Package: fase (via r-universe)

August 27, 2024

Title Functional Adjacency Spectral Embedding

Version 1.0.1.9000

Description Latent process embedding for functional network data with the Functional Adjacency Spectral Embedding. Fits smooth latent processes based on cubic spline bases. Also generates functional network data from three models, and evaluates a network generalized cross-validation criterion for dimension selection. For more information, see MacDonald, Zhu and Levina (2022+) <arXiv:2210.07491>.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL https://github.com/peterwmacd/fase

BugReports https://github.com/peterwmacd/fase/issues

Imports RSpectra (>= 0.16.1), rTensor (>= 1.4.8), splines2 (>= 0.4.7)

Repository https://peterwmacd.r-universe.dev

RemoteUrl https://github.com/peterwmacd/fase

RemoteRef HEAD

RemoteSha 9f8ac1f7dd11f34e03f8766a953bc0c167efdde8

Contents

`fase` *Functional adjacency spectral embedding*

Description

`fase` fits a functional adjacency spectral embedding to snapshots of (undirected) functional network data. The latent processes are fit in a spline basis specified by the user, with additional options for ridge penalization.

Usage

```
fase(A,d,self_loops,spline_design,lambda,optim_options,output_options)
```

Arguments

`A`	An $n \times n \times m$ array containing the snapshots of the functional network.
`d`	A positive integer, the number of latent space dimensions of the functional embedding.
`self_loops`	A Boolean, if `FALSE`, all diagonal entries are ignored in optimization. Defaults to `TRUE`.
`spline_design`	A list, containing the spline design information. For fitting with a B-spline design (the default):

type The string `'bs'`.

q A positive integer, the dimension of the B-spline basis.

x_vec A vector, the snapshot evaluation indices for the data. Defaults to an equally spaced vector of length m from `0` to `1`.

x_max A scalar, the maximum of the index space. Defaults to `max(spline_design$x_vec)`.

x_min A scalar, the minimum of the index space. Defaults to `min(spline_design$x_vec)`.

spline_matrix An $m \times q$ matrix, the B-spline basis evaluated at the snapshot indices. If not specified, it will be calculated internally.

ridge_mat The $m \times m$ matrix for the generalized ridge penalty. If `lambda`> 0, defaults to `diag(m)`.

For fitting with a smoothing spline design:

type The string `'ss'`.

x_vec A vector, the snapshot evaluation indices for the data. Defaults to an equally spaced vector of length m from `0` to `1`.

x_max A scalar, the maximum of the index space. Defaults to `max(spline_design$x_vec)`.

x_min A scalar, the minimum of the index space. Defaults to `min(spline_design$x_vec)`.

spline_matrix An $m \times m$ matrix, the natural cubic spline basis evaluated at the snapshot indices. If not specified, it will be calculated internally.

ridge_mat The $m \times m$ matrix for the generalized ridge penalty. Defaults to the second derivatives of the natural cubic spline basis evaluated at the snapshot indices.

`lambda` A positive scalar, the scale factor for the generalized ridge penalty (see Details). Defaults to `0`.

`optim_options` A list, containing additional optional arguments controlling the gradient descent algorithm.

eps A positive scalar, the convergence threshold for gradient descent in terms of relative change in objective value. Defaults to `1e-5`.

eta A positive scalar, the step size for gradient descent. Defaults to `1/(n*m)`.

K_max A positive integer, the maximum iterations for gradient descent. Defaults to `2e3`.

verbose A Boolean, if `TRUE`, console output will provide updates on the progress of gradient descent. Defaults to `FALSE`.

init_W A 3-dimensional array containing initial basis coordinates for gradient descent. Dimension should be $n \times$ `spline_design$q` $\times d$ for B-spline designs, and $n \times m \times d$ for smoothing spline designs. If included, `init_M`, `init_L` and `init_sigma` are ignored.

init_sigma A positive scalar, the estimated edge dispersion parameter to calibrate initialization. If not provided, it is either estimated using the robust method proposed by Gavish and Donoho (2014) for weighted edge networks, or set to a default value `0.5` for binary edge networks.

init_L A positive integer, the number of contiguous groups used for initialization. Defaults to the floor of $(2nm/\texttt{init\_sigma}^2)^{1/3}$.

init_M A positive integer, the number of snapshots averaged in each group for initialization. Defaults use all snapshots.

