Package ‘ftsspec’

September 8, 2015

Title  Spectral Density Estimation and Comparison for Functional Time Series

Version  1.0.0

Author  Shahin Tavakoli [aut, cre]

Maintainer  Shahin Tavakoli <s.tavakoli@statslab.cam.ac.uk>

Description  Functions for estimating spectral density operator of functional time series (FTS) and comparing the spectral density operator of two functional time series, in a way that allows detection of differences of the spectral density operator in frequencies and along the curve length.

Depends  R (>= 3.2.0)

Imports  sna (>= 2.3-2)

License  GPL-2

LazyData  true

NeedsCompilation  no

Repository  CRAN

Date/Publication  2015-09-08 13:13:41

R topics documented:

Epanechnikov_kernel .................................................. 2
ftsspec ................................................................. 2
Generate_filterMA .................................................... 3
Get_noise_sd ........................................................... 4
lines.SampleSpecDiffFreq ............................................. 4
Marginal_basis_pval .................................................. 5
plot.SampleSpec ...................................................... 5
plot.SampleSpecDiffFreq .......................................... 6
plot.SampleSpecDiffFreqCurvelength .............................. 6
plot.SpecMA ........................................................... 7
print.SampleSpecDiffFreqCurvelength ............................ 7
PvalAdjust ............................................................. 8
Simulate_new_MA ...................................................... 8
Epanechnikov_kernel

The Epanechnikov weight function, with support in $[-1, 1]$.

Description

The Epanechnikov weight function, with support in $[-1, 1]$.

Usage

Epanechnikov_kernel(x)

Arguments

x  argument at which the function is evaluated

ftsspec

ftsspec: collection of functions for estimating spectral density operator of functional time series (FTS) and comparing the spectral density operator of two functional time series, in a way that allows detection of differences of the spectral density operator in frequencies and along the curve length.

Description

ftsspec: collection of functions for estimating spectral density operator of functional time series (FTS) and comparing the spectral density operator of two functional time series, in a way that allows detection of differences of the spectral density operator in frequencies and along the curve length.

References

Tavakoli, Shahin and Panaretos, Victor M. "Detecting and Localizing Differences in Functional Time Series Dynamics: A Case Study in Molecular Biophysics", 2014, under revision
Generate the Filter of a multivariate MA process

Description

Generate the Filter of a multivariate MA process

Usage

```r
Generate_filterMA(d.ts, d.n, MA.len = 3, ma.scale = rep(1, MA.len),
                  a.smooth.coef = 0, seed = 1)
```

Arguments

- `d.ts`: dimension of the (output) time series
- `d.n`: dimension of the noise that is filtered
- `MA.len`: Length of the filter. Set to 3 by default.
- `ma.scale`: scaling factor of each lag matrix. See details.
- `a.smooth.coef`: A coefficient to shrink coefficients of filter. Set to 0 by default.
- `seed`: The random seed used to generate the filter. Set to 1 by default.

Value

A `d.ts x d.n x MA.len` array

Details

Generates a filter (i.e. a `d.ts x d.n x MA.len` array) for a moving average process. The entries of the filter are generate randomly, but can be reproduced by specifying the random seed `seed`.

The `ma.scale` parameter should be a vector of length `MA.len`, and corresponds to a scaling factor applied to each lag of the filter of the MA process that is generated.

Examples

```r
ma.scale1 = c(-1.4, 2.3, -2)
A1 = Generate_filterMA(10, 10, MA.len = 3, ma.scale = ma.scale1, seed = 1)
str(A1)
rm(A1)
```
Get_noise_sd

Get the square root of the covariance matrix associated to a noise type

Description

Get the square root of the covariance matrix associated to a noise type

Usage

Get_noise_sd(noise.type, d.n)

Arguments

- noise.type: the type of noise that is driving the MA process. See Details section.
- d.n: dimension of the noise that is filtered

lines.SampleSpecDiffFreq

Plotting function for SampleSpecDiffFreq class

Description

Plotting function for SampleSpecDiffFreq class

Usage

## S3 method for class 'SampleSpecDiffFreq'
lines(x, method = NA, Kmax = 4, pch = 20, 
...)

Arguments

- x: object of the class SampleSpecDiffFreq
- method: method used to adjust p-values
- Kmax: maximum number of levels K for which the p-values are plotted (used only if autok==0)
- pch: the plot character to be used
- ...: additional parameters to be passed to plot()

