Package 'registr'

October 14, 2022

Title Curve Registration for Exponential Family Functional Data

Version 2.1.0

Description A method for performing joint registration and functional principal component analysis for curves (functional data) that are generated from exponential family distributions. This mainly implements the algorithms described in 'Wrobel et al. (2019)' <doi:10.1111/biom.12963> and further adapts them to potentially incomplete curves where (some) curves are not observed from the beginning and/or until the end of the common domain. Curve registration can be used to better understand patterns in functional data by separating curves into phase and amplitude variability. This software handles both binary and continuous functional data, and is especially applicable in accelerometry and wearable technology.

Depends R (>= 3.2)

Imports tidyr, magrittr, dplyr, pbs, Rcpp (>= 0.11.5), parallel, MASS, utils, gamm4, lme4, mgcv, purrr, Matrix

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, knitr, rmarkdown, cowplot, ggplot2, pbapply, fastglm

SystemRequirements GNU make

VignetteBuilder knitr

NeedsCompilation yes

Author Julia Wrobel [aut, cre] (<https://orcid.org/0000-0001-6783-1421>),
Alexander Bauer [aut],
Erin McDonnell [aut],
Fabian Scheipl [ctb],
Jeff Goldsmith [aut]

Maintainer Julia Wrobel <julia.wrobel@cuanschutz.edu>

Repository CRAN

Date/Publication 2022-10-02 21:40:02 UTC

R topics documented:

amp_curve *Simulate amplitude variance*

Description

This function generates amplitudes for simulated accelerometer data.

Usage

```
amp_curve(grid, period = 2 * pi, spline_based = FALSE)
```

Arguments

grid	Grid of x values over which to evaluate the function.
period	Controls the period of the mean curve
spline_based	If FALSE curve is constructed using sine and cosine functions, if TRUE, curve is constructed using B-spline basis.

Value

A numeric vector.

bfpca *Binary functional principal components analysis*

Description

Function used in the FPCA step for registering binary functional data, called by register_fpca when family = "binomial". This method uses a variational EM algorithm to estimate scores and principal components for binary functional data.

The number of functional principal components (FPCs) can either be specified directly (argument npc) or chosen based on the explained share of variance (npc_varExplained). In the latter case, the explained share of variance and accordingly the number of FPCs is estimated before the main estimation step by once running the FPCA with npc = 20 (and correspondingly Kt = 20). Doing so, we approximate the overall variance in the data Y with the variance represented by the FPC basis with 20 FPCs.

Usage

```
bfpca(
  Y,
  npc = NULL,
  npc_varExplained = NULL,
  Kt = 8,
  maxiter = 50,
  t_min = NULL,
  t_max = NULL,
  print.iter = FALSE,
  row_obj = NULL,
  seed = 1988,
  periodic = FALSE,
  error_thresh = 1e-04,
  verbose = 1,
  subsample = TRUE,
  ...
)
```

Arguments

`Y` Dataframe. Should have variables id, value, index.

`npc` The number of functional principal components (FPCs) has to be specified either directly as `npc` or based on their explained share of variance. In the latter case, `npc_varExplained` has to be set to a share between 0 and 1.

`npc_varExplained` The number of functional principal components (FPCs) has to be specified either directly as `npc` or based on their explained share of variance. In the latter case, `npc_varExplained` has to be set to a share between 0 and 1.

`Kt` Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8. If `npc_varExplained` is used, `Kt` is set to 20.

`maxiter` Maximum number of iterations to perform for EM algorithm. Default is 50.

`t_min` Minimum value to be evaluated on the time domain.

`t_max` Maximum value to be evaluated on the time domain.

`print.iter` Prints current error and iteration

`row_obj` If NULL, the function cleans the data and calculates row indices. Keep this NULL if you are using standalone `register` function.

`seed` Set seed for reproducibility. Defaults to 1988.

`periodic` If TRUE, uses periodic b-spline basis functions. Default is FALSE.

`error_thresh` Error threshold to end iterations. Defaults to 0.0001.

`verbose` Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

`subsample` if the number of rows of the data is greater than 10 million rows, the ‘id‘ values are subsampled to get the mean coefficients.

`...` Additional arguments passed to or from other functions

Value

An object of class `fpca` containing:

`fpca_type` Information that FPCA was performed with the 'variationEM' approach, in contrast to registr::gfpca_twoStep.

`t_vec` Time vector over which the mean `mu` and the functional principal components `efunctions` were evaluated.

`knots` Cutpoints for B-spline basis used to rebuild `alpha`.

`efunctions` $D \times npc$ matrix of estimated FPC basis functions.

`evalues` Estimated variance of the FPC scores.

`evalues_sum` Approximation of the overall variance in `Y`, based on an initial run of the FPCA with `npc = 20`. Is `NULL` if `npc_varExplained` was not specified.

`npc` number of FPCs.

`scores` $I \times npc$ matrix of estimated FPC scores.

`alpha` Estimated population-level mean.

`mu` Estimated population-level mean. Same value as `alpha` but included for compatibility with `refund.shiny` package.

`subject_coefs` B-spline basis coefficients used to construct subject-specific means. For use in `registr()` function.

`Yhat` FPC approximation of subject-specific means, before applying the response function.

`Y` The observed data.

`family` `binomial`, for compatibility with `refund.shiny` package.

`error` vector containing error for each iteration of the algorithm.

Author(s)

Julia Wrobel <`julia.wrobel@cuanschutz.edu`>, Jeff Goldsmith <`ajg2202@cumc.columbia.edu`>, Alexander Bauer <`alexander.bauer@stat.uni-muenchen.de`>

References

Jaakkola, T. S. and Jordan, M. I. (1997). A variational approach to Bayesian logistic regression models and their extensions. *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*.

Tipping, M. E. (1999). Probabilistic Visualisation of High-dimensional binary data. *Advances in neural information processing systems*, 592–598.

Examples

```
Y = simulate_functional_data()$Y

# estimate 2 FPCs
bfpca_obj = bfpca(Y, npc = 2, print.iter = TRUE, maxiter = 25)



plot(bfpca_obj)

# estimate npc adaptively, to explain 90% of the overall variation
bfpca_obj2 = bfpca(Y, npc_varExplained = 0.9, print.iter = TRUE, maxiter = 30)
plot(bfpca_obj2)
```

`bfpca_argPreparation` *Internal main preparation function for bfpca*

Description

Internal main preparation function for bfpca

Usage

```
bfpca_argPreparation(
  Y,
  Kt,
  time,
  t_min,
  t_max,
  periodic,
  seed,
  subsample,
  verbose
)
```

Arguments

`Y, time, t_min, t_max`
Internal objects created in `bfpca`.

`Kt` Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8. If `npc_varExplained` is used, `Kt` is set to 20.

`periodic` If TRUE, uses periodic b-spline basis functions. Default is FALSE.

`seed` Set seed for reproducibility. Defaults to 1988.

`subsample` if the number of rows of the data is greater than 10 million rows, the ‘id‘ values are subsampled to get the mean coefficients.

verbose — Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

Value

List with elements knots, Theta_phi, xi, alpha_coefs.

---

bfpca_optimization *Internal main optimization for bfpca*

Description

Main optimization function for bfpca. If npc_varExplained is specified, the function simply returns a list with elements npc (chosen number of FPCs), evalues (estimated variances of the first 'npc' FPCs) and evalues_sum (sum of the estimated variances of the first 20 FPCs, as approximation of the overall variance).

Usage

```
bfpca_optimization(
  npc,
  npc_varExplained = NULL,
  Kt,
  maxiter,
  print.iter,
  seed,
  periodic,
  error_thresh,
  verbose,
  Y,
  rows,
  I,
  knots,
  Theta_phi,
  xi,
  alpha_coefs
)
```

Arguments

npc — The number of functional principal components (FPCs) has to be specified either directly as npc or based on their explained share of variance. In the latter case, npc_varExplained has to be set to a share between 0 and 1.

npc_varExplained — The number of functional principal components (FPCs) has to be specified either directly as npc or based on their explained share of variance. In the latter case, npc_varExplained has to be set to a share between 0 and 1.

