Package 'mvLSW'

October 24, 2018

Type Package

Title Multivariate, Locally Stationary Wavelet Process Estimation

Version 1.2.2

Date 2018-10-23

Maintainer Simon Taylor <s.taylor2@lancaster.ac.uk>

Description Tools for analysing multivariate time series with wavelets. This includes: simulation of a multivariate locally stationary wavelet (mvLSW) process from a multivariate evolutionary wavelet spectrum (mvEWS); estimation of the mvEWS, local coherence and local partial coherence. See Park, Eckley and Ombao (2014) <doi:10.1109/TSP.2014.2343937> for details.

Depends R (>= 3.2), fields, wavethresh, xts, zoo

License GPL (>= 3)

Author Simon Taylor [aut, cre],
Tim Park [aut],
Idris Eckley [ths],
Rebecca Killick [ctb]

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-10-24 10:20:03 UTC

R topics documented:

ApxCI *Evaluate the Approximate Confidence Interval of a mvEWS Estimate*

Description

Evaluate the approximate confidence interval of a multivariate evolutionary wavelet spectrum.

Usage

```
ApxCI(object, var = NULL, alpha = 0.05, ...)
```

Arguments

object	A mvLSW object containing the multivariate evolutionary wavelet spectrum estimate.
var	A mvLSW object containing the variance estimate of the wavelet spectrum. If this is NULL (default) then the variance is estimates by calling the varEWS and using object.
alpha	Type I error, a single numerical value within (0,0.5].
...	Additional arguments to be passed to the varEWS command.

Details

The command evaluates the approximate Gaussian confidence intervals for the elements of the mvEWS estimate.

Value

Invisibly returns a list containing two mvLSW classed objects with names "L" and "U" that respectively identify the lower and upper interval estimates.

References

Park, T. (2014) Wavelet Methods for Multivariate Nonstationary Time Series, PhD thesis, Lancaster University, pp. 91-111.

See Also

mvEWS, as.mvLSW, varEWS.

Examples

```
## Define evolutionary wavelet spectrum, structure only on level 2
Spec <- array(0, dim = c(3, 3, 8, 256))
Spec[1, 1, 2, ] <- 10
Spec[2, 2, 2, ] <- c(rep(5, 64), rep(0.6, 64), rep(5, 128))
Spec[3, 3, 2, ] <- c(rep(2, 128), rep(8, 128))
Spec[2, 1, 2, ] <- Spec[1, 2, 2, ] <- punif(1:256, 65, 192)
Spec[3, 1, 2, ] <- Spec[1, 3, 2, ] <- c(rep(-1, 128), rep(5, 128))
Spec[3, 2, 2, ] <- Spec[2, 3, 2, ] <- -0.5
EWS <- as.mvLSW(x = Spec, filter.number = 1, family = "DaubExPhase",
  min.eig.val = NA)

## Sample time series and estimate the EWS.
set.seed(10)
X <- rmvLSW(Spectrum = EWS)
EWS_X <- mvEWS(X, kernel.name = "daniell", kernel.param = 20)

## Evaluate asymptotic spectral variance
SpecVar <- varEWS(EWS_X)

## Plot Estimate & 95% confidence interval
CI <- ApxCI(object = EWS_X, var = SpecVar, alpha = 0.05)
plot(x = EWS_X, style = 2, info = 2, Interval = CI)
```

`as.mvLSW` *Multivariate Locally Stationary Wavelet Object*

Description

Constructs a multivariate locally stationary wavelet (mvLSW) object.

Usage

```
as.mvLSW(x, filter.number = 1, family = "DaubExPhase",
  smooth.type = "all", smooth.kernel = kernel("daniell", 0),
  bias.correct = FALSE, min.eig.val = -Inf, names = NULL)

is.mvLSW(object)
```

Arguments

`x`	4D array of order PxPxJxT where P is the number of channels of the time series of length T such that T=2^J for some positive integer J defining the levels of the mvLSW object.
`family`	Character string specifying the wavelet family. Only two options are available, either `"DaubExPhase"` (default) or `"DaubLeAsymm"`.

filter.number	Integer number defining the number of vanishing moments of the wavelet function. By default, filter.number=1 and so defining the Haar wavelet.
smooth.type	What type of smoothing regime has been applied. Either "all" (default) if the smoothing method been applied to all levels. Otherwise "by.level", a different smoothing method is applied to each level.
smooth.kernel	Definition of the smoothing kernel from kernel(). By default, the identity kernel is defined.
bias.correct	Logical, has a bias correction been applied to the data. FALSE by default.
min.eig.val	Minimum eigenvalue from spectral matrices across all levels and locations, set at -Inf by default. If NA, then the minimum eigenvalue is calculated.
names	Character vector containing the channel names of the multivariate time series.
object	Any R object.

