Package 'MFSD'

March 31, 2026

Type Package

Title Multivariate Functional Spatial Data

Version 0.1.0

Description Analysis of multivariate functional spatial data,
including spectral multivariate functional principal
component analysis and related statistical procedures
(Si-Ahmed, Idris, et al. ``Principal component analysis of multivariate spatial functional data." Big Data Research 39 (2025) 100504).
(Kuenzer, T., Hörmann, S., & Kokoszka, P. (2021). ``Principal component analysis of spatially indexed functions." Journal of the American Statistical Association, 116(535), 1444-1456.)
(Happ, C., & Greven, S. (2018). ``Multivariate functional principal component analysis for data observed on different (dimensional) domains." Journal of the American Statistical Association, 113(522), 649-659.)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports fda, graphics, mvtnorm, Rcpp (>= 1.0.0)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author SI-AHMED IDRIS [aut, cre],
T. Kuenzer [ctb] (Code adapted from the fsd package)

Maintainer SI-AHMED IDRIS <i_siahmed@esi.dz>

Repository CRAN

Date/Publication 2026-03-31 10:00:18 UTC

Contents

`*.fsd.fd` *Arithmetic for functional spatial data objects*

Description

Works as one would expect.

Usage

```
## S3 method for class 'fsd.fd'
scalar * fsdobj
```

Arguments

`scalar`	a scalar number
`fsdobj`	a functional spatial data object

Value

An object of class `fsd.fd`. The result corresponds to a scalar multiplication of the functional spatial data object, where all coefficients are multiplied by the given scalar.

`+.fsd.fd` *Arithmetic for functional spatial data objects*

Description

Works as one would expect.

Usage

```
## S3 method for class 'fsd.fd'
fsd1 + fsd2
```

Arguments

`fsd1`, `fsd2`	functional spatial data objects

Value

An object of class `fsd.fd`. The result corresponds to the element-wise sum of the coefficient arrays of the two functional spatial data objects. The spatial structure and basis are preserved.

`-.fsd.fd` *Arithmetic for functional spatial data objects*

Description

Works as one would expect.

Usage

```
## S3 method for class 'fsd.fd'
fsd1 - fsd2 = NULL
```

Arguments

`fsd1`, `fsd2`	functional spatial data objects

Value

An object of class `fsd.fd`. If a single argument is provided, the result is the additive inverse of the input object. Otherwise, it corresponds to the element-wise difference between two functional spatial data objects.

`/.fsd.fd` *Arithmetic for functional spatial data objects*

Description

Works as one would expect.

Usage

```
## S3 method for class 'fsd.fd'
fsdobj / scalar
```

Arguments

`fsdobj`	a functional spatial data object
`scalar`	a scalar number

Value

An object of class `fsd.fd`. The result corresponds to a scalar division of the functional spatial data object, where all coefficients are divided by the given scalar.

as.fd.fsd.fd *Converting a functional spatial data object into a functional data object*

Description

Works as one would expect

Usage

```
## S3 method for class 'fsd.fd'
as.fd(x, ...)
```

Arguments

x	a functional spatial data object
...	currently unused

Value

An object of class `fd` (from the **fda** package). The returned object corresponds to a functional data object obtained by collapsing the spatial dimensions of the input. The coefficients matrix represents functional observations, while the spatial structure is removed (or flattened).

See Also

`fsd.fd`

as.fsd.fd *Converting a functional data object into a functional spatial data object*

Description

Works as one would expect

Usage

```
as.fsd.fd(x, ...)
```

Arguments

x	a functional data object or something that can be converted into one.
...	currently unused

Value

An object of class `fsd.fd`. The returned object represents functional data indexed over a spatial grid. It contains: `coefs`: an array of coefficients where the first dimension corresponds to basis coefficients and the remaining dimensions correspond to spatial locations, `basis`: a basis object describing the functional representation, `fdnames`: a list describing the axes (spatial indices and function domain). This transformation embeds classical functional data into a spatial framework.

See Also

`fsd.fd`

`center.fsd.fd` *Compute the centered version of spatial functional data*

Description

This function is used to center spatial functional data.

Usage

```
center.fsd.fd(X, margins = NULL, na.rm = FALSE)
```

Arguments

`X`	The functional spatial data.
`margins`	The margins for which the centering should be done separately in every entry, in order to remove a trend.
`na.rm`	Whether or not missing values should be ignored.