`Kt` Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8. If `npc_varExplained` is used, `Kt` is set to 20.

`maxiter` Maximum number of iterations to perform for EM algorithm. Default is 50.

`print.iter` Prints current error and iteration

`seed` Set seed for reproducibility. Defaults to 1988.

`periodic` If TRUE, uses periodic b-spline basis functions. Default is FALSE.

`error_thresh` Error threshold to end iterations. Defaults to 0.0001.

`verbose` Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

`Y, rows, I, knots, Theta_phi, xi, alpha_coefs` Internal objects created in `bfpca`.

Value

list with elements `t_vec`, `Theta_phi_mean`, `alpha_coefs`, `efunctions`, `evalues`, `evalues_sum`, `scores`, `subject_coef`, `fittedVals`, `error`. See documentation of `fpca_gauss` for details.

`bs_deriv` *Nth derivative of spline basis*

Description

This function gets derivative of a spline basis. Adapted from `bs()` function in `splines` package.

Usage

```
bs_deriv(
  x,
  knots,
  degree = 3L,
  Boundary.knots = range(x),
  derivative = 1,
  intercept = TRUE
)
```

Arguments

`x` a numeric vector of values at which to evaluate the B-spline functions or derivatives.

`knots` the internal breakpoints that define the spline.

`degree` degree of the piecewise polynomial—default is 3 for cubic splines.

`Boundary.knots` boundary points at which to anchor the B-spline basis. Set to [0,1] if you want this to be your domain.

`derivative` a positive integer value that specifies which derivative to take. Defaults to 1 for 1st derivative. Value of 0 returns the original set of b-spline basis functions.

`intercept` if TRUE, an intercept is included in the basis; default is TRUE

Value

A matrix containing:

`basis` A B-spline basis that can be used to approximate the derivative of a function.

Author(s)

Julia Wrobel `<julia.wrobel@cuanschutz.edu>`

`coarsen_index` *Coarsen an index vector to a given resolution*

Description

Reduce the resolution of a numeric vector by specifying the number of `significant_digits` to which the numbers should be rounded.

Internal function used to coarsen the index vector before estimating the two-step GFPCA with `gfpca_twoStep`.

Usage

```
coarsen_index(index, significant_digits)
```

Arguments

`index` Numeric vector of index values.

`significant_digits`
Positive integer value.

Value

Numeric vector of rounded index values.

Author(s)

Alexander Bauer `<alexander.bauer@stat.uni-muenchen.de>`

Examples

```
index_vector = c(0.7892, 0.2984, 0.328)
registr:::coarsen_index(index_vector, 1)
registr:::coarsen_index(index_vector, 3)

index_vector2 = c(2803, -7639, 13)
registr:::coarsen_index(index_vector2, 1)
registr:::coarsen_index(index_vector2, 3)
```

constraints *Define constraints for optimization of warping functions*

Description

Constraints ensure monotonicity of spline coefficients for warping functions for use with constrOptim() function.

Usage

```
constraints(Kh, t_min = 0, t_max = 1, warping = "nonparametric")
```

Arguments

Kh	Number of B-spline basis functions used to estimate warping functions *h*.
t_min	Minimum value to be evaluated on the time domain.
t_max	Maximum value to be evaluated on the time domain.
warping	If nonparametric (default), inverse warping functions are estimated nonparametrically. If piecewise_linear2 they follow a piecewise linear function with 2 knots.

Value

An list containing:

ui	A constraint matrix.
ci	A constraint vector.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>, Erin McDonnell <eim2117@cumc.columbia.edu>

cov_hall *Covariance estimation after Hall et al. (2008)*

Description

Internal function for the estimation of the covariance matrix of the latent process using the approach of Hall et al. (2008). Used in the two-step GFPCA approach implemented in `gfpca_twoStep`.

This function is an adaptation of the implementation of Jan Gertheiss and Ana-Maria Staicu for Gertheiss et al. (2017), with focus on higher (RAM) efficiency for large data settings.

Usage

```
cov_hall(
  Y,
  index_evalGrid,
  Kt = 25,
  Kc = 10,
  family = "gaussian",
  diag_epsilon = 0.01,
  make_pd = TRUE
)
```

Arguments

`Y`	Dataframe. Should have values id, value, index.
`index_evalGrid`	Grid for the evaluation of the covariance structure.
`Kt`	Number of P-spline basis functions for the estimation of the marginal mean. Defaults to 25.
`Kc`	Number of marginal P-spline basis functions for smoothing the covariance surface. Defaults to 10.
`family`	One of `c("gaussian","binomial","gamma","poisson")`. Poisson data are rounded before performing the GFPCA to ensure integer data, see Details section below. Defaults to `"gaussian"`.
`diag_epsilon`	Small constant to which diagonal elements of the covariance matrix are set if they are smaller. Defaults to 0.01.
`make_pd`	Indicator if positive (semi-)definiteness of the returned latent covariance should be ensured via `Matrix::near_PD()`. Defaults to TRUE.

Details

The implementation deviates from the algorithm described in Hall (2008) in one crucial step – we compute the crossproducts of *centered* observations and smooth the surface of these crossproducts directly instead of computing and smoothing the surface of crossproducts of uncentered observations and subsequently subtracting the (crossproducts of the) mean function. The former seems to yield smoother eigenfunctions and fewer non-positive-definite covariance estimates.

If the data Y or the crossproduct matrix contain more than 100,000 rows or elements, the estimation of the marginal mean or the smoothing step of the covariance matrix are performed by using the discretization-based estimation algorithm in bam rather than the gam estimation algorithm.

Value

Covariance matrix with dimension time_evalGrid x time_evalGrid.

Author(s)

Alexander Bauer <alexander.bauer@stat.uni-muenchen.de> and Fabian Scheipl, based on work of Jan Gertheiss and Ana-Maria Staicu

References

Hall, P., Müller, H. G., & Yao, F. (2008). Modelling sparse generalized longitudinal observations with latent Gaussian processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4), 703–723.

Gertheiss, J., Goldsmith, J., & Staicu, A. M. (2017). A note on modeling sparse exponential-family functional response curves. *Computational statistics & data analysis*, 105, 46–52.

Examples

```
data(growth_incomplete)

index_grid = c(1.25, seq(from = 2, to = 18, by = 1))
cov_matrix = registr:::cov_hall(growth_incomplete, index_evalGrid = index_grid)
```

crossprods_irregular *Crossproduct computation for highly irregular grids*

Description

Compute the crossproduct in a fast way for highly irregular grids (index values are mostly unique). Only used internally in cov_hall().

Usage

```
crossprods_irregular(Y)
```

Arguments

Y	Dataframe with the centered observations. Should have values id, centered, index.

crossprods_regular *Crossproduct computation for mostly regular grids*

Description

Compute the crossproduct in a fast way for mostly regular grids (index values are mostly *not* unique). Only used internally in `cov_hall()`.

Usage

```
crossprods_regular(Y)
```

Arguments

`Y`	Dataframe with the centered observations. Should have values id, centered, index.

data_clean *Convert data to a* `refund` *object*

Description

Function used for data cleaning.

Usage

```
data_clean(data)
```

Arguments

`data`	Dataframe. Should have values id, value, index.

Value

An list containing:

`Y`	The original data sorted by id and index.
`Y_rows`	A dataframe containing the first and last row for each subject.

deriv.inv.logit *Estimate the derivative of the logit function*

Description

Compute the derivative of the logit function for a given point x.

Usage

```
## S3 method for class 'inv.logit'
deriv(x)
```

Arguments

x	Value at which the derivative is computed

Author(s)

Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>

determine_npc *Determine the number of FPCs based on the share of explained variance*

Description

This internal function is called in gfpca_twoStep, fpca_gauss and bfpca to determine the number of functional principal components based on their share of explained variance.

Usage

```
determine_npc(evalues, npc_criterion)
```

Arguments

evalues	Vector of estimated variances of the FPC scores.
npc_criterion	Either (i) a share between 0 and 1, or (ii) a vector with two elements for the targeted explained share of variance and a cut-off scree plot criterion, both between 0 and 1. For the latter, e.g., npc_criterion = c(0.9,0.02) tries to choose a number of FPCs that explains at least 90% of variation, but only includes FPCs that explain at least 2% of variation (even if this means 90% explained variation is not reached).