`output_options` A list, containing additional optional arguments controlling the output of `fase`.

align_output A Boolean, if `TRUE`, the returned latent processes have been aligned according to a Procrustes alignment which minimizes (in terms of Frobenius norm) the overall discrepancies between consecutive snapshots. Defaults to `TRUE`.

return_coords A Boolean, if `TRUE`, the basis coordinates for each latent process component are also returned as an array. Defaults to `FALSE`.

return_ngcv A Boolean, if `TRUE` and `spline_design$type=='bs'`, the network generalized cross validation criterion is returned. Defaults to `TRUE`.

Details

`fase` finds a functional adjacency spectral embedding of an $n \times n \times m$ array A of symmetric adjacency matrices on a common set of nodes, where each $n \times n$ slice is associated to a scalar index x_k for $k = 1, ..., m$. Embedding requires the specification of a latent space dimension d and spline design information (with the argument `spline_design`).

`fase` can fit latent processes using either a cubic B-spline basis with equally spaced knots, or a natural cubic spline basis with a second derivative (generalized ridge) smoothing penalty: a smoothing spline. To fit with a B-spline design (`spline_design$type = 'bs'`), one must minimally provide a basis dimension q of at least `4` and at most m.

When fitting with a smoothing spline design, the generalized ridge penalty is scaled by λ/n, where λ is specified by the argument `lambda`. see MacDonald et al., (2022+), Appendix E for more

details. `lambda` can also be used to introduce a ridge penalty on the basis coordinates when fitting with B-splines.

Fitting minimizes a least squares loss, using gradient descent (Algorithm 2) on the basis coordinates $w_{i,r}$ of each component process

$$z_{i,r}(x) = w_{i,r}^T B(x).$$

Additional options for the fitting algorithm, including initialization, can be specified by the argument `optim_options`. For more details on the fitting and initialization algorithms, see MacDonald et al., (2022+), Section 3.

By default, `fase` will return estimates of the latent processes evaluated at the snapshot indices as an $n \times d \times m$ array, after performing a Procrustes alignment of the consecutive snapshots. This extra alignment step can be skipped. `fase` will also return the spline design information used to fit the embedding, convergence information for gradient descent, and (if specified) the basis coordinates.

When fitting with B-splines, `fase` can return a network generalized cross validation criterion, described in MacDonald et al., (2022+), Section 3.3. This criterion can be minimized to choose appropriate values for q and d.

Value

A list is returned with the functional adjacency spectral embedding, the spline design information, and some additional optimization output:

`Z` An $n \times d \times m$ array containing the latent process embedding evaluated at the indices in `spline_design$x_vec`.

`W` For B-spline designs, an $n \times q \times d$ array; or for smoothing spline designs, an $n \times m \times d$ array of estimated basis coordinates. If `output_options$return_coords` is `FALSE`, this is not returned.

`spline_design` A list, describing the spline design:

- **type** A string, either `'bs'` or `'ss'`.
- **q** A positive integer, the dimension of the B-spline basis. Only returned for B-spline designs.
- **x_vec** A vector, the snapshot evaluation indices for the data.
- **x_max** A scalar, the maximum of the index space.
- **x_min** A scalar, the minimum of the index space.
- **spline_matrix** For B-spline designs, an $m \times q$ matrix; or for smoothing spline designs, an $m \times m$ matrix, the basis evaluated at the snapshot indices.
- **ridge_matrix** An $m \times m$ matrix used in the generalized ridge penalty. Only returned for `lambda > 0`.