Value

The centered data.

See Also

`mean.fsd.fd`

Examples

```
data("temp")

T.centered = center.fsd.fd(temp)
plot(mean(T.centered, margins = 1:2))

T.centered2 = center.fsd.fd(temp, margins = 1:2)
plot(mean(T.centered2, margins = 1:2))
plot(mean(temp, margins = 1:2))
```

diff.fsd.fd *Calculate the differences for a Functional Spatial Data Object*

Description

This function is used to calculate the differences of functional data on a spatial grid.

Usage

```
## S3 method for class 'fsd.fd'
diff(x, lag = 1, differences = 1, ..., dim = 1)
```

Arguments

x	the functional spatial data.
lag	the lag.
differences	the order of the differences.
...	currently unused
dim	the dimension with respect to which the differences are to be computed.

Value

An object of class ‘"fsd.fd"‘ representing the functional difference between the input functional spatial data objects.

See Also

fsd.fd

Examples

```
data("temp")
plot(diff(temp, dim = 1))
```

fsd.covariance *Estimate autocovariance operators*

Description

This function is used to estimate a covariance operator of stationary functional spatial data with respect to some lag h.

Usage

```
fsd.covariance(Y = NULL, h, centered = FALSE, na.rm = FALSE, unbiased = FALSE)
```

Arguments

Y	the functional spatial data. Either an fd object or an array.
h	the spatial lag with respect to which the autocovariance is to be estimated.
centered	a boolean indicating whether or not the data is already centered.
na.rm	whether or not missing values should be ignored.
unbiased	whether or not to scale the estimator to be unbiased. FALSE by default. Is not used if na.rm is TRUE.

Value

the covariance matrix.

---

fsd.fd *Creating a functional spatial data object*

Description

This function extends the possibilities of the function fd found in the package fda to include functional data on a spatial grid of arbitrary dimension r.

Usage

```
fsd.fd(coef = NULL, basisobj = NULL, fdnames = NULL)
```

Arguments

coef	an array of arbitrary dimension 1 + r, where the first dimension corresponds to basis functions.
basisobj	a functional basis object defining the basis.
fdnames	a list of length r with each member containing the labels for the dimensions of the data.

Value

the fsd.fd object.

fsd.filter *Creating a functional spatial filter*

Description

This function creates a functional spatial filter

Usage

```
fsd.filter(laglist, operatorlist, basisobj = NULL)
```

Arguments

laglist	a list of lags.
operatorlist	a list of operators, i.e. matrices.
basisobj	a functional basis object defining the basis.

Value

the fsd.filter object.

fsd.filter.is.unilateral *Check if functional spatial filter is unilateral*

Description

This function is used to verify if a functional spatial filter is unilateral.

Usage

```
fsd.filter.is.unilateral(A, exclude.origin = FALSE)
```

Arguments

A	the functional spatial filter.
exclude.origin	whether to exclude the origin from the definition of unilaterality.

Value

a Boolean indicating whether A is unilateral or not.

See Also

fsd.filter

fsd.fourier *Compute the Fourier transform*

Description

This function is used to calculate the Fourier transform

Usage

```
fsd.fourier(covs, laglist, thlist)
```

Arguments

covs	the covariances to be transformed
laglist	list of lags.
thlist	list of frequencies.

Value

the Fourier transform of the covariances.

fsd.fourier.inverse *Compute the Fourier inverse*

Description

This function is used to calculate the Fourier inverse.

Usage

```
fsd.fourier.inverse(evecs, thlist, laglist)
```

Arguments

evecs	the eigenvectors to be expanded.
thlist	list of frequencies.
laglist	list of lags.

Value

the filter obtained by applying the Fourier inverse on evecs.

`fsd.jb.test` *Perform a Jarque-Bera Test on Normality of Functional Spatial Data*

Description

This function performs a test on normality of some functional spatial data.

Usage

```
fsd.jb.test(X.spca, Npc = NULL, L = NULL, var.method = "integral")
```

Arguments

`X.spca`	a spectral principal components analysis as performed by `fsd.spca`, i.e. a list that in particular contains the estimated SPC scores and the estimated spectral density.
`Npc`	the number of spectral principal components to use for the test.
`L`	the maximum lag to compute the covariance of the scores during the computation of the autocovariances.
`var.method`	the method that is used to calculate the long-run variance of the SFPC scores. Either "direct" or "integral".

