Package ‘waveslim’

January 26, 2019

Version 1.7.5.1

Date 2014-12-21

Title Basic Wavelet Routines for One-, Two- And Three-Dimensional Signal Processing

Author Brandon Whitcher

Maintainer Brandon Whitcher <bwhitcher@gmail.com>

Depends R (>= 2.11.0), graphics, grDevices, stats, utils

Suggests fftw

Description Basic wavelet routines for time series (1D), image (2D) and array (3D) analysis. The code provided here is based on wavelet methodology developed in Percival and Walden (2000); Gencay, Selcuk and Whitcher (2001); the dual-tree complex wavelet transform (DTCWT) from Kingsbury (1999, 2001) as implemented by Selesnick; and Hilbert wavelet pairs (Selesnick 2001, 2002). All figures in chapters 4-7 of GSW (2001) are reproducible using this package and R code available at the book website(s) below.

URL http://waveslim.blogspot.com,
http://www2.imperial.ac.uk/~bwhitche/book

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-01-26 16:48:57 UTC

R topics documented:

`Andel` *Autocovariance and Autocorrelation Sequences for a Seasonal Persistent Process*

Description

The autocovariance and autocorrelation sequences from the time series model in Figures 8, 9, 10, and 11 of Andel (1986). They were obtained through numeric integration of the spectral density function.

Usage

```
data(acvs.andel8)
data(acvs.andel9)
data(acvs.andel10)
data(acvs.andel11)
```

Format

A data frame with 4096 rows and three columns: lag, autocovariance sequence, autocorrelation sequence.

References

Andel, J. (1986) Long memory time series models, *Kypernetika*, **22**, No. 2, 105-123.

ar1 — *Simulated AR(1) Series*

Description

Simulated AR(1) series used in Gencay, Selcuk and Whitcher (2001).

Usage

```
data(ar1)
```

Format

A vector containing 200 observations.

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Band-pass variance — *Bandpass Variance for Long-Memory Processes*

Description

Computes the band-pass variance for fractional difference (FD) or seasonal persistent (SP) processes using numeric integration of their spectral density function.

Usage

```
bandpass.fdp(a, b, d)
bandpass.spp(a, b, d, fG)
bandpass.spp2(a, b, d1, f1, d2, f2)
bandpass.var.spp(delta, fG, J, Basis, Length)
```

Arguments

`a`	Left-hand boundary for the definite integral.
`b`	Right-hand boundary for the definite integral.
`d,delta,d1,d2`	Fractional difference parameter.
`fG,f1,f2`	Gegenbauer frequency.
`J`	Depth of the wavelet transform.
`Basis`	Logical vector representing the adaptive basis.
`Length`	Number of elements in Basis.

Details

See references.

Value

Band-pass variance for the FD or SP process between a and b.

Author(s)

Brandon Whitcher

References

McCoy, E. J., and A. T. Walden (1996) Wavelet analysis and synthesis of stationary long-memory processes, *Journal for Computational and Graphical Statistics*, **5**, No. 1, 26-56.

Whitcher, B. (2001) Simulating Gaussian stationary processes with unbounded spectra, *Journal for Computational and Graphical Statistics*, **10**, No. 1, 112-134.

`barbara` *Barbara Test Image*

Description

The Barbara image comes from Allen Gersho's lab at the University of California, Santa Barbara.

Usage

```
data(barbara)
```

Format

A 256×256 matrix.

Source

Internet.

`basis` *Produce Boolean Vector from Wavelet Basis Names*

Description

Produce a vector of zeros and ones from a vector of basis names.

Usage

```
basis(x, basis.names)
```

Arguments

`x`	Output from the discrete wavelet package transfrom (DWPT).
`basis.names`	Vector of character strings that describe leaves on the DWPT basis tree. See the examples below for appropriate syntax.

Details

None.

Value

Vector of zeros and ones.

See Also

`dwpt`.

Examples

```
data(acvs.andel8)
## Not run:
x <- hosking.sim(1024, acvs.andel8[,2])
x.dwpt <- dwpt(x, "la8", 7)
## Select orthonormal basis from wavelet packet tree
x.basis <- basis(x.dwpt, c("w1.1","w2.1","w3.0","w4.3","w5.4","w6.10",
                           "w7.22","w7.23"))
for(i in 1:length(x.dwpt))
  x.dwpt[[i]] <- x.basis[i] * x.dwpt[[i]]
## Resonstruct original series using selected orthonormal basis
y <- idwpt(x.dwpt, x.basis)
par(mfrow=c(2,1), mar=c(5-1,4,4-1,2))
plot.ts(x, xlab="", ylab="", main="Original Series")
plot.ts(y, xlab="", ylab="", main="Reconstructed Series")

## End(Not run)
```

blocks — *A Piecewise-Constant Function*

Description

$$blocks(x) = \sum_{j=1}^{11} (1 + \text{sign}(x - p_j))h_j/2$$

Usage

```
data(blocks)
```

Format

A vector containing 512 observations.

Source

S+WAVELETS.

References

Bruce, A., and H.-Y. Gao (1996) *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

brick.wall — *Replace Boundary Wavelet Coefficients with Missing Values*

Description

Sets the first n wavelet coefficients to `NA`.

Usage

```
brick.wall(x, wf, method="modwt")
dwpt.brick.wall(x, wf, n.levels, method="modwpt")
```

Arguments

`x`	DWT/MODWT/DWPT/MODWPT object
`wf`	Character string; name of wavelet filter
`method`	Either `dwt` or `modwt` for `brick.wall`, or either `dwpt` or `modwpt` for `dwpt.brick.wall`
`n.levels`	depth of the wavelet transform

Details

The fact that observed time series are finite causes boundary issues. One way to get around this is to simply remove any wavelet coefficient computed involving the boundary. This is done here by replacing boundary wavelet coefficients with `NA`.

Value

Same object as `x` only with some missing values.

Author(s)

B. Whitcher

References

Lindsay, R. W., D. B. Percival and D. A. Rothrock (1996). The discrete wavelet transform and the scale anlaysis of the surface properties of sea ice, *IEEE Transactions on Geoscience and Remote Sensing*, **34**, No.~3, 771-787.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

`convolve2D` *Fast Column-wise Convolution of a Matrix*

Description

Use the Fast Fourier Transform to perform convolutions between a sequence and each column of a matrix.

Usage

```
convolve2D(x, y, conj = TRUE, type = c("circular", "open"))
```

Arguments

`x`	$M \times N$ matrix.
`y`	numeric sequence of length N.
`conj`	logical; if `TRUE`, take the complex *conjugate* before back-transforming (default, and used for usual convolution).
`type`	character; one of `circular`, `open` (beginning of word is ok). For `circular`, the two sequences are treated as *circular*, i.e., periodic. For `open` and `filter`, the sequences are padded with zeros (from left and right) first; `filter` returns the middle sub-vector of `open`, namely, the result of running a weighted mean of `x` with weights `y`.

Details

This is a corrupted version of `convolve` made by replacing `fft` with `mvfft` in a few places. It would be nice to submit this to the R Developers for inclusion.

Author(s)

Brandon Whitcher

See Also

`convolve`

`cpi` *U.S. Consumer Price Index*

Description

Monthly U.S. consumer price index from 1948:1 to 1999:12.

Usage

```
data(cpi)
```

Format

A vector containing 624 observations.

Source

Unknown.

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

dau *Digital Photograph of Ingrid Daubechies*

Description

A digital photograph of Ingrid Daubechies taken at the 1993 AMS winter meetings in San Antonio, Texas. The photograph was taken by David Donoho with a Canon XapShot video still frame camera.

Usage

```
data(dau)
```

Format

A 256×256 matrix.

Source

S+WAVELETS.

References

Bruce, A., and H.-Y. Gao (1996) *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

denoise.2d *Denoise an Image via the 2D Discrete Wavelet Transform*

Description

Perform simple de-noising of an image using the two-dimensional discrete wavelet transform.

Usage

```
denoise.dwt.2d(x, wf = "la8", J = 4, method = "universal", H = 0.5,
               noise.dir = 3, rule = "hard")
denoise.modwt.2d(x, wf = "la8", J = 4, method = "universal", H = 0.5,
                 rule = "hard")
```

Arguments

x	input matrix (image)
wf	name of the wavelet filter to use in the decomposition
J	depth of the decomposition, must be a number less than or equal to $\log_2(\min\{M, N\})$
method	character string describing the threshold applied, only "universal" and "long-memory" are currently implemented
H	self-similarity or Hurst parameter to indicate spectral scaling, white noise is 0.5
noise.dir	number of directions to estimate background noise standard deviation, the default is 3 which produces a unique estimate of the background noise for each spatial direction
rule	either a "hard" or "soft" thresholding rule may be used

Details

See Thresholding.

Value

Image of the same dimension as the original but with high-freqency fluctuations removed.

Author(s)

B. Whitcher

References

See Thresholding for references concerning de-noising in one dimension.

See Also

Thresholding

Examples

```
## Xbox image
data(xbox)
n <- NROW(xbox)
xbox.noise <- xbox + matrix(rnorm(n*n, sd=.15), n, n)
par(mfrow=c(2,2), cex=.8, pty="s")
image(xbox.noise, col=rainbow(128), main="Original Image")
image(denoise.dwt.2d(xbox.noise, wf="haar"), col=rainbow(128),
      zlim=range(xbox.noise), main="Denoised image")
image(xbox.noise - denoise.dwt.2d(xbox.noise, wf="haar"), col=rainbow(128),
      zlim=range(xbox.noise), main="Residual image")

## Daubechies image
data(dau)
n <- NROW(dau)
dau.noise <- dau + matrix(rnorm(n*n, sd=10), n, n)
```

```
par(mfrow=c(2,2), cex=.8, pty="s")
image(dau.noise, col=rainbow(128), main="Original Image")
dau.denoise <- denoise.modwt.2d(dau.noise, wf="d4", rule="soft")
image(dau.denoise, col=rainbow(128), zlim=range(dau.noise),
      main="Denoised image")
image(dau.noise - dau.denoise, col=rainbow(128), main="Residual image")
```

`doppler` *Sinusoid with Changing Amplitude and Frequency*

Description

$$doppler(x) = \sqrt{x(1-x)}\sin\left(\frac{2.1\pi}{x+0.05}\right)$$

Usage

```
data(doppler)
```

Format

A vector containing 512 observations.

Source

S+WAVELETS.

References

Bruce, A., and H.-Y. Gao (1996) *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

`dpss.taper` *Calculating Thomson's Spectral Multitapers by Inverse Iteration*

Description

The following function links the subroutines in "bell-p-w.o" to an R function in order to compute discrete prolate spheroidal sequences (dpss).

Usage

```
dpss.taper(n, k, nw = 4, nmax = 2^(ceiling(log(n, 2))))
```

Arguments

`n`	length of data taper(s)
`k`	number of data tapers; 1, 2, 3, ... (do not use 0!)
`nw`	product of length and half-bandwidth parameter (w)
`nmax`	maximum possible taper length, necessary for FORTRAN code

Details

Spectral estimation using a set of orthogonal tapers is becoming widely used and appreciated in scientific research. It produces direct spectral estimates with more than 2 df at each Fourier frequency, resulting in spectral estimators with reduced variance. Computation of the orthogonal tapers from the basic defining equation is difficult, however, due to the instability of the calculations – the eigenproblem is very poorly conditioned. In this article the severe numerical instability problems are illustrated and then a technique for stable calculation of the tapers – namely, inverse iteration – is described. Each iteration involves the solution of a matrix equation. Because the matrix has Toeplitz form, the Levinson recursions are used to rapidly solve the matrix equation. FORTRAN code for this method is available through the Statlib archive. An alternative stable method is also briefly reviewed.

Value

`v`	matrix of data tapers (cols = tapers)
`eigen`	eigenvalue associated with each data taper
`iter`	total number of iterations performed
`n`	same as input
`w`	half-bandwidth parameter
`ifault`	0 indicates success, see documentation for "bell-p-w" for information on non-zero values

Author(s)

B. Whitcher

References

B. Bell, D. B. Percival, and A. T. Walden (1993) Calculating Thomson's spectral multitapers by inverse iteration, *Journal of Computational and Graphical Statistics*, **2**, No. 1, 119-130.

Percival, D. B. and A. T. Walden (1993) *Spectral Estimation for Physical Applications: Multitaper and Conventional Univariate Techniques*, Cambridge University Press.

