Package ‘multiwave’

November 24, 2016

Version 1.2
Date 2016-11-24
Title Estimation of Multivariate Long-Memory Models Parameters
Maintainer Sophie Achard <sophie.achard@gipsa-lab.fr>
Description Computation of an estimation of the long-memory parameters and
the long-run covariance matrix using a multivariate model
(Lobato (1999) <doi:10.1016/S0304-4076(98)00038-
4>; Shimotsu (2007) <doi:10.1016/j.jeconom.2006.01.003>). Two semi-parametric methods are
implemented: a Fourier based approach (Shi-
motsu (2007) <doi:10.1016/j.jeconom.2006.01.003>) and a wavelet based
License GPL (>= 2)
NeedsCompilation no
Author Sophie Achard [aut, cre],
Irene Gannaz [aut]
Repository CRAN
Date/Publication 2016-11-24 11:58:00

R topics documented:

  multiwave-package .............................................. 2
  brainHCP ....................................................... 4
  compute_nj ..................................................... 5
  DWTexact ....................................................... 6
  fivarma ........................................................ 7
  K_eval .......................................................... 9
  mfw ............................................................ 10
  mfw_cov_eval .................................................. 11
  mfw_eval ....................................................... 12
  mww ............................................................ 14
  mww_cov_eval .................................................. 15
  mww_eval ....................................................... 17
  mww_wav ....................................................... 18
multiwave-package

Estimation of multivariate long-memory models parameters: memory parameters and long-run covariance matrix (also called fractal connectivity).

Description

This package computes an estimation of the long-memory parameters and the long-run covariance matrix using a multivariate model (Lobato, 1999; Shimotsu 2007). Two semi-parametric methods are implemented: a Fourier based approach (Shimotsu 2007) and a wavelet based approach (Achard and Gannaz 2014).

Details

Package: multiwave
Type: Package
Version: 1.0
Date: 2015-09-17
License: GPL (>= 2)

Author(s)

Sophie Achard and Irene Gannaz

Maintainer: Sophie Achard <sophie.achard@gipsa-lab.fr>, Irene Gannaz <irene.gannaz@insa-lyon.fr>

References


Examples
rho <- 0.4
cov <- matrix(c(1, rho, rho, 1), 2, 2)
d <- c(0.4, 0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix = cov)

x <- resp$x
long_run_cov <- resp$long_run_cov

### Compute wavelets this is also included in the functions without _wav
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
M <- res_filter$M
alpha <- res_filter$alpha

LU <- c(1, 11)

if(is.matrix(x)){
  N <- dim(x)[1]
  k <- dim(x)[2]
} else{
  N <- length(x)
  k <- 1
}
mat_x <- as.matrix(x, dim = c(N, k))

## Wavelet decomposition
xwav <- matrix(0, N, k)
for(j in 1:k){
  xx <- mat_x[, j]
  resw <- DWTexact(xx, filter)
  xwav_temp <- resw$dwt
  index <- resw$indmaxband
  Jmax <- resw$Jmax
  xwav[1:index[Jmax], j] <- xwav_temp;
}

## we free some memory
new_xwav <- matrix(0, min(index[Jmax], N), k)
if(index[Jmax] < N){
  new_xwav[1:(index[Jmax]), ] <- xwav[1:(index[Jmax]), ]
}
xwav <- new_xwav
index <- c(0, index)

#### Compute the wavelet functions
res_psi <- psi_hat_exact(filter, J)
psih <- res_psi$psih
grid <- res_psi$grid
### Estimate using Fourier

```r
m <- floor(N^0.65) # default value of Shimotsu
res_mfw <- mfw(x,m)
res_d_mfw <- res_mfw$d
res_rho_mfw <- res_mfw$cov[1,2]

### Eval MFW

res_mfw_eval <- mfw_eval(d,x,m)
res_mfw_cov_eval <- mfw_cov_eval(d,x,m)

### Estimate using Wavelets

## Using xwav

if(dim(xwav)[2]==1) xwav<-as.vector(xwav)
res_mww_wav <- mww_wav(xwav,index,psih,grid,LU)

### Eval MWW_wav

res_mww_wav_eval <- mww_wav_eval(d,xwav,index,LU)
res_mww_wav_cov_eval <- mww_wav_cov_eval(d,xwav,index,psih,grid,LU)

## Using directly the time series

res_mww <- mww(x,filter,LU)
res_d_mww <- res_mww$d
res_rho_mww <- res_mww$ cov[1,2]

### Eval MWW_wav

res_mww_eval <- mww_eval(d,x,filter,LU)
res_mww_cov_eval <- mww_cov_eval(d,x,filter,LU)
```

---

**brainHCP**

*Time series obtained by an fMRI experiment on the brain*

**Description**

Time series for each region of interest in the brain. These series are obtained by SPM preprocessing.