Value

Integer for the number of fucntional principal components

ensure_proper_beta *Correct slightly improper parameter vectors*

Description

Internal function. In the joint iterations between registration and GFPCA, the optimization with `constrOptim()` in the registration step sometimes leads to slightly improper solutions, which cause the optimization to throw an error in the following optimization step. This function corrects the parameter vector if one of the following slight inconsistencies occurs that can mess with the optimization of `constrOptim()`:
- two neighboring values of the parameter vector are too similar
- the initial values of the parameter vector are smaller than `t_min`, the minimum of the underlying time domain
- the last values of the parameter vector are greater than `t_max`, the maximum of the underlying time domain
- one parameter value is slightly greater than its following value, i.e. the parameter vector is not monotone.

Usage

```
ensure_proper_beta(beta, t_min, t_max)
```

Arguments

`beta`	Parameter vector.
`t_min, t_max`	Minimum and maximum of the underlying time domain in the registration step.

Value

A slightly changed parameter vector that ensures a proper solution in the optimization of the registration step.

Author(s)

Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>

Examples

```
beta_improper = c(0.24, 1.000047, 1.000002)
registr:::ensure_proper_beta(beta_improper, t_min = 0, t_max = 1)
```

expectedScores *Calculate expected score and score variance for the current subject.*

Description

Calculations derived using maximum likelihood estimation.

Usage

```
expectedScores(Y, mu, psi, theta, theta_quad)
```

Arguments

`Y`	vector of observations for the current subject.
`mu`	vector of spline coefficients for the population mean.
`psi`	matrix of spline coefficients for the principal component basis functions.
`theta`	spline basis functions for the current subject.
`theta_quad`	quadratic form of theta for the current subject.

Value

A list with expected score mean and variance for the current subject.

expectedXi *Estimate variational parameter for the current subject.*

Description

Function calculates value of variational parameter using maximum likelihood.

Usage

```
expectedXi(theta, mu, mi, psi, Ci)
```

Arguments

`theta`	spline basis functions for the current subject.
`mu`	vector of spline coefficients for the population mean.
`mi`	vector of expected mean scores for the current subject.
`psi`	matrix of spline coefficients for the principal component basis functions.
`Ci`	expected covariance matrix of scores for the current subject.

Value

A vector of variational parameters for the current subject.

fpca_gauss *Functional principal components analysis via variational EM*

Description

Function used in the FPCA step for registering functional data, called by register_fpca when family = "gaussian". Parameters estimated based on probabilistic PCA framework originally introduced by Tipping and Bishop in 1999.

The number of functional principal components (FPCs) can either be specified directly (argument npc) or chosen based on the explained share of variance (npc_varExplained). In the latter case, the explained share of variance and accordingly the number of FPCs is estimated before the main estimation step by once running the FPCA with npc = 20 (and correspondingly Kt = 20). Doing so, we approximate the overall variance in the data Y with the variance represented by the FPC basis with 20 FPCs.

Usage

```
fpca_gauss(
  Y,
  npc = NULL,
  npc_varExplained = NULL,
  Kt = 8,
  maxiter = 20,
  t_min = NULL,
  t_max = NULL,
  print.iter = FALSE,
  row_obj = NULL,
  seed = 1988,
  periodic = FALSE,
  error_thresh = 1e-04,
  subsample = TRUE,
  verbose = 1,
  ...
)
```

Arguments

Y Dataframe. Should have variables id, value, index.

npc, npc_varExplained
The number of functional principal components (FPCs) has to be specified either directly as npc or based on their explained share of variance. In the latter case, npc_varExplained has to be set to a share between 0 and 1.

Kt Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8. If npc_varExplained is used, Kt is set to 20.

`maxiter` Maximum number of iterations to perform for EM algorithm. Default is 50.

`t_min` Minimum value to be evaluated on the time domain.

`t_max` Maximum value to be evaluated on the time domain.

`print.iter` Prints current error and iteration

`row_obj` If NULL, the function cleans the data and calculates row indices. Keep this NULL if you are using standalone `register` function.

`seed` Set seed for reproducibility. Defaults to 1988.

`periodic` If TRUE, uses periodic b-spline basis functions. Default is FALSE.

`error_thresh` Error threshold to end iterations. Defaults to 0.0001.

`subsample` if the number of rows of the data is greater than 10 million rows, the ‘id‘ values are subsampled to get the mean coefficients.

`verbose` Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

`...` Additional arguments passed to or from other functions

Value

An object of class `fpca` containing:

`fpca_type` Information that FPCA was performed with the ’variationEM’ approach, in contrast to registr::gfpca_twoStep.

`t_vec` Time vector over which the mean `mu` and the functional principal components `efunctions` were evaluated.

`knots` Cutpoints for B-spline basis used to rebuild `alpha`.

`efunctions` $D \times npc$ matrix of estimated FPC basis functions.

`evalues` Estimated variance of the FPC scores.

`evalues_sum` Approximation of the overall variance in `Y`, based on an initial run of the FPCA with `npc = 20`. Is `NULL` if `npc_varExplained` was not specified.

`npc` number of FPCs.

`scores` $I \times npc$ matrix of estimated FPC scores.

`alpha` Estimated population-level mean.

`mu` Estimated population-level mean. Same value as `alpha` but included for compatibility with `refund.shiny` package.

`subject_coefs` B-spline basis coefficients used to construct subject-specific means. For use in `registr()` function.

`Yhat` FPC approximation of subject-specific means.

`Y` The observed data.

`family` `gaussian`, for compatibility with `refund.shiny` package.

`sigma2` Estimated error variance

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>, Jeff Goldsmith <ajg2202@cumc.columbia.edu>, Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>

References

Tipping, M. E. and Bishop, C (1999). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society Series B,*, 592–598.

Examples

```
data(growth_incomplete)

# estimate 2 FPCs
fpca_obj = fpca_gauss(Y = growth_incomplete, npc = 2)
plot(fpca_obj)

# estimate npc adaptively, to explain 90% of the overall variation

fpca_obj2 = fpca_gauss(Y = growth_incomplete, npc_varExplained = 0.9)
plot(fpca_obj, plot_FPCs = 1:2)
```

---

fpca_gauss_argPreparation *Internal main preparation function for fpca_gauss*

Description

Internal main preparation function for fpca_gauss

Usage

```
fpca_gauss_argPreparation(
  Y,
  Kt,
  time,
  t_min,
  t_max,
  periodic,
  seed,
  subsample,
  verbose
)
```

Arguments

Y, time, t_min, t_max
: Internal objects created in fpca_gauss.

Kt
: Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8. If npc_varExplained is used, Kt is set to 20.

periodic
: If TRUE, uses periodic b-spline basis functions. Default is FALSE.

seed
: Set seed for reproducibility. Defaults to 1988.

subsample
: if the number of rows of the data is greater than 10 million rows, the ‘id‘ values are subsampled to get the mean coefficients.

verbose
: Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

Value

List with elements knots, Theta_phi, alpha_coefs.

fpca_gauss_optimization

Internal main optimization for fpca_gauss

Description

Main optimization function for fpca_gauss. If npc_varExplained is specified, the function simply returns a list with elements npc (chosen number of FPCs), evalues (estimated variances of the first 'npc' FPCs) and evalues_sum (sum of the estimated variances of the first 20 FPCs, as approximation of the overall variance).

Usage

```
fpca_gauss_optimization(
  npc,
  npc_varExplained = NULL,
  Kt,
  maxiter,
  print.iter,
  seed,
  periodic,
  error_thresh,
  verbose,
  Y,
  rows,
  I,
  knots,
  Theta_phi,
  alpha_coefs
)
```

Arguments

`npc` The number of functional principal components (FPCs) has to be specified either directly as `npc` or based on their explained share of variance. In the latter case, `npc_varExplained` has to be set to a share between 0 and 1.

`npc_varExplained` The number of functional principal components (FPCs) has to be specified either directly as `npc` or based on their explained share of variance. In the latter case, `npc_varExplained` has to be set to a share between 0 and 1.

