Package ‘rwavelet’

October 14, 2022

Type Package

Title Wavelet Analysis

Version 0.4.1

Date 2020-12-12

Author F. Navarro and C. Chesneau

Maintainer Navarro Fabien <fnavarro@math.cnrs.fr>

Description Perform wavelet analysis (orthogonal,translation invariant, tensorial, 1-2-3d transforms, thresholding, block thresholding, linear,...) with applications to data compression or denoising/regression. The core of the code is a port of 'MATLAB' Wavelab toolbox written by D. Donoho, A. Maleki and M. Shahram (<https://statweb.stanford.edu/~wavelab/>).

URL https://github.com/fabnavarro/rwavelet

BugReports https://github.com/fabnavarro/rwavelet/issues

License LGPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports signal

Suggests knitr, rmarkdown, misc3d, imager

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-12-12 15:00:03 UTC

R topics documented:

`aconv` *Convolution tool for two-scale transform*

Description

Filtering by periodic convolution of x with the time-reverse of f.

Usage

```
aconv(f, x)
```

Arguments

`f`	filter.
`x`	1-d signal.

Value

y filtered result.

See Also

`iconvv`, `UpDyadHi`, `UpDyadLo`, `DownDyadHi`, `DownDyadLo`.

Examples

```
qmf <- MakeONFilter('Haar')
x <- MakeSignal('HeaviSine',2^3)
aconv(qmf,x)
```

BlockThresh *1d wavelet Block Thresholding*

Description

This function is used for thresholding coefficients by group (or block) according to the hard or soft thresholding rule.

Usage

```
BlockThresh(wc, j0, hatsigma, L, qmf, thresh = "hard")
```

Arguments

wc	wavelet coefficients.
j0	coarsest decomposition scale.
hatsigma	estimator of noise variance.
L	Block size (n mod L must be 0).
qmf	Orthonormal quadrature mirror filter.
thresh	'hard' or 'soft'.

Value

wcb wavelet coefficient estimators.

See Also

invblock_partition, invblock_partition.

Examples

```
n <- 64
x <- MakeSignal('Ramp', n)
sig <- 0.01
y <- x + rnorm(n, sd=sig)
j0 <- 1
qmf <- MakeONFilter('Daubechies',8)
wc <- FWT_PO(y, j0, qmf)
L <- 2
wcb <- BlockThresh(wc, j0, sig, L, qmf, "hard")
```

`block_partition` *Construct 1d block partition*

Description

This function is used to group the coefficients into blocks (or groups) of size L.

Usage

```
block_partition(x, L)
```

Arguments

x	(noisy) wc at a given scale.
L	block size.

Value

`out` partition of coefficients by block.

See Also

invblock_partition, BlockThresh.

Examples

```
x <- MakeSignal('Ramp', 8)
j0 <- 0
qmf <- MakeONFilter('Haar')
wc <- FWT_PO(x, j0, qmf)
L <- 2
wcb <- block_partition(wc, L)
```

`block_partition2d` *Construct 2d block partition*

Description

Group the coefficients into blocks (or groups) of size L.

Usage

```
block_partition2d(x, L)
```

Arguments

x	(noisy) wc at a given scale.
L	block size.

Value

`out` partition of coefficients by block.

See Also

invblock_partition2d

Examples

```
x <- matrix(rnorm(2^2), ncol=2)
j0 <- 0
qmf <- MakeONFilter('Haar')
wc <- FWT2_PO(x, j0, qmf)
L <- 2
wcb <- block_partition2d(wc, L)
```

CircularShift *Circular Shifting of a matrix/image*

Description

Pixels that get shifted off one side of the image are put back on the other side.

Usage

```
CircularShift(matrix, colshift = 0, rowshift = 0)
```

Arguments

matrix	2-d signal (matrix).
colshift	column shift index (integer).
rowshift	row shift index (integer).

Value

`result` 2-d shifted signal.

See Also

FWT2_TI, IWT2_TI.

Examples

```
A <- matrix(1:4, ncol=2, byrow=TRUE)
CircularShift(A, 0, -1)
```

`cubelength` *Find length and dyadic length of square array*

Description

3d counterpart of Donoho's quadlength utilized by the 2d pair. Original matlab code Vicki Yang and Brani Vidakovic.

Usage

```
cubelength(x)
```

Arguments

`x`	3-d array; dim(n,n,n), n = 2^J (hopefully).