`ngcv` A scalar, the network generalized cross validation criterion (see Details). Only returned for B-spline designs and when `output_options$return_ngcv` is `TRUE`.

`K` A positive integer, the number of iterations run in gradient descent.

`converged` An integer convergence code, `1` if gradient descent converged in fewer than `optim_options$K_max` iterations, `0` otherwise.

Examples

```
# Gaussian edge data with sinusoidal latent processes
set.seed(1)
data <- gaussian_snapshot_ss(n=50,d=2,
                             x_vec=seq(0,1,length.out=50),
                             self_loops=FALSE,sigma_edge=4)


# fase fit with B-spline design
fit_bs <- fase(data$A,d=2,self_loops=FALSE,
               spline_design=list(type='bs',q=9,x_vec=data$spline_design$x_vec),
               optim_options=list(eps=1e-4,K_max=40),
               output_options=list(return_coords=TRUE))

# fase fit with smoothing spline design
fit_ss <- fase(data$A,d=2,self_loops=FALSE,
               spline_design=list(type='ss',x_vec=data$spline_design$x_vec),
               lambda=.5,
               optim_options=list(eta=1e-4,K_max=40,verbose=FALSE),
               output_options=list(align_output=FALSE))

#NOTE: both examples fit with small optim_options$K_max=40 for demonstration
```

`fase_seq` *Functional adjacency spectral embedding (sequential algorithm)*

Description

`fase_seq` fits a functional adjacency spectral embedding to snapshots of (undirected) functional network data, with each of the d latent dimensions fit sequentially. The latent processes are fit in a spline basis specified by the user, with additional options for ridge penalization.

Usage

```
fase_seq(A,d,self_loops,spline_design,lambda,optim_options,output_options)
```

Arguments

`A`	An $n \times n \times m$ array containing the snapshots of the functional network.
`d`	A positive integer, the number of latent space dimensions of the functional embedding.
`self_loops`	A Boolean, if `FALSE`, all diagonal entries are ignored in optimization. Defaults to `TRUE`.
`spline_design`	A list, containing the spline design information. For fitting with a B-spline design (the default):
	type The string `'bs'`.

q A positive integer, the dimension of the B-spline basis.

x_vec A vector, the snapshot evaluation indices for the data. Defaults to an equally spaced vector of length m from `0` to `1`.

x_max A scalar, the maximum of the index space. Defaults to `max(spline_design$x_vec)`.

x_min A scalar, the minimum of the index space. Defaults to `min(spline_design$x_vec)`.

spline_matrix An $m \times q$ matrix, the B-spline basis evaluated at the snapshot indices. If not specified, it will be calculated internally.

ridge_mat The $m \times m$ matrix for the generalized ridge penalty. If `lambda`> 0, defaults to `diag(m)`.

For fitting with a smoothing spline design:

type The string `'ss'`.

x_vec A vector, the snapshot evaluation indices for the data. Defaults to an equally spaced vector of length m from `0` to `1`.

x_max A scalar, the maximum of the index space. Defaults to `max(spline_design$x_vec)`.

x_min A scalar, the minimum of the index space. Defaults to `min(spline_design$x_vec)`.

spline_matrix An $m \times m$ matrix, the natural cubic spline basis evaluated at the snapshot indices. If not specified, it will be calculated internally.

ridge_mat The $m \times m$ matrix for the generalized ridge penalty. Defaults to the second derivatives of the natural cubic spline basis evaluated at the snapshot indices.

`lambda` A positive scalar, the scale factor for the generalized ridge penalty (see Details). Defaults to `0`.

`optim_options` A list, containing additional optional arguments controlling the gradient descent algorithm.

eps A positive scalar, the convergence threshold for gradient descent in terms of relative change in objective value. Defaults to `1e-5`.

eta A positive scalar, the step size for gradient descent. Defaults to `1/(n*m)`.

