Package 'pfica'

September 18, 2020

Version 0.1.0

Title Penalized Independent Component Analysis for Univariate Functional Data

Author Marc Vidal Badia, Ana Mª Aguilera del Pino

Maintainer Marc Vidal Badia <marc.vidalbadia@ugent.be>

Description
Performs penalized independent component analysis for univariate functional data. Two alternative models are implemented, both based on the fourth order blind identification method.

License GPL (>= 2)

Depends R (>= 2.10), fda

Imports moments

Encoding UTF-8

LazyData true

NeedsCompilation no

Repository CRAN

RoxygenNote 7.1.1

Date/Publication 2020-09-18 12:20:02 UTC

R topics documented:

`ffobi` *Smoothed functional ICA in terms of basis functions coordinates*

Description

This function computes the ordinary ICA procedure from a sample represented by basis functions (Fourier, B-splines...). The estimation method is based on the use of fourth moments (FOBI), in which it is assumed that the independent components have different kurtosis values. The proposed algorithm can be considered an extension of the implementation of the kurtosis operator introduced in Peña et. al (2014), whose decomposition is used to identify cluster structures and outliers.

Usage

```
ffobi(fdx, ncomp = fdx$basis$nbasis, eigenfPar = fdPar(fdx),
      center = FALSE, plotfd = FALSE)
```

Arguments

`fdx`	a functional data object obtained from the **fda** package.
`ncomp`	number of independent components to compute.
`eigenfPar`	a functional parameter object, obtained from the **fda** package, that defines the independent component functions to be estimated.
`center`	a logical value indicating whether the mean function has to be subtracted from each functional observation.
`plotfd`	a logical value indicating whether to plot the eigenfunctions of the FOBI operator.

Details

This IC model for functional data consists in performing the multivariate ICA of a transformation of the coordinate vectors associated to a basis of functions. This algorithm also incorporates a continuous penalty in the orthonormality constraint.

Let $x = (x_i(t) : i = 1, \ldots, n; t \in T)$ be a set of functions in a finite-dimensional space spanned by a basis of functions $\{\phi_1(t), \ldots, \phi_p(t)\}$. Assume that each function of x and an independent factor h can be expanded as

$$x_i(t) = \sum_{j=1}^{p} a_{ij}\phi_j(t), \quad h(t) = \sum_{j=1}^{p} c_j\phi_j(t) = \phi(t)'c.$$

Consider also the matrix $G = (\int_T \phi_j(t)\phi_{j'}(t)dt; j, j' = 1, \ldots, p)$ and the continuous matrix of penalties $G_r = \int_T (\mathcal{D}^r\phi_j(t)) (\mathcal{D}^r\phi_{j'}(t))' dt$, where $\mathcal{D}$ denotes *r*-order difference operator.
Then, the followed ICA procedure can be summarized in the following steps:

1. Standardization of the coordinates: $A_{st} = \mathrm{cov}(A)^{-\frac{1}{2}} (A - \mathrm{E}[A])$, where $A = (a_{ij})_{n\times p}$.

2. Find solutions to the matrix generalized eigenproblem:

$$n^{-1}K^{\bullet}c = \lambda(G + \rho G_r)c$$

with $K^{\bullet}$ a fixed FOBI matrix of $A_{st}G$ and ρ a smoothing parameter.

3. Obtain the matrix of coordinates $C_{p\times p}$, whose columns are the eigenvectors of the FOBI matrix $K^{\bullet}$, such that $A_{st}C$ are the independent components of x.

Note that, by applying Cholesky factorization of the form $LL' = G + \rho G_r$, the generalized eigenproblem in step 2 leads to an eigenproblem of a symmetric FOBI matrix of $A_{st}GL^{-1'}$. Then, the basis coefficients of the independent factors are given by $c_j = L^{-1'}v_j$, where v_j are the eigenvectors of the above FOBI matrix.

Value

a list with the following named entries:

`eigenbasis`	a functional data object for the eigenfunctions or independent factors.
`kurtosis`	a numeric vector giving the kurtosis associated to each independent component vector.
`scores`	a matrix whose column vectors are the independent components.

