Package 'pfica'

June 3, 2021

Version 0.1.2

Title Independent Component Analysis for Univariate Functional Data

Author Marc Vidal, Ana Mª Aguilera

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

Description Performs penalized independent component analysis for univariate functional data [<doi:10.3390/math9111243>].

License GPL (>= 2)

Depends R (>= 2.10), fda

Imports corpcor, expm, moments

URL https://github.com/m-vidal/pfica

Encoding UTF-8

NeedsCompilation no

Repository CRAN

RoxygenNote 7.1.1

Date/Publication 2021-06-03 09:50:02 UTC

R topics documented:

pfica-package *Functional independent component analysis*

Description

pfica:: Performs penalized (and non-penalized) independent component analysis for univariate functional data. Two alternative versions are implemented, both based on the spectral decomposition of the kurtosis operator. Our methods are interfaced with the basis systems provided in the **fda** package.

Details

This package contains a set of tools for performing functional independent component analysis (FICA) by analyzing the kurtosis structure of whitened data. Two FICA versions are considered depending on the basis of expansion: `ffobi` computes the independent components from a data representation in a canonical basis of functions, while `kffobi` uses the eigenfunctions of the covariance operator (in terms of basis functions). The application of penalties differs in both algorithms. The former introduces a penalty in the eigenfunctions of the kurtosis operator of a standardized sample; the latter in the eigenfunctions of the covariance operator for a subsequent standardization of the principal component expansion. This algorithm is also extended using a discrete penalty (P-spline) in `pspline.kffobi`, with this function being computationally more efficient. The current FICA routines use Mahalanobis kernel whitening and shrinkage covariance estimators to improve the outcomes in the estimation process.

Author(s)

Marc Vidal <marc.vidalbadia@ugent.be>, Ana Mª Aguilera <aaguiler@ugr.es>

References

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

Schafer, J. and K. Strimmer (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statistical Applications in Genetics and Molecular Biology, 4.32(1).

Vidal, M., M. Rosso and AM. Aguilera. (2021). Bi-Smoothed Functional Independent Component Analysis for EEG Artifact Removal. Mathematics 9(11) 1243.

See Also

Useful links:

- https://github.com/m-vidal/pfica

bcv *Compute baseline cross-validation criterion (P-spline version)*

Description

The generalized cross-validation method is adapted for B-spline representations with large number of knots to select the penalty parameter in `pspline.kffobi`. This function computes the log cross-validation score for a given penalty parameter.

Usage

```
bcv(a, phi, G, r = 2, ncomp, baseline, value = 0.1)
```

Arguments

`a`	the coefficient matrix of a functional data object obtained from the **fda** package.
`phi`	a functional basis object associated to the coefficient matrix `a`.
`G`	the Gramian matrix of inner products between pairs of basis elements in `phi`.
`r`	a number indicating the order of the penalty.
`ncomp`	number of principal components to compute.
`baseline`	a non-negative value (the penalty parameter).
`value`	a number that slightly increases the `baseline` penalty parameter. This value may automatically change depending on the range of `baseline`. The default is set to 0.1.

Value

The baseline log cross-validation score.

Author(s)

Marc Vidal, Ana Mª Aguilera

References

Vidal, M., Rosso, M. and Aguilera, A. M. (2021). Bi-Smoothed Functional Independent Component Analysis for EEG Artifact Removal. Mathematics 9(11) 1243.

See Also

`pspline.kffobi`

`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 (kurtosis), 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),
      pr = c("fdx", "fdx.st"), shrinkage = FALSE,
      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.
`pr`	the functional data object to project into the space spanned by the eigenfunctions of the kurtosis operator. To compute the independent components, the usual procedure is to use `fdx.st`, the standardized basis expansion. Thus, if `pr` is not supplied, `fdx.st` is used.
`shrinkage`	uses shrinkage estimators to compute the covariance matrix of the coordinate vectors.
`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 kurtosis operator.

Details

This functional ICA consists of performing the multivariate ICA of a transformation of the coordinate vectors associated with a basis of functions. The algorithm also incorporates a continuous penalty in the orthonormality constraint of the kurtosis eigenfunctions.

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 proportion of variance kurtosis explained by each eigenfunction.
scores	a matrix whose column vectors are the independent components.

Author(s)

Marc Vidal, Ana Mª Aguilera

References

Peña, C., J. Prieto and C. Rendón (2014). *Independent components techniques based on kurtosis for functional data analysis*. Working paper 14–10 Statistics and Econometric Series (06); Universidad Carlos III de Madrid, Madrid, 2014.

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 in order of maximum kurtosis (outliers)
sc <- ica.fd$scores
plot(sc[,1], sc[,2], ylab = "", xlab = "")
text(sc[,1], sc[,2], CanadianWeather$region, pch=0.5, cex=0.6)
```

---

kd *Kurtosis distance*

Description

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

Usage

```
kd(fdx, hm = fdPar(fdx), pp = NULL, r = 2,
   pr = c("fdx", "fdx.st", "KL", "KL.st"),
   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.
pp	the penalty parameter to perform kd on pspline.kffobi.
r	a number indicating the order of the penalty to perform kd on pspline.kffobi.
pr	the functional data object to project into the space spanned by the eigenfunctions of the kurtosis 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.
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 measures the degree of extremeness in a component space by computing the excess kurtosis on each score vector and the distance between the most extreme kurtosis values using the Frobenius norm.