Details

as.mvLSW constructs a multivariate locally stationary classed object that contains all information about the constructions of various multivariate wavelet estimates.

The command is.mvLSW checks that the supplied R object is a valid mvLSW object in that its structure and contents are as expected.

Value

The as.mvLSW command invisibly returns a list with the following items:

spectrum	A 4D array containing the data relating to the estimate of interest.
Information	List containing information on the estimation procedure.

The list Information contains:

names	Character vector containing the channel names.
dimensions	A list containing items P - the number of channels forming of the time series, T - the length of the time series and J - the number of levels in the wavelet transform of the data.
wavelet	A list containing the filter.number and family of the wavelet used in the transformation.
smooth	A list detailing applied smoothing of the estimate. Items include: smooth.type - name of the smoothing regime. smooth.kernels - a tskernel class from the command kernel(). GCV - generalized cross-validation gamma deviance criterion of the smoothing. smooth.eps - smoothing threshold. If smooth.type="by.level", then smooth.kernels is a list containing the tskernel object for each level from fine to coarse. In addition, GCV is a length J vector containing the criterion estimate for each level from fine to coarse.
correction	A list containing bias.correction and min.eig.val.

The command is.mvLSW returns TRUE if the supplied object is a valid mvLSW object as described above. Otherwise, the command returns FALSE.

See Also

mvEWS, varEWS, kernel.

Examples

```
## Define evolutionary wavelet spectrum, structure only on level 2
Spec <- array(0, dim = c(3, 3, 8, 256))
Spec[1, 1, 2, ] <- 10
Spec[2, 2, 2, ] <- c(rep(5, 64), rep(0.6, 64), rep(5, 128))
Spec[3, 3, 2, ] <- c(rep(2, 128), rep(8, 128))
Spec[2, 1, 2, ] <- Spec[1, 2, 2, ] <- punif(1:256, 65, 192)
Spec[3, 1, 2, ] <- Spec[1, 3, 2, ] <- c(rep(-1, 128), rep(5, 128))
Spec[3, 2, 2, ] <- Spec[2, 3, 2, ] <- -0.5

## Define EWS as mvLSW object
EWS <- as.mvLSW(x = Spec, filter.number = 1, family = "DaubExPhase",
  names = c("A", "B", "C"), min.eig.val = NA)
is.mvLSW(EWS)
plot(EWS, style = 2, info = 2)
```

AutoCorrIP *Wavelet Autocorrelation Inner Product Functions*

Description

Inner product of cross-level wavelet autocorrelation functions.

Usage

```
AutoCorrIP(J, filter.number = 1, family = "DaubExPhase",
  crop = TRUE)
```

Arguments

J	Number of levels.
filter.number	Number of vanishing moments of the wavelet function.
family	Wavelet family, either "DaubExPhase" or "DaubLeAsymm". The Haar wavelet is defined as default.
crop	Logical, should the output of AutoCorrIP be cropped such that the first dimension of the returned array relate to the offset range -2^J:2^J.This is set at TRUE by default.

Details

Let $\psi(x)$ denote the mother wavelet and the wavelet defined for level j as $\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)$. The wavelet autocorrelation function between levels j & l is therefore:

$$\Psi_{j,l}(\tau) = \sum_{\tau} \psi_{j,k}(0)\psi_{l,k-\tau}(0)$$

Here, integer τ defines the offset of the latter wavelet function relative to the first.

The inner product of this wavelet autocorrelation function is defined as follows for level indices j, l & h and offset λ:

$$A^{\lambda}_{j,l,h} = \sum_{\tau} \Psi_{j,l}(\lambda - \tau)\Psi_{h,h}(\tau)$$

Value

A 4D array (invisibly returned) of order LxJxJxJ where L depends on the specified wavelet function. If `crop=TRUE` then L=2^{J+1}+1. The first dimension defines the offset λ, whilst the second to fourth dimensions identify the levels indexed by j, l & h respectively.

References

Fryzlewicz, P. and Nason, G. (2006) HaarFisz estimation of evolutionary wavelet spectra. *Journal of the Royal Statistical Society. Series B*, **68**(4) pp. 611-634.

See Also

`ipndacw`.

