Package ‘registr’

March 16, 2020

Title Curve Registration for Exponential Family Functional Data

Version 1.0.0

Description A method for registering curves (functional data) that are generated from exponential family distributions. This
implements the algorithms described in 'Wrobel et al. (2019)' <doi:10.1111/biom.12963>. Curve registration is an active area of
research in functional data analysis, and 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 ggplot2, tidyr, magrittr, dplyr, gridExtra, Rcpp (>= 0.11.5)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, knitr, rmarkdown

SystemRequirements GNU make

VignetteBuilder knitr

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2020-03-16 11:40:02 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.

Usage

```
bfpca(
  Y,
  npc = 1,
  Kt = 8,
  maxiter = 50,
  t_min = NULL,
  t_max = NULL,
  print.iter = FALSE,
  row_obj = NULL,
  seed = 1988,
  ...
)
```

Arguments

Y	Dataframe. Should have variables id, value, index.
npc	Default is 1. Number of principal components to calculate.
Kt	Number of B-spline basis functions used to estimate mean functions. Default is 8.
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. Default is 1988.
...	Additional arguments passed to or from other functions

Value

An object of class `fpca` containing:

- `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.
- `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` — `binomial`, for compatibility with `refund.shiny` package.
- `error` — vector containing error for each iteration of the algorithm.

Author(s)

Julia Wrobel <jw3134@cumc.columbia.edu>, Jeff Goldsmith <ajg2202@cumc.columbia.edu>

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
bfpca_object = bfpca(Y, npc = 2, print.iter = TRUE)
```

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 `<jw3134@cumc.columbia.edu>`

`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, parametric_warps = FALSE)
```

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.
`parametric_warps`	If FALSE (default), inverse warping functions are estimated nonparametrically. If 'beta_cdf', they are assumed to have the form of a Beta(a,b) CDF. If 'piecewise' they follow a piecewise parameterized function.

Value

An list containing:

`ui`	A constraint matrix.
`ci`	A constraint vector.

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

Description

Function used for data cleaning.

Usage

```
data_clean(data, family = "binomial")
```

Arguments

`data`	Dataframe. Should have values id, value, index.
`family`	`gaussian` or `binomial`.

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.

`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.

Usage

```
fpca_gauss(
  Y,
  npc = 1,
  Kt = 8,
  maxiter = 20,
  t_min = NULL,
  t_max = NULL,
  print.iter = FALSE,
  row_obj = NULL,
  seed = 1988,
  ...
)
```

Arguments

`Y`	Dataframe. Should have variables id, value, index.
`npc`	Defaults to 1. Number of principal components to calculate.
`Kt`	Number of B-spline basis functions used to estimate mean functions. Default is 8.
`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. Default is 1991.
`...`	Additional arguments passed to or from other functions

Value

An object of class `fpca` containing:

`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.
`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 <jw3134@cumc.columbia.edu>, Jeff Goldsmith <ajg2202@cumc.columbia.edu>

References

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

`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.

`h_inv_parametric` *One parameter parametric warping on (0, T)*

Description

One parameter parametric warping on (0, T)

Usage

```
h_inv_parametric(grid, t_max = 1, beta = 0.01)
```

Arguments

`grid`	grid of values over which to evaluate the function.
`t_max`	maximum value to be evaluated on the time domain.
`beta`	parameter that controls shape of warping. Result approaches identity warp as beta approaches zero.

Value

A numeric vector containing values for a single warping function.

Examples

```
x = runif(100)
plot(x, type = 'l')
lines(registr:::h_inv_parametric(grid = x, beta = 0.5), col = "red")
```

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,
  parametric_warps = FALSE
)
```

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`	`gaussian` or `binomial`.
`t_min`	minimum value to be evaluated on the time domain.
`t_max`	maximum value to be evaluated on the time domain.
`parametric_warps`	If FALSE (default), inverse warping functions are estimated nonparametrically. If 'beta_cdf', they are assumed to have the form of a Beta(a,b) CDF. If 'piecewise' they follow a piecewise parameterized function.

Value

The scalar value taken by the loss function.

`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",
  t_min,
  t_max
)
```

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 — gaussian or binomial.

t_min — minimum value to be evaluated on the time domain.

t_max — maximum value to be evaluated on the time domain.