Details

To ensure accuracy of numerical integration during the Fourier transform, the frequencies of `FS` should be a suitably dense grid.

Value

A list with components

`T4`	the test statistic.
`p.value`	the p-value.
`df`	the degrees of freedom.
`T4.vector`	the vector of the test statistics for each single SPC.

See Also

`fsd.spca`, `fsd.spca.cov`

fsd.norm *Compute the norm of functional data*

Description

This function is used to compute the norm of functional data.

Usage

```
fsd.norm(X)
```

Arguments

X	the functional spatial data.

Value

an array of the norms.

fsd.perm *Permute the dimensions of a Functional Spatial Data Grid*

Description

This function is used to permute the order of the dimensions of the spatial grid of some functional spatial data.

Usage

```
fsd.perm(fsdobj, perm = NULL)
```

Arguments

fsdobj	the functional spatial data.
perm	the subscript permutation vector, i.e. a permutation of the integers 1:r.

Value

A numeric vector containing the permutation indices used to reorder the functional spatial data.

See Also

fsd.fd

Examples

```
data("temp")
fsd.plot.data(fsd.perm(temp, c(2, 1, 3)))
```

`fsd.plot.covariance` *Plot the autocovariance operator of functional spatial data*

Description

This function is used to plot the autocovariance operator of functional spatial data along a two-dimensional plane of lags.

Usage

```
fsd.plot.covariance(
  X,
  basisobj = NULL,
  q = 0,
  gridsize = NULL,
  set.dims = NULL,
  plot.cor = FALSE,
  na.rm = FALSE,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

`X`	the functional spatial data. Either an fd object or an array.
`basisobj`	the basis of the functional data.
`q`	the maximal lag to be plotted.
`gridsize`	the resolution of the grid on which the kernel will be evaluated.
`set.dims`	the coordinates to be fixed. A vector of dimension r. NaN represents not fixed components.
`plot.cor`	a boolean indicating whether the correlation or the covariance should be plotted.
`na.rm`	whether or not missing values should be ignored.
`main`	title of the plot.
`xlab`	label for the x-axis.
`ylab`	label for the y-axis.
`...`	other arguments to pass to the plot function

Value

No return value. The function computes the covariance or correlation operator of the functional spatial data

See Also

fsd.plot.data

Examples

```
data("temp")
fsd.plot.covariance(temp, q = 3, plot.cor = TRUE, zlim = c(-1,1))
```

`fsd.plot.data` *Plot functional spatial data*

Description

This function is used to plot functional spatial data on a grid of dimension r, along a two-dimensional plane of indices.

Usage

```
fsd.plot.data(
  X,
  basisobj = NULL,
  gridsize = NULL,
  set.dims = NULL,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`X`	the functional spatial data. Either an fd object or an array, or a list thereof.
`basisobj`	the basis of the functional data.
`gridsize`	the number of grid points on which to evaluate the functional data.
`set.dims`	the coordinates to be fixed. A vector of dimension r. NaN represents not fixed components.
`xlim`	limits for the coordinates on the x-axis.
`ylim`	limits for the coordinates on the y-axis.
`main`	title of the plot.
`xlab`	label for the x-axis.
`ylab`	label for the y-axis.

Value

No return value. This function produces a graphical representation of functional spatial data.

See Also

plot.fsd.fd, fsd.plot.filters, fsd.plot.covariance

Examples

```
data("temp")
fsd.plot.data(temp, set.dims = c(NaN, NaN, 3))
```

`fsd.plot.filters` *Plot functional spatial filters*

Description

This function is used to plot functional spatial filters, along a two-dimensional plane of indices.