Value

`n` length(x).

`J` least power of two greater than n.

See Also

`FWT3_PO`, `IWT3_PO`.

Examples

```
cubelength(array(1:3, c(2,2,2)))
```

`CVlinear` *2-Fold Cross Validation for linear estimator*

Description

Selection of the number of wavelet coefficients to be maintained by the cross validation method proposed by Nason in the case of threshold selection. This method is adapted here to select among linear estimators.

Usage

```
CVlinear(Y, L, qmf, D, wc)
```

Arguments

Y	Noisy observations.
L	Level of coarsest scale.
qmf	Orthonormal quadrature mirror filter.
D	Dimension vector of the models considered.
wc	1-d wavelet coefficients.

Value

CritCV Cross validation criteria.

hat_f_m_2FCV

References

Nason, G. P. (1996). Wavelet shrinkage using cross-validation. *Journal of the Royal Statistical Society: Series B*, 58(2), 463–479.

Navarro, F. and Saumard, A. (2017). Slope heuristics and V-Fold model selection in heteroscedastic regression using strongly localized bases. *ESAIM: Probability and Statistics*, 21, 412–451.

DownDyadHi *Hi-Pass Downsampling operator (periodized)*

Description

Hi-Pass Downsampling operator (periodized)

Usage

```
DownDyadHi(x, qmf)
```

Arguments

x	1-d signal at fine scale.
qmf	filter.

Value

y 1-d signal at coarse scale.

See Also

DownDyadLo, UpDyadHi, UpDyadLo, FWT_PO, iconvv.

Examples

```
qmf <- MakeONFilter('Haar')
x <- MakeSignal('HeaviSine',2^3)
DownDyadHi(x, qmf)
```

DownDyadLo — *Lo-Pass Downsampling operator (periodized)*

Description

Lo-Pass Downsampling operator (periodized)

Usage

```
DownDyadLo(x, qmf)
```

Arguments

x	1-d signal at fine scale.
qmf	filter.

Value

d 1-d signal at coarse scale.

See Also

DownDyadHi, UpDyadHi, UpDyadLo, FWT_PO, aconv.

Examples

```
qmf <- MakeONFilter('Haar')
x <- MakeSignal('HeaviSine',2^3)
DownDyadLo(x,qmf)
```

dyad — *Index entire j-th dyad of 1-d wavelet xform*

Description

Index entire j-th dyad of 1-d wavelet xform

Usage

```
dyad(j)
```

Arguments

j	integer.

Value

`ix` list of all indices of wavelet coeffts at j-th level.

Examples

```
dyad(0)
```

---

dyadlength *Find length and dyadic length of array*

Description

Find length and dyadic length of array

Usage

```
dyadlength(x)
```

Arguments

`x`	array of length n = 2^J (hopefully).

Value

`n` length(x).

`J` least power of two greater than n.

See Also

`quadlength`, `dyad`

Examples

```
x <- MakeSignal('Ramp', 8)
dyadlength(x)
```

FTWT2_PO *2-d tensor wavelet transform (periodized, orthogonal).*

Description

A two-dimensional Wavelet Transform is computed for the array x. qmf filter may be obtained from MakeONFilter. To reconstruct, use ITWT2_PO.

Usage

```
FTWT2_PO(x, L, qmf)
```

Arguments

x	2-d image (n by n array, n dyadic).
L	coarse level.
qmf	quadrature mirror filter.

Value

wc 2-d wavelet transform.

See Also

ITWT2_PO, MakeONFilter.

Examples

```
qmf <- MakeONFilter('Daubechies', 10)
L <- 0
x <- matrix(rnorm(2^2), ncol=2)
wc <- FTWT2_PO(x, L, qmf)
```

FWT2_PO *2-d MRA Forwad Wavelet Transform (periodized, orthogonal)*

Description

A two-dimensional wavelet transform is computed for the array x. qmf filter may be obtained from MakeONFilter. To reconstruct, use IWT2_PO.

Usage

```
FWT2_PO(x, L, qmf)
```

Arguments

x	2-d image (n by n array, n dyadic).
L	coarse level.
qmf	quadrature mirror filter.

Value

wc 2-d wavelet transform.

See Also

IWT2_PO, MakeONFilter.