`Kt` Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8. If `npc_varExplained` is used, `Kt` is set to 20.

`maxiter` Maximum number of iterations to perform for EM algorithm. Default is 50.

`print.iter` Prints current error and iteration

`seed` Set seed for reproducibility. Defaults to 1988.

`periodic` If TRUE, uses periodic b-spline basis functions. Default is FALSE.

`error_thresh` Error threshold to end iterations. Defaults to 0.0001.

`verbose` Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

`Y, rows, I, knots, Theta_phi, alpha_coefs` Internal objects created in `fpca_gauss`.

Value

list with elements `t_vec`, `Theta_phi_mean`, `alpha_coefs`, `efunctions`, `evalues`, `evalues_sum`, `scores`, `subject_coef`, `fittedVals`, `sigma2`. See documentation of `fpca_gauss` for details.

`gfpca_twoStep` *Generalized functional principal component analysis*

Description

Function for applying FPCA to different exponential family distributions. Used in the FPCA step for registering functional data, called by `register_fpca` when `fpca_type = "two-step"`.

The method implements the ‘two-step approach‘ of Gertheiss et al. (2017) and is based on the approach of Hall et al. (2008) to estimate functional principal components.

The number of functional principal components (FPCs) can either be specified directly (argument `npc`) or chosen based on the explained share of variance (`npc_criterion`). Using the latter, we approximate the overall variance in the data Y with the variance represented by the smoothed covariance surface estimated with cov_hall. Note that the Eigenvalue decomposition of this covariance surface sometimes leads to a long tail of subordinate FPCs with small eigenvalues. Such subordinate dimensions seem to often represent phase rather than amplitude variation, and can be cut off by specifying the second element of argument npc_criterion.

This function is an adaptation of the implementation of Jan Gertheiss for Gertheiss et al. (2017), with focus on higher (RAM) efficiency for large data settings.

Usage

```
gfpca_twoStep(
  Y,
  family = "gaussian",
  npc = NULL,
  npc_criterion = NULL,
  Kt = 8,
  t_min = NULL,
  t_max = NULL,
  row_obj = NULL,
  index_significantDigits = 4L,
  estimation_accuracy = "high",
  start_params = NULL,
  periodic = FALSE,
  verbose = 1,
  ...
)
```

Arguments

Y Dataframe. Should have values id, value, index.

family One of c("gaussian","binomial","gamma","poisson"). Poisson data are rounded before performing the GFPCA to ensure integer data, see Details section below. Defaults to "gaussian".

npc, npc_criterion
The number of functional principal components (FPCs) has to be specified either directly as npc or based on their explained share of variance. In the latter case, npc_criterion can either be set to (i) a share between 0 and 1, or (ii) a vector with two elements comprising the targeted explained share of variance and a cut-off scree plot criterion, both between 0 and 1. As an example for the latter, npc_criterion = c(0.9,0.02) tries to choose a number of FPCs that explains at least 90% of variation, but only includes FPCs that explain at least 2% of variation (even if this means 90% explained variation is not reached).

Kt Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8.

t_min Minimum value to be evaluated on the time domain.

t_max Maximum value to be evaluated on the time domain.

row_obj
: If NULL, the function cleans the data and calculates row indices. Keep this NULL if you are using standalone register function.

index_significantDigits
: Positive integer >= 2, stating the number of significant digits to which the index grid should be rounded. Coarsening the index grid is necessary since otherwise the covariance surface matrix explodes in size in the presence of too many unique index values (which is always the case after some registration step). Defaults to 4. Set to NULL to prevent rounding.

estimation_accuracy
: One of c("high","low"). When set to "low", the mixed model estimation step in lme4 is performed with lower accuracy, reducing computation time. Defaults to "high".

start_params
: Optional start values for gamm4. Not used if npc_criterion is specified.

periodic
: Only contained for full consistency with fpca_gauss and bfpca. If TRUE, returns the knots vector for periodic b-spline basis functions. Defaults to FALSE. This parameter does not change the results of the two-step GFPCA.

verbose
: Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

...
: Additional arguments passed to cov_hall.

Details

For family = "poisson" the values in Y are rounded before performing the GFPCA to ensure integer data. This is done to ensure reasonable computation times. Computation times tend to explode when estimating the underlying high-dimensional mixed model with continuous Poisson data based on the gamm4 package.

If negative eigenvalues are present, the respective eigenfunctions are dropped and not considered further.

Value

An object of class fpca containing:

fpca_type
: Information that FPCA was performed with the 'two-step' approach, in contrast to registr::fpca_gauss or registr::bfpca.

t_vec
: Time vector over which the mean mu was evaluated. The resolution is can be specified by setting index_significantDigits.

knots
: Cutpoints for B-spline basis used to rebuild alpha.

efunctions
: $D \times npc$ matrix of estimated FPC basis functions.

evalues
: Estimated variance of the FPC scores.

evalues_sum
: Sum of all (nonnegative) eigenvalues of the smoothed covariance surface estimated with cov_hall. Can be used as an approximation for the total variance present in Y to compute the shares of explained variance of the FPC scores.

npc
: number of FPCs.

scores $I \times npc$ matrix of estimated FPC scores.

alpha Estimated population-level mean.

mu Estimated population-level mean. Same value as alpha but included for compatibility with refund.shiny package.

subject_coefs Always NA but included for full consistency with fpca_gauss and bfpca.

Yhat FPC approximation of subject-specific means, before applying the response function.

Y The observed data.

family binomial, for compatibility with refund.shiny package.

gamm4_theta Estimated parameters of the mixed model.

Author(s)

Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>, based on work of Jan Gertheiss

References

Gertheiss, J., Goldsmith, J., & Staicu, A. M. (2017). A note on modeling sparse exponential-family functional response curves. *Computational statistics & data analysis*, 105, 46–52.

Hall, P., Müller, H. G., & Yao, F. (2008). Modelling sparse generalized longitudinal observations with latent Gaussian processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4), 703–723.

Examples

```
data(growth_incomplete)

# estimate 2 FPCs
fpca_obj = gfpca_twoStep(Y = growth_incomplete, npc = 2, family = "gaussian")
plot(fpca_obj)

# estimate npc adaptively, to explain 90% of the overall variation
fpca_obj2 = gfpca_twoStep(Y = growth_incomplete, npc_criterion = 0.9, family = "gaussian")
plot(fpca_obj2, plot_FPCs = 1:2)
```

grid_subj_create *Generate subject-specific grid (t_star)*

Description

This function creates subject-specific time grid

Usage

```
grid_subj_create(coefs, D)
```

Arguments

`coefs`	Spline basis coefficients for reconstructing the subject-specific grid.
`D`	Number of grid points per subject.

Value

A numeric vector.

`growth_incomplete` *Berkeley Growth Study data with simulated incompleteness*

Description

This dataset from the Berkeley Growth Study comprises the height development of 39 boys and 54 girls between ages 1 and 18. It is based on the dataset `fda::growth` and focuses not on the observed heights, but on the first derivatives of the curves. Before taking the first derivative, the curves were slightly smoothed.

To showcase the functionality of the `registr` package regarding the analysis of incomplete curves, the growth curves were artificially made incomplete. For each child, leading incompleteness was simulated by drawing a random initial age in the first quarter of the domain. Also, trailing incompleteness was simulated by drawing a random cut-off age in the second half of the domain.

Usage

```
data(growth_incomplete)
```

Format

A dataframe made up of

id A unique subject identifier;

index Observed age of the child's height;

value First derivative of the height development in the given age.

References

Ramsay, J. O., and Silverman, B. W. (2006), *Functional Data Analysis, 2nd ed.*, Springer, New York.

Tuddenham, R. D., and Snyder, M. M. (1954). Physical growth of California boys and girls from birth to age 18. *University of California Publications in Child Development*, 1, 183-364.

initial_params *Create initial parameters for (inverse) warping functions*

Description

Dependent on the specific type of warping functions, this function creates a vector of initial parameters. For "nonparametric" warpings that are based on a given spline basis matrix, the initial parameters are defined s.t. the resulting (inverse) warping function equals a diagonal line. For "piecewise_linear2" warpings a fixed parameter vector is returned.