K_max A positive integer, the maximum iterations for gradient descent. Defaults to `2e3`.

verbose A Boolean, if `TRUE`, console output will provide updates on the progress of gradient descent. Defaults to `FALSE`.

init_W A 3-dimensional array containing initial basis coordinates for gradient descent. Dimension should be $n \times$`spline_design$q`$\times d$ for B-spline designs, and $n \times m \times d$ for smoothing spline designs. If included, `init_M`, `init_L` and `init_sigma` are ignored.

init_sigma A positive scalar, the estimated edge dispersion parameter to calibrate initialization. If not provided, it is either estimated using the robust method proposed by Gavish and Donoho (2014) for weighted edge networks, or set to a default value `0.5` for binary edge networks.

init_L A positive integer, the number of contiguous groups used for initialization. Defaults to the floor of $(2nm/$`init_sigma`$^2)^{1/3}$.

init_M A positive integer, the number of snapshots averaged in each group for initialization. Defaults use all snapshots.

`output_options` A list, containing additional optional arguments controlling the output of `fase`.

return_coords A Boolean, if TRUE, the basis coordinates for each latent process component are also returned as an array. Defaults to FALSE.

return_ngcv A Boolean, if TRUE and spline_design$type=='bs', the network generalized cross validation criterion is returned. Defaults to TRUE.

Details

Note that fase_seq is a wrapper for fase. When $d = 1$, fase_seq coincides with fase.

fase_seq finds a functional adjacency spectral embedding of an $n \times n \times m$ array A of symmetric adjacency matrices on a common set of nodes, where each $n \times n$ slice is associated to a scalar index x_k for $k = 1, ..., m$. Embedding requires the specification of a latent space dimension d and spline design information (with the argument spline_design).

fase_seq can fit latent processes using either a cubic B-spline basis with equally spaced knots, or a natural cubic spline basis with a second derivative (generalized ridge) smoothing penalty: a smoothing spline. To fit with a B-spline design (spline_design$type = 'bs'), one must minimally provide a basis dimension q of at least 4 and at most m.

When fitting with a smoothing spline design, the generalized ridge penalty is scaled by λ/n, where λ is specified by the argument lambda. see MacDonald et al., (2022+), Appendix E for more details. lambda can also be used to introduce a ridge penalty on the basis coordinates when fitting with B-splines.

Fitting minimizes a least squares loss, using gradient descent (Algorithm 1) on the basis coordinates $w_{i,r}$ of each component process

$$z_{i,r}(x) = w_{i,r}^T B(x).$$

Additional options for the fitting algorithm, including initialization, can be specified by the argument optim_options. For more details on the fitting and initialization algorithms, see MacDonald et al., (2022+), Section 3.

By default, fase_seq will return estimates of the latent processes evaluated at the snapshot indices as an $n \times d \times m$ array, after performing a Procrustes alignment of the consecutive snapshots. This extra alignment step can be skipped. fase_seq will also return the spline design information used to fit the embedding, convergence information for gradient descent, and (if specified) the basis coordinates.

When fitting with B-splines, fase_seq can return a network generalized cross validation criterion, described in MacDonald et al., (2022+), Section 3.3. This criterion can be minimized to choose appropriate values for q and d.

Value

A list is returned with the functional adjacency spectral embedding, the spline design information, and some additional optimization output:

Z	An $n \times d \times m$ array containing the latent process embedding evaluated at the indices in spline_design$x_vec.
W	For B-spline designs, an $n \times q \times d$ array; or for smoothing spline designs, an $n \times m \times d$ array of estimated basis coordinates. If output_options$return_coords is FALSE, this is not returned.
spline_design	A list, describing the spline design:

type A string, either 'bs' or 'ss'.

q A positive integer, the dimension of the B-spline basis. Only returned for B-spline designs.

x_vec A vector, the snapshot evaluation indices for the data.

x_max A scalar, the maximum of the index space.

x_min A scalar, the minimum of the index space.

spline_matrix For B-spline designs, an $m \times q$ matrix; or for smoothing spline designs, an $m \times m$ matrix, the basis evaluated at the snapshot indices.

ridge_matrix An $m \times m$ matrix used in the generalized ridge penalty. Only returned for lambda > 0.

ngcv — A scalar, the network generalized cross validation criterion (see Details). Only returned for B-spline designs and when output_options$return_ngcv is TRUE.