Value

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

Author(s)

Julia Wrobel <jw3134@cumc.columbia.edu>

---

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_parametric_hinv
Create two-parameter piecewise (inverse) warping functions

Description

This function uses a parametric model to calculate inverse warping functions for registration. The parameter `beta` controls the shape of warping, and the parameter `midpoint_percentile` control where the warping function crosses the identity line. 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_parametric_hinv(grid, beta = 0.01, midpoint_percentile = 0.5)
```

Arguments

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

`beta` parameter that controls shape of warping. Result approaches identity warp as beta approaches zero.

`midpoint_percentile` controls where the result crosses the identity warp. Default is 0.5, which forces result to cross identity line at median of grid.

Value

A numeric vector containing values for a single warping function.

Author(s)

Julia Wrobel `<jw3134@cumc.columbia.edu>`

Examples

```
x = runif(100)
plot(x, type = 'l')
lines(piecewise_parametric_hinv(grid = x, beta = 0.5), col = "red")
```

`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 FPCA to estimate warping functions for exponential family curves. The FPCA step is performed through the function `bfpca` if `family = "binomial"` or the function `fpca_gauss` if `family = "gaussian"`. Warping functions are calculated by the function `registr`.

Usage

```
register_fpca(
  Y,
  Kt = 8,
  Kh = 4,
  family = "binomial",
  max_iterations = 10,
  npc = 1,
  ...
)
```

Arguments

`Y`	Dataframe. Should have values id, value, index.
`Kt`	Number of B-spline basis functions used to estimate mean functions. Defaults to 8.
`Kh`	Number of B-spline basis functions used to estimate warping functions *h*. Defaults to 4.
`family`	`gaussian` or `binomial`.
`max_iterations`	Number of iterations for overall algorithm. Defaults to 10.
`npc`	Number of principal components to calculate. Defaults to 1.
`...`	Additional arguments passed to registr and fpca functions.

Details

Requires input data `Y` to be a dataframe in long format with variables `id`, `index`, and `value` to indicate subject IDs, times, and observations, respectively. The code calls two

Value

An object of class `registration` containing:

`fpca_obj`	List of items from FPCA step.
`Y`	The observed data plus variables `t_star` and `t_hat` which are the unregistered grid and registered grid, respectively.
`time_warps`	List of time values for each iteration of the algorithm. `time_warps[1]` is the original (observed) time and `time_warps[n]` provides time values for the nth iteration.
`loss`	Loss for each iteration of the algorithm, calculated in the registration step using an exponential family likelihood with natural parameter from the FPCA step.

family `gaussian` or `binomial`.

Author(s)

Julia Wrobel `<jw3134@cumc.columbia.edu>` Jeff Goldsmith `<ajg2202@cumc.columbia.edu>`

Examples

```
Y = simulate_unregistered_curves(I = 20, D = 200)
registr_object = register_fpca(Y, family = "binomial", max_iterations = 5)


# example using accelerometer data from nhanes 2003-2004 study
data(nhanes)
register_nhanes = register_fpca(nhanes, npc = 2, family = "binomial", max_iterations = 5)
```

---

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 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.

Usage

```
registr(
  obj = NULL,
  Y = NULL,
  Kt = 8,
  Kh = 4,
  family = "binomial",
  gradient = TRUE,
  beta = NULL,
  t_min = NULL,
  t_max = NULL,
  row_obj = NULL,
  parametric_warps = 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` `gaussian` or `binomial`.
- `gradient` if `TRUE`, uses analytic gradient to calculate derivative. If `FALSE`, calculates gradient numerically.
- `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.
- `parametric_warps` If FALSE (default), inverse warping functions are estimated nonparametrically. If 'beta_cdf', they are assumed to have the form of a Beta(a,b) CDF. If 'piecewise' they follow a piecewise parameterized function.
- `...` additional arguments passed to or from other functions

Value

An object of class `fpca` containing:

- `Y` The observed data. The variable index is the new estimated time domain.
- `loss` Value of the loss function after registraton.
- `beta` Matrix of B-spline basis coefficients used to construct subject-specific warping functions.

Author(s)

Julia Wrobel

Julia Wrobel <jw3134@cumc.columbia.edu>

Examples

```
Y = simulate_unregistered_curves()
register_step = registr(obj = NULL, Y = Y, Kt = 6, Kh = 3, family = "binomial",
   gradient = TRUE)
testthat::expect_error({
registr(obj = list(Y = Y), Kt = 6, Kh = 3, family = "binomial",
   gradient = TRUE)
```

```
})
testthat::expect_error({
registr(obj = NULL, Y = Y, Kt = 2, Kh = 3)
})
testthat::expect_error({
registr(obj = NULL, Y = Y, Kt = 6, Kh = 2)
})

Y = simulate_unregistered_curves()
register_step = registr(obj = NULL, Y = Y, Kt = 6, Kh = 3, family = "binomial",
   gradient = TRUE)
```

---

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 <jw3134@cumc.columbia.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 `<jw3134@cumc.columbia.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