Usage

```
initial_params(warping = "nonparametric", K, t_vec)
```

Arguments

warping	If nonparametric (default), inverse warping functions are estimated nonparametrically. If piecewise_linear2 they follow a piecewise linear function with 2 knots.
K	Spline basis matrix defined over the interval c(t_min, t_max).
t_vec	Vector of the observed and potentially irregular time grid.

lambdaF *Apply lambda transformation of variational parameter.*

Description

Simple function for use within other C++ functions.

Usage

```
lambdaF(x)
```

Arguments

x	The value to which you apply the function

Value

A numeric value that has been transformed.

loss_h *Loss function for registration step optimization*

Description

Loss function for registration step optimization

Usage

```
loss_h(
  Y,
  Theta_h,
  mean_coefs,
  knots,
  beta.inner,
  family,
  t_min,
  t_max,
  t_min_curve,
  t_max_curve,
  incompleteness = NULL,
  lambda_inc = NULL,
  periodic = FALSE,
  Kt = 8,
  warping = "nonparametric",
  priors = FALSE,
  prior_sd = NULL
)
```

Arguments

`Y`	vector of observed points.
`Theta_h`	B-spline basis for inverse warping functions.
`mean_coefs`	spline coefficient vector for mean curve.
`knots`	knot locations for B-spline basis used to estimate mean and FPC basis function.
`beta.inner`	spline coefficient vector to be estimated for warping function h.
`family`	One of `c("gaussian","binomial","gamma","poisson")`. For internal purposes, can also be set to `"gamma-varEM"` and `"poisson-varEM"` if the preceding FPCA step in `register_fpca` was performed with `fpca_type = "variationalEM"` which uses Gaussian family.
`t_min, t_max`	minimum and maximum value to be evaluated on the time domain.
`t_min_curve, t_max_curve`	minimum and maximum value of the observed time domain of the (potentially incomplete) curve.

`incompleteness` Optional specification of incompleteness structure. One of `c("leading","trailing","full")`, specifying that incompleteness is present only in the initial measurements, only in the trailing measurements, or in both, respectively. For details see the accompanying vignette. Defaults to NULL, i.e. no incompleteness structure. Can only be set when `warping = "nonparametric"`.

`lambda_inc` Penalization parameter to control the amount of overall dilation of the domain. The higher this lambda, the more the registered domains are forced to have the same length as the observed domains. Only used if `incompleteness` is not NULL.

`periodic` If `TRUE` uses periodic b-spline basis functions. Default is `FALSE`.

`Kt` Number of B-spline basis functions used to estimate mean functions. Default is 8.

`warping` If `nonparametric` (default), inverse warping functions are estimated nonparametrically. If `piecewise_linear2` they follow a piecewise linear function with 2 knots.

`priors` For `warping = "piecewise_linear2"` only. Logical indicator of whether to add Normal priors to pull the knots toward the identity line.

`prior_sd` For `warping = "piecewise_linear2"` with `priors = TRUE` only. User-specified standard deviation for the Normal priors (single value applied to all 4 knot priors).

Value

The scalar value taken by the loss function.

Author(s)

Julia Wrobel <`julia.wrobel@cuanschutz.edu`>, Erin McDonnell <`eim2117@cumc.columbia.edu`>, Alexander Bauer <`alexander.bauer@stat.uni-muenchen.de`>

`loss_h_gradient` *Gradient of loss function for registration step*

Description

Gradient of loss function for registration step

Usage

```
loss_h_gradient(
  Y,
  Theta_h,
  mean_coefs,
  knots,
  beta.inner,
  family = "gaussian",
  incompleteness = NULL,
  lambda_inc = NULL,
  t_min,
  t_max,
  t_min_curve,
  t_max_curve,
  Kt = 8,
  periodic = FALSE,
  warping = "nonparametric"
)
```

Arguments

`Y` vector of observed points.

`Theta_h` B-spline basis for inverse warping functions.

`mean_coefs` spline coefficient vector for mean curve.

`knots` knot locations for B-spline basis used to estimate mean and FPC basis function.

`beta.inner` spline coefficient vector to be estimated for warping function h.

`family` One of `c("gaussian","binomial")`. Defaults to `"gaussian"`.

`incompleteness` Optional specification of incompleteness structure. One of `c("leading","trailing","full")`, specifying that incompleteness is present only in the initial measurements, only in the trailing measurements, or in both, respectively. For details see the accompanying vignette. Defaults to NULL, i.e. no incompleteness structure. Can only be set when `warping = "nonparametric"`.

`lambda_inc` Penalization parameter to control the amount of overall dilation of the domain. The higher this lambda, the more the registered domains are forced to have the same length as the observed domains. Only used if `incompleteness` is not NULL.

`t_min` minimum and maximum value to be evaluated on the time domain.

`t_max` minimum and maximum value to be evaluated on the time domain.

`t_min_curve` minimum and maximum value of the observed time domain of the (potentially incomplete) curve.

`t_max_curve` minimum and maximum value of the observed time domain of the (potentially incomplete) curve.

`Kt` Number of B-spline basis functions used to estimate mean functions. Default is 8.

`periodic` If `TRUE`, uses periodic b-spline basis functions. Default is `FALSE`. `loss_h_gradient()` is currently only available for `periodic = FALSE`.

`warping` If `nonparametric` (default), inverse warping functions are estimated nonparametrically. If `piecewise_linear2` they follow a piecewise linear function with 2 knots. `loss_h_gradient()` is currently only available for `warping = "nonparametric"`.

Value

A numeric vector of spline coefficients for the gradient of the loss function.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>, Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>

---

mean_curve *Simulate mean curve*

Description

This function generates mean for simulated accelerometer data.

Usage

```
mean_curve(grid, period = 2 * pi, spline_based = FALSE)
```

Arguments

`grid`	Grid of x values over which to evaluate the function.
`period`	Controls the period of the mean curve
`spline_based`	If FALSE curve is constructed using sine and cosine functions, if TRUE, curve is constructed using B-spline basis.

Value

A numeric vector.

---

mean_sim *Simulate mean*

Description

This function generates mean for simulated functional data.

Usage

```
mean_sim(grid)
```

Arguments

`grid`	Grid of x values over which to evaluate the function.

`nhanes` *NHANES activity data*

Description

Subset of 24 hours of activity data for 50 subjects from 2003-2004 National Health and Nutrition Examination Survey (NHANES). Each subject is observed over 24 hours on a Sunday and wore the activity collection device for a minimum of 10 hours. Activity is measured each minute over 24 hours.

Usage

```
data(nhanes)
```

Format

A dataframe made up of

id A unique subject identifier;

age Age of survey participant;

gender Gender of survey participant;

index Observed time of activity measurement. Integers from 1 to 1440, indicating minutes from midnight to midnight;

value Binary value of zero or one indicating inactivity or activity;

raw_activity Raw activity count.

`piecewise_linear2_hinv` *Create two-parameter piecewise linear (inverse) warping functions*

Description

This function uses a 2-knot piecewise linear model to calculate inverse warping functions for registration. The parameters `knot1_x` and `knot1_y` control the x and y locations of the first knot, and the parameters `knot1_x` and `knot1_y` control the x and y locations of the second knot. The designation (inverse) is intended to communicate that these functions take data from the unregistered space to the registered space, consistent with functional data literature on registration.

Usage

```
piecewise_linear2_hinv(grid, knot_locations = c(0.25, 0.3, 0.75, 0.9))
```

Arguments

grid grid of values over which to evaluate the function.

knot_locations controls the x and y locations of the two knots.

Author(s)

Erin McDonnell <eim2117@cumc.columbia.edu>

plot.fpca *Plot the results of a functional PCA*

Description

S3 plot method for class fpca. Plot FPCA results by visualizing the variation of the individual FPCs around the global mean. based on an object created with function fpca_gauss, bfpca or gfpca_twoStep.

The shares of explained variance are included in the plot titles if x contains an element evalues_sum.