See Also

Spec_compare_localize_freq
Marginal_basis_pval

Compute the marginal p-values at each basis coefficients of for testing the equality of two spectral density kernels

Description

Compute the marginal p-values at each basis coefficients of for testing the equality of two spectral density kernels

Usage

Marginal_basis_pval(spec1, spec2, m, kappa.square, is.pi.multiple)

Arguments

spec1 The two sample spectral densities (at the same frequency \( \omega \)) to be compared.
spec2 The two sample spectral densities (at the same frequency \( \omega \)) to be compared.
m The number of Fourier frequencies over which the periodogram operator was smoothed.
kappa.square the L2-norm of the weight function used to estimate the spectral density operator
is.pi.multiple A logical variable, to specify if \( \omega = 0, \pi \) or not.

plot.SampleSpec

Plotting method for object inheriting from class SampleSpec

Description

Plotting method for object inheriting from class SampleSpec

Usage

## S3 method for class 'SampleSpec'
plot(x, ...)

Arguments

x An object of the class SampleSpec
... additional parameters to be passed to plot()
plot.SampleSpecDiffFreq

Plotting function for SampleSpecDiffFreq class

Description

Plotting function for SampleSpecDiffFreq class

Usage

## S3 method for class 'SampleSpecDiffFreq'
plot(x, method = NA, Kmax = 4, pch = 20, ...)

Arguments

- **x**: object of the class SampleSpecDiffFreq
- **method**: method used to adjust p-values
- **Kmax**: maximum number of levels \( K \) for which the p-values are plotted (used only if autok==0)
- **pch**: the plot character to be used
- **...** additional parameters to be passed to plot()

See Also

Spec_compare_localize_freq

plot.SampleSpecDiffFreqCurvelength

Plotting method for class SampleSpecDiffFreqCurvelength

Description

Plotting method for class SampleSpecDiffFreqCurvelength

Usage

## S3 method for class 'SampleSpecDiffFreqCurvelength'
plot(x, ncolumns = 3, ...)

Arguments

- **x**: Object of the class SampleSpecDiffFreqCurvelength
- **ncolumns**: number of columns for the plots
- **...**: additional parameters to be passed to plot()
### plot.SpecMA

Plotting method for object inheriting from class SpecMA

#### Description
Plotting method for object inheriting from class SpecMA

#### Usage
```r
## S3 method for class 'SpecMA'
plot(x, ...)
```

#### Arguments
- `x`: A object of the class SpecMA
- `...`: additional parameters to be passed to plot()

### print.SampleSpecDiffFreqCurvelength

Printing method for class SampleSpecDiffFreqCurvelength

#### Description
Printing method for class SampleSpecDiffFreqCurvelength

#### Usage
```r
## S3 method for class 'SampleSpecDiffFreqCurvelength'
print(x, ...)
```

#### Arguments
- `x`: Object of the class SampleSpecDiffFreqCurvelength
- `...`: Additional arguments for print
PvalAdjust

Generic function to adjust pvalues

Description

Generic function to adjust pvalues

function to adjust pvalues for class SampleSpecDiffFreq

Usage

PvalAdjust(sample.spec.diff, method)

## S3 method for class 'SampleSpecDiffFreq'
PvalAdjust(sample.spec.diff, method)

Arguments

- **sample.spec.diff**
  - Object of the class SampleSpecDiffFreq

- **method**
  - Method used to adjust p-values

See Also

Spec_compare_localize_freq

Simulate_new_MA

Simulate a new Moving Average (MA) vector time series and return the time series

Description

Simulate a new Moving Average (MA) vector time series and return the time series

Usage

Simulate_new_MA(a, T.len, noise.type, DEBUG = FALSE)

Arguments

- **a**
  - Array, returned by Generate_filterMA, containing the filter of the MA process

- **T.len**
  - Numeric, the length of the time series to generate

- **noise.type**
  - The type of noise that is driving the MA process. See Details section.