Usage

```
fsd.plot.filters(
  A,
  basisobj = NULL,
  gridsize = NULL,
  Npc = 3,
  Lmax = 3,
  set.dims = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

A	the filters.
basisobj	the basis of the functional data.
gridsize	the number of grid points on which to evaluate the functional data.
Npc	number of filters to plot.
Lmax	maximum lag to plot.
set.dims	the coordinates to be fixed. A vector of dimension r. NaN represents not fixed components.
main	title of the plot.
xlab	label for the x-axis.
ylab	label for the y-axis.

Value

No return value. This function produces a graphical representation of functional spatial filters.

`fsd.sfarma` *Simulate a spatial functional ARMA process*

Description

This function is used to simulate a sample from a spatial functional ARMA (SFARMA) process using a normal or t-distributed noise process.

Usage

```
fsd.sfarma(
  n,
  Sigma = NULL,
  ARfilter = NULL,
  MAfilter = NULL,
  burnin = 30,
  basisobj = NULL,
  noise = "normal",
  do.fixed.point = FALSE,
  max.iter = 50,
  eps = 1e-05
)
```

Arguments

`n`	the sample size.
`Sigma`	the covariance matrix of the noise.
`ARfilter`	the functional spatial filter of the autoregressive part.
`MAfilter`	the functional spatial filter of the moving-average part.
`burnin`	the number of grid points to add on each side of the sample to achieve stationarity.
`basisobj`	the basis of the functional data.
`noise`	the noise distribution to use. Either "normal" or an integer for the degrees of freedom of a multivariate Student's t-distribution.
`do.fixed.point`	whether to always use a fixed point iteration.
`max.iter`	the maximum number of iterations in the fixed point iteration.
`eps`	the stopping criterion for the fixed point iteration.

Details

Please note that convergence in the fixed point iteration depends strongly on the sample size and the structure of the AR filter. For certain choices of parameters, even more than `n` iterations may be needed.

Value

one sample of the SFARMA process.

See Also

fsd.fd

`fsd.spca` *Perform Spectral Principal Components Analysis on Spatial Functional Data*

Description

This function performs spectral PCA on functional spatial data on a grid of dimension r.

Usage

```
fsd.spca(
  X,
  freq.res = 100,
  Npc = 3,
  L = 3,
  q = NULL,
  na.ignore = TRUE,
  return.F = FALSE,
  only.filters = FALSE
)
```

Arguments

`X`	the functional spatial data. Either an fd object or an array.
`freq.res`	the resolution for the computation of the spectral density.
`Npc`	the number of principal components to be computed.
`L`	the maximum lag for the filters. An integer or vector of integers.
`q`	a tuning parameter for the estimation of the the spectral density operator. An integer or vector of integers.
`na.ignore`	whether to ignore missing data points in the computation of the scores.
`return.F`	a boolean indicating whether to return the spectral density.
`only.filters`	a boolean indicating whether to compute only the filters, leaving out the scores.

Details

This function can be used to compute spectral PCA. By setting `q = 0`, it can also be used to perform static PCA.

Setting a higher value for `freq.res` increases the runtime significantly, but yields a more accurate result because of reduced integration errors.

Value

A list with components

F	the spectral density.
tuning.params	a list of the used tuning parameters freq.res, q and Lmax.
filters	the SPC filters.
scores	the SPC scores.
var	the theoretical fractions of variance explained by each PC.
X.mean	the mean of X.

See Also

fsd.spca.inverse, fsd.spectral.density

---

fsd.spca.cov *Calculate the Covariance of the Spectral Principal Component Scores*

Description

This function estimates the (theoretical) autocovariance for each spectral principal component score for some spatial functional data.

Usage

```
fsd.spca.cov(FS, L = 3)
```

Arguments

FS	the spectral density.
L	the maximum lag to compute the covariance of the scores

Details

To ensure accuracy of numerical integration during the Fourier transform, the frequencies of F should be a suitably dense grid.

Value

A list with components

laglist	the list of lags.
cov	a matrix with the autocovariances of the principal component scores.

See Also

fsd.spca, fsd.spca.var

fsd.spca.filters — *Calculate the Spectral Principal Components Filters Spatial Functional Data*

Description

This function calculates the SPC filters from a given spectral density operator.

Usage

```
fsd.spca.filters(FS, Npc = 1, L = 3)
```

Arguments

FS	the spectral density.
Npc	the number of principal components to be computed.
L	the maximum lag for the filters, as an integer or vector of integers.

Details

The eigenvectors used for the calculation of the Fourier expansion are oriented such that the sum of their coordinates over the basis of the fd object (i.e. the entries in the coefs array) is a non-negative real number. If the basis is not orthonormal, this is done with respect to a virtual orthonormal basis in the background.

Value

the SPC filters.

See Also

fsd.spca

fsd.spca.inverse — *Reconstruct the original functional data from the filters and the score*

Description

This function is used to reconstruct the original functional data from the filters and the score. It applies the transposed matrices of the filter A to the scores.

Usage