K — A positive integer, the number of iterations run in gradient descent.

converged — An integer convergence code, 1 if gradient descent converged in fewer than optim_options$K_max iterations, 0 otherwise.

Examples

```
# Gaussian edge data with sinusoidal latent processes
set.seed(1)
data <- gaussian_snapshot_ss(n=50,d=2,
                             x_vec=seq(0,1,length.out=50),
                             self_loops=FALSE,sigma_edge=4)


# fase fit with B-spline design
fit_bs <- fase_seq(data$A,d=2,self_loops=FALSE,
                   spline_design=list(type='bs',q=9,x_vec=data$spline_design$x_vec),
                   optim_options=list(eps=1e-4,K_max=40),
                   output_options=list(return_coords=TRUE))

# fase fit with smoothing spline design
fit_ss <- fase_seq(data$A,d=2,self_loops=FALSE,
                   spline_design=list(type='ss',x_vec=data$spline_design$x_vec),
                   lambda=.5,
                   optim_options=list(eta=1e-4,K_max=40,verbose=FALSE))

#NOTE: both models fit with small optim_options$K_max=40 for demonstration
```

---

gaussian_snapshot_bs *Simulate Gaussian edge networks with B-spline latent processes*

Description

gaussian_snapshot_bs simulates a realization of a functional network with Gaussian edges, according to an inner product latent process model. The latent processes are generated from a B-spline basis with equally spaced knots.

Usage

```
gaussian_snapshot_bs(n,d,m,self_loops=TRUE,
                     spline_design,sigma_edge=1,
                     process_options)
```

Arguments

`n`	A positive integer, the number of nodes.
`d`	A positive integer, the number of latent space dimensions.
`m`	A positive integer, the number of snapshots. If this argument is not specified, it is determined from the snapshot index vector `spline_design$x_vec`.
`self_loops`	A Boolean, if `FALSE`, all diagonal adjacency matrix entries are set to zero. Defaults to `TRUE`.
`spline_design`	A list, describing the B-spline design:
	q A positive integer, the dimension of the B-spline basis. Must be at least `4` and at most `m`.
	x_vec A vector, the snapshot evaluation indices for the data. Defaults to an equally spaced sequence of length `m` from `0` to `1`.
	x_max A scalar, the maximum of the index space. Defaults to `max(spline_design$x_vec)`.
	x_min A scalar, the minimum of the index space. Defaults to `min(spline_design$x_vec)`.
`sigma_edge`	A positive scalar, the entry-wise standard deviation for the Gaussian edge variables. Defaults to `1`.
`process_options`	
	A list, containing additional optional arguments:
	sigma_coord A positive scalar, or a vector of length d. If it is a vector, the entries correspond to the standard deviation of the randomly generated basis coordinates for each latent dimension. If is is a scalar, it corresponds to the standard deviation of the basis coordinates in all dimensions. Defaults to `1`.

Details

The spline design of the functional network data (snapshot indices, basis dimension) is generated using the information provided in `spline_design`, producing a q-dimensional cubic B-spline basis with equally spaced knots.

The latent process basis coordinates are generated as iid Gaussian random variables with standard deviation `process_options$sigma_coord`. Each latent process is given by

$$z_{i,r}(x) = w_{i,r}^T B(x).$$

Then, the $n \times n$ symmetric adjacency matrix for snapshot $k = 1, ..., m$ has independent Gaussian entries with standard deviation `sigma_edge` and mean

$$E([A_k]_{ij}) = z_i(x_k)^T z_j(x_k)$$

for $i \leq j$ (or $i < j$ with no self loops).