**Usage**

```r
data(brainHCP)
```

**Format**

A data frame with 1200 observations on the following 89 variables.
compute_nj

Source

contact S. Achard (sophie.achard@gipsa-lab.fr)

References


Examples

data(brainHCP)

```r
## maybe str(brainHCP) ; plot(brainHCP) ... 
```

<table>
<thead>
<tr>
<th>compute_nj</th>
<th>Wavelets coefficients utilities</th>
</tr>
</thead>
</table>

Description

Computes the number of wavelet coefficients at each scale.

Usage

`compute_nj(n, N)`

Arguments

- `n`: sample size.
- `N`: filter length.

Value

- `nj`: number of coefficients at each scale.
- `J`: Number of scales.

Author(s)

S. Achard and I. Gannaz

References


See Also

DWTexact, scaling_filter

Examples

res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
n <- 5^10
N <- length(filter)
compute_nj(n, N)

Description

Computes the discrete wavelet transform of the data using the pyramidal algorithm.

Usage

DWTexact(x, filter)

Arguments

x  vector of raw data
filter  Quadrature mirror filter (also called scaling filter, as returned by the scaling_filter function)

Value

dwt  computable Wavelet coefficients without taking into account the border effect.
indmaxband  vector containing the largest index of each band, i.e. for \( j > 1 \) the wavelet coefficients of scale \( j \) are \( \text{dwt}(k) \) for \( k \in [\text{indmaxband}(j-1) + 1, \text{indmaxband}(j)] \) and for \( j = 1 \), \( \text{dwt}(k) \) for \( k \in [1, \text{indmaxband}(1)] \).
Jmax  largest available scale index (=length of indmaxband).

Note

This function was rewritten from an original matlab version by Fay et al. (2009)

Author(s)

S. Achard and I. Gannaz
References


See Also

scaling_filter

Examples

```r
res_filter <- scaling_filter('Daubechies',8);
filter <- res_filter$h
u <- rnorm(2*10,0,1)
x <- vfracdiff(u,d=0.2)

resw <- DWTexact(x,filter)
xwav <- resw$dwt
index <- resw$indmaxband
Jmax <- resw$Jmax

# Wavelet scale 1
ws_1 <- xwav[1:index[1]]
# Wavelet scale 2
# Wavelet scale 3
### upto Jmax
```

fivarma

*simulation of FIVARMA process*

Description

Generates N observations of a realisation of a multivariate FIVARMA process X.

Usage

```r
fivarma(N, d = 0, cov_matrix = diag(length(d)), VAR = NULL,
        VMA = NULL, skip = 2000)
```
Arguments

N number of time points.

d vector of parameters of long-memory.

cov_matrix matrix of correlation between the innovations (optional, default is identity).

VAR array of VAR coefficient matrices (optional).

VMA array of VMA coefficient matrices (optional).

skip number of initial observations omitted, after applying the ARMA operator and the fractional integration (optional, the default is 2000).

Details

Let \((e(t))_t\) be a multivariate gaussian process with a covariance matrix cov_matrix. The values of the process \(X\) are given by the equations:

\[ \text{VAR}(L)U(t) = \text{VMA}(L)e(t), \]

and

\[ \text{diag}((1 - L)^d)X(t) = U(t) \]

where \(L\) is the lag-operator.

Value

x vector containing the \(N\) observations of the vector ARFIMA(arlags, \(d\), malags) process.

long_run_cov matrix of covariance of the spectral density of \(x\) around the zero frequency.

d vector of parameters of long-range dependence, modified in case of cointegration.