- **DEBUG**
  - Logical, for outputting information on the progress of the function
Spec

Value

A $T \times \text{dim}(a)[1]$ matrix, where each column corresponds to a coordinate of the vector time series.

Details

The function simulates a moving average process of dimension $\text{dim}(a)[1]$, defined by

$$X[t] = a[1]*\epsilon[t-1] + a[2]*\epsilon[t-2] + \ldots + a[\text{dim}(a)[3]]*\epsilon[t-\text{dim}(a)[3]]$$

`noise.type` specifies the nature and internal correlation of the noise that is driving the MA process. It can take the values:

- `white_noise` the noise is Gaussian with covariance matrix identity
- `white_noise` the noise is Gaussian with diagonal covariance matrix, whose j-th diagonal entry is $((j - 0.5) * \pi)^{(j - 1)}$
- `studentk` the coordinates of the noise are independent and have a student t distribution with 'k' degrees of freedom, standardized to have variance 1

Examples

```r
ma.scale=c(-1.4,2.3,-2)
a1=Generate_filterMA(6, 6, MA.len=3, ma.scale=ma.scale1)
X=Simulate_new_MA(a1, T.len=512, noise.type='wiener')
plot.ts(X)
```

---

**Compute Spectral Density of Functional Time Series**

Description

This function estimates the spectral density operator of a Functional Time Series (FTS).

Usage

```r
Spec(X, W = Epanechnikov_kernel, B.T = (dim(X)[1])^(-1/5),
only.diag = FALSE, trace = FALSE, demean = TRUE, subgrid = FALSE,
subgrid.density = 10, verbose = 0,
subgrid.density.relative.to.bandwidth = TRUE)
```

Arguments

- **X**
  - A $T \times \text{nbasis}$ matrix of containing the coordinates of the FTS expressed in a basis. Each row corresponds to a time point, and each column corresponds to the coefficient of the corresponding basis function of the FTS.

- **W**
  - The weight function used to smooth the periodogram operator. Set by default to be the Epanechnikov kernel
**B.T**  The bandwidth of frequencies over which the periodogram operator is smoothed. If B.T=0, the periodogram operator is returned.

**only.diag**  A logical variable to choose if the function only computes the marginal spectral density of each basis coordinate (only.diag=TRUE). only.diag=FALSE by default, the full spectral density operator is computed.

**trace**  A logical variable to choose if only the trace of the spectral density operator is computed. trace=FALSE by default.

**demean**  A logical variable to choose if the FTS is centered before computing its spectral density operator.

**subgrid**  A logical variable to choose if the spectral density operator is only returned for a subgrid of the Fourier frequencies, which can be useful in large datasets to reduce memory usage. subgrid=FALSE by default.

**subgrid.density**  Only used if subgrid=TRUE. Specifies the approximate number of frequencies within the bandwidth over which the periodogram operator is smoothed.

**verbose**  A variable to show the progress of the computations. By default, verbose=0.

**Value**

A list containing the following elements:

- **spec**  The estimated spectral density operator. The first dimension corresponds to the different frequencies over which the spectral density operators are estimated.
- **omega**  The frequencies over which the spectral density is estimated.
- **m**  The number of Fourier frequencies over which the periodogram operator was smoothed.
- **bw**  The equivalent Bandwidth used in the weight function W(), as defined in Bloomfield (1976, p.201).
- **weight**  The weight function used to smooth the periodogram operator.
- **kappa.square**  The L2 norm of the weight function W.

**References**

spec.pgram function of R.


**Examples**

ma.scale1=c(-1.4,2.3,-2)
a1=Generate_filterMA(10, 10, MA.len=3, ma.scale=ma.scale1)
X=Simulate_new_MA(a1, T.len=512, noise.type='wiener')
ans=Spec(X, trace=FALSE, only.diag=FALSE)
SpecMA

'Spectral density operator of a MA vector process' Object

Description
'Spectral density operator of a MA vector process' Object

Usage
SpecMA(a, nfreq = 2^9, noise.type)

Arguments
- a: the filter of the moving average
- nfreq: the number of frequencies between 0 and pi at which the spectral density has to be computed
- noise.type: the type of noise that is driving the MA process. See Simulate_new_MA

Examples
ma.scale1=c(-1.4,2.3,-2)
a1=generate_filterMA(6, 6, MA.len=3, ma.scale=ma.scale1)
a1.spec=SpecMA(a1, nfreq=512, noise.type='wiener')
plot(a1.spec)
rm(a1, a1.spec)

Spec_compare_fixed_freq
Test if two spectral density operators at some fixed frequency are equal.