Usage

```
## S3 method for class 'fpca'
plot(
  x,
  plot_FPCs = 1:x$npc,
  sd_factor = 2,
  response_function = NULL,
  add_symbols = TRUE,
  subtitle = TRUE,
  xlim = NULL,
  ylim = NULL,
  xlab = "t [registered]",
  ylab = "y",
  ...
)
```

Arguments

x Object of class "fpca".

plot_FPCs Optional index vector of the FPCs to be plotted. Defaults to all FPCs contained in x.

sd_factor Numeric factor with which the standard deviations of each FPC's scores are multiplied to display its variation in the plots. Defaults to 2.

response_function
Optional response function to be applied before plotting the curves. Defaults to NULL, i.e. the identity function if x$family is one of c("gaussian","binomial") or exp() if x$family is one of c("gamma","poisson").

`add_symbols` Indicator if '+' and '-' symbols should be added to the plot to highlight the direction of the displayed FPCs. Defaults to TRUE.

`subtitle` If TRUE (default) the parameter `sd_factor` is displayed in the plot subtitle.

`xlim, ylim` Optional numeric vectors with limits for the x and y axis.

`xlab, ylab` Optional titles for the x and y axis.

`...` Additional arguments passed to `theme`.

Value

@return If multiple FPCs are plotted, returns a grid of `ggplot` plots, created with `cowplot::plot_grid`. If only one FPC is plotted, returns a single `ggplot` plot.

Author(s)

Alexander Bauer <`alexander.bauer@stat.uni-muenchen.de`>

Examples

```
data(growth_incomplete)

fpca_obj = fpca_gauss(Y = growth_incomplete, npc = 2)
if (requireNamespace("ggplot2", quietly = TRUE) &&
requireNamespace("cowplot", quietly = TRUE)) {
library(ggplot2)
plot(fpca_obj)
}
```

`psi1_sim` *Simulate PC1*

Description

This function generates the first principal component for simulated functional data.

Usage

```
psi1_sim(grid)
```

Arguments

`grid` Grid of x values over which to evaluate the function.

psi2_sim *Simulate PC2*

Description

This function generates the second principal component for simulated functional data.

Usage

```
psi2_sim(grid)
```

Arguments

`grid`	Grid of x values over which to evaluate the function.

register_fpca *Register curves using constrained optimization and GFPCA*

Description

Function combines constrained optimization and GFPCA to estimate warping functions for exponential family curves. See argument `family` for which families are supported. Warping functions are calculated by the function `registr`. The GFPCA step can be performed either using the variational EM-based GFPCA approaches of Wrobel et al. (2019) (`fpca_type = "variationalEM"`, default) or the mixed model-based two-step approach of Gertheiss et al. (2017) (`fpca_type = "two-step"`).

Warping functions by default are forced to start and end on the diagonal to be domain-preserving. This behavior can be changed by setting `incompleteness` to some other value than NULL and a reasonable `lambda_inc` value. For further details see the accompanying vignette.

The number of functional principal components (FPCs) can either be specified directly (argument `npc`) or chosen based on the explained share of variance in each iteration (argument `npc_criterion`).

By specifying `cores > 1` the registration call can be parallelized.

Usage

```
register_fpca(
  Y,
  Kt = 8,
  Kh = 4,
  family = "gaussian",
  incompleteness = NULL,
  lambda_inc = NULL,
  Y_template = NULL,
  max_iterations = 10,
  npc = NULL,
  npc_criterion = NULL,
  fpca_type = "variationalEM",
  fpca_maxiter = 50,
  fpca_seed = 1988,
  fpca_error_thresh = 1e-04,
  fpca_index_significantDigits = 4L,
  cores = 1L,
  verbose = 1,
  ...
)
```

Arguments

`Y` Dataframe. Should have values id, value, index.

`Kt` Number of B-spline basis functions used to estimate mean functions and functional principal components. Default is 8. If `fpca_type = "variationalEM"` and `npc_criterion` is used, `Kt` is set to 20.

`Kh` Number of B-spline basis functions used to estimate warping functions *h*. Default is 4.

`family` One of `c("gaussian","binomial","gamma","poisson")`. Families `"gamma"` and `"poisson"` are only supported by `fpca_type = "two-step"`. Defaults to `"gaussian"`.

`incompleteness` Optional specification of incompleteness structure. One of `c("leading","trailing","full")`, specifying that incompleteness is present only in the initial measurements, only in the trailing measurements, or in both, respectively. For details see the accompanying vignette. Defaults to NULL, i.e. no incompleteness structure. Can only be set when `warping = "nonparametric"`.

`lambda_inc` Penalization parameter to control the amount of overall dilation of the domain. The higher this lambda, the more the registered domains are forced to have the same length as the observed domains. Only used if `incompleteness` is not NULL.

`Y_template` Optional dataframe with the same structure as `Y`. Only used for the initial registration step. If NULL, curves are registered to the overall mean of all curves in `Y` as template function. If specified, the template function is taken as the mean of all curves in `Y_template`. Defaults to NULL.

`max_iterations` Number of iterations for overall algorithm. Defaults to 10.

`npc, npc_criterion`

The number of functional principal components (FPCs) has to be specified either directly as `npc` or based on their explained share of variance. In the latter case, `npc_criterion` has to be set to a number between 0 and 1. For `fpca_type = "two-step"`, it is also possible to cut off potential tails of subordinate FPCs (see `gfpca_twoStep` for details).

`fpca_type` One of `c("variationalEM","two-step")`. Defaults to `"variationalEM"`.

`fpca_maxiter` Only used if `fpca_type = "variationalEM"`. Number to pass to the `maxiter` argument of ‘bfpca()‘ or ‘fpca_gauss()‘. Defaults to 50.

`fpca_seed` Only used if `fpca_type = "variationalEM"`. Number to pass to the `seed` argument of ‘bfpca()‘ or ‘fpca_gauss()‘. Defaults to 1988.

`fpca_error_thresh`
Only used if `fpca_type = "variationalEM"`. Number to pass to the `error_thresh` argument of ‘bfpca()‘ or ‘fpca_gauss()‘. Defaults to 0.0001.

`fpca_index_significantDigits`
Only used if `fpca_type = "two-step"`. Positive integer `>= 2`, stating the number of significant digits to which the index grid should be rounded in the GFPCA step. Coarsening the index grid is necessary since otherwise the covariance surface matrix explodes in size in the presence of too many unique index values (which is the case after some registration step). Defaults to 4. Set to `NULL` to prevent rounding.

`cores` Number of cores to be used. If `cores > 1`, the registration call is parallelized by using `parallel::mclapply` (for Unix-based systems) or `parallel::parLapply` (for Windows). Defaults to 1, no parallelized call.

`verbose` Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

`...` Additional arguments passed to registr and to the gfpca functions (if `fpca_type = "variationalEM"`).

Details

Requires input data `Y` to be a dataframe in long format with variables `id`, `index`, and `value` to indicate subject IDs, observation times on the domain, and observations, respectively.

One joint iteration consists of a GFPCA step and a registration step. As preprocessing, one initial registration step is performed. The template function for this registration step is defined by argument `Y_template`. After convergence or `max_iterations` is reached, one final GFPCA step is performed.

Value

An object of class `registration` containing:

`Y` The observed data plus variables `t_star` and `t_hat` which are the unregistered grid and registered grid, respectively.

`fpca_obj` List of items from FPCA step.

`family` Used exponential family.

`index_warped` List of the (warped) index values for each iteration. Has `'convergence$iterations + 2'` elements since the first two elements contain the original (observed) index and the warped index values from the preprocessing registration step (see Details), respectively.

`hinv_innerKnots`
List of inner knots for setting up the spline bases for the inverse warping functions. Only contains `NULL` values for `Kh <= 4`.

`hinv_beta` Matrix of B-spline basis coefficients used to construct the subject-specific inverse warping functions. From the last performed registration step. For details see `?registr`.

`convergence` List with information on the convergence of the joint approach. Containing the following elements:

converged
Indicator if the joint algorithm converged or if not (i.e., `max_iterations` was reached)

iterations
Number of joint iterations that were performed.

delta_index
Vector of mean squared differences between the (warped) index values (scaled to [0,1] based on the size of the observed domain) in the current and the previous iteration. Convergence is reached if this measure drops below 0.0001.

registration_loss
Vector of the loss in each iteration of the algorithm. Calculated in the registration step using the exponential family likelihood with natural parameter from the FPCA step. Has `'iterations + 1'` elements since the first element contains the loss of the preprocessing registration step (see Details).