Author(s)

S. Achard and I. Gannaz

References


See Also

varma, vfracdiff
\textbf{K\_eval}

\textbf{Examples}

\begin{verbatim}
 rho1 <- 0.3
 rho2 <- 0.8
 cov <- matrix(c(1, rho1, rho2, rho1, 1, rho1, rho2, rho1, 1), 3, 3)
 d <- c(0.2, 0.3, 0.4)

 J <- 9
 N <- 2^J
 VMA <- diag(c(0.4, 0.1))
 ### or another example
 VAR <- array(c(0.8, 0, 0, 0.6, 0, 0, 0.2, 0, 0, 0.4, 0, 0, 0.5), dim = c(3, 3, 2))
 VAR <- diag(c(0.8, 0.6, 0))
 resp <- fivarma(N, d, cov_matrix = cov, VAR = VAR, VMA = VMA)
 x <- resp$x
 long_run_cov <- resp$long_run_cov
d <- resp$d
\end{verbatim}

\begin{tabular}{ll}
\textbf{K\_eval} & \textit{Evaluation of function $K$} \\
\end{tabular}

\textbf{Description}

Computes the function $K$ as defined in \cite{Achard2014}.

\textbf{Usage}

\texttt{K\_eval(psi\_hat, u, d)}

\textbf{Arguments}

- \texttt{psi\_hat} \hspace{1cm} Fourier transform of the wavelet mother at values $u$
- \texttt{u} \hspace{1cm} grid for the approximation of the integral
- \texttt{d} \hspace{1cm} vector of long-memory parameters.

\textbf{Details}

\texttt{K\_eval} computes the matrix $K$ with elements

$$K(d_i, d_m) = \int u^{(d_i + d_m)} |\text{psi\_hat}(u)|^2 du$$

\textbf{Value}

value of function $K$ as a matrix.
Author(s)

S. Achard and I. Gannaz

References


See Also

psi_hat_exact

Examples

```r
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
M <- res_filter$M
alpha <- res_filter$alpha
res_psi <- psi_hat_exact(filter, J=10)
K_eval(res_psi$psih, res_psi$grid, d=c(0.2,0.2))
```

Description

Computes the multivariate Fourier Whittle estimators of the long-memory parameters and the long-run covariance matrix also called fractal connectivity.

Usage

```r
mfw(x, m)
```

Arguments

- `x` : data (matrix with time in rows and variables in columns).
- `m` : truncation number used for the estimation of the periodogram.

Details

The choice of `m` determines the range of frequencies used in the computation of the periodogram, \( \lambda_j = 2\pi j / N, j = 1, \ldots, m \). The optimal value depends on the spectral properties of the time series such as the presence of short range dependence. In Shimotsu (2007), `m` is chosen to be equal to \( N^{0.65} \).

Value

- `d` : estimation of the vector of long-memory parameters.
Author(s)
S. Achard and I. Gannaz

References

See Also
mfw_eval, mfw_cov_eval

Examples
```r
### Simulation of ARFIMA(0,d,0)
rho <- 0.4
cov <- matrix(c(1,rho,rho,1),2,2)
d <- c(0.4,0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix=cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

m <- 57  # default value of Shimotsu 2007
res_mfw <- mfw(x, m)
```

Description
Computes the multivariate Fourier Whittle estimator of the long-run covariance matrix (also called fractal connectivity) for a given value of long-memory parameters \(d\).

Usage
```r
mfw_cov_eval(d, x, m)
```

Arguments
- **d**: vector of long-memory parameters (dimension should match dimension of \(x\))
- **x**: data (matrix with time in rows and variables in columns)
- **m**: truncation number used for the estimation of the periodogram
Details

The choice of m determines the range of frequencies used in the computation of the periodogram, $\lambda_j = 2\pi j/N$, $j = 1, \ldots, m$. The optimal value depends on the spectral properties of the time series such as the presence of short range dependence. In Shimotsu (2007), m is chosen to be equal to $\sqrt{\lambda}^{0.65}$.

Value

long-run covariance matrix estimation.

Author(s)

S. Achard and I. Gannaz

References


See Also

mfw_eval, mfw

Examples

```r
### Simulation of ARFIMA(0,\text{code}(d),0)
rho <- 0.4
cov <- matrix(c(1,rho,rho,1),2,2)
d <- c(0.4,0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix= cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

m <- 57 # default value of Shimotsu
G <- mfw_cov_eval(d, x, m) # estimation of the covariance matrix when d is known
```

Description

Evaluates the multivariate Fourier Whittle criterion at a given long-memory parameter value d.
Usage

mfw_eval(d, x, m)

Arguments

d vector of long-memory parameters (dimension should match dimension of x).
x data (matrix with time in rows and variables in columns).
m truncation number used for the estimation of the periodogram.

