

`pfit_method` (experimental) "smooth" or "polygon"

`smooth_warp` (experimental) controls the weighting of original and smoothed observations over the iterations, if pfit_method == "smooth".

`eps` the algorithm stops if L2 norm of coefficients changes by less than eps

`max_iter` maximal number of iterations

`verbose` print iterations

`cluster` (experimental) use the parallel package for faster computation

Value

an object of class `elastic_shape_mean`, which is a `list` with entries

`type` "smooth" if mean was modeled using linear srv-splines, "polygon" if constant srv-splines

`coefs` spline coefficients

`knots` spline knots

`variance` sample elastic shape variance

`data_curves` list of `data.frames` with observed points in each row. First variable `t` gives the initial parametrization, second variable `t_optim` the optimal parametrization when the curve is aligned to the mean. Has the attributes 'rotation', 'scaling', 'translation' and 'dist_to_mean'. Use `get_procrustes_fit` to get the elastic full Procrustes fit.

`fit` see `fit_mean`

Examples

```
curve <- function(t){
  rbind(t*cos(13*t), t*sin(13*t))
}
set.seed(18)
data_curves <- lapply(1:4, function(i){
  m <- sample(10:15, 1)
  delta <- abs(rnorm(m, mean = 1, sd = 0.05))
  t <- cumsum(delta)/sum(delta)
  data.frame(t(curve(t)) + 0.07*t*matrix(cumsum(rnorm(2*length(delta))),
             ncol = 2))
})

#randomly rotate and scale curves
rand_scale <- function(curve){ ( 0.5 + runif(1) ) * curve }
rand_rotate <- function(curve){
  names <- colnames(curve)
  theta <- 2*pi*runif(1)
  mat <- matrix(c(cos(theta), sin(theta), -sin(theta), cos(theta)), nrow = 2, ncol = 2)
  curve.rot <- as.matrix(curve) %*% t(mat)
  curve.rot <- as.data.frame(curve.rot)
  colnames(curve.rot) <- names
  return(curve.rot)
}
data_curves <- lapply(data_curves, rand_scale)
data_curves <- lapply(data_curves, rand_rotate)

#compute smooth procrustes mean with 2 order penalty
knots <- seq(0,1, length = 11)
elastic_shape_mean <- compute_elastic_shape_mean(
    data_curves,
    knots = knots,
    type = "smooth",
    penalty = 2
    )
plot(elastic_shape_mean)
```

fit_alignment_proc2d *Optimal rotation and scaling alignment to a smooth curve*

Description

Finds optimal rotation and scaling alignment for a discrete open srv curve to a smooth curve

Usage

```
fit_alignment_proc2d(
  q,
  type,
  knots,
  var_type,
  coefs.compl,
  method,
  cov_fit,
  pca,
  L
)
```

Arguments

q	complex srv curve with parametrization, needs to be vectorized. The result of a call to get_model_data_complex
type	spline degree
knots	basis knots
var_type	either "smooth" or "constant" measurement error in cov_fit object

`coefs.compl` complex coefficients of smooth curve

`method` temp

`cov_fit` temp

`pca` temp

`L` temp

Value

optimal rotation G and scaling b

`fit_mean` *Mean estimation for open planar curves.*

Description

Fits an elastic full Procrustes mean for open, planar curves. Is usually called from `compute_elastic_shape_mean`.

Usage

```
fit_mean(
  srv_data_curves,
  knots,
  penalty,
  var_type,
  pfit_method,
  max_iter,
  type,
  eps,
  cluster,
  verbose,
  smooth_warp
)
```

Arguments

`srv_data_curves`
list of `data.frames` with srv vectors in each row.curves

`knots` set of knots for the mean spline curve

`penalty` the penalty to use in the covariance smoothing step. use '-1' for no penalty.

`var_type` (experimental) assume "smooth", "constant" or "zero" measurement-error variance along t

`pfit_method` (experimental) "smooth" or "polygon"

`max_iter` maximal number of iterations

`type` if "smooth" linear srv-splines are used which results in a differentiable mean curve if "polygon" the mean will be piecewise linear.

`eps` the algorithm stops if L2 norm of coefficients changes less

`cluster` a cluster object for use in the `bam` call

`verbose` print iterations

`smooth_warp` (experimental) controls the weighting of original and smoothed observations over the iterations, if pfit_method == "smooth".

Value

a `list` with entries

`type` "smooth" or "polygon"

`coefs` `coefs` srv spline coefficients of the estimated mean

`knots` spline knots

`penalty` penalty used in the covariance estimation

`distances` distances to mean

`fit` a `list` containing `t_optims`optimal parametrizations `G_optims`optimal rotations `b_optims`optimal scalings `n_optims`optimal re-normalization `n_iter`number of iterations until convergence `gram` the mean basis Gram matrix, `cov_fit` the covariance smoothing objects in the final iteration, `cov_pca` cov coef matrix pca object in the final iteration and `pfit_coefs` the mean basis coefs of smoothed pfits in the final iteration

---

`get_center` *Calculate the center of a curve*

Description

Calculate the center of a curve

Usage

```
get_center(curve)
```

Arguments

`curve` a `data.frame` with observed points in each row. Each variable is one coordinate direction. If there is a variable `t`, `t_optim` or `id`, it is treated as the time parametrization, not as an additional coordinate.