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu> Jeff Goldsmith <ajg2202@cumc.columbia.edu>, Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>

Examples

```
### complete binomial curves
Y = simulate_unregistered_curves(I = 20, D = 200)

# estimation based on Wrobel et al. (2019)
reg = register_fpca(Y, npc = 2, family = "binomial",
                    fpca_type = "variationalEM", max_iterations = 5)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)

  ggplot(reg$Y, aes(x = tstar, y = t_hat, group = id)) +
    geom_line(alpha = 0.2) + ggtitle("Estimated warping functions")

  plot(reg$fpca_obj, response_function = function(x) { 1 / (1 + exp(-x)) })
}
```

```
# estimation based on Gertheiss et al. (2017)
reg2 = register_fpca(Y, npc = 2, family = "binomial",
                     fpca_type = "two-step", max_iterations = 5,
                     fpca_index_significantDigits = 4)

# example using accelerometer data from nhanes 2003-2004 study
data(nhanes)
nhanes_short = nhanes[nhanes$id %in% unique(nhanes$id)[1:5],]
reg_nhanes  = register_fpca(nhanes_short, npc = 2, family = "binomial", max_iterations = 5)


### incomplete Gaussian curves
data(growth_incomplete)

# Force the warping functions to start and end on the diagonal
reg2a = register_fpca(growth_incomplete, npc = 2, family = "gaussian",
                      incompleteness = NULL, max_iterations = 5)
if (requireNamespace("ggplot2", quietly = TRUE)) {

  ggplot(reg2a$Y, aes(x = tstar, y = t_hat, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Estimated warping functions")
  ggplot(reg2a$Y, aes(x = t_hat, y = value, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Registered curves")
}
# Allow the warping functions to not start / end on the diagonal.
# The higher lambda_inc, the more the starting points and endpoints are forced
# towards the diagonal.
reg2b = register_fpca(growth_incomplete, npc = 2, family = "gaussian",
                      incompleteness = "full", lambda_inc = 0.1,
                      max_iterations = 5)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot(reg2b$Y, aes(x = tstar, y = t_hat, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Estimated warping functions")
  ggplot(reg2b$Y, aes(x = t_hat, y = value, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Registered curves")
}

### complete Gamma curves
Y             = simulate_unregistered_curves(I = 20, D = 100)
Y$value       = exp(Y$latent_mean)
registr_gamma = register_fpca(Y, npc = 2, family = "gamma", fpca_type = "two-step",
                              gradient = FALSE, max_iterations = 3)
```

registr *Register Exponential Family Functional Data*

Description

Software for registering functional data from the exponential family of distributions.

Function used in the registration step of an FPCA-based approach for registering exponential-family, potentially incomplete functional data, called by `register_fpca`. This method uses constrained optimization to estimate spline coefficients for warping functions, where the objective function for optimization comes from maximizing the EF likelihood subject to monotonicity constraints on the warping functions. You have to either specify `obj`, which is a fpca object from an earlier step, or `Y`, a dataframe in long format with variables `id`, `index`, and `value` to indicate subject IDs, times, and observations, respectively.

Warping functions by default are forced to start and end on the diagonal to be domain-preserving. This behavior can be changed by setting `incompleteness` to some other value than NULL and a reasonable `lambda_inc` value. For further details see the accompanying vignette.

By specifying `cores` > `1` the registration call can be parallelized.

Usage

```
registr(
  obj = NULL,
  Y = NULL,
  Kt = 8,
  Kh = 4,
  family = "gaussian",
  gradient = TRUE,
  incompleteness = NULL,
  lambda_inc = NULL,
  Y_template = NULL,
  beta = NULL,
  t_min = NULL,
  t_max = NULL,
  row_obj = NULL,
  periodic = FALSE,
  warping = "nonparametric",
  gamma_scales = NULL,
  cores = 1L,
  subsample = TRUE,
  verbose = 1,
  ...
)
```

Arguments

`obj` Current estimate of FPC object. Can be NULL only if Y argument is selected.

`Y` Dataframe. Should have values id, value, index.

`Kt` Number of B-spline basis functions used to estimate mean functions. Default is 8.

Kh Number of B-spline basis functions used to estimate warping functions *h*. Default is 4.

family One of c("gaussian","binomial","gamma","poisson"). Defaults to "gaussian".

gradient If TRUE, uses analytic gradient to calculate derivative. If FALSE, calculates gradient numerically. Not available for families "gamma","poisson".

incompleteness Optional specification of incompleteness structure. One of c("leading","trailing","full"), specifying that incompleteness is present only in the initial measurements, only in the trailing measurements, or in both, respectively. For details see the accompanying vignette. Defaults to NULL, i.e. no incompleteness structure. Can only be set when warping = "nonparametric".

lambda_inc Penalization parameter to control the amount of overall dilation of the domain. The higher this lambda, the more the registered domains are forced to have the same length as the observed domains. Only used if incompleteness is not NULL.

Y_template Optional dataframe with the same structure as Y. Only used if obj is NULL. If Y_template is NULL, curves are registered to the overall mean of all curves in Y as template function. If Y_template is specified, the template function is taken as the mean of all curves in Y_template. Default is NULL.

beta Current estimates for beta for each subject. Default is NULL.

t_min Minimum value to be evaluated on the time domain. if 'NULL', taken to be minimum observed value.

t_max Maximum value to be evaluated on the time domain. if 'NULL', taken to be maximum observed value.

row_obj If NULL, the function cleans the data and calculates row indices. Keep this NULL if you are using standalone registr function.

periodic If TRUE, uses periodic b-spline basis functions. Default is FALSE.

warping If nonparametric (default), inverse warping functions are estimated nonparametrically. If piecewise_linear2 they follow a piecewise linear function with 2 knots.

gamma_scales Only used for family = "gamma". Vector with one entry for each subject, containing the current estimate for the scale parameter of its gamma distribution. Default is NULL, which sets the starting value for the scale parameter to 1.5.

cores Number of cores to be used. If cores > 1, the registration call is parallelized by using parallel::mclapply (for Unix-based systems) or parallel::parLapply (for Windows). Defaults to 1, no parallelized call.

subsample if the number of rows of the data is greater than 10 million rows, the 'id' values are subsampled to get the mean coefficients.

verbose Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

... additional arguments passed to or from other functions

Details

The template function for the registration is defined by argument obj or Y_template, depending on if obj is NULL or not, respectively.

Value

An list containing:

Y The observed data. The variables index and index_scaled contain the new estimated time domain.

loss Value of the loss function after registraton.

hinv_innerKnots

List of inner knots for setting up the spline bases for the inverse warping functions. Only contains NULL values for Kh <= 4.

hinv_beta Matrix of B-spline basis coefficients used to construct subject-specific inverse warping functions. See examples on how to reconstruct a warping function based on hinv_innerKnots and hinv_beta.