Examples

```
## Plot Haar autocorrelation wavelet functions inner product
AInnProd <- AutoCorrIP(J = 8, filter.number = 1, family = "DaubExPhase")
## Not run:
MaxOffset <- 2^8
for(h in 6:8){
  x11()
  par(mfrow = c(3, 3))
  for(l in 6:8){
    for(j in 6:8){
      plot(-MaxOffset:MaxOffset, AInnProd[, j, l, h], type = "l",
        xlab = "lambda", ylab = "Autocorr Inner Prod",
        main = paste("j :", j, "- l :", l, "- h :", h))
    }
  }
}

## End(Not run)

## Special case relating to ipndacw function from wavethresh package
```

```
Amat <- matrix(NA, ncol = 8, nrow = 8)
for(j in 1:8) Amat[, j] <- AInnProd[2^8 + 1, j, j, ]
round(Amat, 5)
round(ipndacw(J = -8, filter.number = 1, family = "DaubExPhase"), 5)
```

`coherence` *Local Wavelet Coherence and Partial Coherence*

Description

Wavelet coherence and partial coherence of an evolutionary wavelet spectrum.

Usage

```
coherence(object, partial = FALSE)
```

Arguments

`object`	Multivariate evolutionary wavelet spectrum as a `mvLSW` object.
`partial`	Logical, should the partial coherence be calculated. Set as `FALSE` by default.

Details

Given the evolutionary wavelet spectrum of a multivariate locally stationary time series, denoted by the matrix sequence $S_{j,k}$, then the coherence matrix for level j and location k is:

$$R_{j,k} = D_{j,k} S_{j,k} D_{j,k}$$

where $D_{j,k} = diag\{(S_{j,k}^{(p,p)})^{-0.5} : p = 1, \ldots, P\}$. This measures the linear cross-dependence between different channels at a particular level.

Notate the inverse spectrum matrix as $G_{j,k} = S_{j,k}^{-1}$, then the partial coherence matrix for level j and location k is derived as follows:

$$\Gamma_{j,k} = -H_{j,k} G_{j,k} H_{j,k}$$

where $H_{j,k} = diag\{(G_{j,k}^{(p,p)})^{-0.5} : p = 1, \ldots, P\}$. This measures the coherence between channels after removing the linear effects if all other channels and so enable the distinction between direct and indirect linear dependency between channels.

For valid calculations of (partial) coherence, values within [-1,1], it is important that the spectral matrices are positive definite.

Value

An object of class `mvLSW`, invisibly.

References

Park, T., Eckley, I. and Ombao, H.C. (2014) Estimating time-evolving partial coherence between signals via multivariate locally stationary wavelet processes. *Signal Processing, IEEE Transactions on* **62**(20) pp. 5240-5250.

See Also

as.mvLSW, mvEWS.

Examples

```
## Sample tri-variate time series
##  Series 2 & 3 are dependent indirectly via Series 1
set.seed(100)
X <- matrix(rnorm(3 * 2^8), ncol = 3)
X[1:192, 2] <- X[1:192, 2] + 0.95 * X[1:192, 1]
X[65:256, 3] <- X[65:256, 3] - 0.95 * X[65:256, 1]
X <- as.ts(X)

## Evolutionary Wavelet Spectrum
EWS <- mvEWS(X, filter.number = 4, kernel.name = "daniell",
  kernel.param = 20)

## Coherence
RHO <- coherence(EWS, partial = FALSE)
plot(RHO, style = 2, info = 1, ylab = "Coherence", diag = FALSE)

## Partial Coherence
PRHO <- coherence(EWS, partial = TRUE)
plot(PRHO, style = 2, info = 1, ylab = "P. Coh.", diag = FALSE)
#series 2&3 are closer to 0
```

---

mvEWS *Multivariate Evolutionary Wavelet Spectrum*

Description

Calculates the multivariate Evolutionary Wavelet Spectrum (mvEWS) of a multivariate locally stationary time series.

