Package ‘fgm’

October 13, 2022

Type Package

Title Partial Separability and Functional Gaussian Graphical Models

Description Estimates a functional graphical model and a partially separable Karhunen-Loève decomposition for a multivariate Gaussian process. See Zapata J., Oh S. and Petersen A. (2019) <arXiv:1910.03134>.

Version 1.0

Maintainer Javier Zapata <jzapata@ucsb.edu>

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports JGL, fdapace

Suggests mvtnorm, fda, knitr, rmarkdown

RoxygenNote 6.1.1

NeedsCompilation no

Author Javier Zapata [cre],
Sang-Yun Oh [aut],
Alexander Petersen [aut]

Repository CRAN

Date/Publication 2019-10-22 12:20:02 UTC

R topics documented:

fgm *Functional Gaussian Graphical Model*

Description

Estimates a sparse adjacency matrix representing the conditional dependency structure between features of a multivariate Gaussian process

Usage

```
fgm(y, L, alpha, gamma, t = seq(0, 1, length.out = dim(y[[1]])[2]),
  thr.FVE = 95, include.Omega = FALSE)
```

Arguments

`y`	list of length p containing densely observed multivariate (p-dimensional) functional data. `y[[j]]` is an nxm matrix of functional data for n subjects observed on a grid of length m
`L`	the number of eigenfunctions used for dimension reduction using the partially separable Karhunen-Loeve (PSKL) expansion obtained using 'pfpca()'. This argument can take positive integer values greater or equal to 1.
`alpha`	penalty parameter for the common sparsity pattern taking values in `[0,1]`.
`gamma`	penalty parameter for the overall sparsity pattern taking positive values.
`t`	(optional) grid on which functional data is observed, defaults to `seq(0, 1, m)` where `m = dim(data[[1]])[2]`.
`thr.FVE`	this parameter sets a threshold for the minimum percentage of functional variance explained (FVE) by the PSKL eigenfunctions (obtained using 'pfpca()'). This criterion is used only if a value for L is not provided or is greater than the maximum possible number of eigenfunctions estimated from `y` using `pfpca()`.
`include.Omega`	logical variable indicating wheter to include the list of precision matrices in the output. Default value is FALSE.

Details

This function implements the functional graphical model in Zapata, Oh, and Petersen (2019). The arguments `alpha` and `gamma` are a reparameterization of the Group Graphical Lasso tuning parameters when using the `JGL` package. When using `JGL::JGL`, the tuning parameters are computed as `lambda1 = alpha*gamma` and `lambda2 = (1-alpha)*gamma`

Value

A list with letters and numbers.

`A` Resulting adjacency matrix as the union of all the Omega matrices

`L` number of PSKL expansion eigenfunctions considered for the estimation of the graphical model.

`Omega` list of of precision matrices obtained using the multivariate functional principal component scores theta obtained using 'fpca()'

Author(s)

Javier Zapata, Sang-Yun Oh and Alexander Petersen

References

Zapata J., Oh S. and Petersen A. (2019) - Partial Separability and Functional Graphical Models for Multivariate Gaussian Processes. Available at https://arxiv.org/abs/1910.03134.

Examples

```
## Variables
# Omega - list of precision matrices, one per eigenfunction
# Sigma - list of covariance matrices, one per eigenfunction
# theta - list of functional  principal component scores
# phi - list of eigenfunctions densely observed on a time grid
# y - list containing densely observed multivariate (p-dimensional) functional data

library(mvtnorm)
library(fda)

## Generate data y
 source(system.file("exec", "getOmegaSigma.R", package = "fgm"))
 theta = lapply(1:nbasis, function(b) t(rmvnorm(n = 100, sigma = Sigma[[b]])))
 theta.reshaped = lapply( 1:p, function(j){
     t(sapply(1:nbasis, function(i) theta[[i]][j,]))
 })
 phi.basis=create.fourier.basis(rangeval=c(0,1), nbasis=21, period=1)
 t = seq(0, 1, length.out = time.grid.length)
 chosen.basis = c(2, 3, 6, 7, 10, 11, 16, 17, 20, 21)
 phi = t(predict(phi.basis, t))[chosen.basis,]
 y = lapply(theta.reshaped, function(th) t(th)%*%phi)

## Solve
 fgm(y, alpha=0.5, gamma=0.8)
```

---

pfpca *Partially Separable Karhunen-Loeve Expansion*

Description

Estimates the Karhunen-Loeve expansion for a partially separable multivariate Gaussian process.

Usage

```
pfpca(y, t = seq(0, 1, length.out = dim(y[[1]])[2]))
```

Arguments

y
: list of length p containing densely observed multivariate (p-dimensional) functional data . `y[[j]]` is an nxm matrix of functional data for n subjects observed on a grid of length m

t
: (optional) grid on which functional data is observed, defaults to seq(0, 1, m) where `m = dim(data[[1]])[2]`

Details

This function implements the functional graphical model in Zapata, Oh, and Petersen (2019). This code uses functions from the testing version of fdapace available at: https://github.com/functionaldata/tPACE.

Value

A list with three variables:

- `phi` Lxm matrix where each row denotes the value of a basis function evaluated at a grid of length m
- `theta` list of length L of functional principal component scores. `theta[[l]]` is an nxp matrix of vector scores corresponding to the basis function `phi[l,]`
- `FVE` fraction of functional variance explained (FVE) by the first L components

Author(s)

Javier Zapata, Sang-Yun Oh and Alexander Petersen

References

Zapata J., Oh S. and Petersen A. (2019) - Partial Separability and Functional Graphical Models for Multivariate Gaussian Processes. Available at https://arxiv.org/abs/1910.03134.

Examples

```
## Variables
# Omega - list of precision matrices, one per eigenfunction
# Sigma - list of covariance matrices, one per eigenfunction
# theta - list of functional  principal component scores
# phi - list of eigenfunctions densely observed on a time grid
# y - list containing densely observed multivariate (p-dimensional) functional data

library(mvtnorm)
library(fda)

## Generate data y
 source(system.file("exec", "getOmegaSigma.R", package = "fgm"))
 theta = lapply(1:nbasis, function(b) t(rmvnorm(n = 100, sigma = Sigma[[b]])))
 theta.reshaped = lapply( 1:p, function(j){
     t(sapply(1:nbasis, function(i) theta[[i]][j,]))
 })
```

```
 phi.basis=create.fourier.basis(rangeval=c(0,1), nbasis=21, period=1)
 t = seq(0, 1, length.out = time.grid.length)
 chosen.basis = c(2, 3, 6, 7, 10, 11, 16, 17, 20, 21)
 phi = t(predict(phi.basis, t))[chosen.basis,]
 y = lapply(theta.reshaped, function(th) t(th)%*%phi)

## Solve
 pfpca(y)
```

Index