Author(s)

Julia Wrobel

Julia Wrobel <julia.wrobel@cuanschutz.edu>, Erin McDonnell <eim2117@cumc.columbia.edu>, Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>

Examples

```
### complete binomial curves
Y = simulate_unregistered_curves()
register_step = registr(obj = NULL, Y = Y, Kt = 6, Kh = 4, family = "binomial",
                        gradient = TRUE)

### incomplete Gaussian curves
data(growth_incomplete)

# Force the warping functions to start and end on the diagonal to preserve the domain
register_step2a = registr(obj = NULL, Y = growth_incomplete, Kt = 6, Kh = 4,
                          family = "gaussian", gradient = TRUE,
                          incompleteness = NULL)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)

  ggplot(register_step2a$Y, aes(x = tstar, y = index, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Estimated warping functions")
  ggplot(register_step2a$Y, aes(x = index, y = value, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Registered curves")
}

# Example for how to recreate an estimated inverse warping function given
# the output of registr(). Focus on id "boy01".
id         = "boy01"
index_obsRange_i = range(growth_incomplete$index[growth_incomplete$id == id])
index      = seq(min(index_obsRange_i), max(index_obsRange_i), length.out = 100)
# (note that 'index' must contain both the observed min and max in index_obsRange_i)
```

```
Theta_h_i  = splines::bs(index, knots = register_step2a$hinv_innerKnots[[id]], intercept = TRUE)
index_reg  = as.vector(Theta_h_i %*% register_step2a$hinv_beta[,id])
warp_dat_i = data.frame(index_observed   = index,
                        index_registered = index_reg)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot(warp_dat_i, aes(x = index_observed, y = index_registered)) + geom_line() +
    ggtitle("Extracted warping function for id 'boy01'")
}

# Allow the warping functions to not start / end on the diagonal.
# The higher lambda_inc, the more the starting points and endpoints are
# forced towards the diagonal.
register_step2b = registr(obj = NULL, Y = growth_incomplete, Kt = 6, Kh = 4,
                          family = "gaussian", gradient = TRUE,
                          incompleteness = "full", lambda_inc = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot(register_step2b$Y, aes(x = tstar, y = index, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Estimated warping functions")
  ggplot(register_step2b$Y, aes(x = index, y = value, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Registered curves")
}

# Define the template function only over a subset of the curves
# (even though not very reasonable in this example)
template_ids    = c("girl12","girl13","girl14")
Y_template      = growth_incomplete[growth_incomplete$id %in% template_ids,]
register_step2c = registr(obj = NULL, Y = growth_incomplete, Kt = 6, Kh = 4,
                          family = "gaussian", gradient = TRUE,
                          Y_template = Y_template,
                          incompleteness = "full", lambda_inc = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot(register_step2c$Y, aes(x = index, y = value, group = id)) +
    geom_line(alpha = 0.2) +
    ggtitle("Registered curves")
}
```

`registr_oneCurve` *Internal function to register one curve*

Description

This internal function is only to be used from within `registr`. It performs the main optimization step with `constrOptim` for the registration of one curve.

Usage

```
registr_oneCurve(
  obj = NULL,
  Y = NULL,
  Kt = 8,
  Kh = 4,
  family = "gaussian",
  gradient = TRUE,
  incompleteness = NULL,
  lambda_inc = NULL,
  beta = NULL,
  t_min = NULL,
  t_max = NULL,
  periodic = FALSE,
  warping = "nonparametric",
  gamma_scales = NULL,
  global_knots = NULL,
  mean_coefs = NULL,
  ...,
  verbose = 1,
  just_return_list = FALSE
)
```

Arguments

`obj` Current estimate of FPC object. Can be NULL only if Y argument is selected.

`Y` Dataframe. Should have values id, value, index.

`Kt` Number of B-spline basis functions used to estimate mean functions. Default is 8.

`Kh` Number of B-spline basis functions used to estimate warping functions *h*. Default is 4.

`family` One of `c("gaussian","binomial","gamma","poisson")`. Defaults to `"gaussian"`.

`gradient` If `TRUE`, uses analytic gradient to calculate derivative. If `FALSE`, calculates gradient numerically. Not available for families `"gamma","poisson"`.

`incompleteness` Optional specification of incompleteness structure. One of `c("leading","trailing","full")`, specifying that incompleteness is present only in the initial measurements, only in the trailing measurements, or in both, respectively. For details see the accompanying vignette. Defaults to NULL, i.e. no incompleteness structure. Can only be set when `warping = "nonparametric"`.

`lambda_inc` Penalization parameter to control the amount of overall dilation of the domain. The higher this lambda, the more the registered domains are forced to have the same length as the observed domains. Only used if `incompleteness` is not NULL.

`beta` Current estimates for beta for each subject. Default is NULL.

`t_min` Minimum value to be evaluated on the time domain. if ‘NULL‘, taken to be minimum observed value.

`t_max` Maximum value to be evaluated on the time domain. if ‘NULL‘, taken to be maximum observed value.

`periodic` If `TRUE`, uses periodic b-spline basis functions. Default is `FALSE`.

`warping` If `nonparametric` (default), inverse warping functions are estimated nonparametrically. If `piecewise_linear2` they follow a piecewise linear function with 2 knots.

`gamma_scales` Only used for `family = "gamma"`. Vector with one entry for each subject, containing the current estimate for the scale parameter of its gamma distribution. Default is NULL, which sets the starting value for the scale parameter to 1.5.

`global_knots` knots for the basis/splines, passed to [pbs::pbs()] or [stats::bs()]

`mean_coefs` Mean coefficients for the mean of all curves or GFPCA based. May extract from ‘obj‘ object

`...` additional arguments passed to or from other functions

`verbose` Can be set to integers between 0 and 4 to control the level of detail of the printed diagnostic messages. Higher numbers lead to more detailed messages. Defaults to 1.

`just_return_list` Do not use. For developers only

Value

An list containing:

`hinv_innerKnots` Inner knots for setting up the spline basis for the inverse warping function.

`hinv_beta` Estimated B-spline basis coefficients used to construct subject-specific inverse warping functions.

`t_hat` Vector of registered time domain.

`loss` Loss of the optimal solution.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>, Erin McDonnell <eim2117@cumc.columbia.edu>, Alexander Bauer <alexander.bauer@stat.uni-muenchen.de>

`simulate_functional_data` *Simulate functional data*

Description

This function simulates functional data. The data it outputs is generated from a mean function and two orthogonal principal component basis functions. The mean and principal components are based on sine and cosine functions. Subject-specific scores for each PC are drawn from normal distributions with standard deviation lambda1 and lambda2.

Usage

```
simulate_functional_data(
  lambda1 = 2,
  lambda2 = 1,
  I = 50,
  D = 100,
  seed = 1988,
  vary_D = FALSE
)
```

Arguments

`lambda1` Standard deviation for PC1 scores.

`lambda2` Standard deviation for PC2 scores.

`I` Number of subjects. Defaults is 50.

`D` Number of grid points per subject. Default is 100.

`seed` Seed for reproducibility. Default is 1988.

`vary_D` Indicates if grid length vary by subject. If FALSE all subjects have grid length D.

Value

A list containing:

`Y` Simulated dataframe with variables id, value, index, and latent_mean.

`psi1` True values for first principal component.

`psi2` True values for second principal component.

`alpha` True values for population-level mean.

A list containing:

`Y` A dataframe of simulated data.

`psi1` The first simulated eigenfunction.

`psi2` The second simulated eigenfunction.

`alpha` The population mean.

Author(s)

Julia Wrobel <`julia.wrobel@cuanschutz.edu`>

simulate_unregistered_curves

Simulate unregistered curves

Description

This function simulates unregistered curves, providing the time values for both the unregistered curves (t_star) and the registered curves (t). Curves all have one peak, the location of which is shifted on the unregistered domain, meant to mimic accelerometer data.

Usage

```
simulate_unregistered_curves(
  I = 50,
  D = 100,
  lambda = 15,
  seed = 1988,
  period = 2 * pi,
  spline_based = FALSE,
  phase_variation = TRUE
)
```

Arguments

`I`	Number of subjects. Defaults is 50.
`D`	Number of grid points per subject. Default is 100.
`lambda`	Standard deviation for subject-specific amplitudes.
`seed`	Seed for reproducibility. Default is 1988.
`period`	Controls the period of the mean curve
`spline_based`	If FALSE curve is constructed using sine and cosine functions, if TRUE, curve is constructed using B-spline basis.
`phase_variation`	If TRUE, creates phase variation (registered curves are observed on uneven grid). If FALSE, no phase variation.

Value

A simulated dataframe with variables id, value, index, latent_mean, and t. Index is the domain on which curves are unregistered and t is the domain on which curves are registered.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>, Jeff Goldsmith <ajg2202@cumc.columbia.edu>

squareTheta *Calculate quadratic form of spline basis functions for the current subject.*

Description

Calculations quadratic form of theta with diagonalized variational parameter in the center.

Usage

```
squareTheta(xi, theta)
```

Arguments

`xi`	vector of variational parameters for the current subject.
`theta`	spline basis functions for the current subject.

Value

A matrix of the quadratic form of theta for the current subject.

Index