Value

The average of observed points in `curve`.

get_distance *Distance to a smooth curve*

Description

Finds the distance of a discrete open srv curve to a smooth curve

Usage

```
get_distance(srv_curve, s, q, eps = 10 * .Machine$double.eps)
```

Arguments

srv_curve	srv transformation of the smooth curve, needs to be vectorized
s	time points for q, first has to be 0, last has to be 1
q	square root velocity vectors, one less than time points in s
eps	convergence tolerance

Value

distance between srv_curve and q

get_evals *Evaluate a curve on a grid*

Description

Evaluate a curve on a grid

Usage

```
get_evals(curve, t_grid = NULL, ...)

## S3 method for class 'data.frame'
get_evals(curve, t_grid = NULL, ...)

## S3 method for class 'elastic_shape_mean'
get_evals(curve, t_grid = NULL, centering = TRUE, srv = FALSE, ...)
```

Arguments

curve	a one parameter function which is to be evaluated on a grid
t_grid	the curve is evaluated at the values in t_grid, first value needs to be 0, last value needs to be 1. If t_grid = NULL, a default regular grid with grid length 0.01 is chosen
...	other arguments
centering	TRUE if curves shall be centered
srv	TRUE if SRV curve shall be evaluated

Value

a `data.frame` with evaluations of the curve at the values in `t_grid` in its rows.

See Also

See `get_evals` for the original code.

Examples

```
curve <- function(t){c(t*sin(10*t), t*cos(10*t))}
plot(get_evals(curve), type = "b")
```

---

get_optimal_t *Finds optimal alignment for discrete open curves*

Description

Finds optimal aligned time points for srv curve q to srv curve p using coordinate wise optimization.

Usage

```
get_optimal_t(srv_procrustes_curves, coefs, t_optims, type, knots, eps, i)
```

Arguments

srv_procrustes_curves	scaling and rotation aligned srv curves
coefs	mean coefficients
t_optims	current optimal parametrization
type	"smooth" or "polygon"
knots	mean basis knots
eps	convergence tolerance
i	current iteration

Value

optimal time points for srv_data_curves, without first value 0 and last value 1 optimal time points have the distance of the observation to the srv_curve as an attribute

`get_polygon_length` *Calculate the polygon length of a curve*

Description

Calculate the polygon length of a curve

Usage

```
get_polygon_length(curve)
```

Arguments

`curve` a `data.frame` with observed points in each row. Each variable is one coordinate direction. If there is a variable `t`, `t_optim` or `id`, it is treated as the time parametrization, not as an additional coordinate.

Value

The length of `curve`, treating it as a polygon.

`get_procrustes_fit` *Get Procrustes data curve from mean object.*

Description

Compute the Procrustes aligned data curve...

Usage

```
get_procrustes_fit(data_curve)
```

Arguments

`data_curve` A `data.frame` in an `elastic_shape_mean` object.

Value

Aligned data_curve as a `data.frame`.

`get_Procrustes_fit_from_param` *Helper functions for calculating Procrustes data curve from rotation, scaling and translation parameters.*

Description

Compute the Procrustes fit given optimal rotation, scaling and translation.

Usage

```
get_procrustes_fit_from_param(
  data_curve,
  rot,
  scale,
  plength,
  trans,
  norm_factor
)
```

Arguments

`data_curve`	A `data.frame` with observed points on a curve. Each row is one point, each variable one coordinate direction. If there is a variable `t`, it is treated as the time parametrization, not as an additional coordinate.
`rot`	The rotation (in radian).
`scale`	The scaling.
`plength`	The polygon length of the original curve.
`trans`	The translation.
`norm_factor`	The normalization factor from the smooth curve estimate.

`plot.elastic_shape_mean` *Plot method for planar elastic Procrustes mean curves*

Description

Plots objects of class `elastic_shape_mean`.

Usage

```
## S3 method for class 'elastic_shape_mean'
plot(x, srv = FALSE, centering = TRUE, asp = 1, col = "red", ...)
```

Arguments

`x` object of class `elastic_shaped_mean`, usually a result of a call to `compute_elastic_shape_mean`

`srv` TRUE if the SRV curve should be plotted

`centering` TRUE if mean and pfits should be centered

`asp` numeric, giving the aspect ratio of the two coordinates, see `plot.window` for details.

`col` color of the mean curve.

`...` further plotting parameters.

Value

No return value, called for side effects.

See Also

For examples see documentation of `compute_elastic_shape_mean`. See `plot.elastic_mean` for the original code.

Index