Details

The choice of m determines the range of frequencies used in the computation of the periodogram, \( \lambda_j = \frac{2\pi j}{N} \), \( j = 1, \ldots, m \). The optimal value depends on the spectral properties of the time series such as the presence of short range dependence. In Shimotsu (2007), m is chosen to be equal to \( N^{0.65} \).

Value

multivariate Fourier Whittle estimator computed at point d.

Author(s)

S. Achard and I. Gannaz

References


See Also

mfw_cov_eval, mfw

Examples

```R
### Simulation of ARFIMA(0,d,0)
rho <- 0.4
cov <- matrix(c(1, rho, rho, 1), 2, 2)
d <- c(0.4, 0.2)
j <- 9
N <- 2^j

resp <- fivarma(N, d, cov_matrix=cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

m <- 57 # default value of Shimotsu
res_mfw <- mfw(x, m)
```
multivariate wavelet Whittle estimation

Description

Computes the multivariate wavelet Whittle estimation for the long-memory parameter vector \( d \) and the long-run covariance matrix, using \texttt{DWTexact} for the wavelet decomposition.

Usage

\texttt{mww(x, filter, LU = NULL)}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} data (matrix with time in rows and variables in columns).
  \item \texttt{filter} \hspace{1cm} wavelet filter as obtain with \texttt{scaling_filter}.
  \item \texttt{LU} \hspace{1cm} bivariate vector (optional) containing \( L \), the lowest resolution in wavelet decomposition and \( U \), the maximal resolution in wavelet decomposition. (Default values are set to \( L=1 \), and \( U=J_{\max} \).)
\end{itemize}

Details

\( L \) is fixing the lower limit of wavelet scales. \( L \) can be increased to avoid finest frequencies that can be corrupted by the presence of high frequency phenomena.

\( U \) is fixing the upper limit of wavelet scales. \( U \) can be decreased when highest frequencies have to be discarded.

Value

\begin{itemize}
  \item \texttt{d} \hspace{1cm} estimation of vector of long-memory parameters.
  \item \texttt{cov} \hspace{1cm} estimation of long-run covariance matrix.
\end{itemize}

Author(s)

S. Achard and I. Gannaz

References


\texttt{d <- res_mfw$d}
\texttt{G <- mfw_eval(d,x,m)}
\texttt{k <- length(d)}
\texttt{res_d <- optim(rep(0,k),mfw_eval,x=x,m=m,method='Nelder-Mead',lower=-Inf,upper=Inf)$par}
**mww_cov_eval**

**See Also**

`mww_eval`, `mww_cov_eval`, `mww_wav`, `mww_wav_eval`, `mww_wav_cov_eval`

**Examples**

```r
### Simulation of ARFIMA(0,d,0)
rho <- 0.4
cov <- matrix(c(1, rho, rho, 1), 2, 2)
d <- c(0.4, 0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix=cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

### Wavelet coefficients definition
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
M <- res_filter$M
alpha <- res_filter$alpha
LU <- c(2, 11)

res_mww <- mww(x, filter, LU)
```

**Description**

Computes the multivariate wavelet Whittle estimation of the long-run covariance matrix given the long-memory parameter vector `d`, using DWTeexact for the wavelet decomposition.

**Usage**

`mww_cov_eval(d, x, filter, LU)`

**Arguments**

- `d`: vector of long-memory parameters (dimension should match dimension of `x`).
- `x`: data (matrix with time in rows and variables in columns).
- `filter`: wavelet filter as obtain with `scaling_filter`.
- `LU`: bivariate vector (optional) containing `L`, the lowest resolution in wavelet decomposition, `U`, the maximal resolution in wavelet decomposition.
Details

L is fixing the lower limit of wavelet scales. L can be increased to avoid finest frequencies that can be corrupted by the presence of high frequency phenomena.

\( L \) can be increased to avoid finest frequencies that can be corrupted by the presence of high frequency phenomena.

\( U \) is fixing the upper limit of wavelet scales. \( U \) can be decreased when highest frequencies have to be discarded.

Value

long-run covariance matrix estimation.