```
fsd.spca.inverse(A, scores, mean.X = NULL)
```

Arguments

A	the filters.
scores	the scores as an array.
mean.X	the mean of X.

Value

X the reconstructed functional data

See Also

fsd.spca, fsd.spca.scores

---

fsd.spca.scores *Compute the Scores for Spatial Functional Data*

Description

This function computes the scores. It applies the matrices of the filter A to the data X.

Usage

```
fsd.spca.scores(X, A, na.ignore = FALSE)
```

Arguments

X	the functional spatial data. Either an fd object or an array.
A	the filter used to compute the scores.
na.ignore	whether missing values in the data X should be ignored.

Value

the array of the scores.

See Also

fsd.spca.inverse

fsd.spca.var *Calculate the Variance explained by Spectral Principal Components*

Description

This function estimates the (theoretical) fraction of the variance explained by each spectral principal component for some spatial functional data.

Usage

```
fsd.spca.var(FS)
```

Arguments

FS	the spectral density.

Details

To ensure accuracy of numerical integration, the frequencies of FS should be a suitably dense grid. In order for the estimations to conform with the actual variance explained (1 - NMSE), the tuning parameters in the estimation of the spectral density need to be chosen carefully.

Value

the fraction of the variance explained by each PC.

See Also

fsd.spca

fsd.spectral.density *Estimate the spectral density operator*

Description

This function is used to estimate the spectral density operator of stationary functional spatial data on a spatial frequency grid.

Usage

```
fsd.spectral.density(X, freq, q = NULL, na.rm = FALSE)
```

Arguments

X	the functional spatial data. Either an fd object or an array.
freq	the spatial frequencies with respect to which the spectral density is to be estimated. A vector or a list of vectors defining the grid.
q	the size of the window of the kernel used for estimation. An integer or a vector. For any dimension, the value q = 1 is equivalent to q = 0 and means that no lags in this direction are taken into account.
na.rm	whether or not missing values should be ignored.

Details

The spectral density is estimated employing a Bartlett kernel on the Euclidean norm of the lags. Therefore, the lag q is the smallest lag not to be taken into consideration.

Value

the spectral density, i.e. a list consisting of

operators	a list of the the spectral density operator at different spatial frequencies.
freq	a list of the spatial frequencies.
basis	the basis object of the functional data.
estimation.q	the tuning parameter q used in the estimation process.

---

fsd.z.plot *Plot functional data on a grid*

Description

This function plots functional data on a grid.

Usage

```
fsd.z.plot(Zlist, positions, basisobj = NULL, gridsize = 50, ...)
```

Arguments

Zlist	list of arrays with coefficients.
positions	the positions of the single curves in space.
basisobj	the basis of the functional data.
gridsize	the number of grid points on which to evaluate the curves.
...	other arguments to pass to the plot function

Value

No return value. This function is called for its side effect of producing a graphical representation of functional data evaluated on a spatial grid.

Llist *a list containing the maximum lag for the filters for each variable.*

Description

an example taken from the variable indian which represents a list containing the maximum lag for the filters for each variable. An integer or vector of #' integers.

Usage

```
data(Llist)
```

Format

Llist has a dimension of [1:22]

An object of class `int [1:22]`.

Examples

```
data(Llist)
```

mean.fsd.fd *Compute the mean of spatial functional data*

Description

This function is used to calculate the mean on functional data on a spatial grid.

Usage

```
## S3 method for class 'fsd.fd'
mean(x, trim = 0, na.rm = FALSE, ..., margins = NULL)
```

Arguments

x	the functional spatial data.
trim	currently unused
na.rm	whether or not missing values should be ignored.
...	currently unused
margins	the margins that should not be averaged over.

Value

the mean of x

See Also

center.fsd.fd

Examples

```
data("temp")
plot(mean(temp))
plot(mean(temp, margins = 1:2))
```

`mfsd` *Perform Multivariate Spectral Principal Components Analysis on Spatial Functional Data.*

Description

This function performs multivariate PCA on functional spatial data on a grid of dimension r.

Usage