Author(s)

Marc Vidal, Ana Mª Aguilera

References

Aguilera, AM. and MC. Aguilera-Morillo (2013). "Penalized PCA approaches for B-spline expansions of smooth functional data". In: *Applied Mathematics and Computation* 219(14), pp. 7805–7819.

Peña, C., J. Prieto, and C. Rendón (2014). *Independent components techniques based on kurtosis for functional data analysis*. Working paper 14-10. Universidad Carlos III de Madrid.

Ramsay, J. and B. Silverman (2005). *Functional Data Analysis*. Springer.

Vidal, M. (2020). *Functional Independent Component Analysis in Bioelectrical Signal Processing*. MA thesis. Universidad de Granada.

See Also

`kffobi`

Examples

```
## Canadian Weather data
library(fda)
arg <- 1:365
Temp <- CanadianWeather$dailyAv[,,1]
B <- create.bspline.basis(rangeval=c(min(arg),max(arg)), nbasis=16)
x <- Data2fd(Temp, argvals = arg, B)
```

```
Lfdobj <- int2Lfd(max(0, norder(B)-2))
penf <- fdPar(B, Lfdobj, lambda=10^4)
ica.fd <- ffobi(x, 16, penf)
## Plot by region: classification in order of maximum kurtosis
k1 <- which.max(ica.fd$kurtosis)
k2 <- which.max(ica.fd$kurtosis[c(-k1)])+1
sc <- ica.fd$scores
plot(sc[,c(k1)], sc[,c(k2)], ylab = "", xlab = "")
text(sc[,c(k1)], sc[,c(k2)], CanadianWeather$region, pch=0.5, cex=0.6)
```

---

kd *Kurtosis distance*

Description

This function calculates the kurtosis distance (Vidal, 2020), which is an heuristic measure to select the number of components to be computed in `kffobi` and `pspline.kffobi`.

Usage

```
kd(fdx, hm = fdPar(fdx), rho = NULL, r = 2, centerfd = FALSE, qmin = 2, qmax = 5)
```

Arguments

`fdx`	a functional data object obtained from the **fda** package.
`hm`	a functional parameter object, obtained from the **fda** package, that defines the independent component functions to be estimated in `kffobi`.
`rho`	the smoothing parameter to perform `kd` on `pspline.kffobi`.
`r`	a number indicating the order of the penalty to perform `kd` on `pspline.kffobi`.
`centerfd`	a logical value indicating whether the mean function has to be subtracted from each functional observation.
`qmin`	the minimum allowable q degree.
`qmax`	the maximum allowable q degree.

Details

The kurtosis distance (`KD`) measures the degree of extremeness in an independent component coordinate space of degree q. For a fixed q, the kurtosis distance is defined as

$$\texttt{KD}(z_j) = \max\left[\sum_{j=1}^{q}\text{kurt}(z_j)\right] - \min\left[\sum_{j=1}^{q}\text{kurt}(z_j)\right],$$

where z_j is the vector of independent components calculated from $\int_T x_i^{st}(t)h_j(t)dt$. Thus, `KD` is calculated from the the standardized original sample instead of the standardized principal component expansion. Then, the `kd` function calculates `KD` for different values of $q = 2, \ldots, n-1$. The user can then consider choosing a value from those amongst the first relative maxima of the vector of `KD` values. If the value of q is increased too much, there is a risk of losing accurate estimates, as more of the independent part (noise) of the model is induced.

Value

A vector of KD values.

Author(s)

Marc Vidal

References

Vidal, M. (2020). *Functional Independent Component Analysis in Bioelectrical Signal Processing*. MA thesis. Universidad de Granada.

See Also

kffobi

---

kffobi *Smoothed functional ICA in terms of principal components*

Description

This function computes the ordinary ICA procedure from a penalized principal component expansion (also known as Karhunen-Loève expansion) whose basis, the eigenfunctions of the covariance matrix operator, is expressed in terms of basis functions (Fourier, B-splines...). The estimation method is based on the use of fourth moments (FOBI), in which it is assumed that the independent components have different kurtosis values. The proposed algorithm can be considered an extension of the IC model proposed in Li et al. (2015). This function provides more accurate estimates than ffobi and was used in Vidal (2020) to identify artifactual independent curves in bioelectrical signals.