Usage

```
mvEWS(X, filter.number = 1, family = "DaubExPhase",
  smooth = TRUE, type = "all", kernel.name = "daniell",
  kernel.param = floor(sqrt(nrow(X))), optimize = FALSE,
  smooth.Jset = NA, bias.correct = TRUE, tol = 1e-10,
  verbose = FALSE)
```

Arguments

`X`	A multivariate time series object of class `ts`, `zoo`, `xts` or `matrix`. The length of the time series must be 2^J for positive integer J.
`filter.number`	Integer number defining the number of vanishing moments of the wavelet function. By default, `filter.number=1` and so defining the Haar wavelet.
`family`	Character string specifying the wavelet family. Only two options are available, either `"DaubExPhase"` (default) or `"DaubLeAsymm"`.
`smooth`	Logical, should the mvEWS should be smoothed?
`type`	How should the smoothing be performed? If `"all"` (default) then the same smoothing kernel is applied to all levels, else if `"by.level"` then a different smoothing kernel is applied to each level.
`kernel.name`	Name of smoothing kernel to be supplied to `kernel()`. Kernel `"daniell"` is defined by default.
`kernel.param`	Parameters to be passed to `kernel()`. This argument must be a vector if `type="all"`, otherwise it must be a matrix with each column defining the kernel parameters for each `log2(nrow(X))` levels from coarse to fine. If the name is `"dirichlet"` or `"fejer"` then `kernel.param` must have length 2 (or a matrix with 2 rows) which are supplied to `kernel()` as arguments `m` and `r` respectively. Note that the width of the kernel cannot be larger than the time series length. This is set by default as the square root of the length of the time series.
`optimize`	Logical, should the smoothing be optimized? If `FALSE` (default) then smoothing is performed as specified with `kernel.name` and `kernel.param`. Otherwise, `kernel.param` defines the upper parameter bound in determining the optimal kernel is determined by minimising the generalized cross-validation gamma deviance criterion.
`smooth.Jset`	Integer vector indicating with levels to be included in the calculation of the generalized cross-validation gamma deviance criterion. This argument is only used if `type="all"` and is set as `NA` by default, implying that all levels should be used.
`bias.correct`	Logical, should the correction be applied to address the bias in the raw mvEWS estimator.
`tol`	Tolerance in applying matrix regularisation to ensure each mvEWS matrix per location and level to be strictly positive definite. If `NA` or `-Inf` then the threshold is not applied. This is 1e-10 by default.
`verbose`	Logical. Controls the printing of messages whist the computations progress. Set as `FALSE` as default.

Details

This command evaluates the multivariate evolutionary wavelet spectrum of a multivariate locally stationary wavelet time series. The order of operations are as follows:

Calculate the non-decimated wavelet coefficients $\{d_{j,k}^{(p)}\}$ for levels j = 1,...,J, locations k = 0,...,T-1 (T=2^J) and channels p = 1,...,P(=`ncol(X)`). The raw periodogram matrices are then evaluated by $I_{j,k}^{(p,q)} = d_{j,k}^{p} d_{j,k}^{q}$ between any channel pair p & q.

The above estimator is inconsistent and so the matrix sequence is smoothed: $\tilde{I}_{j,k}^{(p,q)} = \sum_i W_i I_{j,k+i}^{(p,q)}$. The kernel weights W_i are derived from the `kernel` command and satisfy $W_i = W_{-i}$ and $\sum_i W_i = 1$. The optimal parameter for the smoothing kernel is determined by minimising the generalized cross-validation gamma deviance criterion (see Ombao et al., 2005).

The raw wavelet periodogram is also a biased estimator. A correction is subsequently applied to the smoothed estimate as follows:

$$\hat{S}_{j,k} = \sum_{l=1}^{J} (A^{-1})_{j,l} \hat{I}_{l,k}$$

Here, A denotes the wavelet autocorrelation inner product matrix.

If chosen to, the mvEWS matrices at each level and location, $\hat{S}_{j,k}$, is regularised to ensure positive definiteness.

Value

An object of class `mvLSW`, invisibly.

References

Park, T., Eckley, I. and Ombao, H.C. (2014) Estimating time-evolving partial coherence between signals via multivariate locally stationary wavelet processes. *Signal Processing, IEEE Transactions on* **62**(20) pp. 5240-5250.

Ombao, H., von Sachs, R. and Guo, W. (2005) SLEX analysis of multivariate nonstationary time series. *Journal of the American Statistical Association* **100**(470) pp.519-531.

See Also

`ts`, `wd`, `kernel`, `as.mvLSW`, `ipndacw`.

Examples

```
## Sample bivariate locally stationary time series
set.seed(100)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

## Haar wavelet, apply same smoothing to all levels & optimize
EWS <- mvEWS(X, kernel.name = "daniell", kernel.param = 20,
  optimize = TRUE)
summary(EWS)
plot(EWS, style = 2, info = 1)

## Over smoothed EWS
EWS_smooth <- mvEWS(X, filter.number = 10, family = "DaubLeAsymm",
  kernel.name = "modified.daniell", kernel.param = c(5, 5),
```

```
  optimize = FALSE)
summary(EWS_smooth)
plot(EWS_smooth, style = 2, info = 1)
```

---

mvLSW *Multivariate, Locally Stationary Wavelet Process Estimation*

Description

The mvLSW package provides an implementation of the multivariate locally stationary time series modelling approach proposed by Park, Eckley and Ombao (2014).