```
mfsd(
  X,
  freq.res = 100,
  SM_Npc = 3,
  L = 3,
  q = NULL,
  na.ignore = TRUE,
  return.F = FALSE,
  only.filters = FALSE
)
```

Arguments

`X`	a list containing the functional spatial data. Either an fd object or an array.
`freq.res`	the resolution for the computation of the spectral density.
`SM_Npc`	the number of multivariat spatial functional principal components to be computed.
`L`	a list containing the maximum lag for the filters for each variable. An integer or vector of integers.
`q`	a list containing tuning parameter for the estimation of the the spectral density operator for each variable. An integer or vector of integers.
`na.ignore`	whether to ignore missing data points in the computation of the scores.
`return.F`	a boolean indicating whether to return the spectral density.
`only.filters`	a boolean indicating whether to compute only the filters, leaving out the scores.

Details

This function can be used to compute spectral PCA. By setting `q = 0`, it can also be used to perform static MFPCA.

Setting a higher value for `freq.res` increases the runtime significantly, but yields a more accurate result because of reduced integration errors.

Value

A list with components

`Value`	eigenvalue for each multivariate PC
`tuning.params`	a list of the used tuning parameters `freq.res`, `q` and `Lmax`.
`filters`	the multivariate SPC filters.
`scores`	the multivariate SPC scores.
`vm`	the theoretical fractions of variance explained by each multivariate PC.
`SML_Mean`	the mean of X.
`SM_FSD`	list of univariate spatial functional principal component analysis.

Examples

```
data("X1999")
data("X2000")
data("qlist")
data("Llist")
X = list(X1999, X2000)
q = list(qlist[,17], qlist[,18])
L = list(L1 = Llist[17], L2 = Llist[18])
mfsd(X = X, L = L, q = q, SM_Npc = 4)
```

---

mfsd.evaluation *Perform A Comparative Evaluation of Dimension Reduction: SMF-PCA versus MFPCA in Multivariate Spatial Data Reconstruction*

Description

To assess the effectiveness of integrating the spatial and multivariate aspects through SMFPCA, it is common to reconstruct the original functional data from the already computed scores and filters. For each instance, we employ both the novel SMFPCA and the conventional MFPCA. We assess the effectiveness of dimension reduction using the metric known as the normalized mean squared error (NMSE) for each univariate component. NMSE* evaluates the quality of the approximation without being influenced by grid boundary effects, unlike NMSE, where the approximation is less accurate at the boundary.

Usage

```
mfsd.evaluation(Xspec, Xspecc, X00, Npc = 3)
```

Arguments

Xspec	an object of class SM_object, this function is obtained by applying the mfsd function with a non-zero value of q.
Xspecc	an object of class M_object, this object is obtained by applying the mfsd function with q equal to zero.
X00	a list containing the functional spatial data. Either an fd object or an array.
Npc	the number of multivariat spatial functional principal components to be computed.

Value

a display of the NMSE and NMSE* results from print.

---

mfsd.nmse *Perform A Evaluation of Dimension Reduction Using NMSE and NMSE**

Description

To assess the effectiveness of integrating the spatial and multivariate aspects through SMFPCA, one must evaluate the effectiveness of dimension reduction using the metric known as the normalized mean squared error (NMSE) for each univariate component. NMSE* evaluates the quality of the approximation without being influenced by grid boundary effects, unlike NMSE, where the approximation is less accurate at the boundary.

Usage

```
mfsd.nmse(Xspec, X, Npc = 3)
```

Arguments

Xspec	an object of class M_object, This object is obtained by applying the mfsd function with q equal to zero.
X	a list containing the functional spatial data. Either an fd object or an array.
Npc	the number of multivariat spatial functional principal components to be computed.

Value

a display of the NMSE and NMSE* results from print.

mfsd_Filter — *Calculate the Multivariate Spectral Principal Components Filters Spatial Functional Data*

Description

This function calculates the multivariate SPC filters from a given spectral density operator.

Usage

```
mfsd_Filter(SM_FSD, SM_vec, J)
```

Arguments

SM_FSD	list of univariate spatial functional principal component analysis.
SM_vec	list of variable eigenvectors.
J	number of functional multivariate spatial principal components.

Details

The univariate eigenvectors and filters are used to compute multivariate spatial function filters.

Value

the Multivariate SPC filters.

mfsd_Scores — *Compute the Multivariate Scores for Spatial Functional Data.*

Description

Compute the Multivariate Scores for Spatial Functional Data.

Usage