Value

A list is returned with the realizations of the basis coordinates, spline design, and the multiplex network snapshots:

`A` An array of dimension $n \times n \times m$, the realized functional network data.

`W` An array of dimension $n \times q \times d$, the realized basis coordinates.

`spline_design` A list, describing the B-spline design:

type The string `'bs'`.

q A positive integer, the dimension of the B-spline basis.

x_vec A vector, the snapshot evaluation indices for the data.

x_max A scalar, the maximum of the index space.

x_min A scalar, the minimum of the index space.

spline_matrix An $m \times q$ matrix, the B-spline basis evaluated at the snapshot indices.

Examples

```
# Gaussian edge data with B-spline latent processes, Gaussian coordinates
# NOTE: x_vec is automatically populated given m

data <- gaussian_snapshot_bs(n=100,d=4,m=100,
                             self_loops=FALSE,
                             spline_design=list(q=12),
                             sigma_edge=3,
                             process_options=list(sigma_coord=.75))
```

---

gaussian_snapshot_ss *Simulate Gaussian edge networks with nonparametric latent processes*

Description

`gaussian_snapshot_ss` simulates a realization of a functional network with Gaussian edges, according to an inner product latent process model. The latent processes are randomly generated sinusoidal functions.

Usage

```
gaussian_snapshot_ss(n,d,m,x_vec,self_loops=TRUE,
                     sigma_edge=1,process_options)
```

Arguments

n	A positive integer, the number of nodes.
d	A positive integer, the number of latent space dimensions.
m	A positive integer, the number of snapshots. If this argument is not specified, it is determined from the snapshot index vector x_vec.
x_vec	A vector, the snapshot evaluation indices for the data. Defaults to an equally spaced sequence of length m from 0 to 1.
self_loops	A Boolean, if FALSE, all diagonal adjacency matrix entries are set to zero. Defaults to TRUE.
sigma_edge	A positive scalar, the entry-wise standard deviation for the Gaussian edge variables. Defaults to 1.
process_options	A list, containing additional optional arguments:

amplitude A positive scalar, the maximum amplitude of the randomly generated latent processes. Defaults to 3.

frequency A positive scalar, frequency of the randomly generated latent processes. Defaults to 2.

sigma_int A positive scalar, or a vector of length d. If it is a vector, the entries correspond to the standard deviation of the random intercepts of the node processes for each latent dimension. If is is a scalar, it corresponds to the standard deviation of the random intercepts in all dimensions. Defaults to 0.5.

return_fn A Boolean, if TRUE, then the latent processes are returned as a function which takes a vector of indices and returns the corresponding evaluations of the latent process matrices. Otherwise, the latent processes are returned as an $n \times d \times m$ array evaluated at the prespecified snapshot indices. Defaults to FALSE.

Details

The the latent process for node i in latent dimension r is given independently by

$$z_{i,r}(x) = \frac{a \sin[2f\pi(x - U)/(x_{max} - x_{min})]}{1 + (2a - 1)[x + B(x_{max} - 2x)]} + G$$

Where G is Gaussian with mean 0 and standard deviation $\sigma_{int,r}$, B is Bernoulli with mean 1/2, and U is uniform with minimum spline_design$x_min and maximum spline_design$x_max. f is a frequency parameter specified with process_options$frequency, and a is a maximum amplitude parameter specified with process_options$amplitude. Roughly, each process is a randomly shifted sine function which goes through f cycles on the index set, with amplitude either increasing or decreasing between $1/2$ and a.

Then, the $n \times n$ symmetric adjacency matrix for snapshot $k = 1, ..., m$ has independent Gaussian entries with standard deviation sigma_edge and mean

$$E([A_k]_{ij}) = z_i(x_k)^T z_j(x_k)$$

for $i \leq j$ (or $i < j$ with no self loops).