The approach extends the locally stationary wavelet time series work of Nason, von Sachs and Kroisandt (2000) to a multivariate setting, introducing wavelet-based measures of local coherence and local partial coherence. The package implements the estimation scheme by Park et al. (2014) for such processes. Note that mvLSW should be used in conjunction with the `wavethresh` package developed by Nason (2016).

Details

Package: mvLSW

Type: Package

Version: 1.2

Date: 2017-08-04

License: GPL(>=3)

Author(s)

Simon Taylor, <s.taylor2@lancaster.ac.uk>

References

Park, T.A., Eckley, I. and Ombao, H.C. (2014) Estimating time-evolving partial coherence between signals via multivariate locally stationary wavelet processes *IEEE Transactions on Signal Processing* **62**(20), pp. 5240–5250.

Nason, G.P., von Sachs, R. and Kroisandt, G. (2000) Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum *Journal of the Royal Statistical Society B* **62**, pp. 271–292.

Nason, G. (2016) wavethresh: Wavelets Statistics and Transforms. R package version 4.6.8.

https://CRAN.R-project.org/package=wavethresh

See Also

`mvEWS`, `coherence`, `rmvLSW`, `as.mvLSW`, `mvEWS`

Examples

```
#
# See examples in individual help pages
#
```

`plot.mvLSW` *Plot mvLSW Object*

Description

Plot the data contained within a `mvLSW` object based on the requested format.

Usage

```
  ## S3 method for class 'mvLSW'
plot(x, style = 1, info = NULL, Interval = NULL,
    diag = TRUE, sub = "Spectrum", ...)
```

Arguments

`x`	A `mvLSW` object.
`style`	Index stating the type of plotting format for the `mvLSW` object. (See details.)
`info`	Vector containing the channel and/or level indices defining the slice through `x` according to the requested plotting `style`. (See details.)
`Interval`	A list containing two items, both `mvLSW` objects with names "L" and "U" that respectively define the lower and upper pointwise interval values. If `NULL`, default, then no interval is plotted.
`diag`	Logical, should the diagonal panels be drawn when `style=2`. Ideally this should be `FALSE` if `object` contains the coherence. Set to `TRUE` by default.
`sub`	Plot subtitle. Set to `"Spectrum"` by default.
`...`	Additional graphical parameters.

Details

This command plots the data contained within the `mvLSW` based on requested plotting style.

Plotting style `style=1` with information `info=c(p,q,j)` generates a single plot for a specified channel pair `p` & `q` and level `j`.

Plotting style `style=2` with information `info=j` creates a set of plots from `x` for all channel pairs in a lower-triangular panel corresponding to the specified level `j`. If `diag=FALSE` then the plots along the diagonal are suppressed, which is ideal when `x` contain coherence estimates.

Plotting style `style=3` with information `info=c(p,q)` creates a set of plots from `x` for all levels (from fine to coarse) for channel pair `p` and `q`.

Finally, the plotting style `style=4` with information `info=c(p,q)` presents the same information as for the previous case, but in a compact matrix format. Please refer to `image.plot` from the `fields` library for additional information on this plotting style.

The argument Interval must be supplied in order to draw a polygon depicting the pointwise interval. See ApxCI for deriving an approximate confidence interval for the evolutionary wavelet spectrum estimate. This argument is ignored in the case style=4.

Value

Generates a plot. No data is returned.

See Also

plot.default, image.plot, as.mvLSW, mvEWS, coherence, ApxCI.