See Also

`sine.taper`.

Dual-tree Filter Banks *Filter Banks for Dual-Tree Wavelet Transforms*

Description

Analysis and synthesis filter banks used in dual-tree wavelet algorithms.

Usage

```
afb(x, af)
afb2D(x, af1, af2 = NULL)
afb2D.A(x, af, d)
sfb(lo, hi, sf)
sfb2D(lo, hi, sf1, sf2 = NULL)
sfb2D.A(lo, hi, sf, d)
```

Arguments

`x`	vector or matrix of observations
`af`	analysis filters. First element of the list is the low-pass filter, second element is the high-pass filter.
`af1,af2`	analysis filters for the first and second dimension of a 2D array.
`sf`	synthesis filters. First element of the list is the low-pass filter, second element is the high-pass filter.
`sf1,sf2`	synthesis filters for the first and second dimension of a 2D array.
`d`	dimension of filtering (d = 1 or 2)
`lo`	low-frequecy coefficients
`hi`	high-frequency coefficients

Details

The functions `afb2D.A` and `sfb2D.A` implement the convolutions, either for analysis or synthesis, in one dimension only. Thus, they are the workhorses of `afb2D` and `sfb2D`. The output for the analysis filter bank along one dimension (`afb2D.A`) is a list with two elements

lo low-pass subband

hi high-pass subband

where the dimension of analysis will be half its original length. The output for the synthesis filter bank along one dimension (`sfb2D.A`) will be the output array, where the dimension of synthesis will be twice its original length.

Value

In one dimension the output for the analysis filter bank (`afb`) is a list with two elements

`lo`	Low frequecy output
`hi`	High frequency output

and the output for the synthesis filter bank (`sfb`) is the output signal.

In two dimensions the output for the analysis filter bank (`afb2D`) is a list with four elements

`lo`	low-pass subband
`hi[[1]]`	'lohi' subband
`hi[[2]]`	'hilo' subband
`hi[[3]]`	'hihi' subband

and the output for the synthesis filter bank (`sfb2D`) is the output array.

Author(s)

Matlab: S. Cai, K. Li and I. Selesnick; R port: B. Whitcher

References

WAVELET SOFTWARE AT POLYTECHNIC UNIVERSITY, BROOKLYN, NY
http://taco.poly.edu/WaveletSoftware/

Examples

```
## EXAMPLE: afb, sfb
af = farras()$af
sf = farras()$sf
x = rnorm(64)
x.afb = afb(x, af)
lo = x.afb$lo
hi = x.afb$hi
y = sfb(lo, hi, sf)
err = x - y
max(abs(err))

## EXAMPLE: afb2D, sfb2D
x = matrix(rnorm(32*64), 32, 64)
af = farras()$af
sf = farras()$sf
x.afb2D = afb2D(x, af, af)
lo = x.afb2D$lo
hi = x.afb2D$hi
y = sfb2D(lo, hi, sf, sf)
err = x - y
max(abs(err))

## Example: afb2D.A, sfb2D.A
```

```
x = matrix(rnorm(32*64), 32, 64)
af = farras()$af
sf = farras()$sf
x.afb2D.A = afb2D.A(x, af, 1)
lo = x.afb2D.A$lo
hi = x.afb2D.A$hi
y = sfb2D.A(lo, hi, sf, 1)
err = x - y
max(abs(err))
```

`dualfilt1` *Kingsbury's Q-filters for the Dual-Tree Complex DWT*

Description

Kingsbury's Q-filters for the dual-tree complex DWT.

Usage

```
dualfilt1()
```

Arguments

None.

Details

These cofficients are rounded to 8 decimal places.

Value

`af`	List $(i = 1, 2)$ - analysis filters for tree i
`sf`	List $(i = 1, 2)$ - synthesis filters for tree i

Note: `af[[2]]` is the reverse of `af[[1]]`.

Author(s)

Matlab: S. Cai, K. Li and I. Selesnick; R port: B. Whitcher

References

Kingsbury, N.G. (2000). A dual-tree complex wavelet transform with improved orthogonality and symmetry properties, *Proceedings of the IEEE Int. Conf. on Image Proc.* (ICIP).

WAVELET SOFTWARE AT POLYTECHNIC UNIVERSITY, BROOKLYN, NY
http://taco.poly.edu/WaveletSoftware/

See Also

dualtree

Dualtree *Dual-tree Complex Discrete Wavelet Transform*

Description

One- and two-dimensional dual-tree complex discrete wavelet transforms developed by Kingsbury and Selesnick *et al.*

Usage

```
dualtree(x, J, Faf, af)
idualtree(w, J, Fsf, sf)
dualtree2D(x, J, Faf, af)
idualtree2D(w, J, Fsf, sf)
```

Arguments

x	N-point vector or $M \times N$ matrix.
w	DWT coefficients.
J	number of stages.
Faf	analysis filters for the first stage.
af	analysis filters for the remaining stages.
Fsf	synthesis filters for the last stage.
sf	synthesis filters for the preceeding stages.

Details

In one dimension N is divisible by 2^J and $N \geq 2^{J-1} \cdot \text{length}(\texttt{af})$.

In two dimensions, these two conditions must hold for both M and N.

Value

For the analysis of `x`, the output is

w	DWT coefficients. Each wavelet scale is a list containing the real and imaginary parts. The final scale $(J + 1)$ contains the low-pass filter coefficients.

For the synthesis of `w`, the output is

y	output signal

Author(s)

Matlab: S. Cai, K. Li and I. Selesnick; R port: B. Whitcher

References

WAVELET SOFTWARE AT POLYTECHNIC UNIVERSITY, BROOKLYN, NY
http://taco.poly.edu/WaveletSoftware/

See Also

FSfarras, farras, convolve, cshift, afb, sfb.

Examples

```
## EXAMPLE: dualtree
x = rnorm(512)
J = 4
Faf = FSfarras()$af
Fsf = FSfarras()$sf
af = dualfilt1()$af
sf = dualfilt1()$sf
w = dualtree(x, J, Faf, af)
y = idualtree(w, J, Fsf, sf)
err = x - y
max(abs(err))

## Example: dualtree2D
x = matrix(rnorm(64*64), 64, 64)
J = 3
Faf = FSfarras()$af
Fsf = FSfarras()$sf
af = dualfilt1()$af
sf = dualfilt1()$sf
w = dualtree2D(x, J, Faf, af)
y = idualtree2D(w, J, Fsf, sf)
err = x - y
max(abs(err))

## Display 2D wavelets of dualtree2D.m

J <- 4
L <- 3 * 2^(J+1)
N <- L / 2^J
Faf <- FSfarras()$af
Fsf <- FSfarras()$sf
af <- dualfilt1()$af
sf <- dualfilt1()$sf
x <- matrix(0, 2*L, 3*L)
w <- dualtree2D(x, J, Faf, af)
w[[J]][[1]][[1]][N/2, N/2+0*N] <- 1
w[[J]][[1]][[2]][N/2, N/2+1*N] <- 1
w[[J]][[1]][[3]][N/2, N/2+2*N] <- 1
w[[J]][[2]][[1]][N/2+N, N/2+0*N] <- 1
w[[J]][[2]][[2]][N/2+N, N/2+1*N] <- 1
w[[J]][[2]][[3]][N/2+N, N/2+2*N] <- 1
y <- idualtree2D(w, J, Fsf, sf)
```

```
image(t(y), col=grey(0:64/64), axes=FALSE)
```

`Dualtree Complex` *Dual-tree Complex 2D Discrete Wavelet Transform*

Description

Dual-tree complex 2D discrete wavelet transform (DWT).

Usage

```
cplxdual2D(x, J, Faf, af)
icplxdual2D(w, J, Fsf, sf)
```

Arguments

`x`	2D array.
`w`	wavelet coefficients.
`J`	number of stages.
`Faf`	first stage analysis filters for tree i.
`af`	analysis filters for the remaining stages on tree i.
`Fsf`	last stage synthesis filters for tree i.
`sf`	synthesis filters for the preceeding stages.

Value

For the analysis of `x`, the output is

`w`	wavelet coefficients indexed by `[[j]][[i]][[d1]][[d2]]`, where $j = 1, \ldots, J$ (scale), $i = 1$ (real part) or $i = 2$ (imag part), $d1 = 1, 2$ and $d2 = 1, 2, 3$ (orientations).

For the synthesis of `w`, the output is

`y`	output signal.

Author(s)

Matlab: S. Cai, K. Li and I. Selesnick; R port: B. Whitcher

References

WAVELET SOFTWARE AT POLYTECHNIC UNIVERSITY, BROOKLYN, NY
http://taco.poly.edu/WaveletSoftware/

See Also

FSfarras, farras, afb2D, sfb2D.

Examples

```
## Not run:
## EXAMPLE: cplxdual2D
x = matrix(rnorm(32*32), 32, 32)
J = 5
Faf = FSfarras()$af
Fsf = FSfarras()$sf
af = dualfilt1()$af
sf = dualfilt1()$sf
w = cplxdual2D(x, J, Faf, af)
y = icplxdual2D(w, J, Fsf, sf)
err = x - y
max(abs(err))

## End(Not run)
```

---

dwpt *(Inverse) Discrete Wavelet Packet Transforms*

Description

All possible filtering combinations (low- and high-pass) are performed to decompose a vector or time series. The resulting coefficients are associated with a binary tree structure corresponding to a partitioning of the frequency axis.

Usage

```
dwpt(x, wf="la8", n.levels=4, boundary="periodic")
idwpt(y, y.basis)
modwpt(x, wf = "la8", n.levels = 4, boundary = "periodic")
```

Arguments

x	a vector or time series containing the data be to decomposed. This must be a dyadic length vector (power of 2).
wf	Name of the wavelet filter to use in the decomposition. By default this is set to "la8", the Daubechies orthonormal compactly supported wavelet of length $L = 8$ (Daubechies, 1992), least asymmetric family.
n.levels	Specifies the depth of the decomposition. This must be a number less than or equal to $\log(\text{length}(x), 2)$.
boundary	Character string specifying the boundary condition. If boundary=="periodic" the default, then the vector you decompose is assumed to be periodic on its defined interval, if boundary=="reflection", the vector beyond its boundaries is assumed to be a symmetric reflection of itself.
y	Object of S3 class dwpt.
y.basis	Vector of character strings that describe leaves on the DWPT basis tree.

Details

The code implements the one-dimensional DWPT using the pyramid algorithm (Mallat, 1989).

Value

Basically, a list with the following components

`w?.?`	Wavelet coefficient vectors. The first index is associated with the scale of the decomposition while the second is associated with the frequency partition within that level.
`wavelet`	Name of the wavelet filter used.
`boundary`	How the boundaries were handled.

Author(s)

B. Whitcher

References

Mallat, S. G. (1989) A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, No. 7, 674-693.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

Wickerhauser, M. V. (1994) *Adapted Wavelet Analysis from Theory to Software*, A K Peters.

See Also

`dwt`, `modwpt`, `wave.filter`.

Examples

```
data(mexm)
J <- 4
mexm.mra <- mra(log(mexm), "mb8", J, "modwt", "reflection")
mexm.nomean <- ts(
  apply(matrix(unlist(mexm.mra), ncol=J+1, byrow=FALSE)[,-(J+1)], 1, sum),
  start=1957, freq=12)
mexm.dwpt <- dwpt(mexm.nomean[-c(1:4)], "mb8", 7, "reflection")
```

dwpt.2d *(Inverse) Discrete Wavelet Packet Transforms in Two Dimensions*

Description

All possible filtering combinations (low- and high-pass) are performed to decompose a matrix or image. The resulting coefficients are associated with a quad-tree structure corresponding to a partitioning of the two-dimensional frequency plane.

Usage

```
dwpt.2d(x, wf="la8", J=4, boundary="periodic")
idwpt.2d(y, y.basis)
```

Arguments

`x`	a matrix or image containing the data be to decomposed. This ojbect must be dyadic (power of 2) in length in each dimension.
`wf`	Name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ (Daubechies, 1992), least asymmetric family.
`J`	Specifies the depth of the decomposition. This must be a number less than or equal to $\log(\text{length}(x), 2)$.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the default, then the vector you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the vector beyond its boundaries is assumed to be a symmetric reflection of itself.
`y`	`dwpt.2d` object (list-based structure of matrices)
`y.basis`	Boolean vector, the same length as y, where `TRUE` means the basis tensor should be used in the reconstruction.