Description
A test for the null hypothesis that two spectral density operators (at the same frequency $\omega$) are equal, using a pseudo-AIC criterion for the choice of the truncation parameter. (used in Spec_compare_localize_freq)

Usage
Spec_compare_fixed_freq(spec1, spec2, is.pi.multiple, m, kappa.square, autok = 2, K.fixed = NA)
Arguments

spec1,spec2  The two sample spectral densities (at the same frequency $\omega$) to be compared.

is.pi.multiple  A logical variable, to specify if $\omega = 0, \pi$ or not.

m  The number of Fourier frequencies over which the periodogram operator was smoothed.

kappa.square  the $L_2$-norm of the weight function used to estimate the spectral density operator

autok  A variable used to specify if (and which) pseudo-AIC criterion is used to select the truncation parameter $K$.

K.fixed  The value of $K$ used if autok=0.

References

Tavakoli, Shahin and Panaretos, Victor M. "Detecting and Localizing Differences in Functional Time Series Dynamics: A Case Study in Molecular Biophysics", 2014, under revision


See Also

Spec_compare_localize_freq

Examples

```r
ma.scale2=ma.scale1=c(-1.4,2.3,-2)
a1=Generate_filterMA(10, 10, MA.len=3, ma.scale=ma.scale1)
a2=Generate_filterMA(10, 10, MA.len=3, ma.scale=ma.scale2)
X=Simulate_new_MA(a1, T.len=512, noise.type='wiener')
Y=Simulate_new_MA(a2, T.len=512, noise.type='wiener')
spec.X = Spec(X)
spec.Y = Spec(Y)
Spec_compare_fixed_freq(spec.X$spec[,], spec.Y$spec[,],
is.pi.multiple=TRUE, spec.X$m, spec.X$kappa.square)
```

Spec_compare_localize_freq

*Compare the spectral density operator of two Functional Time Series and localize frequencies at which they differ.*

Description

Compare the spectral density operator of two Functional Time Series and localize frequencies at which they differ.
Usage

Spec_compare_localize_freq(xL, yL, B.T = (dim(xL)[1])^(-1/5), W, autok = 2,
subgrid.density, verbose = 0, demean = FALSE, K.fixed = NA,
subgrid.density.relative.to.bandwidth)

Arguments

X, Y
The \( T \times nbasis \) matrices of containing the coordinates, expressed in some func-
tional basis, of the two FTS that to be compared. expressed in a basis.

B.T
The bandwidth of frequencies over which the periodogram operator is smoothed.
If B.T=0, the periodogram operator is returned.

W
The weight function used to smooth the periodogram operator. Set by default to be the Epanechnikov kernel

autok
A variable used to specify if (and which) pseudo-AIC criterion is used to select
the truncation parameter \( K \).

subgrid.density
Only used if subgrid=True. Specifies the approximate number of frequencies
within the bandwidth over which the periodogram operator is smoothed.

verbose
A variable to show the progress of the computations. By default, verbose=0.

demean
A logical variable to choose if the FTS is centered before computing its spectral
density operator.

K.fixed
The value of K used if autok=0.

subgrid.density.relative.to.bandwidth
 logical parameter to specify if subgrid.density is specified relative to the
bandwidth parameter B.T

Details

X,Y must be of equal size \( T\cdot\text{len} \times d \), where \( T\cdot\text{len} \) is the length of the time series, and \( d \) is the
number of basis functions. Each row corresponds to a time point, and each column corresponds to
the coefficient of the corresponding basis function of the FTS.

autok=0 returns the p-values for \( K = 1, \ldots, K.fixed \). autok=1 uses the AIC criterion of Tavakoli
\& Panaretos (2015), which is a generalization of the pseudo-AIC introduced in Panaretos et al
(2010). autok=2 uses the AIC* criterion of Tavakoli \& Panaretos (2015), which is an extension of
the AIC criterion that takes into account the difficulty associated with the estimation of eigenvalues
of a compact operator.