Author(s)

S. Achard and I. Gannaz

References


See Also

`mww, mww_eval, mww_wav, mww_wav_eval, mww_wav_cov_eval`

Examples

```r
### Simulation of ARFIMA(0,d,0)
rho <- 0.4
cov <- matrix(c(1,rho,rho,1),2,2)
d <- c(0.4,0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix=cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

### Wavelet coefficients definition
res_filter <- scaling_filter('Daubchies',8);
filter <- res_filter$h
M <- res_filter$M
alpha <- res_filter$alpha

LU <- c(2,11)

res_mww <- mww_cov_eval(d,x,filter,LU)
```
Description

Evaluates the multivariate wavelet Whittle criterion at a given long-memory parameter vector \( d \) using \texttt{DWTexact} for the wavelet decomposition.

Usage

\begin{verbatim}
mww_eval(d, x, filter, LU = NULL)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{d} vector of long-memory parameters (dimension should match dimension of \texttt{x}).
  \item \texttt{x} data (matrix with time in rows and variables in columns).
  \item \texttt{filter} wavelet filter as obtain with \texttt{scaling_filter}.
  \item \texttt{LU} bivariate vector (optional) containing \( L \), the lowest resolution in wavelet decomposition \( U \), the maximal resolution in wavelet decomposition. (Default values are set to \( L = 1 \), and \( U = J_{\text{max}} \)).
\end{itemize}

Details

\( L \) is fixing the lower limit of wavelet scales. \( L \) can be increased to avoid finest frequencies that can be corrupted by the presence of high frequency phenomena.

\( U \) is fixing the upper limit of wavelet scales. \( U \) can be decreased when highest frequencies have to be discarded.

Value

multivariate wavelet Whittle criterion.

Author(s)

S. Achard and I. Gannaz

References


See Also

\texttt{mww, mww_cov_eval,mww_wav,mww_wav_eval,mww_wav_cov_eval}
### Examples

```r
### Simulation of ARFIMA(0,d,0)
rho <- 0.4
cov <- matrix(c(1,rho,rho,1),2,2)
d <- c(0.4,0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix=cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

### wavelet coefficients definition
res_filter <- scaling_filter('Daubechies',8);
filter <- res_filter$h
M <- res_filter$M
alpha <- res_filter$alpha

LU <- c(2,11)

res_mww <- mww_eval(d,x,filter,LU)
k <- length(d)
res_d <- optim(rep(0,k),mww_eval,x=x,filter=filter,
               LU=LU,method='Nelder-Mead',lower=-Inf,upper=Inf)$par
```

---

**mww_wav**

*multiwavelet wavelet Whittle estimation for data as wavelet coefficients*

### Description

Computes the multivariate wavelet Whittle estimation of the long-memory parameter vector `d` and the long-run covariance matrix for the already wavelet decomposed data.

### Usage

```r
mww_wav(xwav, index, psih, grid_K, LU = NULL)
```

### Arguments

- **xwav**
  - wavelet coefficients matrix (with scales in rows and variables in columns).
- **index**
  - vector containing the largest index of each band, i.e. for `j > 1` the wavelet coefficients of scale `j` are `dwt(k)` for `k ∈ [indmaxband(j-1) + 1, indmaxband(j)]` and for `j = 1`, `dwt(k)` for `k ∈ [1, indmaxband(1)]`.
- **psih**
  - the Fourier transform of the wavelet mother at values `grid_K`.
- **grid_K**
  - the grid for the approximation of the integral in `K`.
- **LU**
  - bivariate vector (optional) containing `L`, the lowest resolution in wavelet decomposition `U`, the maximal resolution in wavelet decomposition. (Default values are set to `L=1`, and `U=Jmax`).
Details

L is fixing the lower limit of wavelet scales. L can be increased to avoid finest frequencies that can be corrupted by the presence of high frequency phenomena.

U is fixing the upper limit of wavelet scales. U can be decreased when highest frequencies have to be discarded.

Value

d estimation of the vector of long-memory parameters.

 cov estimation of the long-run covariance matrix.

Author(s)

S. Achard and I. Gannaz

References


See Also

mww_eval, mww_cov_eval, mww_wav_eval, mww_wav_cov_eval

Examples

```r
### Simulation of ARFIMA(0, d, 0)
rho <- 0.4
cov <- matrix(c(1, rho, rho, 1), 2, 2)
d <- c(0.4, 0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix=cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

### wavelet coefficients definition
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
LU <- c(2, 11)

### wavelet decomposition

if(is.matrix(x)){
   N <- dim(x)[1]
   k <- dim(x)[2]
} else{
   N <- length(x)
```
mww_wav_cov_eval

multivariate wavelet Whittle estimation of the long-run covariance matrix

Description

Computes the multivariate wavelet Whittle estimation of the long-run covariance matrix given the long-memory parameter vector \( \mathbf{d} \) for the already wavelet decomposed data.