Examples

```
## Define evolutionary wavelet spectrum, structure only on level 2
Spec <- array(0, dim=c(3, 3, 8, 256))
Spec[1, 1, 2, ] <- 10
Spec[2, 2, 2, ] <- c(rep(5, 64), rep(0.6, 64), rep(5, 128))
Spec[3, 3, 2, ] <- c(rep(2, 128), rep(8, 128))
Spec[2, 1, 2, ] <- Spec[1, 2, 2, ] <- punif(1:256, 65, 192)
Spec[3, 1, 2, ] <- Spec[1, 3, 2, ] <- c(rep(-1, 128), rep(5, 128))
Spec[3, 2, 2, ] <- Spec[2, 3, 2, ] <- -0.5
EWS <- as.mvLSW(x = Spec, filter.number = 1, family = "DaubExPhase",
  min.eig.val = NA)

## Sample time series and estimate the EWS and coherence.
set.seed(10)
X <- rmvLSW(Spectrum = EWS)
EWS_X <- mvEWS(X, kernel.name = "daniell", kernel.param = 20)
RHO_X <- coherence(EWS_X, partial = FALSE)

## Evaluate asymptotic spectral variance
SpecVar <- varEWS(EWS_X)

## Evaluate 95% approximate confidence interval
CI <- ApxCI(object = EWS_X, var = SpecVar, alpha=0.05)

## Plot mvEWS between channels 1 & 3 at level 2
plot(x = EWS_X, style = 1, info = c(1, 3, 2), Interval = CI)

## Plot coherence between channels 1 & 3 at level 2
plot(x = RHO_X, style = 1, info = c(1, 3, 2), ylab = "Coherence")

## mvEWS panel plot for level 2
plot(x = EWS_X, style = 2, info = 2, Interval = CI)

## Panel plot of coherence for level 2
plot(x = RHO_X, style = 2, info = 2, diag = FALSE, ylab = "Coherence")

## Plot mvEWS for channel pair 1 & 3 at all levels
plot(x = EWS_X, style = 3, info = c(1, 3), Interval = CI)
```

```
## Plot coherence for channel pair 1 & 3 at all levels
plot(x = RHO_X, style = 3, info = c(1, 3), ylab = "Coherence")

## Image plot for coherence between channels 1 & 3
plot(x = RHO_X, style = 4, info = c(1, 3), sub = "Coherence")
```

---

rmvLSW *Sample a Multivariate Locally Stationary Wavelet Process*

Description

Sample a multivariate locally stationary wavelet process.

Usage

```
rmvLSW(Transfer = NULL, Spectrum = NULL, noiseFN = rnorm, ...)

## S3 method for class 'mvLSW'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

Transfer	A mvLSW object containing the set of transfer function matrices of the process.
Spectrum, object	A mvLSW object containing the multivariate evolutionary wavelet spectrum of the process. This argument is only used if Transfer is not supplied.
noiseFN	The function for sampling the innovations.
nsim	Number of mvLSW time series to draw. Only nsim = 1 is accepted.
seed	Seed for the random number generator.
...	Optional arguments to be passed to the function for sampling the innovation process.

Details

Samples a single multivariate locally stationary wavelet time series for the given set of transfer function matrices. These are assumed to be lower-triangular (including diagonal) matrices. If the mvEWS is supplied instead, then this is pre-processed by Spectrum2Transfer() to obtain the transfer function matrices.

The Transfer and Spectrum are both mvLSW objects and therefore contain information about defining the wavelet function.

The innovation process is assumed to be second order stationary with expectation zero, orthogonal and unit variance. The first argument of noiseFN must be n and define the number of samples to generate. The function must also return a numerical vector of length n.

The simulate command implements rmvLSW under default arguments unless specified via ....

Value

A ts matrix object of a multivariate locally stationary time series. The columns of the matrix correspond to different channels and the rows identify the time axis.

References

Park, T., Eckley, I. and Ombao, H.C. (2014) Estimating time-evolving partial coherence between signals via multivariate locally stationary wavelet processes. *Signal Processing, IEEE Transactions on* **62**(20) pp. 5240-5250.

See Also

mvLSW, Spectrum2Transfer, rnorm, AvBasis, ts.

Examples

```
## Define evolutionary wavelet spectrum, structure only on level 2
Spec <- array(0, dim = c(3, 3, 8, 256))
Spec[1, 1, 2, ] <- 10
Spec[2, 2, 2, ] <- c(rep(5, 64), rep(0.6, 64), rep(5, 128))
Spec[3, 3, 2, ] <- c(rep(2, 128), rep(8, 128))
Spec[2, 1, 2, ] <- Spec[1, 2, 2, ] <- punif(1:256, 65, 192)
Spec[3, 1, 2, ] <- Spec[1, 3, 2, ] <- c(rep(-1, 128), rep(5, 128))
Spec[3, 2, 2, ] <- Spec[2, 3, 2, ] <- -0.5

## Define Haar wavelet function and create mvLSW object
EWS <- as.mvLSW(x = Spec, filter.number = 1, family = "DaubExPhase",
  min.eig.val = NA)
plot(EWS, style = 2, info = 2)