Details

The code implements the two-dimensional DWPT using the pyramid algorithm of Mallat (1989).

Value

Basically, a list with the following components

`w?.?-w?.?`	Wavelet coefficient matrices (images). The first index is associated with the scale of the decomposition while the second is associated with the frequency partition within that level. The left and right strings, separated by the dash '-', correspond to the first (x) and second (y) dimensions.
`wavelet`	Name of the wavelet filter used.
`boundary`	How the boundaries were handled.

Author(s)

B. Whitcher

References

Mallat, S. G. (1989) A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, No. 7, 674-693.

Wickerhauser, M. V. (1994) *Adapted Wavelet Analysis from Theory to Software*, A K Peters.

See Also

`dwt.2d`, `modwt.2d`, `wave.filter`.

---

dwpt.boot — *Bootstrap Time Series Using the DWPT*

Description

An adaptive orthonormal basis is selected in order to perform the naive bootstrap within nodes of the wavelet packet tree. A bootstrap realization of the time series is produce by applying the inverse DWPT.

Usage

```
dwpt.boot(y, wf, J=log(length(y),2)-1, p=1e-04, frac=1)
```

Arguments

`y`	Not necessarily dyadic length time series.
`wf`	Name of the wavelet filter to use in the decomposition. See `wave.filter` for those wavelet filters available.
`J`	Depth of the discrete wavelet packet transform.
`p`	Level of significance for the white noise testing procedure.
`frac`	Fraction of the time series that should be used in constructing the likelihood function.

Details

A subroutines is used to select an adaptive orthonormal basis for the piecewise-constant approximation to the underlying spectral density function (SDF). Once selected, sampling with replacement is performed within each wavelet packet coefficient vector and the new collection of wavelet packet coefficients are reconstructed into a bootstrap realization of the original time series.

Value

Time series of length \$N\$, where \$N\$ is the length of `y`.

Author(s)

B. Whitcher

References

Percival, D.B., S. Sardy, A. Davision (2000) Wavestrapping Time Series: Adaptive Wavelet-Based Bootstrapping, in B.J. Fitzgerald, R.L. Smith, A.T. Walden, P.C. Young (Eds.) *Nonlinear and Nonstationary Signal Processing*, pp. 442-471.

Whitcher, B. (2001) Simulating Gaussian Stationary Time Series with Unbounded Spectra, *Journal of Computational and Graphical Statistics*, **10**, No. 1, 112-134.

Whitcher, B. (2004) Wavelet-Based Estimation for Seasonal Long-Memory Processes, *Technometrics*, **46**, No. 2, 225-238.

See Also

`dwpt.sim`, `spp.mle`

`dwpt.sim` *Simulate Seasonal Persistent Processes Using the DWPT*

Description

A seasonal persistent process may be characterized by a spectral density function with an asymptote occuring at a particular frequency in $[0, \frac{1}{2})$. It's time domain representation was first noted in passing by Hosking (1981). Although an exact time-domain approach to simulation is possible, this function utilizes the discrete wavelet packet transform (DWPT).

Usage

```
dwpt.sim(N, wf, delta, fG, M=2, adaptive=TRUE, epsilon=0.05)
```

Arguments

`N`	Length of time series to be generated.
`wf`	Character string for the wavelet filter.
`delta`	Long-memory parameter for the seasonal persistent process.
`fG`	Gegenbauer frequency.
`M`	Actual length of simulated time series.
`adaptive`	Logical; if `TRUE` the orthonormal basis used in the DWPT is adapted to the ideal spectrum, otherwise the orthonormal basis is performed to a maximum depth.
`epsilon`	Threshold for adaptive basis selection.

Details

Two subroutines are used, the first selects an adaptive orthonormal basis for the true spectral density function (SDF) while the second computes the bandpass variances associated with the chosen orthonormal basis and SDF. Finally, when

$$M > N$$

a uniform random variable is generated in order to select a random piece of the simulated time series. For more details see Whitcher (2001).

Value

Time series of length `N`.

Author(s)

B. Whitcher

References

Hosking, J. R. M. (1981) Fractional Differencing, *Biometrika*, **68**, No. 1, 165-176.

Whitcher, B. (2001) Simulating Gaussian Stationary Time Series with Unbounded Spectra, *Journal of Computational and Graphical Statistics*, **10**, No. 1, 112-134.

See Also

`hosking.sim` for an exact time-domain method and `wave.filter` for a list of available wavelet filters.

Examples

```
## Generate monthly time series with annual oscillation
## library(ts) is required in order to access acf()
x <- dwpt.sim(256, "mb16", .4, 1/12, M=4, epsilon=.001)
par(mfrow=c(2,1))
plot(x, type="l", xlab="Time")
acf(x, lag.max=128, ylim=c(-.6,1))
data(acvs.andel8)
lines(acvs.andel8$lag[1:128], acvs.andel8$acf[1:128], col=2)
```

---

dwt *Discrete Wavelet Transform (DWT)*

Description

This function performs a level J decomposition of the input vector or time series using the pyramid algorithm (Mallat 1989).

Usage

```
dwt(x, wf="la8", n.levels=4, boundary="periodic")
dwt.nondyadic(x)
```

Arguments

`x`	a vector or time series containing the data be to decomposed. This must be a dyadic length vector (power of 2).
`wf`	Name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ (Daubechies, 1992), least asymmetric family.
`n.levels`	Specifies the depth of the decomposition. This must be a number less than or equal to $\log(\text{length}(x), 2)$.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the default, then the vector you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the vector beyond its boundaries is assumed to be a symmetric reflection of itself.

Details

The code implements the one-dimensional DWT using the pyramid algorithm (Mallat, 1989). The actual transform is performed in C using pseudocode from Percival and Walden (2001). That means convolutions, not inner products, are used to apply the wavelet filters.

For a non-dyadic length vector or time series, `dwt.nondyadic` pads with zeros, performs the orthonormal DWT on this dyadic length series and then truncates the wavelet coefficient vectors appropriately.

Value

Basically, a list with the following components

`d?`	Wavelet coefficient vectors.
`s?`	Scaling coefficient vector.
`wavelet`	Name of the wavelet filter used.
`boundary`	How the boundaries were handled.

Author(s)

B. Whitcher

References

Daubechies, I. (1992) *Ten Lectures on Wavelets*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM: Philadelphia.

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Mallat, S. G. (1989) A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, No. 7, 674-693.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

See Also

modwt, mra.

Examples

```
## Figures 4.17 and 4.18 in Gencay, Selcuk and Whitcher (2001).
data(ibm)
ibm.returns <- diff(log(ibm))
## Haar
ibmr.haar <- dwt(ibm.returns, "haar")
names(ibmr.haar) <- c("w1", "w2", "w3", "w4", "v4")
## plot partial Haar DWT for IBM data
par(mfcol=c(6,1), pty="m", mar=c(5-2,4,4-2,2))
plot.ts(ibm.returns, axes=FALSE, ylab="", main="(a)")
for(i in 1:4)
  plot.ts(up.sample(ibmr.haar[[i]], 2^i), type="h", axes=FALSE,
          ylab=names(ibmr.haar)[i])
plot.ts(up.sample(ibmr.haar$v4, 2^4), type="h", axes=FALSE,
        ylab=names(ibmr.haar)[5])
axis(side=1, at=seq(0,368,by=23),
     labels=c(0,"",46,"",92,"",138,"",184,"",230,"",276,"",322,"",368))
## LA(8)
ibmr.la8 <- dwt(ibm.returns, "la8")
names(ibmr.la8) <- c("w1", "w2", "w3", "w4", "v4")
## must shift LA(8) coefficients
ibmr.la8$w1 <- c(ibmr.la8$w1[-c(1:2)], ibmr.la8$w1[1:2])
ibmr.la8$w2 <- c(ibmr.la8$w2[-c(1:2)], ibmr.la8$w2[1:2])
for(i in names(ibmr.la8)[3:4])
  ibmr.la8[[i]] <- c(ibmr.la8[[i]][-c(1:3)], ibmr.la8[[i]][1:3])
ibmr.la8$v4 <- c(ibmr.la8$v4[-c(1:2)], ibmr.la8$v4[1:2])
## plot partial LA(8) DWT for IBM data
par(mfcol=c(6,1), pty="m", mar=c(5-2,4,4-2,2))
plot.ts(ibm.returns, axes=FALSE, ylab="", main="(b)")
for(i in 1:4)
  plot.ts(up.sample(ibmr.la8[[i]], 2^i), type="h", axes=FALSE,
          ylab=names(ibmr.la8)[i])
plot.ts(up.sample(ibmr.la8$v4, 2^4), type="h", axes=FALSE,
        ylab=names(ibmr.la8)[5])
axis(side=1, at=seq(0,368,by=23),
  labels=c(0,"",46,"",92,"",138,"",184,"",230,"",276,"",322,"",368))
```

`dwt.2d` *Two-Dimensional Discrete Wavelet Transform*

Description

Performs a separable two-dimensional discrete wavelet transform (DWT) on a matrix of dyadic dimensions.

Usage

```
dwt.2d(x, wf, J = 4, boundary = "periodic")
idwt.2d(y)
```

Arguments

`x`	input matrix (image)
`wf`	name of the wavelet filter to use in the decomposition
`J`	depth of the decomposition, must be a number less than or equal to $\log_2(\min\{M, N\})$
`boundary`	only `"periodic"` is currently implemented
`y`	an object of class `dwt.2d`

Details

See references.

Value

List structure containing the $3J + 1$ sub-matrices from the decomposition.

Author(s)

B. Whitcher

References

Mallat, S. (1998) *A Wavelet Tour of Signal Processing*, Academic Press.

Vetterli, M. and J. Kovacevic (1995) *Wavelets and Subband Coding*, Prentice Hall.

See Also

`modwt.2d`.

Examples

```
## Xbox image
data(xbox)
xbox.dwt <- dwt.2d(xbox, "haar", 3)
par(mfrow=c(1,1), pty="s")
plot.dwt.2d(xbox.dwt)
par(mfrow=c(2,2), pty="s")
image(1:dim(xbox)[1], 1:dim(xbox)[2], xbox, xlab="", ylab="",
      main="Original Image")
image(1:dim(xbox)[1], 1:dim(xbox)[2], idwt.2d(xbox.dwt), xlab="", ylab="",
      main="Wavelet Reconstruction")
image(1:dim(xbox)[1], 1:dim(xbox)[2], xbox - idwt.2d(xbox.dwt),
      xlab="", ylab="", main="Difference")

## Daubechies image
data(dau)
par(mfrow=c(1,1), pty="s")
image(dau, col=rainbow(128))
sum(dau^2)
dau.dwt <- dwt.2d(dau, "d4", 3)
plot.dwt.2d(dau.dwt)
sum(plot.dwt.2d(dau.dwt, plot=FALSE)^2)
```

`dwt.3d` *Three Dimensional Separable Discrete Wavelet Transform*

Description

Three-dimensional separable discrete wavelet transform (DWT).

Usage

```
dwt.3d(x, wf, J=4, boundary="periodic")
idwt.3d(y)
```

Arguments

`x`	input array
`wf`	name of the wavelet filter to use in the decomposition
`J`	depth of the decomposition, must be a number less than or equal to $\log_2(\min\{X,Y,Z\})$
`boundary`	only `"periodic"` is currently implemented
`y`	an object of class `dwt.3d`

Author(s)

B. Whitcher

`exchange`	*Exchange Rates Between the Deutsche Mark, Japanese Yen and U.S. Dollar*

Description

Monthly foreign exchange rates for the Deutsche Mark - U.S. Dollar (DEM-USD) and Japanese Yen - U.S. Dollar (JPY-USD) starting in 1970.

Usage

```
data(exchange)
```

Format

A bivariate time series containing 348 observations.

Source

Unknown.

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

`Farras`	*Farras nearly symmetric filters*

Description

Farras nearly symmetric filters for orthogonal 2-channel perfect reconstruction filter bank and Farras filters organized for the dual-tree complex DWT.

Usage

```
farras()
FSfarras()
```

Arguments

None.