```
mfsd_Scores(EEM, SM_vec, SM_FSD, X, J)
```

Arguments

EEM	concatenated scores.
SM_vec	list of variable eigenvectors.
SM_FSD	list of univariate spatial functional principal component analysis.
X	list of univariate spatial functional data.
J	number of functional multivariate spatial principal components.

Value

a list of the multivariate scores.

`plot.fsd.fd` *Plot functional spatial data*

Description

This function is used to plot functional spatial data on a grid of dimension r, along a two-dimensional plane of indices.

Usage

```
## S3 method for class 'fsd.fd'
plot(x, ..., gridsize = NULL, set.dims = NULL)
```

Arguments

`x`	the functional spatial data.
`...`	graphical parameters.
`gridsize`	the number of grid points on which to evaluate the functional data.
`set.dims`	the coordinates to be fixed. A vector of dimension r. NaN represents not fixed components.

Value

No return value. This function is called for its side effect of producing graphical displays of functional spatial data.

See Also

`fsd.plot.data`

Examples

```
data("temp")
plot(temp)
```

`plot.fsd.filter` *Plot functional spatial filters*

Description

This function plots functional spatial filters along a two-dimensional plane of indices.

Usage

```
## S3 method for class 'fsd.filter'
plot(
  x,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  gridsize = NULL,
  Npc = 3,
  Lmax = 3,
  set.dims = NULL,
  ...
)
```

Arguments

`x`	the filter object.
`xlim`	limits of the x-axis.
`ylim`	limits of the y-axis.
`main`	main title of the graph.
`xlab`	label of the x-axis.
`ylab`	label of the y-axis.
`gridsize`	number of grid points used to evaluate the functional data.
`Npc`	number of filters to plot.
`Lmax`	maximum lag to plot.
`set.dims`	coordinates to be fixed. A vector of dimension r. 'NaN' represents components that are not fixed.
`...`	additional graphical parameters.

Value

No return value. This function is called for its side effect of producing graphical displays of functional spatial filters.

See Also

fsd.plot.filters

`plot.mfsd.fd` *Plot method for mfsd.fd objects*

Description

Plot method for mfsd.fd objects

Usage

```
## S3 method for class 'mfsd.fd'
plot(
  x,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  gridsize = NULL,
  set.dims = NULL,
  ...
)
```

Arguments

`x`	object of class mfsd.fd
`xlim`	limits of x axis
`ylim`	limits of y axis
`main`	title
`xlab`	x label
`ylab`	y label
`gridsize`	grid size
`set.dims`	dimension selection
`...`	additional graphical parameters

Value

No return value. This function is called for its side effect of producing graphical plots of the functional spatial data.

qlist *a list containing tuning parameter for the estimation of the the spectral density of indian variable*

Description

a list containing tuning parameter for the estimation of the the spectral density of indian variable

Usage

```
data(qlist)
```

Format

qlist has a dimension of 2 22

An object of class "matrix" "array".

Examples

```
data(qlist)
```

summary.fsd.filter *Analyse functional spatial filters*

Description

This function is used to analyse functional spatial filters, indicating how the squared norms lie.

Usage

```
## S3 method for class 'fsd.filter'
summary(object, use.norm = Inf, make.plot = FALSE)
```

Arguments

object	the filter.
use.norm	which norm to use for the lags.
make.plot	a boolean indicating whether to plot the result.

Value

object list with components

balls	a data frame containing the cumulative sums of the squared norms of the filter coefficients up to some maximum lag.
zeroplanes	a matrix containing the sums of the squared norms of the filter coefficients lying on the (hyper-) planes with one dimension of the lag equals zero, along with the sum over all lags as comparison.
abs.sum	a data frame containing the cumulative sums of the norms of the filter coefficients.

See Also

fsd.plot.filters

---

summary.mfsd.filter *Analyse multivariate functional spatial filters*

Description

This function is used to analyse multivariate functional spatial filters, indicating how the squared norms lie.

Usage

