Package: SLFPCA (via r-universe)

August 30, 2024

Title Sparse Logistic Functional Principal Component Analysis

Version 3.0

Description Implementation for sparse logistic functional principal component analysis (SLFPCA). SLFPCA is specifically developed for functional binary data, and the estimated eigenfunction can be strictly zero on some sub-intervals, which is helpful for interpretation. The crucial function of this package is SLFPCA().

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.1

Imports fda, fdapace, psych, splines, stats

NeedsCompilation no

Author Rou Zhong [aut, cre], Jingxiao Zhang [aut]

Maintainer Rou Zhong <zhong_rou@163.com>

Repository CRAN

Date/Publication 2022-12-13 08:20:05 UTC

Contents

GenBinaryFD *Generate binary functional data*

Description

Generate binary functional data through latent process.

Usage

```
GenBinaryFD(n, interval, sparse, regular, meanfun, score, eigfd)
```

Arguments

`n`	An integer denoting the number of sample size.
`interval`	A `vector` of length two denoting the supporting interval.
`sparse`	A `vector` denoting the possible numbers of observation size. The elements are chosen with equal chance. The length of `sparse` must be one if `regular = TRUE`.
`regular`	Logical; If `TRUE`, the observation grids are equally-spaced.
`meanfun`	A function for the mean.
`score`	A n by `npc` `matrix` containing the FPC scores, where `npc` is the number of FPCs.
`eigfd`	A `list` containing functional objects for the eigenfunctions.

Value

A `list` containing the following components:

`Lt`	A `list` of n vectors, where n is the sample size. Each entry contains the observation time in ascending order for each subject.
`Lx`	A `list` of n vectors, where n is the sample size. Each entry contains vales of the latent process of each subject at the observation time correspond to `Lt`.
`Ly`	A `list` of n vectors, where n is the sample size. Each entry contains the binary measurements of each subject at the observation time correspond to `Lt`.

Examples

```
n <- 100
npc <- 2
interval <- c(0, 10)
gridequal <- seq(0, 10, length.out = 51)
basis <- fda::create.bspline.basis(c(0, 10), nbasis = 13, norder = 4,
         breaks = seq(0, 10, length.out = 11))
meanfun <- function(t){2 * sin(pi * t/5)/sqrt(5)}
lambda_1 <- 3^2 #the first eigenvalue
lambda_2 <- 2^2 #the second eigenvalue
score <- cbind(rnorm(n, 0, sqrt(lambda_1)), rnorm(n, 0, sqrt(lambda_2)))
```

```
eigfun <- list()
eigfun[[1]] <- function(t){cos(pi * t/5)/sqrt(5)}
eigfun[[2]] <- function(t){sin(pi * t/5)/sqrt(5)}
eigfd <- list()
for(i in 1:npc){
  eigfd[[i]] <- fda::smooth.basis(gridequal, eigfun[[i]](gridequal), basis)$fd
}
DataNew <- GenBinaryFD(n, interval, sparse = 8:12, regular = FALSE,
           meanfun = meanfun, score, eigfd)
```

SLFPCA *Sparse logistic functional principal component analysis*

Description

Sparse logistic functional principal component analysis (SLFPCA) for binary data. The estimated eigenfunctions from SLFPCA can be strictly zero on some sub-intervals, which is helpful for interpretation.

Usage

```
SLFPCA(
  Ly,
  Lt,
  interval,
  npc,
  L_list,
  norder,
  kappa_theta,
  sparse_pen,
  nRegGrid = 51,
  bwmu_init = 0.5,
  bwcov_init = 1,
  kappa_mu,
  itermax = 100,
  tol = 10
)
```

Arguments

`Ly`	A `list` of *n* vectors, where *n* is the sample size. Each entry contains the binary measurements of each subject at the observation time correspond to `Lt`.
`Lt`	A `list` of *n* vectors, where *n* is the sample size. Each entry contains the observation time in ascending order for each subject.
`interval`	A `vector` of length two denoting the supporting interval.
`npc`	An integer denoting the number of FPCs.

`L_list`	A `vector` denoting the candidates for the number of B-spline basis functions.
`norder`	An integer denoting the order of the using B-spline basis, which is one higher than their degree.
`kappa_theta`	A `vector` denoting the smoothing parameters for eigenfunctions, the optimal tuning parameter is chosen from them.
`sparse_pen`	A `vector` denoting the sparseness parameters for eigenfunctions, the optimal tuning parameter is chosen from them.
`nRegGrid`	An integer denoting the number of equally spaced time points in the supporting interval. The eigenfunctions and mean function are estimated at these equally spaced time points first, before transforming into functional data object. (default: 51)
`bwmu_init`	A scalar denoting the bandwidth for mean function estimation in the setting of initial values. (default: 0.5)
`bwcov_init`	A scalar denoting the bandwidth for covariance function estimation in the setting of initial values. (default: 1)
`kappa_mu`	A `vector` denoting the smoothing parameters for mean function, the optimal tuning parameter is chosen from them.
`itermax`	An integer denoting the maximum number of iterations. (default: 100)
`tol`	A scalar. When difference of the loglikelihood functions between two consecutive iteration is less than `tol`, the convergence is supposed to be reached. (default: 10)

Value

A `list` containing the following components:

`mufd`	A functional data object for the mean function estimate.
`eigfd_list`	A `list` containing `npc` functional data objects, which are the eigenfunction estimates.
`score`	A n by `npc` `matrix` containing the estimates of the FPC scores, where n is the sample size.
`kappa_mu`	A scalar denoting the selected smoothing parameter for mean function.
`kappa_theta`	A scalar denoting the selected smoothing parameter for eigenfunctions.
`sparse_pen`	A scalar denoting the selected sparseness parameter for eigenfunctions.
`L_select`	A scalar denoting the selected number of B-spline basis functions.
`EBICscore`	A `vector` denoting the selected EBIC scores corresponding to different numbers of B-spline basis functions in `L_list`.

References

Rou Zhong, Shishi Liu, Haocheng Li, Jingxiao Zhang. Sparse logistic functional principal component analysis for binary data. Statistics and Computing, 33, 15 (2023). https://doi.org/10.1007/s11222-022-10190-3

Examples

```
#Generate data
n <- 100
npc <- 1
interval <- c(0, 10)
gridequal <- seq(0, 10, length.out = 51)
basis <- fda::create.bspline.basis(c(0, 10), nbasis = 13, norder = 4,
         breaks = seq(0, 10, length.out = 11))
meanfun <- function(t){2 * sin(pi * t/5)/sqrt(5)}
lambda_1 <- 3^2 #the first eigenvalue
score <- cbind(rnorm(n, 0, sqrt(lambda_1)))
eigfun <- list()
eigfun[[1]] <- function(t){cos(pi * t/5)/sqrt(5)}
eigfd <- list()
for(i in 1:npc){
  eigfd[[i]] <- fda::smooth.basis(gridequal, eigfun[[i]](gridequal), basis)$fd
}
DataNew <- GenBinaryFD(n, interval, sparse = 8:12, regular = FALSE,
           meanfun = meanfun, score, eigfd)
SLFPCA_list <- SLFPCA(DataNew$Ly, DataNew$Lt, interval, npc, L_list = 13,
               norder = 4, kappa_theta = 0.2, sparse_pen = 0, kappa_mu = 0)
plot(SLFPCA_list$eigfd_list[[1]])
```

Index