Value

`af`	List ($i = 1, 2$) - analysis filters for tree i
`sf`	List ($i = 1, 2$) - synthesis filters for tree i

Author(s)

Matlab: S. Cai, K. Li and I. Selesnick; R port: Brandon Whitcher

References

A. F. Abdelnour and I. W. Selesnick. "Nearly symmetric orthogonal wavelet bases", Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP), May 2001.

WAVELET SOFTWARE AT POLYTECHNIC UNIVERSITY, BROOKLYN, NY
`http://taco.poly.edu/WaveletSoftware/`

See Also

`afb`, `dualtree`, `dualfilt1`.

`fdp.mle` *Wavelet-based Maximum Likelihood Estimation for a Fractional Difference Process*

Description

Parameter estimation for a fractional difference (long-memory, self-similar) process is performed via maximum likelihood on the wavelet coefficients.

Usage

```
fdp.mle(y, wf, J=log(length(y),2))
```

Arguments

`y`	Dyadic length time series.
`wf`	Name of the wavelet filter to use in the decomposition. See `wave.filter` for those wavelet filters available.
`J`	Depth of the discrete wavelet transform.

Details

The variance-covariance matrix of the original time series is approximated by its wavelet-based equivalent. A Whittle-type likelihood is then constructed where the sums of squared wavelet coefficients are compared to bandpass filtered version of the true spectrum. Minimization occurs only for the fractional difference parameter d, while variance is estimated afterwards.

Value

List containing the maximum likelihood estimates (MLEs) of d and σ^2, along with the value of the likelihood for those estimates.

Author(s)

B. Whitcher

References

M. J. Jensen (2000) An alternative maximum likelihood estimator of long-memory processes using compactly supported wavelets, *Journal of Economic Dynamics and Control*, **24**, No. 3, 361-387.

McCoy, E. J., and A. T. Walden (1996) Wavelet analysis and synthesis of stationary long-memory processes, *Journal for Computational and Graphical Statistics*, **5**, No. 1, 26-56.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

Examples

```
## Figure 5.5 in Gencay, Selcuk and Whitcher (2001)
fdp.sdf <- function(freq, d, sigma2=1)
  sigma2 / ((2*sin(pi * freq))^2)^d
dB <- function(x) 10 * log10(x)
per <- function(z) {
  n <- length(z)
  (Mod(fft(z))**2/(2*pi*n))[1:(n %/% 2 + 1)]
}
data(ibm)
ibm.returns <- diff(log(ibm))
ibm.volatility <- abs(ibm.returns)
ibm.vol.mle <- fdp.mle(ibm.volatility, "d4", 4)
freq <- 0:184/368
ibm.vol.per <- 2 * pi * per(ibm.volatility)
ibm.vol.resid <- ibm.vol.per/ fdp.sdf(freq, ibm.vol.mle$parameters[1])
par(mfrow=c(1,1), las=0, pty="m")
plot(freq, dB(ibm.vol.per), type="l", xlab="Frequency", ylab="Spectrum")
lines(freq, dB(fdp.sdf(freq, ibm.vol.mle$parameters[1],
                       ibm.vol.mle$parameters[2]/2)), col=2)
```

---

find.adaptive.basis *Determine an Orthonormal Basis for the Discrete Wavelet Packet Transform*

Description

Subroutine for use in simulating seasonal persistent processes using the discrete wavelet packet transform.

Usage

```
find.adaptive.basis(wf, J, fG, eps)
```

Arguments

- `wf` Character string; name of the wavelet filter.
- `J` Depth of the discrete wavelet packet transform.
- `fG` Gegenbauer frequency.
- `eps` Threshold for the squared gain function.

Details

The squared gain functions for a Daubechies (extremal phase or least asymmetric) wavelet family are used in a filter cascade to compute the value of the squared gain function for the wavelet packet filter at the Gengenbauer frequency. This is done for all nodes of the wavelet packet table.

The idea behind this subroutine is to approximate the relationship between the discrete wavelet transform and long-memory processes, where the squared gain function is zero at frequency zero for all levels of the DWT.

Value

Boolean vector describing the orthonormal basis for the DWPT.

Author(s)

B. Whitcher

See Also

Used in `dwpt.sim`.

`heavisine` *Sine with Jumps at 0.3 and 0.72*

Description

$$heavisine(x) = 4\sin(4\pi x) - \mathrm{sign}(x - 0.3) - \mathrm{sign}(0.72 - x)$$

Usage

```
data(heavisine)
```

Format

A vector containing 512 observations.

Source

S+WAVELETS.

References

Bruce, A., and H.-Y. Gao (1996) *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

`Hilbert` *Discrete Hilbert Wavelet Transforms*

Description

The discrete Hilbert wavelet transforms (DHWTs) for seasonal and time-varying time series analysis. Transforms include the usual orthogonal (decimated), maximal-overlap (non-decimated) and maximal-overlap packet transforms.

Usage

```
dwt.hilbert(x, wf, n.levels=4, boundary="periodic", ...)
dwt.hilbert.nondyadic(x, ...)
idwt.hilbert(y)
modwt.hilbert(x, wf, n.levels=4, boundary="periodic", ...)
imodwt.hilbert(y)
modwpt.hilbert(x, wf, n.levels=4, boundary="periodic")
```

Arguments

`x`	Real-valued time series or vector of observations.
`wf`	Hilbert wavelet pair
`n.levels`	Number of levels (depth) of the wavelet transform.
`boundary`	Boundary treatment, currently only `periodic` and `reflection`.
`y`	Hilbert wavelet transform object (list).
`...`	Additional parametes to be passed on.

Author(s)

B. Whitcher

References

Selesnick, I. (200X). *IEEE Signal Processing Magazine*

Selesnick, I. (200X). *IEEE Transactions in Signal Processing*

Whither, B. and P.F. Craigmile (2004). Multivariate Spectral Analysis Using Hilbert Wavelet Pairs, *International Journal of Wavelets, Multiresolution and Information Processing*, to appear.

See Also

`hilbert.filter`

`hilbert.filter` *Select a Hilbert Wavelet Pair*

Description

Converts name of Hilbert wavelet pair to filter coefficients.

Usage

```
hilbert.filter(name)
```

Arguments

`name`	Character string of Hilbert wavelet pair, see acceptable names below (e.g., `"k3l3"`).

Details

Simple `switch` statement selects the appropriate HWP. There are two parameters that define a Hilbert wavelet pair using the notation of Selesnick (2001,2002), K and L. Currently, the only implemented combinations (K, L) are (3,3), (3,5), (4,2) and (4,4).

Value

List containing the following items:

`L`	length of the wavelet filter
`h0,g0`	low-pass filter coefficients
`h1,g1`	high-pass filter coefficients

Author(s)

B. Whitcher

References

Selesnick, I.W. (2001). Hilbert transform pairs of wavelet bases. *IEEE Signal Processing Letters*\/~**8**(6), 170–173.

Selesnick, I.W. (2002). The design of approximate Hilbert transform pairs of wavelet bases. *IEEE Transactions on Signal Processing*\/~**50**(5), 1144–1152.

See Also

`wave.filter`

Examples

```
hilbert.filter("k3l3")
hilbert.filter("k3l5")
hilbert.filter("k4l2")
hilbert.filter("k4l4")
```

---

hosking.sim *Generate Stationary Gaussian Process Using Hosking's Method*

Description

Uses exact time-domain method from Hosking (1984) to generate a simulated time series from a specified autocovariance sequence.

Usage

```
hosking.sim(n, acvs)
```

Arguments

n	Length of series.
acvs	Autocovariance sequence of series with which to generate, must be of length at least n.

Value

Length n time series from true autocovariance sequence acvs.

Author(s)

Brandon Whitcher

References

Hosking, J. R. M. (1984) Modeling persistence in hydrological time series using fractional differencing, *Water Resources Research*, **20**, No. 12, 1898-1908.

Percival, D. B. (1992) Simulating Gaussian random processes with specified spectra, *Computing Science and Statistics*, **22**, 534-538.

Examples

```
dB <- function(x) 10 * log10(x)
per <- function (z) {
  n <- length(z)
  (Mod(fft(z))^2/(2 * pi * n))[1:(n%/%2 + 1)]
}
spp.sdf <- function(freq, delta, omega)
  abs(2 * (cos(2*pi*freq) - cos(2*pi*omega)))^(-2*delta)
```

```
data(acvs.andel8)
n <- 1024
## Not run:
z <- hosking.sim(n, acvs.andel8[,2])
per.z <- 2 * pi * per(z)
par(mfrow=c(2,1), las=1)
plot.ts(z, ylab="", main="Realization of a Seasonal Long-Memory Process")
plot(0:(n/2)/n, dB(per.z), type="l", xlab="Frequency", ylab="dB",
     main="Periodogram")
lines(0:(n/2)/n, dB(spp.sdf(0:(n/2)/n, .4, 1/12)), col=2)

## End(Not run)
```

HWP Analysis *Time-varying and Seasonal Analysis Using Hilbert Wavelet Pairs*

Description

Performs time-varying or seasonal coherence and phase anlaysis between two time seris using the maximal-overlap discrete Hilbert wavelet transform (MODHWT).

Usage

```
modhwt.coh(x, y, f.length = 0)
modhwt.phase(x, y, f.length = 0)
modhwt.coh.seasonal(x, y, S = 10, season = 365)
modhwt.phase.seasonal(x, y, season = 365)
```

Arguments

x	MODHWT object.
y	MODHWT object.
f.length	Length of the rectangular filter.
S	Number of "seasons".
season	Length of the "season".

Details

The idea of seasonally-varying spectral analysis (SVSA, Madden 1986) is generalized using the MODWT and Hilbert wavelet pairs. For the seasonal case, S seasons are used to produce a consistent estimate of the coherence and phase. For the non-seasonal case, a simple rectangular (moving-average) filter is applied to the MODHWT coefficients in order to produce consistent estimates.

Value

Time-varying or seasonal coherence and phase between two time series. The coherence estimates are between zero and one, while the phase estimates are between $-\pi$ and π.

Author(s)

B. Whitcher

References

Madden, R.A. (1986). Seasonal variation of the 40–50 day oscillation in the tropics. *Journal of the Atmospheric Sciences*\/ **43**\/(24), 3138–3158.

Whither, B. and P.F. Craigmile (2004). Multivariate Spectral Analysis Using Hilbert Wavelet Pairs, *International Journal of Wavelets, Multiresolution and Information Processing*, to appear.

See Also

`hilbert.filter`

`ibm` *Daily IBM Stock Prices*

Description

Daily IBM stock prices spanning May~17, 1961 to November~2, 1962.

Usage

```
data(ibm)
```

Format

A vector containing 369 observations.

Source

Box, G. E.~P. and Jenkins, G.~M. (1976) *Time Series Analysis: Forecasting and Control*, Holden Day, San Francisco, 2 edition.

References

http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/

`japan` *Japanese Gross National Product*

Description

Quarterly Japanese gross national product from 1955:1 to 1996:4.

Usage

```
data(japan)
```

Format

A vector containing 169 observations.

Source

Unknown.

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Hecq, A. (1998) Does seasonal adjustment induce common cycles?, *Empirical Economics*, **59**, 289-297.

`jumpsine` *Sine with Jumps at 0.625 and 0.875*

Description

$$jumpsine(x) = 10\left(\sin(4\pi x) + I_{[0.625 < x \leq 0.875]}\right)$$

Usage

```
data(jumpsine)
```

Format

A vector containing 512 observations.

Source

S+WAVELETS.

References

Bruce, A., and H.-Y. Gao (1996) *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

`kobe` *1995 Kobe Earthquake Data*

Description

Seismograph (vertical acceleration, nm/sq.sec) of the Kobe earthquake, recorded at Tasmania University, HobarTRUE, Australia on 16 January 1995 beginning at 20:56:51 (GMTRUE) and continuing for 51 minutes at 1 second intervals.

Usage

```
data(kobe)
```

Format

A vector containing 3048 observations.

Source

Data management centre, Washington University.

References

http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/

`linchirp` *Linear Chirp*

Description

$$linchirp(x) = \sin(0.125\pi n x^2)$$

Usage

```
data(linchirp)
```

Format

A vector containing 512 observations.

Source

S+WAVELETS.

References

Bruce, A., and H.-Y. Gao (1996) *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

---

mexm — *Mexican Money Supply*

Description

Percentage changes in monthly Mexican money supply.

Usage

```
data(mexm)
```

Format

A vector containing 516 observations.

Source

Unknown.