```
## S3 method for class 'mfsd.filter'
summary(object, use.norm = Inf, make.plot = FALSE)
```

Arguments

object	containing a list of variable filters.
use.norm	which norm to use for the lags.
make.plot	a boolean indicating whether to plot the result.

Value

object list with components

balls	a data frame containing the cumulative sums of the squared norms of the filter coefficients up to some maximum lag.
zeroplanes	a matrix containing the sums of the squared norms of the filter coefficients lying on the (hyper-) planes with one dimension of the lag equals zero, along with the sum over all lags as comparison.
abs.sum	a data frame containing the cumulative sums of the norms of the filter coefficients.

temp *Temperature Data in Wyoming*

Description

CPC daily Temperature data of mean temperatures in Wyoming projected onto a B-spline basis of 36 functions for each year, from 1979 to 2017 and with a spatial resolution of 0.5 degrees.

Usage

```
data(temp)
```

Format

An object of class `"fsd.fd"`.

Details

The gridded data has been obtained by Shepard's method.

Examples

```
data(temp)
plot(temp)
```

unfold *Unfold indices into a high-dimensional grid*

Description

This function is used to expand indices into a grid. Works similar to `expand.grid`.

Usage

```
unfold(indexlist, r = NULL, return.array = FALSE)
```

Arguments

`indexlist`	indices to be expanded. Either a list with each member representing a dimension, or a vector.
`r`	the dimension of the grid to be created. This is only needed if indexlist is a vector.
`return.array`	a boolean indicating whether an array should be returned or a list.

Value

a list or an array with the coordinates of the grid.

Examples

```
unfold(list(11:12, 21:23), return.array = TRUE)
```

X1999 *Spatial functional data of sea surface temperature (SST) data for the year 1999, derived from NOAA optimal sea surface temperature interpolation data.*

Description

Spatial Functional Data of sea surface temperature (SST) data from the NOAA Optimum Interpolation Sea Surface Temperature dataset. The dataset exhibiting typical spatial dependence characterized by an exponential decay.

Usage

```
data(X1999)
```

Format

An object of class `"fsd.fd"`.

Examples

```
data(X1999)
```

X2000 *Spatial functional data of sea surface temperature (SST) data for the year 2000, derived from NOAA optimal sea surface temperature interpolation data.*

Description

Spatial Functional Data of sea surface temperature (SST) data from the NOAA Optimum Interpolation Sea Surface Temperature dataset. The dataset exhibiting typical spatial dependence characterized by an exponential decay.

Usage

```
data(X2000)
```

Format

An object of class "fsd.fd".

Examples

```
data(X2000)
```

X2001 — *Spatial functional data of sea surface temperature (SST) data for the year 2001, derived from NOAA optimal sea surface temperature interpolation data.*

Description

Spatial Functional Data of sea surface temperature (SST) data from the NOAA Optimum Interpolation Sea Surface Temperature dataset. The dataset exhibiting typical spatial dependence characterized by an exponential decay.

Usage

```
data(X2001)
```

Format

An object of class "fsd.fd".

Examples

```
data(X2001)
```

[.fsd.fd — *Subsetting a functional spatial data object*

Description

Subsetting works as with conventional arrays. By default, dimensions can be dropped.

Usage

```
## S3 method for class 'fsd.fd'
fsdobj[..., drop = TRUE]
```

Arguments

fsdobj	a functional spatial data object
...	the indices to take
drop	whether to flatten the spatial grid by dropping dimensions of length one.

Value

An object of class `fsd.fd`. The returned object is a subset of the original functional spatial data object. The structure of the coefficient array and the associated spatial grid is reduced according to the selected indices. If `drop = TRUE`, dimensions of length one in the spatial grid may be removed.

Examples

```
data("temp")
plot(temp[1:2, 1:2, 1, drop = TRUE])
```

`%/%.fsd.fd` *Subsetting a functional spatial data object*

Description

Take only every `s`-th element

Usage

```
## S3 method for class 'fsd.fd'
fsdobj %/% s
```

Arguments

`fsdobj`	a functional spatial data object
`s`	an integer or a vector with one entry for every dimension of the spatial grid.

Value

An object of class `fsd.fd`. The returned object is a downsampled version of the input functional spatial data object, where only every `s`-th spatial location is retained along each spatial dimension.

Examples

```
data("temp")
plot(temp %/% 3)
```

Index