Usage

```
kffobi(fdx, ncomp = fdx$basis$nbasis, eigenfPar = fdPar(fdx),
       pr = c("fdx", "fdx.st", "KL", "KL.st"),
       center = FALSE, plotfd = FALSE)
```

Arguments

fdx	a functional data object obtained from the **fda** package.
ncomp	number of independent components to compute.
eigenfPar	a functional parameter object, obtained from the **fda** package, that defines the principal component functions to be estimated.
pr	the functional data object to project into the space spanned by the eigenfunctions of the FOBI operator. To compute the independent components, the usual procedure is to use KL.st, the standardized principal component expansion. Thus, if pr is not supplied, KL.st is used.

`center`	a logical value indicating whether the mean function has to be subtracted from each functional observation.
`plotfd`	a logical value indicating whether to plot the eigenfunctions of the FOBI operator.

Details

Note that `kffobi` first computes the (penalized) functional PCA; see Aguilera and Aguilera-Morillo (2013) for a detailed discussion. Thus here, the IC model for functional data consists in performing the multivariate ICA of a transformation of the principal component coordinate vectors.

Let $x = (x_i(t): i = 1, \ldots, n; t \in T)$ be a set of functions in a finite-dimensional space spanned by a basis of functions $\{\phi_1(t), \ldots, \phi_p(t)\}$. Assume that the principal component expansion of x is given by

$$x_i^{(q)}(t) = \sum_{j=1}^{q} z_{ij} f_j(t)$$

where $(z_{ij})_{n \times q} = Z$ is the matrix of principal components and f_j the principal factors of x. Notice that f_j can be expressed in terms of basis functions, i.e. $f_j(t) = \phi(t)' b_j$, where the coordinates b_j are computed from the FPCA of x. The ICA procedure consists of the same steps as the ones described in `ffobi`, but replacing the matrix A by Z. As the orthogonalization of the basis functions is implemented in the FPCA algorithm, now $\int_T f_j(t) f_{j'}(t) dt = I_q; \forall j, j' = 1, \ldots, q.$, where I_q denotes the identity matrix. As such, the ICA procedure from a principal component expansion is easy to compute.

Value

a list with the following named entries:

`eigenbasis`	a functional data object for the eigenfunctions or independent factors.
`kurtosis`	a numeric vector giving the kurtosis associated to each independent component vector.
`scores`	a matrix whose column vectors are the independent components.

Author(s)

Marc Vidal, Ana Mª Aguilera

References

Aguilera, AM. and MC. Aguilera-Morillo (2013). "Penalized PCA approaches for B-spline expansions of smooth functional data". In: *Applied Mathematics and Computation* 219(14), pp. 7805–7819.

Li, B., G. Van Bever, H. Oja, R. Sabolová, and F. Critchley (2015). "Functional independent component analysis: an extension of the fourth-orderblind identification." *Submitted*.

Miettinen, J., K. Nordhausen, and S. Taskinen (2017). "Blind source separation based on joint diagonalization in R: The packages JADE and BSSasymp". In: *Journal of Statistical Software* 76.2, pp. 1–31.

Ramsay, J. and B. Silverman (2005). *Functional Data Analysis*. Springer.

Vidal, M. (2020). *Functional Independent Component Analysis in Bioelectrical Signal Processing*. MA thesis. Universidad de Granada.

See Also

kd, ffobi

Examples

```
## foetal_ecg data
library(fda)
dataset <- matrix(
 scan("https://www.jstatsoft.org/index.php/jss/article/downloadSuppFile/v076i02/foetal_ecg.dat"),
  2500, 9, byrow = TRUE);
X <- dataset[1:1000, 2:9]
arg <- 1:1000
basis <- create.fourier.basis(rangeval=c(min(arg), max(arg)), nbasis=301, basisvalues=TRUE)
x <- Data2fd(X, argvals=arg, basis)
## Penalization can be considered:
#Lfdobj <- vec2Lfd(c(0,(2*pi/diff(4))^2,0), 4)
#hm <- fdPar(base, Lfdobj, lambda=2)
## Select the number of components with the kurtosis distance:
#kurt.dist <- kd(x, qmax = 8)
aci <- kffobi(x, 7, plotfd = TRUE)
```