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

---

modwt — *(Inverse) Maximal Overlap Discrete Wavelet Transform*

Description

This function performs a level J decomposition of the input vector using the non-decimated discrete wavelet transform. The inverse transform performs the reconstruction of a vector or time series from its maximal overlap discrete wavelet transform.

Usage

```
modwt(x, wf = "la8", n.levels = 4, boundary = "periodic")
imodwt(y)
```

Arguments

`x`	a vector or time series containing the data be to decomposed. There is **no** restriction on its length.
`y`	Object of class `"modwt"`.
`wf`	Name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ (Daubechies, 1992), least asymmetric family.
`n.levels`	Specifies the depth of the decomposition. This must be a number less than or equal to $\log_2(\text{length}(x))$.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the defaulTRUE, then the vector you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the vector beyond its boundaries is assumed to be a symmetric reflection of itself.

Details

The code implements the one-dimensional non-decimated DWT using the pyramid algorithm. The actual transform is performed in C using pseudocode from Percival and Walden (2001). That means convolutions, not inner products, are used to apply the wavelet filters.

The MODWT goes by several names in the statistical and engineering literature, such as, the “stationary DWT”, “translation-invariant DWT”, and “time-invariant DWT”.

The inverse MODWT implements the one-dimensional inverse transform using the pyramid algorithm (Mallat, 1989).

Value

Object of class `"modwt"`, basically, a list with the following components

`d?`	Wavelet coefficient vectors.
`s?`	Scaling coefficient vector.
`wavelet`	Name of the wavelet filter used.
`boundary`	How the boundaries were handled.

Author(s)

B. Whitcher

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Percival, D. B. and P. Guttorp (1994) Long-memory processes, the Allan variance and wavelets, In *Wavelets and Geophysics*, pages 325-344, Academic Press.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

See Also

dwt, idwt, mra.

Examples

```
## Figure 4.23 in Gencay, Selcuk and Whitcher (2001)
data(ibm)
ibm.returns <- diff(log(ibm))
# Haar
ibmr.haar <- modwt(ibm.returns, "haar")
names(ibmr.haar) <- c("w1", "w2", "w3", "w4", "v4")
# LA(8)
ibmr.la8 <- modwt(ibm.returns, "la8")
names(ibmr.la8) <- c("w1", "w2", "w3", "w4", "v4")
# shift the MODWT vectors
ibmr.la8 <- phase.shift(ibmr.la8, "la8")
## plot partial MODWT for IBM data
par(mfcol=c(6,1), pty="m", mar=c(5-2,4,4-2,2))
plot.ts(ibm.returns, axes=FALSE, ylab="", main="(a)")
for(i in 1:5)
  plot.ts(ibmr.haar[[i]], axes=FALSE, ylab=names(ibmr.haar)[i])
axis(side=1, at=seq(0,368,by=23),
  labels=c(0,"",46,"",92,"",138,"",184,"",230,"",276,"",322,"",368))
par(mfcol=c(6,1), pty="m", mar=c(5-2,4,4-2,2))
plot.ts(ibm.returns, axes=FALSE, ylab="", main="(b)")
for(i in 1:5)
  plot.ts(ibmr.la8[[i]], axes=FALSE, ylab=names(ibmr.la8)[i])
axis(side=1, at=seq(0,368,by=23),
  labels=c(0,"",46,"",92,"",138,"",184,"",230,"",276,"",322,"",368))
```

---

modwt.2d *Two-Dimensional Maximal Overlap Discrete Wavelet Transform*

Description

Performs a separable two-dimensional maximal overlap discrete wavelet transform (MODWT) on a matrix of arbitrary dimensions.

Usage

```
modwt.2d(x, wf, J = 4, boundary = "periodic")
imodwt.2d(y)
```

Arguments

x	input matrix
wf	name of the wavelet filter to use in the decomposition
J	depth of the decomposition

boundary	only "periodic" is currently implemented
y	an object of class dwt.2d

Details

See references.

Value

List structure containing the $3J + 1$ sub-matrices from the decomposition.

Author(s)

B. Whitcher

References

Liang, J. and T. W. Parks (1994) A two-dimensional translation invariant wavelet representation and its applications, *Proceedings ICIP-94*, Vol. 1, 66-70.

Liang, J. and T. W. Parks (1994) Image coding using translation invariant wavelet transforms with symmetric extensions, *IEEE Transactions on Image Processing*, **7**, No. 5, 762-769.

See Also

dwt.2d, shift.2d.

Examples

```
## Xbox image
data(xbox)
xbox.modwt <- modwt.2d(xbox, "haar", 2)
## Level 1 decomposition
par(mfrow=c(2,2), pty="s")
image(xbox.modwt$LH1, col=rainbow(128), axes=FALSE, main="LH1")
image(xbox.modwt$HH1, col=rainbow(128), axes=FALSE, main="HH1")
frame()
image(xbox.modwt$HL1, col=rainbow(128), axes=FALSE, main="HL1")
## Level 2 decomposition
par(mfrow=c(2,2), pty="s")
image(xbox.modwt$LH2, col=rainbow(128), axes=FALSE, main="LH2")
image(xbox.modwt$HH2, col=rainbow(128), axes=FALSE, main="HH2")
image(xbox.modwt$LL2, col=rainbow(128), axes=FALSE, main="LL2")
image(xbox.modwt$HL2, col=rainbow(128), axes=FALSE, main="HL2")
sum((xbox - imodwt.2d(xbox.modwt))^2)

data(dau)
par(mfrow=c(1,1), pty="s")
image(dau, col=rainbow(128), axes=FALSE, main="Ingrid Daubechies")
sum(dau^2)
dau.modwt <- modwt.2d(dau, "d4", 2)
## Level 1 decomposition
```

```
par(mfrow=c(2,2), pty="s")
image(dau.modwt$LH1, col=rainbow(128), axes=FALSE, main="LH1")
image(dau.modwt$HH1, col=rainbow(128), axes=FALSE, main="HH1")
frame()
image(dau.modwt$HL1, col=rainbow(128), axes=FALSE, main="HL1")
## Level 2 decomposition
par(mfrow=c(2,2), pty="s")
image(dau.modwt$LH2, col=rainbow(128), axes=FALSE, main="LH2")
image(dau.modwt$HH2, col=rainbow(128), axes=FALSE, main="HH2")
image(dau.modwt$LL2, col=rainbow(128), axes=FALSE, main="LL2")
image(dau.modwt$HL2, col=rainbow(128), axes=FALSE, main="HL2")
sum((dau - imodwt.2d(dau.modwt))^2)
```

`modwt.3d` *Three Dimensional Separable Maximal Ovelrap Discrete Wavelet Transform*

Description

Three-dimensional separable maximal overlap discrete wavelet transform (MODWT).

Usage

```
modwt.3d(x, wf, J = 4, boundary = "periodic")
imodwt.3d(y)
```

Arguments

`x`	input array
`wf`	name of the wavelet filter to use in the decomposition
`J`	depth of the decomposition
`boundary`	only `"periodic"` is currently implemented
`y`	an object of class `modwt.3d`

Author(s)

B. Whitcher

`mra` *Multiresolution Analysis of Time Series*

Description

This function performs a level J additive decomposition of the input vector or time series using the pyramid algorithm (Mallat 1989).

Usage

```
mra(x, wf = "la8", J = 4, method = "modwt", boundary = "periodic")
```

Arguments

`x`	A vector or time series containing the data be to decomposed. This must be a dyadic length vector (power of 2) for `method="dwt"`.
`wf`	Name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ least asymmetric family.
`J`	Specifies the depth of the decomposition. This must be a number less than or equal to $\log(\mathrm{length}(x), 2)$.
`method`	Either `"dwt"` or `"modwt"`.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the default, then the vector you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the vector beyond its boundaries is assumed to be a symmetric reflection of itself.

Details

This code implements a one-dimensional multiresolution analysis introduced by Mallat (1989). Either the DWT or MODWT may be used to compute the multiresolution analysis, which is an additive decomposition of the original time series.

Value

Basically, a list with the following components

`D?`	Wavelet detail vectors.
`S?`	Wavelet smooth vector.
`wavelet`	Name of the wavelet filter used.
`boundary`	How the boundaries were handled.

Author(s)

B. Whitcher

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Mallat, S. G. (1989) A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, No. 7, 674-693.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

See Also

dwt, modwt.

Examples

```
## Easy check to see if it works...
x <- rnorm(32)
x.mra <- mra(x)
sum(x - apply(matrix(unlist(x.mra), nrow=32), 1, sum))^2

## Figure 4.19 in Gencay, Selcuk and Whitcher (2001)
data(ibm)
ibm.returns <- diff(log(ibm))
ibm.volatility <- abs(ibm.returns)
## Haar
ibmv.haar <- mra(ibm.volatility, "haar", 4, "dwt")
names(ibmv.haar) <- c("d1", "d2", "d3", "d4", "s4")
## LA(8)
ibmv.la8 <- mra(ibm.volatility, "la8", 4, "dwt")
names(ibmv.la8) <- c("d1", "d2", "d3", "d4", "s4")
## plot multiresolution analysis of IBM data
par(mfcol=c(6,1), pty="m", mar=c(5-2,4,4-2,2))
plot.ts(ibm.volatility, axes=FALSE, ylab="", main="(a)")
for(i in 1:5)
  plot.ts(ibmv.haar[[i]], axes=FALSE, ylab=names(ibmv.haar)[i])
axis(side=1, at=seq(0,368,by=23),
  labels=c(0,"",46,"",92,"",138,"",184,"",230,"",276,"",322,"",368))
par(mfcol=c(6,1), pty="m", mar=c(5-2,4,4-2,2))
plot.ts(ibm.volatility, axes=FALSE, ylab="", main="(b)")
for(i in 1:5)
  plot.ts(ibmv.la8[[i]], axes=FALSE, ylab=names(ibmv.la8)[i])
axis(side=1, at=seq(0,368,by=23),
  labels=c(0,"",46,"",92,"",138,"",184,"",230,"",276,"",322,"",368))
```

---

mra.2d *Multiresolution Analysis of an Image*

Description

This function performs a level J additive decomposition of the input matrix or image using the pyramid algorithm (Mallat 1989).

Usage

```
mra.2d(x, wf = "la8", J = 4, method = "modwt", boundary = "periodic")
```

Arguments

`x`	A matrix or image containing the data be to decomposed. This must be have dyadic length in both dimensions (but not necessarily the same) for `method="dwt"`.
`wf`	Name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ least asymmetric family.
`J`	Specifies the depth of the decomposition. This must be a number less than or equal to $\log(\text{length}(x), 2)$.
`method`	Either `"dwt"` or `"modwt"`.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the default, then the matrix you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the matrix beyond its boundaries is assumed to be a symmetric reflection of itself.

Details

This code implements a two-dimensional multiresolution analysis by performing the one-dimensional pyramid algorithm (Mallat 1989) on the rows and columns of the input matrix. Either the DWT or MODWT may be used to compute the multiresolution analysis, which is an additive decomposition of the original matrix (image).

Value

Basically, a list with the following components

`LH?`	Wavelet detail image in the horizontal direction.
`HL?`	Wavelet detail image in the vertical direction.
`HH?`	Wavelet detail image in the diagonal direction.
`LL`J	Wavelet smooth image at the coarsest resolution.
J	Depth of the wavelet transform.
`wavelet`	Name of the wavelet filter used.
`boundary`	How the boundaries were handled.

Author(s)

B. Whitcher

References

Mallat, S. G. (1989) A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, No. 7, 674-693.

Mallat, S. G. (1998) *A Wavelet Tour of Signal Processing*, Academic Press.

See Also

`dwt.2d`, `modwt.2d`

Examples

```
## Easy check to see if it works...
## --------------------------------

x <- matrix(rnorm(32*32), 32, 32)
# MODWT
x.mra <- mra.2d(x, method="modwt")
x.mra.sum <- x.mra[[1]]
for(j in 2:length(x.mra))
  x.mra.sum <- x.mra.sum + x.mra[[j]]
sum((x - x.mra.sum)^2)

# DWT
x.mra <- mra.2d(x, method="dwt")
x.mra.sum <- x.mra[[1]]
for(j in 2:length(x.mra))
  x.mra.sum <- x.mra.sum + x.mra[[j]]
sum((x - x.mra.sum)^2)
```

`mra.3d` *Three Dimensional Multiresolution Analysis*

Description

This function performs a level J additive decomposition of the input array using the pyramid algorithm (Mallat 1989).