Usage

\[
\text{mww_wav_cov_eval}(\mathbf{d}, \text{xwav}, \text{index}, \text{psih}, \text{grid}_K, \text{LU})
\]

Arguments

\( \mathbf{d} \) 
vector of long-memory parameters (dimension should match dimension of \( \text{xwav} \)).

\( \text{xwav} \) 
wavelet coefficients matrix (with scales in rows and variables in columns).
The index vector containing the largest index of each band, i.e., for \( j > 1 \) the wavelet coefficients of scale \( j \) are \( \text{dwt}(k) \) for \( k \in [\text{indmaxband}(j - 1) + 1, \text{indmaxband}(j)] \) and for \( j = 1 \), \( \text{dwt}(k) \) for \( k \in [1, \text{indmaxband}(1)] \).

The \( \text{psih} \) is the Fourier transform of the wavelet mother at values \( \text{grid}_K \).

The \( \text{grid}_K \) is the grid for the approximation of the integral in \( K \).

The \( \text{LU} \) is a bivariate vector (optional) containing \( L \), the lowest resolution in wavelet decomposition, and \( U \), the maximal resolution in wavelet decomposition.

**Details**

- \( L \) is fixing the lower limit of wavelet scales. \( L \) can be increased to avoid finest frequencies that can be corrupted by the presence of high frequency phenomena.
- \( U \) is fixing the upper limit of wavelet scales. \( U \) can be decreased when highest frequencies have to be discarded.

**Value**

- Long-run covariance matrix estimation.

**Author(s)**

S. Achard and I. Gannaz

**References**


**See Also**

- `mww`, `mww_eval`, `mww_wav`, `mww_wav_eval`, `mww_cov_eval`

**Examples**

```r
# Simulation of ARFIMA(0,d,0)
rho <- 0.4
cov <- matrix(c(1, rho, rho, 1), 2, 2)
d <- c(0.4, 0.2)
J <- 9
N <- 2^J

resp <- fivarma(N, d, cov_matrix=cov)
x <- resp$x
long_run_cov <- resp$long_run_cov

# wavelet coefficients definition
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
M <- res_filter$M
alpha <- res_filter$alpha
```
mww_wav_eval

multivariate wavelet Whittle estimation for data as wavelet coefficients

Description

Evaluates the multivariate wavelet Whittle criterion at a given long-memory parameter vector d for the already wavelet decomposed data.
Usage

mww_wav_eval(d, xwav, index, LU = NULL)

Arguments

d vector of long-memory parameters (dimension should match dimension of x).

xwav wavelet coefficients matrix (with scales in rows and variables in columns).

index vector containing the largest index of each band, i.e. for j > 1 the wavelet coefficients of scale j are dwt(k) for k ∈ [indmaxband(j-1) + 1, indmaxband(j)] and for j = 1, dwt(k) for k ∈ [1, indmaxband(1)].

LU bivariate vector (optional) containing L, the lowest resolution in wavelet decomposition U, the maximal resolution in wavelet decomposition. (Default values are set to L=1, and U=Jmax.)

Details

L is fixing the lower limit of wavelet scales. L can be increased to avoid finest frequencies that can be corrupted by the presence of high frequency phenomena.

U is fixing the upper limit of wavelet scales. U can be decreased when highest frequencies have to be discarded.

Value

multivariate wavelet Whittle criterion.

Author(s)

S. Achard and I. Gannaz

References


See Also

mww, mww_cov_eval,mww_wav,mww_wav_eval,mww_wav_cov_eval

Examples

### Simulation of ARFIMA(0,d,0)

rho <- 0.4
cov <- matrix(c(1,rho,rho,1),2,2)
d <- c(0.4,0.2)
j <- 9
N <- 2^j
resp <- fivarma(N, d, cov_matrix= cov)
X <- resp$x
long_run_cov <- resp$long_run_cov

## wavelet coefficients definition
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
LU <- c(2, 11)

### wavelet decomposition
if(is.matrix(x)){
  N <- dim(x)[1]
  k <- dim(x)[2]
} else{
  N <- length(x)
  k <- 1
}
x <- as.matrix(x, dim=c(N, k))