This function may return the latent processes as an $n \times d \times m$ array evaluated at the prespecified snapshot indices, or as a function which takes a vector of indices and returns the corresponding evaluations of the latent process matrices. It also returns the spline design information required to fit a FASE embedding to this data with a natural cubic spline.

Value

A list is returned with the realizations of the basis coordinates, spline design, and the multiplex network snapshots:

`A` An array of dimension $n \times n \times m$, the realized functional network data.

`Z` If `process_options$return_fn` is `TRUE`, a function, which takes a vector of indices and returns the corresponding evaluations of the latent process matrices. Otherwise, an array of dimension $n \times d \times m$, the latent processes evaluated at the prespecified snapshot indices.

`spline_design` A list, describing the B-spline design:

type The string `'ss'`.

x_vec A vector, the snapshot evaluation indices for the data.

Examples

```
# Gaussian edge data with sinusoidal latent processes
# NOTE: latent processes are returned as a function

data <- gaussian_snapshot_ss(n=100,d=2,
                             x_vec=seq(0,3,length.out=80),
                             self_loops=TRUE,
                             sigma_edge=4,
                             process_options=list(amplitude=4,
                                                  frequency=3,
                                                  return_fn=TRUE))
```

`proc_align` *Procrustes alignment*

Description

`proc_align` orthogonally transforms the columns of a matrix A to find the best approximation (in terms of Frobenius norm) to a second matrix B. Optionally, it may also return the optimal transformation matrix.

Usage

```
proc_align(A,B,return_orth=FALSE)
```

Arguments

A	An $n \times d$ matrix.
B	An $n \times d$ matrix.
return_orth	A Boolean which specifies whether to return the orthogonal transformation. Defaults to FALSE.

Value

If return_orth is FALSE, returns the $n \times d$ matrix resulting from applying the optimal aligning transformation to the columns of A. Otherwise, returns a list with two entries:

Ao	The $n \times d$ matrix resulting from applying the optimal aligning transformation to the columns of A.
orth	The $d \times d$ optimal aligning orthogonal transformation matrix.

---

proc_align3 *Procrustes alignment for 3-mode tensors*

Description

proc_align3 applies one orthogonal transformation to the columns of each of the $n \times d$ slices of an $n \times d \times m$ array A to find the best approximation (in terms of matrix Frobenius norm, averaged over the $n \times d$ slices) to a second $n \times d \times m$ array B. Optionally, it may also return the optimal transformation matrix.

Usage

```
proc_align3(A,B,return_orth=FALSE)
```

Arguments

A	An $n \times d \times m$ array.
B	An $n \times d \times m$ array.
return_orth	A Boolean which specifies whether to return the orthogonal transformation. Defaults to FALSE.

Value

If return_orth is FALSE, returns the $n \times d \times m$ array resulting from applying the optimal aligning transformation to the columns of the $n \times d$ slices of A. Otherwise, returns a list with two entries:

Ao	The $n \times d$ matrix resulting from applying the optimal aligning transformation to the columns of the $n \times d$ slices of A.
orth	The $d \times d$ optimal aligning orthogonal transformation matrix.

proc_align_slicewise3 *Slicewise Procrustes alignment for 3-mode tensors*

Description

proc_align_slicewise3 applies an orthogonal transformation to the columns of each of the $n \times d$ slices of an $n \times d \times m$ array A to find the best approximation (in terms of matrix Frobenius norm) to the corresponding $n \times d$ slice of a second $n \times d \times m$ array B.

Usage

```
proc_align_slicewise3(A,B)
```

Arguments

A	An $n \times d \times m$ array.
B	An $n \times d \times m$ array.

Value

Returns the $n \times d \times m$ array resulting from applying the optimal aligning transformations to the columns of the $n \times d$ slices of A.

rdpg_snapshot_bs *Simulate binary edge networks with B-spline latent processes*

Description

rdpg_snapshot_bs simulates a realization of a functional network with Bernoulli edges, according to an inner product latent process model. The latent processes are generated from a B-spline basis with equally spaced knots.