Usage

```
mra.3d(x, wf, J=4, method="modwt", boundary="periodic")
```

Arguments

`x`	A three-dimensional array containing the data be to decomposed. This must be have dyadic length in all three dimensions (but not necessarily the same) for `method="dwt"`.

`wf` Name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ least asymmetric family.

`J` Specifies the depth of the decomposition. This must be a number less than or equal to $\log(\text{length}(x), 2)$.

`method` Either `"dwt"` or `"modwt"`.

`boundary` Character string specifying the boundary condition. If `boundary=="periodic"` the default and only method implemented, then the matrix you decompose is assumed to be periodic on its defined interval.

Details

This code implements a three-dimensional multiresolution analysis by performing the one-dimensional pyramid algorithm (Mallat 1989) on each dimension of the input array. Either the DWT or MODWT may be used to compute the multiresolution analysis, which is an additive decomposition of the original array.

Value

List structure containing the filter triplets associated with the multiresolution analysis.

Author(s)

B. Whitcher

References

Mallat, S. G. (1989) A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, No. 7, 674-693.

Mallat, S. G. (1998) *A Wavelet Tour of Signal Processing*, Academic Press.

See Also

`dwt.3d`, `modwt.3d`

---

mult.loc *Wavelet-based Testing and Locating for Variance Change Points*

Description

This is the major subroutine for `testing.hov`, providing the workhorse algorithm to recursively test and locate multiple variance changes in so-called long memory processes.

Usage

```
mult.loc(dwt.list, modwt.list, wf, level, min.coef, debug)
```

Arguments

`dwt.list`	List of wavelet vector coefficients from the `dwt`.
`modwt.list`	List of wavelet vector coefficients from the `modwt`.
`wf`	Name of the wavelet filter to use in the decomposition.
`level`	Specifies the depth of the decomposition.
`min.coef`	Minimum number of wavelet coefficients for testing purposes.
`debug`	Boolean variable: if set to `TRUE`, actions taken by the algorithm are printed to the screen.

Details

For details see Section 9.6 of Percival and Walden (2000) or Section 7.3 in Gencay, Selcuk and Whitcher (2001).

Value

Matrix.

Author(s)

B. Whitcher

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

See Also

`rotcumvar`, `testing.hov`.

`my.acf` *Autocovariance Functions via the Discrete Fourier Transform*

Description

Computes the autocovariance function (ACF) for a time series or the cross-covariance function (CCF) between two time series.

Usage

```
my.acf(x)
my.ccf(a, b)
```

Arguments

`x,a,b` time series

Details

The series is zero padded to twice its length before the discrete Fourier transform is applied. Only the values corresponding to nonnegative lags are provided (for the ACF).

Value

The autocovariance function for all nonnegative lags or the cross-covariance function for all lags.

Author(s)

B. Whitcher

Examples

```
data(ibm)
ibm.returns <- diff(log(ibm))
plot(1:length(ibm.returns) - 1, my.acf(ibm.returns), type="h",
     xlab="lag", ylab="ACVS", main="Autocovariance Sequence for IBM Returns")
```

---

nile *Nile River Minima*

Description

Yearly minimal water levels of the Nile river for the years 622 to 1281, measured at the Roda gauge near Cairo (Tousson, 1925, p. 366-385). The data are listed in chronological sequence by row.

The original Nile river data supplied by Beran only contained only 500 observations (622 to 1121). However, the book claimed to have 660 observations (622 to 1281). The remaining observations from the book were added, by hand, but the series still only contained 653 observations (622 to 1264).

Note, now the data consists of 663 observations (spanning the years 622-1284) as in original source (Toussoun, 1925).

Usage

```
data(nile)
```

Format

A length 663 vector.

Source

Toussoun, O. (1925) M\'emoire sur l'Histoire du Nil, Volume 18 in *M\'emoires a l'Institut d'Egypte*, pp. 366-404.

References

Beran, J. (1994) *Statistics for Long-Memory Processes*, Chapman Hall: Englewood, NJ.

`ortho.basis` *Derive Orthonormal Basis from Wavelet Packet Tree*

Description

An orthonormal basis for the discrete wavelet transform may be characterized via a disjoint partitioning of the frequency axis that covers $[0, \frac{1}{2})$. This subroutine produces an orthonormal basis from a full wavelet packet tree.

Usage

```
ortho.basis(xtree)
```

Arguments

`xtree`	is a vector whose entries are associated with a wavelet packet tree.

Details

A wavelet packet tree is a binary tree of Boolean variables. Parent nodes are removed if any of their children exist.

Value

Boolean vector describing the orthonormal basis for the DWPT.

Author(s)

B. Whitcher

Examples

```
data(japan)
J <- 4
wf <- "mb8"
japan.mra <- mra(log(japan), wf, J, boundary="reflection")
japan.nomean <-
  ts(apply(matrix(unlist(japan.mra[-(J+1)]), ncol=J, byrow=FALSE), 1, sum),
     start=1955, freq=4)
japan.nomean2 <- ts(japan.nomean[42:169], start=1965.25, freq=4)
```

```
plot(japan.nomean2, type="l")
japan.dwpt <- dwpt(japan.nomean2, wf, 6)
japan.basis <-
  ortho.basis(portmanteau.test(japan.dwpt, p=0.01, type="other"))
# Not implemented yet
# par(mfrow=c(1,1))
# plot.basis(japan.basis)
```

`per` *Periodogram*

Description

Computation of the periodogram via the Fast Fourier Transform (FFT).

Usage

```
per(z)
```

Arguments

`z`	time series

Author(s)

Author: Jan Beran; modified: Martin Maechler, Date: Sep 1995.

`phase.shift` *Phase Shift Wavelet Coefficients*

Description

Wavelet coefficients are circularly shifted by the amount of phase shift induced by the wavelet transform.

Usage

```
phase.shift(z, wf, inv = FALSE)
phase.shift.packet(z, wf, inv = FALSE)
```

Arguments

`z`	DWT object
`wf`	character string; wavelet filter used in DWT
`inv`	Boolean variable; if `inv=TRUE` then the inverse phase shift is applied

Details

The center-of-energy argument of Hess-Nielsen and Wickerhauser (1996) is used to provide a flexible way to circularly shift wavelet coefficients regardless of the wavelet filter used. The results are not identical to those used by Percival and Walden (2000), but are more flexible.

`phase.shift.packet` is not yet implemented fully.

Value

DWT (DWPT) object with coefficients circularly shifted.

Author(s)

B. Whitcher

References

Hess-Nielsen, N. and M. V. Wickerhauser (1996) Wavelets and time-frequency analysis, *Proceedings of the IEEE*, **84**, No. 4, 523-540.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

`phase.shift.hilbert` *Phase Shift for Hilbert Wavelet Coefficients*

Description

Wavelet coefficients are circularly shifted by the amount of phase shift induced by the discrete Hilbert wavelet transform.

Usage

```
phase.shift.hilbert(x, wf)
phase.shift.hilbert.packet(x, wf)
```

Arguments

`x`	Discete Hilbert wavelet transform (DHWT) object.
`wf`	character string; Hilbert wavelet pair used in DHWT

Details

The "center-of-energy" argument of Hess-Nielsen and Wickerhauser (1996) is used to provide a flexible way to circularly shift wavelet coefficients regardless of the wavelet filter used.

Value

DHWT (DHWPT) object with coefficients circularly shifted.

Author(s)

B. Whitcher

References

Hess-Nielsen, N. and M. V. Wickerhauser (1996) Wavelets and time-frequency analysis, *Proceedings of the IEEE*, **84**, No. 4, 523-540.

See Also

`phase.shift`

`plot.dwt.2d` *Plot Two-dimensional Discrete Wavelet Transform*

Description

Organizes the wavelet coefficients from a 2D DWT into a single matrix and plots it. The coarser resolutions are nested within the lower-lefthand corner of the image.

Usage

```
## S3 method for class 'dwt.2d'
plot(x, cex.axis = 1, plot = TRUE, ...)
```

Arguments

`x`	input matrix (image)
`cex.axis`	`par` plotting parameter that controls the size of the axis text
`plot`	if `plot = FALSE` then the matrix of wavelet coefficients is returned, the default is `plot = TRUE`
`...`	additional graphical parameters if necessary

Details

The wavelet coefficients from the DWT object (a list) are reorganized into a single matrix of the same dimension as the original image and the result is plotted.

Value

Image plot.

Author(s)

B. Whitcher

See Also

`dwt.2d`.

qmf *Quadrature Mirror Filter*

Description

Computes the quadrature mirror filter from a given filter.

Usage

```
qmf(g, low2high=TRUE)
```

Arguments

g	Filter coefficients.
low2high	Logical, default is `TRUE` which means a low-pass filter is input and a high-pass filter is output. Setting `low2high=F` performs the inverse.

Details

None.

Value

Quadrature mirror filter.

Author(s)

B. Whitcher

References

Any basic signal processing text.

See Also

`wave.filter`.

Examples

```
## Haar wavelet filter
g <- wave.filter("haar")$lpf
qmf(g)
```

`rotcumvar` *Rotated Cumulative Variance*

Description

Provides the normalized cumulative sums of squares from a sequence of coefficients with the diagonal line removed.

Usage

```
rotcumvar(x)
```

Arguments

`x`	vector of coefficients to be cumulatively summed (missing values excluded)

Details

The rotated cumulative variance, when plotted, provides a qualitative way to study the time dependence of the variance of a series. If the variance is stationary over time, then only small deviations from zero should be present. If on the other hand the variance is non-stationary, then large departures may exist. Formal hypothesis testing may be performed based on boundary crossings of Brownian bridge processes.

Value

Vector of coefficients that are the sumulative sum of squared input coefficients.

Author(s)

B. Whitcher

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

Selesnick *Miscellaneous Functions for Dual-Tree Wavelet Software*

Description

Miscellaneous functions for dual-tree wavelet software.

Usage

```
cshift(x, m)
cshift2D(x, m)
pm(a, b)
```

Arguments

x	N-point vector
m	amount of shift
a,b	input parameters

Value

y	vector x will be shifed by m samples to the left or matrix x will be shifed by m samples down.
u	$(a + b)/sqrt(2)$
v	$(a - b)/sqrt(2)$

Author(s)

Matlab: S. Cai, K. Li and I. Selesnick; R port: B. Whitcher

References

WAVELET SOFTWARE AT POLYTECHNIC UNIVERSITY, BROOKLYN, NY
http://taco.poly.edu/WaveletSoftware/

`shift.2d` *Circularly Shift Matrices from a 2D MODWT*

Description

Compute phase shifts for wavelet sub-matrices based on the "center of energy" argument of Hess-Nielsen and Wickerhauser (1996).

Usage

```
shift.2d(z, inverse=FALSE)
```

Arguments

`z`	Two-dimensional MODWT object
`inverse`	Boolean value on whether to perform the forward or inverse operation.

Details

The "center of energy" technique of Wickerhauser and Hess-Nielsen (1996) is employed to find circular shifts for the wavelet sub-matrices such that the coefficients are aligned with the original series. This corresponds to applying a (near) linear-phase filtering operation.

Value

Two-dimensional MODWT object with circularly shifted coefficients.

Author(s)

Brandon Whitcher

References

Hess-Nielsen, N. and M. V. Wickerhauser (1996) Wavelets and time-frequency analysis, *Proceedings of the IEEE*, **84**, No. 4, 523-540.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

See Also

`phase.shift`, `modwt.2d`.

Examples

```
n <- 512
G1 <- G2 <- dnorm(seq(-n/4, n/4, length=n))
G <- 100 * zapsmall(outer(G1, G2))
G <- modwt.2d(G, wf="la8", J=6)
k <- 50
xr <- yr <- trunc(n/2) + (-k:k)
par(mfrow=c(3,3), mar=c(1,1,2,1), pty="s")
for (j in names(G)[1:9]) {
  image(G[[j]][xr,yr], col=rainbow(64), axes=FALSE, main=j)
}
Gs <- shift.2d(G)
for (j in names(G)[1:9]) {
  image(Gs[[j]][xr,yr], col=rainbow(64), axes=FALSE, main=j)
}
```

`sine.taper` *Computing Sinusoidal Data Tapers*

Description

Computes sinusoidal data tapers directly from equations.