## Sample with Gaussian innovations
set.seed(10)
X <- rmvLSW(Spectrum = EWS)
plot(X)

## Alternatively:
X1 <- simulate(object = EWS)
plot(X1)

## Define smoother wavelet function and create mvLSW object
EWS2 <- as.mvLSW(x = Spec, filter.number = 10, family = "DaubExPhase")

## Sample with logistic innovations
set.seed(10)
X2 <- rmvLSW(Spectrum = EWS2, noiseFN = rlogis, scale = sqrt(3)/pi)
plot(X2)
```

`Spectrum2Transfer` *Convert Between mvEWS and Transfer Function Matrices*

Description

Convert between multivariate evolutionary wavelet spectrum and the set of transfer function matrices.

Usage

```
Spectrum2Transfer(object, S2V = TRUE)
```

Arguments

`object`	A `mvLSW` object containing either the mvEWS or matrix transfer function.
`S2V`	Logical, if `TRUE` (default) then `object` is the mvEWS and the set of transfer function matrices are to be derived. If `FALSE` the `object` is the set of transfer function matrices and the converse transformation is derived.

Details

If the mvEWS is supplied, then the set of transfer function matrices are derived by the Choleski factorization of a real symmetric semi-positive definite square matrix. In the cases where the matrix is semi-definite, then the Choleski factorization is applied to the submatrix that is positive definite and the remaining lower triangular elements are populated such that the resulting matrix is a valid factorization.

Conversely, if the set of transfer function matrices are supplied, then the EWS are derived by squaring the matrices.

Value

A `mvLSW` object containing either the mvEWS or set of transfer function matrices depending on the specified transformation direction.

References

Park, T., Eckley, I. and Ombao, H.C. (2014) Estimating time-evolving partial coherence between signals via multivariate locally stationary wavelet processes. *Signal Processing, IEEE Transactions on* **62**(20) pp. 5240-5250.

See Also

`chol`, `as.mvLSW`, `mvEWS`.

Examples

```
## Define evolutionary wavelet spectrum, structure only on level 2
Spec <- array(0, dim=c(3, 3, 8, 256)) ## Ensure all are positive def.
Spec[1, 1, 2, ] <- 10
Spec[2, 2, 2, ] <- c(rep(5, 64), rep(0.6, 64), rep(5, 128))
Spec[3, 3, 2, ] <- c(rep(2, 128), rep(8, 128))
Spec[2, 1, 2, ] <- Spec[1, 2, 2, ] <- punif(1:256, 65, 192)
Spec[3, 1, 2, ] <- Spec[1, 3, 2, ] <- c(rep(-1, 128), rep(5, 128))
Spec[3, 2, 2, ] <- Spec[2, 3, 2, ] <- -0.5

## Define EWS as mvLSW object
EWS <- as.mvLSW(x = Spec, filter.number = 1, family = "DaubExPhase",
  min.eig.val = NA)
plot(EWS, style = 2, info = 2)

## EWS to Transfer function matrices
Transfer <- Spectrum2Transfer(object = EWS, S2V = TRUE)

## Transfer function matrices to EWS
EWS2 <- Spectrum2Transfer(object = Transfer, S2V = FALSE)
plot(EWS2, style = 2, info = 2)
```

`summary.mvLSW` *Print a Summary of mvLSW Object*

Description

Prints a summary of the information contained within a `mvLSW` classed object.

Usage

```
  ## S3 method for class 'mvLSW'
summary(object, ...)
  ## S3 method for class 'mvLSW'
print(x, ...)
```

Arguments

`object, x`	A `mvLSW` object.
`...`	Additional arguments.

Details

The command prints to screen a summary of the information contained within a `mvLSW` object. Information printed includes: dimensions, wavelet function, the smoothing regime applied, smoothing kernel(s), generalized cross-validation gamma deviance criteria score, application of the bias correction and minimum eigenvalue from across all spectral matrices.

Value

This command returns nothing, only prints a summary to the console.

See Also

mvEWS, as.mvLSW.

Examples

```
## Generate a bivariate time series
set.seed(100)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

## Haar wavelet, apply same smoothing to all levels & optimize
EWS <- mvEWS(X, kernel.name = "daniell", kernel.param = 20,
  optimize = TRUE)
summary(EWS)
print(EWS)
plot(EWS, style = 2, info = 1)
```

---

varEWS *Asymptotic Variance of the mvEWS Estimate*

Description

Calculates the asymptotic variance of a multivariate evolutionary wavelet spectrum estimate.

Usage