## Wavelet decomposition
xwav <- matrix(0, N, k)
for(j in 1:k){
  xx <- x[, j]
  resw <- DWTexact(xx, filter)
xwav_temp <- resw$dwt
  index <- resw$indmaxband
  Jmax <- resw$Jmax
  xwav[1:index[Jmax], j] <- xwav_temp;
}

## we free some memory
new_xwav <- matrix(0, min(index[Jmax], N), k)
if(index[Jmax]<N){
  new_xwav[1:index[Jmax],] <- xwav[1:index[Jmax],]
}
xwav <- new_xwav
index <- c(0, index)

res_mww <- mww_wav_eval(d, xwav, index, LU)
res_d <- optim(rep(0, k), mww_wav_eval, xwav=xwav, index=index, LU=LU,
  method='Nelder-Mead', lower=-Inf, upper=Inf)$par

### psi_hat_exact: discrete Fourier transform of the wavelet
**psi_hat_exact**

**Description**

Computes the discrete Fourier transform of the wavelet associated to the given filter using `scaling_function`. The length of the Fourier transform is equal to the length of the grid where the wavelet is evaluated.

**Usage**

```r
psi_hat_exact(filter, J=10)
```

**Arguments**

- `filter`:
  - wavelet filter as obtained with `scaling_filter`.
- `J`:
  - $2^J$ corresponds to the size of the grid for the discretisation of the wavelet. The default value is set to 10.

**Value**

- `psih`:
  - Values of the discrete Fourier transform of the wavelet.
- `grid`:
  - Frequencies where the Fourier transform is evaluated.

**Author(s)**

S. Achard and I. Gannaz

**References**


**See Also**

`DWTexact`, `scaling_filter`

**Examples**

```r
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
psi_hat_exact(filter, J=6)
```
Description
Computes the filter coefficients of the Haar or Daubechies wavelet family with a specific order

Usage
scaling_filter(family, parameter)

Arguments
family Wavelet family, 'Haar' or 'Daubechies'
parameter Order of the Daubechies wavelet (equal to twice the number of vanishing moments). The value of parameter can be 2, 4, 8, 10, 12, 14 and 16.

Value
h Vector of scaling filter coefficients.
M Number of vanishing moments.
alpha Fourier decay exponent.

Author(s)
S. Achard and I. Gannaz

References

See Also
dwtexact

Examples
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
M <- res_filter$M
alpha <- res_filter$alpha
scaling_function

scaling_function and the wavelet function

Description
Computes the scaling function and the wavelet function (for compactly supported wavelet) using the cascade algorithm on the grid of dyadic integer \(2^{-J}\).

Usage
scaling_function(filter, J)

Arguments
- filter: wavelet filter as obtained with scaling_filter.
- J: value of the largest scale.

Value
- phi: Scaling function.
- psi: Wavelet function.

Note
This function was rewritten from an original matlab version by Fay et al. (2009)

Author(s)
S. Achard and I. Gannaz

References

See Also
dWTexact, scaling_filter

Examples
res_filter <- scaling_filter('Daubechies', 8);
filter <- res_filter$h
scaling_function(filter, J=6)
toeplitz_nonsym  

*Transform a vector in a non symmetric Toeplitz matrix*

---

**Description**

Transform a vector in a non symmetric Toeplitz matrix

**Usage**

```r
toeplitz_nonsym(vec)
```

**Arguments**

- `vec`  
  input vector.

**Value**

the corresponding matrix.

**Author(s)**

S. Achard and I. Gannaz

**References**


**See Also**

- `scaling_function`

**Examples**

```r
res_filter <- scaling_filter('Daubechies',8);
filter <- res_filter$h
htmp <- toeplitz_nonsym(filter)
```
varma simulation of multivariate ARMA process

Description
generates N observations of a k-vector ARMA process

Usage

\[
\text{varma}(N, k = 1, \text{VAR} = \text{NULL}, \text{VMA} = \text{NULL}, \text{cov\_matrix} = \text{diag}(k), \text{innov} = \text{NULL})
\]

Arguments

- \(N\) number of time points.
- \(k\) dimension of the vector ARMA (optional, default is univariate).
- \(\text{VAR}\) array of VAR coefficient matrices (optional).
- \(\text{VMA}\) array of VMA coefficient matrices (optional).
- \(\text{cov\_matrix}\) matrix of correlation between the innovations (optional, default is identity).
- \(\text{innov}\) matrix of the innovations (optional, default is a gaussian process).