Usage

```
sine.taper(n, k)
```

Arguments

n	length of data taper(s)
k	number of data tapers

Details

See reference.

Value

A vector or matrix of data tapers (cols = tapers).

Author(s)

B. Whitcher

References

Riedel, K. S. and A. Sidorenko (1995) Minimum bias multiple taper spectral estimation, *IEEE Transactions on Signal Processing*, **43**, 188-195.

See Also

dpss.taper.

Spectral Density Functions *Spectral Density Functions for Long-Memory Processes*

Description

Draws the spectral density functions (SDFs) for standard long-memory processes including fractional difference (FD), seasonal persistent (SP), and seasonal fractional difference (SFD) processes.

Usage

```
fdp.sdf(freq, d, sigma2 = 1)
spp.sdf(freq, d, fG, sigma2 = 1)
spp2.sdf(freq, d1, f1, d2, f2, sigma2 = 1)
sfd.sdf(freq, s, d, sigma2 = 1)
```

Arguments

freq	vector of frequencies, normally from 0 to 0.5
d,d1,d2	fractional difference parameter
fG,f1,f2	Gegenbauer frequency
s	seasonal parameter
sigma2	innovations variance

Value

The power spectrum from an FD, SP or SFD process.

Author(s)

Brandon Whitcher

See Also

fdp.mle, spp.mle.

Examples

```
dB <- function(x) 10 * log10(x)

fdp.main <- expression(paste("FD", group("(",d==0.4,")")))
sfd.main <- expression(paste("SFD", group("(",list(s==12, d==0.4),")")))
spp.main <- expression(paste("SPP",
    group("(",list(delta==0.4, f[G]==1/12),")")))

freq <- 0:512/1024

par(mfrow=c(2,2), mar=c(5-1,4,4-1,2), col.main="darkred")
plot(freq, dB(fdp.sdf(freq, .4)), type="l", xlab="frequency",
     ylab="spectrum (dB)", main=fdp.main)
plot(freq, dB(spp.sdf(freq, .4, 1/12)), type="l", xlab="frequency",
     ylab="spectrum (dB)", font.main=1, main=spp.main)
plot(freq, dB(sfd.sdf(freq, 12, .4)), type="l", xlab="frequency",
     ylab="spectrum (dB)", main=sfd.main)
```

---

spin.covariance *Compute Wavelet Cross-Covariance Between Two Time Series*

Description

Computes wavelet cross-covariance or cross-correlation between two time series.

Usage

```
spin.covariance(x, y, lag.max = NA)
spin.correlation(x, y, lag.max = NA)
```

Arguments

`x`	first time series
`y`	second time series, same length as `x`
`lag.max`	maximum lag to compute cross-covariance (correlation)

Details

See references.

Value

List structure holding the wavelet cross-covariances (correlations) according to scale.

Author(s)

B. Whitcher

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Whitcher, B., P. Guttorp and D. B. Percival (2000) Wavelet analysis of covariance with application to atmospheric time series, *Journal of Geophysical Research*, **105**, No. D11, 14,941-14,962.

See Also

`wave.covariance`, `wave.correlation`.

Examples

```
## Figure 7.9 from Gencay, Selcuk and Whitcher (2001)
data(exchange)
returns <- diff(log(exchange))
returns <- ts(returns, start=1970, freq=12)
wf <- "d4"
demusd.modwt <- modwt(returns[,"DEM.USD"], wf, 8)
demusd.modwt.bw <- brick.wall(demusd.modwt, wf)
jpyusd.modwt <- modwt(returns[,"JPY.USD"], wf, 8)
jpyusd.modwt.bw <- brick.wall(jpyusd.modwt, wf)
n <- dim(returns)[1]
J <- 6
lmax <- 36
returns.cross.cor <- NULL
for(i in 1:J) {
  blah <- spin.correlation(demusd.modwt.bw[[i]], jpyusd.modwt.bw[[i]], lmax)
  returns.cross.cor <- cbind(returns.cross.cor, blah)
}
returns.cross.cor <- ts(as.matrix(returns.cross.cor), start=-36, freq=1)
dimnames(returns.cross.cor) <- list(NULL, paste("Level", 1:J))
lags <- length(-lmax:lmax)
lower.ci <- tanh(atanh(returns.cross.cor) - qnorm(0.975) /
                 sqrt(matrix(trunc(n/2^(1:J)), nrow=lags, ncol=J, byrow=TRUE)
                      - 3))
upper.ci <- tanh(atanh(returns.cross.cor) + qnorm(0.975) /
                 sqrt(matrix(trunc(n/2^(1:J)), nrow=lags, ncol=J, byrow=TRUE)
                      - 3))
par(mfrow=c(3,2), las=1, pty="m", mar=c(5,4,4,2)+.1)
for(i in J:1) {
  plot(returns.cross.cor[,i], ylim=c(-1,1), xaxt="n", xlab="Lag (months)",
       ylab="", main=dimnames(returns.cross.cor)[[2]][i])
  axis(side=1, at=seq(-36, 36, by=12))
  lines(lower.ci[,i], lty=1, col=2)
  lines(upper.ci[,i], lty=1, col=2)
  abline(h=0,v=0)
}
```

spp.mle — *Wavelet-based Maximum Likelihood Estimation for Seasonal Persistent Processes*

Description

Parameter estimation for a seasonal persistent (seasonal long-memory) process is performed via maximum likelihood on the wavelet coefficients.

Usage

```
spp.mle(y, wf, J=log(length(y),2)-1, p=0.01, frac=1)
spp2.mle(y, wf, J=log(length(y),2)-1, p=0.01, dyadic=TRUE, frac=1)
```

Arguments

`y`	Not necessarily dyadic length time series.
`wf`	Name of the wavelet filter to use in the decomposition. See `wave.filter` for those wavelet filters available.
`J`	Depth of the discrete wavelet packet transform.
`p`	Level of significance for the white noise testing procedure.
`dyadic`	Logical parameter indicating whether or not the original time series is dyadic in length.
`frac`	Fraction of the time series that should be used in constructing the likelihood function.

Details

The variance-covariance matrix of the original time series is approximated by its wavelet-based equivalent. A Whittle-type likelihood is then constructed where the sums of squared wavelet coefficients are compared to bandpass filtered version of the true spectral density function. Minimization occurs for the fractional difference parameter d and the Gegenbauer frequency f_G, while the innovations variance is subsequently estimated.

Value

List containing the maximum likelihood estimates (MLEs) of δ, f_G and σ^2, along with the value of the likelihood for those estimates.

Author(s)

B. Whitcher

References

Whitcher, B. (2004) Wavelet-based estimation for seasonal long-memory processes, *Technometrics*, **46**, No. 2, 225-238.

See Also

fdp.mle

---

spp.var *Variance of a Seasonal Persistent Process*

Description

Computes the variance of a seasonal persistent (SP) process using a hypergeometric series expansion.

Usage

```
spp.var(d, fG, sigma2 = 1)
Hypergeometric(a, b, c, z)
```

Arguments

d	Fractional difference parameter.
fG	Gegenbauer frequency.
sigma2	Innovations variance.
a,b,c,z	Parameters for the hypergeometric series.

Details

See Lapsa (1997). The subroutine to compute a hypergeometric series was taken from *Numerical Recipes in C*.

Value

The variance of an SP process.

Author(s)

B. Whitcher

References

Lapsa, P.M. (1997) Determination of Gegenbauer-type random process models. *Signal Processing* **63**, 73-90.

Press, W.H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery (1992) *Numerical Recipes in C*, 2nd edition, Cambridge University Press.

squared.gain — *Squared Gain Function of a Filter*

Description

Produces the modulus squared of the Fourier transform for a given filtering sequence.

Usage

```
squared.gain(wf.name, filter.seq = "L", n = 512)
```

Arguments

`wf.name`	Character string of wavelet filter.
`filter.seq`	Character string of filter sequence. `H` means high-pass filtering and `L` means low-pass filtering. Sequence is read from right to left.
`n`	Length of zero-padded filter. Frequency resolution will be `n`/2+1.

Details

Uses `cascade` subroutine to compute the squared gain function from a given filtering sequence.

Value

Squared gain function.

Author(s)

B. Whitcher

See Also

`wave.filter`, `wavelet.filter`.

Examples

```
par(mfrow=c(2,2))
f.seq <- "H"
plot(0:256/512, squared.gain("d4", f.seq), type="l", ylim=c(0,2),
     xlab="frequency", ylab="L = 4", main="Level 1")
lines(0:256/512, squared.gain("fk4", f.seq), col=2)
lines(0:256/512, squared.gain("mb4", f.seq), col=3)
abline(v=c(1,2)/4, lty=2)
legend(-.02, 2, c("Daubechies", "Fejer-Korovkin", "Minimum-Bandwidth"),
       lty=1, col=1:3, bty="n", cex=1)
f.seq <- "HL"
plot(0:256/512, squared.gain("d4", f.seq), type="l", ylim=c(0,4),
     xlab="frequency", ylab="", main="Level 2")
```

```
lines(0:256/512, squared.gain("fk4", f.seq), col=2)
lines(0:256/512, squared.gain("mb4", f.seq), col=3)
abline(v=c(1,2)/8, lty=2)
f.seq <- "H"
plot(0:256/512, squared.gain("d8", f.seq), type="l", ylim=c(0,2),
     xlab="frequency", ylab="L = 8", main="")
lines(0:256/512, squared.gain("fk8", f.seq), col=2)
lines(0:256/512, squared.gain("mb8", f.seq), col=3)
abline(v=c(1,2)/4, lty=2)
f.seq <- "HL"
plot(0:256/512, squared.gain("d8", f.seq), type="l", ylim=c(0,4),
     xlab="frequency", ylab="", main="")
lines(0:256/512, squared.gain("fk8", f.seq), col=2)
lines(0:256/512, squared.gain("mb8", f.seq), col=3)
abline(v=c(1,2)/8, lty=2)
```

---

stackPlot *Stack Plot*

Description

Stack plot of an object. This function attempts to mimic a function called `stack.plot` in S+WAVELETS. It is mostly a hacked version of `plot.ts` in R.

Usage

```
stackPlot(x, plot.type = c("multiple", "single"), panel = lines,
    log = "", col = par("col"), bg = NA, pch = par("pch"), cex = par("cex"),
    lty = par("lty"), lwd = par("lwd"), ann = par("ann"), xlab = "Time",
    main = NULL, oma = c(6, 0, 5, 0), layout = NULL,
    same.scale = 1:dim(x)[2], ...)
```

Arguments

`x`	`ts` object
`layout`	Doublet defining the dimension of the panel. If not specified, the dimensions are chosen automatically.
`same.scale`	Vector the same length as the number of series to be plotted. If not specified, all panels will have unique axes.
`plot.type,panel,log,col,bg,pch,cex,lty,lwd,ann,xlab,main,oma,...`	See `plot.ts`.

Details

Produces a set of plots, one for each element (column) of `x`.

Author(s)

Brandon Whitcher

---

testing.hov *Testing for Homogeneity of Variance*

Description

A recursive algorithm for detecting and locating multiple variance change points in a sequence of random variables with long-range dependence.

Usage

```
testing.hov(x, wf, J, min.coef=128, debug=FALSE)
```

Arguments

`x`	Sequence of observations from a (long memory) time series.
`wf`	Name of the wavelet filter to use in the decomposition.
`J`	Specifies the depth of the decomposition. This must be a number less than or equal to $\log(\text{length}(x), 2)$.
`min.coef`	Minimum number of wavelet coefficients for testing purposes. Empirical results suggest that 128 is a reasonable number in order to apply asymptotic critical values.
`debug`	Boolean variable: if set to `TRUE`, actions taken by the algorithm are printed to the screen.

Details

For details see Section 9.6 of Percival and Walden (2000) or Section 7.3 in Gencay, Selcuk and Whitcher (2001).

Value

Matrix whose columns include (1) the level of the wavelet transform where the variance change occurs, (2) the value of the test statistic, (3) the DWT coefficient where the change point is located, (4) the MODWT coefficient where the change point is located. Note, there is currently no checking that the MODWT is contained within the associated support of the DWT coefficient. This could lead to incorrect estimates of the location of the variance change.

Author(s)

B. Whitcher

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

See Also

dwt, modwt, rotcumvar, mult.loc.