References

Tavakoli, Shahin and Panaretos, Victor M. "Detecting and Localizing Differences in Functional
Time Series Dynamics: A Case Study in Molecular Biophysics", 2014, under revision
Panaretos, Victor M., David Kraus, and John H. Maddocks. "Second-order comparison of Gauss-
ian random functions and the geometry of DNA minicircles." Journal of the American Statistical
Spec_compare_localize_freq_curvelength

**Examples**

```r
data1 = c(-1.4, 2.3, -2)
data2 = c(10, 10, 10)

# Generate filter MA
filter1 = Generate_filter_MA(10, 10)
filter2 = Generate_filter_MA(10, 10)

# Simulate new MA
sim1 = Simulate_new_MA(data1, noise.type = 'wiener')
sim2 = Simulate_new_MA(data2, noise.type = 'wiener')

# Compare spectral density operator
spec_comp1 = Spec_compare_localize_freq(sim1, sim2, w = 'epanechnikov', autok = 2,
                                       demean = FALSE, subgrid_density = 10,
                                       subgrid_density_relative.to.bandwidth = TRUE)

plot(spec_comp1, method = 'fdr')
```

**Description**

Compare the spectral density operator of two Functional Time Series and localize frequencies at which they differ, and (spatial) regions where they differ.

**Usage**

```r
Spec_compare_localize_freq_curvelength(X, Y, t = (dim(X)[1])^(-1/5), W,
                      alpha = 0.05, accept = 0, reject = 1, verbose = 0, demean = FALSE)
```

**Arguments**

- **X**: The $T \times nbasis$ matrices of containing the coordinates, expressed in some functional basis, of the two FTS that to be compared. expressed in a basis.
- **Y**: The $T \times nbasis$ matrices of containing the coordinates, expressed in some functional basis, of the two FTS that to be compared. expressed in a basis.
- **B.T**: The bandwidth of frequencies over which the periodogram operator is smoothed. If B.T=0, the periodogram operator is returned.
The weight function used to smooth the periodogram operator. Set by default to be the Epanechnikov kernel

alpha  level for the test

accept, reject  values for accepted, rejected regions

verbose  A variable to show the progress of the computations. By default, verbose=0.

demean  A logical variable to choose if the FTS is centered before computing its spectral density operator.

Examples

ma.scale2=ma.scale1=c(-1.4,2.3,-2)
a1=Generate_filterMA(10,10, MA.len=3, ma.scale=ma.scale1)
a2=Generate_filterMA(10,10, MA.len=3, ma.scale=ma.scale2)
X=Simulate_new_MA(a1, T.len=2^9, noise.type='wiener')
Y=Simulate_new_MA(a2, T.len=2^9, noise.type='wiener')
ans0=Spec_compare_localize_freq_curvelength(X, Y, W=Epanechnikov_kernel, alpha=.01, demean=TRUE)
print(ans0)
plot(ans0)
rm(ma.scale1, ma.scale2, a1, a2, X, Y, ans0)
Index

Epanechnikov_kernal, 2
ftsspec, 2
ftsspec-package (ftsspec), 2
Generate_filterMA, 3
Get_noise_sd, 4
lines.SampleSpecDiffFreq, 4
Marginal_basis_pval, 5
plot.SampleSpec, 5
plot.SampleSpecDiffFreq, 6
plot.SampleSpecDiffFreqCurvelength, 6
plot.SpecMA, 7
print.SampleSpecDiffFreqCurvelength, 7
PvalAdjust, 8
Simulate_new_MA, 8, 11
Spec, 9
spec.pgram, 10
Spec_compare_fixed_freq, 11
Spec_compare_localize_freq, 4, 6, 8, 11, 12, 12
Spec_compare_localize_freq_curvelength, 14
SpecMA, 11