Value

vector containing the N observations of the k-vector ARMA process.

Author(s)

S. Achard and I. Gannaz

References


See Also

fivarma, vfracdiff

Examples

```r
rho1 <- 0.3
rho2 <- 0.8
cov <- matrix(c(1, rho1, rho2, rho1, 1, rho1, rho2, rho1, 1), 3, 3)

J <- 9
N <- 2^J
VMA <- diag(c(0.4, 0.1))
```
### or another example

```
VAR <- array(c(0.8,0,0,0.6,0,0,2,0,0,0,0,0,0,0,0,0,0.5),dim=c(3,3,2))
VAR <- diag(c(0.8,0.6,0))
x <- varma(N, k=3, cov_matrix=cov, VAR=VAR, VMA=VMA)
```

---

**vfracdiff**

*simulation of vector fractional differencing process*

**Description**

Given a vector process \(x\) and a vector of long memory parameters \(d\), this function is producing the corresponding fractional differencing process.

**Usage**

```r
vfracdiff(x, d)
```

**Arguments**

- **x**: initial process.
- **d**: vector of long-memory parameters

**Details**

Given a process \(x\), this function applied a fractional difference procedure using the formula:

\[
\text{diag}((1 - L)^d)x,
\]

where \(L\) is the lag operator.

**Value**

vector fractional differencing of \(x\).

**Author(s)**

S. Achard and I. Gannaz

**References**


**See Also**

`varma`, `fivarma`
Examples

```r
rho1 <- 0.3
rho2 <- 0.8
cov <- matrix(c(1,rho1,rho2,rho1,1,rho1,rho2,rho1,1),3,3)
d <- c(0.2,0.3,0.4)

J <- 9
N <- 2^J
VMA <- diag(c(0.4,0.1,0))
### or another example
VAR <- array(c(0.8,0,0,0.6,0,0,0.2,0,0,0,0.4,0,0,0.05),dim=c(3,3,2))
VMA <- diag(c(0.8,0.6,0))
x <- varma(N, k=3, cov_matrix=cov, VAR=VAR, VMA=VMA)
x<-vfracdiff(x,d)
```
## Index

**Topic datagen**
- fivarma, 7
- varma, 29
- vfracdiff, 30

**Topic datasets**
- brainHCP, 4

**Topic nonparametric**
- DWTexact, 6
- mfw, 10
- mfw_cov_eval, 11
- mfw_eval, 12
- mww, 14
- mww_cov_eval, 15
- mww_eval, 17
- mww_wav, 18
- mww_wav_cov_eval, 20
- mww_wav_eval, 22

**Topic package**
- multiwaveMpackage, 2

**Topic ts**
- compute_nj, 5
- DWTexact, 6
- fivarma, 7
- K_eval, 9
- mfw, 10
- mfw_cov_eval, 11
- mfw_eval, 12
- mww, 14
- mww_cov_eval, 15
- mww_eval, 17
- mww_wav, 18
- mww_wav_cov_eval, 20
- mww_wav_eval, 22
- psi_hat_exact, 10, 24
- scaling_filter, 6, 7, 25, 26, 27
- scaling_function, 27, 28
- toeplitz nonsym, 28
- varma, 8, 29, 30
- vfracdiff, 8, 29, 30

brainHCP, 4
compute_nj, 5
DWTexact, 6
fivarma, 7, 29, 30
K_eval, 9
mfw, 10, 12, 13
mfw_cov_eval, 11, 11, 13
mfw_eval, 11, 12, 12
multiwave (multiwave-package), 2
multiwave-package, 2
mww, 14, 16, 17, 19, 21, 23
mww_cov_eval, 15, 15, 17, 19, 21, 23
mww_eval, 15, 16, 17, 19, 21, 23
mww_wav, 15–17, 18, 21, 23
mww_wav_cov_eval, 15–17, 19, 20, 23
mww_wav_eval, 15–17, 19, 21, 22
psi_hat_exact, 10, 24
scaling_filter, 6, 7, 25, 26, 27
scaling_function, 27, 28
toeplitz nonsym, 28
varma, 8, 29, 30
vfracdiff, 8, 29, 30