Thresholding *Wavelet Shrinkage via Thresholding*

Description

Perform wavelet shrinkage using data-analytic, hybrid SURE, manual, SURE, or universal thresholding.

Usage

```
da.thresh(wc, alpha = .05, max.level = 4, verbose = FALSE, return.thresh = FALSE)
hybrid.thresh(wc, max.level = 4, verbose = FALSE, seed = 0)
manual.thresh(wc, max.level = 4, value, hard = TRUE)
sure.thresh(wc, max.level = 4, hard = TRUE)
universal.thresh(wc, max.level = 4, hard = TRUE)
universal.thresh.modwt(wc, max.level = 4, hard = TRUE)
```

Arguments

`wc`	wavelet coefficients
`alpha`	level of the hypothesis tests
`max.level`	maximum level of coefficients to be affected by threshold
`verbose`	if `verbose=TRUE` then information is printed to the screen
`value`	threshold value (only utilized in `manual.thresh`)
`hard`	Boolean value, if `hard=F` then soft thresholding is used
`seed`	sets random seed (only utilized in `hybrid.thresh`)
`return.thresh`	if `return.thresh=TRUE` then the vector of threshold values is returned, otherwise the surviving wavelet coefficients are returned

Details

An extensive amount of literature has been written on wavelet shrinkage. The functions here represent the most basic approaches to the problem of nonparametric function estimation. See the references for further information.

Value

The default output is a list structure, the same length as was input, containing only those wavelet coefficients surviving the threshold.

Author(s)

B. Whitcher (some code taken from R. Todd Ogden)

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Ogden, R. T. (1996) *Essential Wavelets for Statistical Applications and Data Analysis*, Birkhauser.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

Vidakovic, B. (1999) *Statistical Modeling by Wavelets*, John Wiley \& Sons.

`tourism` *U.S. Tourism*

Description

Quarterly U.S. tourism figures from 1960:1 to 1999:4.

Usage

```
data(tourism)
```

Format

A vector containing 160 observations.

Source

Unknown.

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

`unemploy` *U.S. Unemployment*

Description

Monthly U.S. unemployment figures from 1948:1 to 1999:12.

Usage

```
data(unemploy)
```

Format

A vector containing 624 observations.

Source

Unknown.

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

`up.sample` *Upsampling of a vector*

Description

Upsamples a given vector.

Usage

```
up.sample(x, f, y = NA)
```

Arguments

`x`	vector of observations
`f`	frequency of upsampling; e.g, 2, 4, etc.
`y`	value to upsample with; e.g., NA, 0, etc.

Value

A vector twice its length.

Author(s)

B. Whitcher

References

Any basic signal processing text.

`wave.filter` *Select a Wavelet Filter*

Description

Converts name of wavelet filter to filter coefficients.

Usage

```
wave.filter(name)
```

Arguments

`name`	Character string of wavelet filter.

Details

Simple `switch` statement selects the appropriate filter.

Value

List containing the following items:

`L`	Length of the wavelet filter.
`hpf`	High-pass filter coefficients.
`lpf`	Low-pass filter coefficients.

Author(s)

B. Whitcher

References

Daubechies, I. (1992) *Ten Lectures on Wavelets*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM: Philadelphia.

Doroslovacki (1998) On the least asymmetric wavelets, *IEEE Transactions for Signal Processing*, **46**, No. 4, 1125-1130.

Morris and Peravali (1999) Minimum-bandwidth discrete-time wavelets, *Signal Processing*, **76**, No. 2, 181-193.

Nielsen, M. (2000) On the Construction and Frequency Localization of Orthogonal Quadrature Filters, *Journal of Approximation Theory*, **108**, No. 1, 36-52.

See Also

`wavelet.filter`, `squared.gain`.

`wave.variance` *Wavelet Analysis of Univariate/Bivariate Time Series*

Description

Produces an estimate of the multiscale variance, covariance or correlation along with approximate confidence intervals.

Usage

```
wave.variance(x, type="eta3", p=0.025)
wave.covariance(x, y)
wave.correlation(x, y, N, p=0.975)
```

Arguments

`x`	first time series
`y`	second time series
`type`	character string describing confidence interval calculation; valid methods are `gaussian`, `eta1`, `eta2`, `eta3`, `nongaussian`
`p`	(one minus the) two-sided p-value for the confidence interval
`N`	length of time series

Details

The time-independent wavelet variance is basically the average of the squared wavelet coefficients across each scale. As shown in Percival (1995), the wavelet variance is a scale-by-scale decomposition of the variance for a stationary process, and certain non-stationary processes.

Value

Matrix with as many rows as levels in the wavelet transform object. The first column provides the point estimate for the wavelet variance, covariance, or correlation followed by the lower and upper bounds from the confidence interval.

Author(s)

B. Whitcher

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Percival, D. B. (1995) *Biometrika*, **82**, No. 3, 619-631.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

Whitcher, B., P. Guttorp and D. B. Percival (2000) Wavelet Analysis of Covariance with Application to Atmospheric Time Series, *Journal of Geophysical Research*, **105**, No. D11, 14,941-14,962.

Examples

```
## Figure 7.3 from Gencay, Selcuk and Whitcher (2001)
data(ar1)
ar1.modwt <- modwt(ar1, "haar", 6)
ar1.modwt.bw <- brick.wall(ar1.modwt, "haar")
ar1.modwt.var2 <- wave.variance(ar1.modwt.bw, type="gaussian")
ar1.modwt.var <- wave.variance(ar1.modwt.bw, type="nongaussian")
par(mfrow=c(1,1), las=1, mar=c(5,4,4,2)+.1)
matplot(2^(0:5), ar1.modwt.var2[-7,], type="b", log="xy",
        xaxt="n", ylim=c(.025, 6), pch="*LU", lty=1, col=c(1,4,4),
        xlab="Wavelet Scale", ylab="")
matlines(2^(0:5), as.matrix(ar1.modwt.var)[-7,2:3], type="b",
         pch="LU", lty=1, col=3)
axis(side=1, at=2^(0:5))
legend(1, 6, c("Wavelet variance", "Gaussian CI", "Non-Gaussian CI"),
       lty=1, col=c(1,4,3), bty="n")

## Figure 7.8 from Gencay, Selcuk and Whitcher (2001)
data(exchange)
returns <- diff(log(as.matrix(exchange)))
returns <- ts(returns, start=1970, freq=12)
wf <- "d4"
J <- 6
demusd.modwt <- modwt(returns[,"DEM.USD"], wf, J)
demusd.modwt.bw <- brick.wall(demusd.modwt, wf)
jpyusd.modwt <- modwt(returns[,"JPY.USD"], wf, J)
jpyusd.modwt.bw <- brick.wall(jpyusd.modwt, wf)
returns.modwt.cov <- wave.covariance(demusd.modwt.bw, jpyusd.modwt.bw)
par(mfrow=c(1,1), las=0, mar=c(5,4,4,2)+.1)
matplot(2^(0:(J-1)), returns.modwt.cov[-(J+1),], type="b", log="x",
        pch="*LU", xaxt="n", lty=1, col=c(1,4,4), xlab="Wavelet Scale",
        ylab="Wavelet Covariance")
axis(side=1, at=2^(0:7))
abline(h=0)

returns.modwt.cor <- wave.correlation(demusd.modwt.bw, jpyusd.modwt.bw,
                                      N = dim(returns)[1])
par(mfrow=c(1,1), las=0, mar=c(5,4,4,2)+.1)
matplot(2^(0:(J-1)), returns.modwt.cor[-(J+1),], type="b", log="x",
        pch="*LU", xaxt="n", lty=1, col=c(1,4,4), xlab="Wavelet Scale",
        ylab="Wavelet Correlation")
axis(side=1, at=2^(0:7))
abline(h=0)
```

`wavelet.filter` *Higher-Order Wavelet Filters*

Description

Create a wavelet filter at arbitrary scale.

Usage

```
wavelet.filter(wf.name, filter.seq = "L", n = 512)
```

Arguments

`wf.name`	Character string of wavelet filter.
`filter.seq`	Character string of filter sequence. `H` means high-pass filtering and `L` means low-pass filtering. Sequence is read from right to left.
`n`	Length of zero-padded filter. Frequency resolution will be `n/2+1`.

Details

Uses `cascade` subroutine to compute higher-order wavelet coefficient vector from a given filtering sequence.

Value

Vector of wavelet coefficients.

Author(s)

B. Whitcher

References

Bruce, A. and H.-Y. Gao (1996). *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

Doroslovacki, M. L. (1998) On the least asymmetric wavelets, *IEEE Transactions on Signal Processing*, **46**, No. 4, 1125-1130.

Daubechies, I. (1992) *Ten Lectures on Wavelets*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM: Philadelphia.

Morris and Peravali (1999) Minimum-bandwidth discrete-time wavelets, *Signal Processing*, **76**, No. 2, 181-193.

Nielsen, M. (2001) On the Construction and Frequency Localization of Finite Orthogonal Quadrature Filters, *Journal of Approximation Theory*, **108**, No. 1, 36-52.

See Also

`squared.gain`, `wave.filter`.

Examples

```
## Figure 4.14 in Gencay, Selcuk and Whitcher (2001)
par(mfrow=c(3,1), mar=c(5-2,4,4-1,2))
f.seq <- "HLLLLL"
plot(c(rep(0,33), wavelet.filter("mb4", f.seq), rep(0,33)), type="l",
     xlab="", ylab="", main="D(4) in black, MB(4) in red")
lines(c(rep(0,33), wavelet.filter("d4", f.seq), rep(0,33)), col=2)
plot(c(rep(0,35), -wavelet.filter("mb8", f.seq), rep(0,35)), type="l",
     xlab="", ylab="", main="D(8) in black, -MB(8) in red")
lines(c(rep(0,35), wavelet.filter("d8", f.seq), rep(0,35)), col=2)
plot(c(rep(0,39), wavelet.filter("mb16", f.seq), rep(0,39)), type="l",
     xlab="", ylab="", main="D(16) in black, MB(16) in red")
lines(c(rep(0,39), wavelet.filter("d16", f.seq), rep(0,39)), col=2)
```

`wpt.test` *Testing the Wavelet Packet Tree for White Noise*

Description

A wavelet packet tree, from the discrete wavelet packet transform (DWPT), is tested node-by-node for white noise. This is the first step in selecting an orthonormal basis for the DWPT.

Usage

```
cpgram.test(y, p = 0.05, taper = 0.1)
css.test(y)
entropy.test(y)
portmanteau.test(y, p = 0.05, type = "Box-Pierce")
```

Arguments

`y`	wavelet packet tree (from the DWPT)
`p`	significance level
`taper`	weight of cosine bell taper (`cpgram.test` only)
`type`	`"Box-Pierce"` and `other` recognized (`portmanteau.test` only)

Details

Top-down recursive testing of the wavelet packet tree is

Value

Boolean vector of the same length as the number of nodes in the wavelet packet tree.

Author(s)

B. Whitcher

References

Brockwell and Davis (1991) *Time Series: Theory and Methods*, (2nd. edition), Springer-Verlag.

Brown, Durbin and Evans (1975) Techniques for testing the constancy of regression relationships over time, *Journal of the Royal Statistical Society B*, **37**, 149-163.

Percival, D. B., and A. T. Walden (1993) *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques*, Cambridge University Press.

See Also

ortho.basis.

Examples

```
data(mexm)
J <- 6
wf <- "la8"
mexm.dwpt <- dwpt(mexm[-(1:4)], wf, J)
## Not implemented yet
## plot.dwpt(x.dwpt, J)
mexm.dwpt.bw <- dwpt.brick.wall(mexm.dwpt, wf, 6, method="dwpt")
mexm.tree <- ortho.basis(portmanteau.test(mexm.dwpt.bw, p=0.025))
## Not implemented yet
## plot.basis(mexm.tree)
```

---

xbox *Image with Box and X*

Description

$$xbox(i,j) = I_{[i=n/4,\ 3n/4,\ j;\ n/4 \le j \le 3n/4]} + I_{[n/4 \le i \le 3n/4;\ j=n/4,\ 3n/4,\ i]}$$

Usage

```
data(xbox)
```

Format

A 128×128 matrix.

Source

S+WAVELETS.

References

Bruce, A., and H.-Y. Gao (1996) *Applied Wavelet Analysis with S-PLUS*, Springer: New York.

Index