```
varEWS(object, ACWIP = NULL, verbose = FALSE)
```

Arguments

object	A mvLSW object containing the multivariate evolutionary wavelet spectrum. Matrices must be positive definite, i.e. information item min.eig.val must be greater than zero.
ACWIP	4D array containing the wavelet autocorrelation inner product functions. Set to NULL by default and therefore evaluated within the command based on the information supplied by object.
verbose	Logical. Controls the printing of messages whist the computations progress. Set as FALSE as default.

Details

The varEWS commands evaluate the asymptotic variance of a multivariate evolutionary wavelet spectrum (mvEWS) estimate. Note, the variance is only applicable when the mvEWS is smoothed consistently across all levels with list item smooth.type="all". This can be written in terms of the smoothed periodogram relating to the bias correction of the mvEWS estimate, where $A_{j,k}$ is the inner product matrix of the wavelet autocorrelation function:

$$Var(\hat{S}_{j,k}^{(p,q)}) = \sum_{l_1,l_2=1}^{J} (A^{-1})_{j,l_1}(A^{-1})_{j,l_2} Cov(\tilde{I}_{l_1,k}^{(p,q)}, \tilde{I}_{l_1,k}^{(p,q)})$$

The covariance between elements of the smoothed periodogram can also be expressed in terms of the raw wavelet periodogram:

$$Cov(\tilde{I}_{l_1,k}^{(p,q)}, \tilde{I}_{l_1,k}^{(p,q)}) = \sum_{m_1,m_2} W_{m_1} W_{m_2} Cov(I_{l_1,m_1}^{(p,q)}, I_{l_2,m_2}^{(p,q)})$$

The weights W_i, for integer i, define the smoothing kernel function that is evaluated by the kernel command. Note that $W_i = W_{-i}$ and $\sum_i W_i = 1$.

The final step is to derive the covariance of the raw periodogram. This has a long derivation, which can be concisely calculated by:

$$Cov(I_{j,k}^{(p,q)}, I_{l,m}^{(p,q)}) = E(p,j,k,q,l,m)^2 + E(p,j,k,p,l,m)E(q,j,k,q,l,m)$$

where

$$E(p,j,k,q,l,m) = \sum_{h=1}^{J} A_{j,l,h}^{k-m} S_h^{(p,q)}((k+m)/2T)$$

Here, $A_{j,l,h}^{\lambda}$ defines the autocorrelation wavelet inner product function and $S_j^{(p,q)}(k/T)$ is the true spectrum of the process between channels p & q, level j and location k. The true spectrum is not always available and so this may be substituted with the smoothed and bias corrected mvEWS estimate. For practical purposes, if k+m is odd then the average between the available spectrum values at neighbouring locations are substituted.

For efficiency purpose, if the varEWS command is going to be called multiple times then it is highly recommended that the autocorrelation wavelet inner product should be evaluated beforehand by AutoCorrIP and supplied via the ACWIP argument.

Value

Invisibly returns a mvLSW object containing the asymptotic variance of the multivariate evolutionary wavelet spectrum.

References

Park, T. (2014) Wavelet Methods for Multivariate Nonstationary Time Series, PhD thesis, Lancaster University, pp. 91-111.

See Also

ipndacw, AutoCorrIP, as.mvLSW, mvEWS

Examples

```
## Define evolutionary wavelet spectrum, structure only on level 2
Spec <- array(0, dim=c(3, 3, 8, 256))
Spec[1, 1, 2, ] <- 10
Spec[2, 2, 2, ] <- c(rep(5, 64), rep(0.6, 64), rep(5, 128))
Spec[3, 3, 2, ] <- c(rep(2, 128), rep(8, 128))
Spec[2, 1, 2, ] <- Spec[1, 2, 2, ] <- punif(1:256, 65, 192)
Spec[3, 1, 2, ] <- Spec[1, 3, 2, ] <- c(rep(-1, 128), rep(5, 128))
Spec[3, 2, 2, ] <- Spec[2, 3, 2, ] <- -0.5
EWS <- as.mvLSW(x = Spec, filter.number = 1, family = "DaubExPhase",
  min.eig.val = NA)

## Sample time series and estimate the EWS.
set.seed(10)
X <- rmvLSW(Spectrum = EWS)
EWS_X <- mvEWS(X, kernel.name = "daniell", kernel.param = 20)

## Evaluate asymptotic spectral variance
SpecVar <- varEWS(EWS_X)

## Plot Estimate & 95% confidence interval
CI <- ApxCI(object = EWS_X, var = SpecVar, alpha = 0.05)
plot(x = EWS_X, style = 2, info = 2, Interval = CI)
```

Index