---

pspline.kffobi *P-Spline smoothed functional ICA*

Description

This function provides an alternative form of computing the smoothed functional ICA in terms of principal components (function kffobi), using a discrete penalty that measures the roughness of principal factors by summing squared r-order differences between adjacent B-spline coefficients (P-spline penalty); see Aguilera and Aguilera-Morillo (2013) for a detailed discussion.

Usage

```
pspline.kffobi(fdx, ncomp = fdx$basis$nbasis, rho = 0, r = 2,
               pr = c("fdx", "fdx.st", "KL", "KL.st"),
               center = FALSE, plotfd = FALSE)
```

Arguments

fdx	a functional data object obtained from the **fda** package.
ncomp	number of independent components to compute.
rho	the smoothing parameter. It can be estimated by *leave-one-out* cross-validation.
r	a number indicating the order of the penalty.

`pr`	the functional data object to project into the space spanned by the eigenfunctions of the FOBI operator. To compute the independent components, the usual procedure is to use `KL.st`, the standardized principal component expansion. Thus, if `pr` is not supplied, then `KL.st` is used.
`center`	a logical value indicating whether the mean function has to be subtracted from each functional observation.
`plotfd`	a logical value indicating whether to plot the eigenfunctions of the FOBI operator.

Details

The smooth FPCA can be summarized in the following generalized matrix eigenproblem:

$$n^{-1}Vc = \lambda(G + \rho G_r)c,$$

with V as the covariance matrix of A; see *Details* in `kffobi`. Then, the matrix of penalties G_r is defined as $G_r = (\Delta^r)' \Delta^r$ with Δ^r as the matrix representation of the r-order difference operator. For $r = 2$ this matrix is given by

$$\Delta^2 = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots \\ 0 & 1 & -2 & 1 & \cdots \\ 0 & 0 & 1 & -2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

In **R**, this can be translated into the following code: $\Delta^2 =$ `diff(diag(nknots + 2), differences = 2)`, where `nknots` is the number of basis knots.

Once the P-spline approximation of eigenfunctions has been computed and the smooth principal component expansion of P-splines is obtained, then the ICA procedure is conducted. As in `kffobi`, the functional ICA of the principal component expansion is equivalent to the ICA of the matrix Z; see *Details* in `kffobi`.

Value

a list with the following named entries:

`eigenbasis`	a functional data object for the eigenfunctions or independent factors.
`kurtosis`	a numeric vector giving the kurtosis associated to each independent component vector.
`scores`	a matrix whose column vectors are the independent components.

Author(s)

Marc Vidal, Ana Mª Aguilera

References

Aguilera, AM. and MC. Aguilera-Morillo (2013). "Penalized PCA approaches for B-spline expansions of smooth functional data". In: *Applied Mathematics and Computation* 219(14), pp. 7805–7819.

Ramsay, J. and B. Silverman (2005). *Functional Data Analysis*. Springer.

Vidal, M. (2020). *Functional Independent Component Analysis in Bioelectrical Signal Processing*. MA thesis. Universidad de Granada.

See Also

kffobi

Examples

```
## Canadian Weather data
library(fda)
arg <- 1:365
Temp <- CanadianWeather$dailyAv[,,1]
B <- create.bspline.basis(rangeval=c(min(arg),max(arg)), nbasis=16)
x <- Data2fd(Temp, argvals = arg, B)
ica.fd <- pspline.kffobi(x, 16, rho = 10)
## Plot by region: classification in order of maximum kurtosis
k1 <- which.max(ica.fd$kurtosis)
k2 <- which.max(ica.fd$kurtosis[c(-k1)])+1
sc <- ica.fd$scores
plot(sc[,c(k1)], sc[,c(k2)], ylab = "", xlab = "")
text(sc[,c(k1)], sc[,c(k2)], CanadianWeather$region, pch=0.5, cex=0.6)
```

Index