Value

A vector of kurtosis distance 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-Loeve expansion) whose eigenbasis is expressed in terms of basis functions (Fourier, B-splines...). The estimation method is based on the use of fourth moments (kurtosis), in which it is assumed that the independent components have different kurtosis values. The proposed algorithm can be considered an extension of the functional ICA proposed in Li et al. (2019).

Usage

```
kffobi(fdx, ncomp = fdx$basis$nbasis, eigenfPar = fdPar(fdx),
       pr = c("fdx", "fdx.st", "KL", "KL.st"),
       shrinkage = FALSE, 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 kurtosis 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.
`shrinkage`	uses shrinkage estimators to compute the covariance matrix of the coordinate vectors.
`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 kurtosis operator.

Details

Note that `kffobi` first computes the (penalized) functional PCA; see Aguilera and Aguilera-Morillo (2013) for a detailed discussion. Thus, this functional ICA consists of performing the multivariate ICA of the PC coordinate vectors in terms of the PC weight functions.

Value

a list with the following named entries:

`PCA.eigv`	a numeric vector giving the eigenvalues of the covariance operator.
`PCA.basis`	a functional data object for the eigenfunctions of the covariance operator.
`PCA.scores`	a matrix whose column vectors are the principal components.
`ICA.eigv`	a numeric vector giving the eigenvalues of the kurtosis operator.
`ICA.basis`	a functional data object for the eigenfunctions of the covariance operator.
`ICA.scores`	a matrix whose column vectors are the projection coefficients for `fdx`,`fdx.st`,`KL` or `KL.st`.
`ICA.kurtosis`	a numeric vector giving the kurtosis of each component vector.

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. *Applied Mathematics and Computation* 219(14), pp. 7805–7819.

Li, B., G. Van Bever, H. Oja, R. Sabolova and F. Critchley (2019). *Functional independent component analysis: an extension of the fourth-orderblind identification*. Technical Report, Univ. Namur.

Miettinen, J., K. Nordhausen and S. Taskinen (2017). Blind source separation based on joint diagonalization in R: The packages JADE and BSSasymp. *Journal of Statistical Software* 76(2), pp. 1–31.

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(basis, 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). A discrete penalty that measures the roughness of principal factors by summing squared *r*-order differences between adjacent B-spline coefficients (P-spline penalty) is used; see Aguilera and Aguilera-Morillo (2013) for a detailed discussion.

Usage

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

Arguments

fdx	a functional data object obtained from the **fda** package.
ncomp	number of independent components to compute.
pp	the penalty parameter. It can be estimated by *leave-one-out* cross-validation or *baseline* cross-validation (see bcv).
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 kurtosis 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.
shrinkage	uses shrinkage estimators to compute the covariance matrix of the coordinate vectors.
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 kurtosis operator.

Details

To compute the penalty matrix, the following code is used: $\Delta^2 = \texttt{diff}(\texttt{diag}(\texttt{nknots} + 2), \texttt{differences} = 2)$, where nknots is the number of basis knots. As in kffobi, the functional ICA of the principal component expansion is equivalent to the multivariate ICA of the principal coordinate vectors; see *Details* in kffobi.

Value

a list with the following named entries:

`PCA.eigv`	a numeric vector giving the eigenvalues of the covariance operator.
`PCA.basis`	a functional data object for the eigenfunctions of the covariance operator.
`PCA.scores`	a matrix whose column vectors are the principal components.
`ICA.eigv`	a numeric vector giving the eigenvalues of the kurtosis operator.
`ICA.basis`	a functional data object for the eigenfunctions of the covariance operator.
`ICA.scores`	a matrix whose column vectors are the projection coefficients for `fdx`,`fdx.st`,`KL` or `KL.st`.
`ICA.kurtosis`	a numeric vector giving the kurtosis of each component vector.

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.

Vidal, M., M. Rosso and AM. Aguilera. (2021). Bi-Smoothed Functional Independent Component Analysis for EEG Artifact Removal. Mathematics 9(11) 1243.

See Also

`kffobi`

Examples

```
## Canadian Weather data
library(fda)
arg <- 1:365
Temp <- CanadianWeather$dailyAv[,,1]
B <- create.bspline.basis(rangeval=c(min(arg),max(arg)), nbasis=16)
x <- Data2fd(Temp, argvals = arg, B)
ica.fd <- pspline.kffobi(x, 16, pp = 10)
## Plot by region in order of maximum kurtosis (outliers)
sc <- ica.fd$ICA.scores
plot(sc[,1], sc[,2], ylab = "", xlab = "")
text(sc[,1], sc[,2], CanadianWeather$region, pch=0.5, cex=0.6)
```

Index