Usage

```
rdpg_snapshot_bs(n,d,m,self_loops=TRUE,
                 spline_design,process_options)
```

Arguments

n	A positive integer, the number of nodes.
d	A positive integer, the number of latent space dimensions.
m	A positive integer, the number of snapshots. If this argument is not specified, it is determined from the snapshot index vector spline_design$x_vec.
self_loops	A Boolean, if FALSE, all diagonal adjacency matrix entries are set to zero. Defaults to TRUE.

`spline_design` A list, describing the B-spline design:

q A positive integer, the dimension of the B-spline basis. Must be at least `4` and at most `m`.

x_vec A vector, the snapshot evaluation indices for the data. Defaults to an equally spaced sequence of length `m` from `0` to `1`.

x_max A scalar, the maximum of the index space. Defaults to `max(spline_design$x_vec)`.

x_min A scalar, the minimum of the index space. Defaults to `min(spline_design$x_vec)`.

`process_options`

A list, containing additional optional arguments:

alpha_coord A positive scalar, or a vector of length d. If it is a vector, it corresponds to the Dirichlet parameter of the basis coordinates. If is is a scalar, the basis coordinates have Dirichlet parameter `rep(alpha_coord,d)`. Defaults to `0.1`.

density A scalar between `0` and `1`, which controls the approximate overall edge density of the resulting multiplex matrix. Defaults to `1/d`. If specified larger than `1/d`, this argument is reset to `1/d` and a warning is given.

Details

The spline design of the functional network data (snapshot indices, basis dimension) is generated using the information provided in `spline_design`, producing a q-dimensional cubic B-spline basis with equally spaced knots.

The $(q \times d)$ latent process basis coordinates W_i for each node are generated as q iid Dirichlet random variables with d-dimensional parameter `process_options$alpha_coord` or `rep(process_options$alpha_coord,d)` depending on the dimension of `process_options$alpha_coord`. Roughly, smaller values of `process_options$alpha_coo` will tend to generate latent positions closer to the corners of the simplex.

W_i is then rescaled so the overall network density is approximately `process_options$density`, and the Euclidean norm of $z_i(x)$ never exceeds `1`. If the density requested is too high, it will revert to the maximum density under this model $(1/d)$. Then each latent process is given by

$$z_i(x) = W_i^T B(x).$$

The $n \times n$ symmetric adjacency matrix for snapshot $k = 1, ..., m$ has independent Bernoulli entries with mean

$$E([A_k]_{ij}) = z_i(x_k)^T z_j(x_k)$$

for $i \leq j$ (or $i < j$ with no self loops).

Value

A list is returned with the realizations of the basis coordinates, spline design, and the multiplex network snapshots:

`A` An array of dimension $n \times n \times m$, the realized functional network data.

`W` An array of dimension $n \times q \times d$, the realized basis coordinates.

`spline_design` A list, describing the B-spline design:

type The string `'bs'`.

q A positive integer, the dimension of the B-spline basis.

x_vec A vector, the snapshot evaluation indices for the data.

x_max A scalar, the maximum of the index space.

x_min A scalar, the minimum of the index space.

spline_matrix An $m \times q$ matrix, the B-spline basis evaluated at the snapshot indices.

Examples

```
# Bernoulli edge data with B-spline latent processes, Dirichlet coordinates
# NOTE: for B-splines, x_max and x_min do not need to coincide with the
# max and min snapshot times.

data <- rdpg_snapshot_bs(n=100,d=10,
                         self_loops=FALSE,
                         spline_design=list(q=8,
                                            x_vec=seq(-1,1,length.out=50),
                                            x_min=-1.1,x_max=1.1),
                         process_options=list(alpha_coord=.2,
                         density=1/10))
```

Index