Examples

```
qmf <- MakeONFilter('Daubechies', 10)
L <- 3
x <- matrix(rnorm(128^2),ncol=128)
wc <- FWT2_PO(x, L, qmf)
```

FWT2_TI — *2-d Translation Invariant Forward Wavelet Transform*

Description

1. qmf filter may be obtained from MakeONFilter. 2. usually, length(qmf) < 2^(L+1). 3. To reconstruct use IWT_TI.

Usage

```
FWT2_TI(x, L, qmf)
```

Arguments

x	2-d image (n by n real array, n dyadic).
L	degree of coarsest scale.
qmf	orthonormal quadrature mirror filter.

Value

TIWT translation-invariant wavelet transform table, (3(J-L)+1)n by n.

Examples

```
x <- matrix(rnorm(2^2), ncol=2)
L <- 0
qmf <- MakeONFilter('Haar')
TIWT <- FWT2_TI(x, L, qmf)
```

FWT3_PO *3-d MRA Forward Wavelet Transform (periodized, orthogonal)*

Description

A three-dimensional wavelet transform is computed for the array x. qmf filter may be obtained from MakeONFilter. To reconstruct, use IWT3_PO.

Usage

```
FWT3_PO(x, L, qmf)
```

Arguments

x	3-d array (n by n by n array, n dyadic).
L	coarse level.
qmf	quadrature mirror filter.

Details

3-D counterpart of Donoho's FWT2_PO, original matlab code Vicki Yang and Brani Vidakovic.

Value

wc 3-d wavelet transform.

See Also

IWT3_PO, MakeONFilter.

Examples

```
qmf <- MakeONFilter('Daubechies', 10)
L <- 3
x <- array(rnorm(32^3), c(32,32,32))
wc <- FWT3_PO(x, L, qmf)
```

FWT_PO *Forward Wavelet Transform (periodized, orthogonal)*

Description

1. qmf filter may be obtained from MakeONFilter. 2. usually, length(qmf) < 2^(L+1). 3. To reconstruct use IWT_PO.

Usage

```
FWT_PO(x, L, qmf)
```

Arguments

x	1-d signal; length(x) = 2^J.
L	Coarsest Level of V_0; L « J.
qmf	quadrature mirror filter (orthonormal).

Value

wc 1-d wavelet transform of x.

See Also

IWT_PO, MakeONFilter.

Examples

```
x <- MakeSignal('Ramp', 8)
L <- 0
qmf <- MakeONFilter('Haar')
wc <- FWT_PO(x, L, qmf)
```

FWT_TI *Translation Invariant Forward Wavelet Transform*

Description

1. qmf filter may be obtained from MakeONFilter. 2. usually, length(qmf) < 2^(L+1). 3. To reconstruct use IWT_TI.

Usage

```
FWT_TI(x, L, qmf)
```

Arguments

x	array of dyadic length n=2^J.
L	degree of coarsest scale.
qmf	orthonormal quadrature mirror filter.

Value

TIWT stationary wavelet transform table.

See Also

IWT_TI, MakeONFilter.

Examples

```
x <- MakeSignal('Ramp', 8)
L <- 0
qmf <- MakeONFilter('Haar')
TIWT <- FWT_TI(x, L, qmf)
```

GWN *Generation of Gaussian White Noise*

Description

Generation of Gaussian White Noise

Usage

```
GWN(n, sigma)
```

Arguments

n	sample size.
sigma	standard deviation.

Value

epsilon resulting noise.

Examples

```
GWN(10, 0.1)
```

HardThresh *Apply Hard Threshold*

Description

Apply Hard Threshold

Usage

```
HardThresh(y, t)
```

Arguments

y	Noisy Data.
t	Threshold.

Value

x filtered result (y 1_|y|>t).

See Also

SoftThresh.

Examples

```
f <- MakeSignal('HeaviSine',2^3)
qmf <- MakeONFilter('Daubechies', 10)
L <- 0
wc <- FWT_PO(f, L, qmf)
thr <- 2
wct <- HardThresh(wc, thr)
fhard <- IWT_PO(wct, L, qmf)
```

iconvv *Convolution tool for two-scale transform*

Description

Filtering by periodic convolution of x with f.

Usage

```
iconvv(f, x)
```

Arguments

f	filter.
x	1-d signal.

Value

y filtered result.

See Also

aconv, UpDyadHi, UpDyadLo, DownDyadHi, DownDyadLo.

Examples

```
qmf <- MakeONFilter('Haar')
x <- MakeSignal('HeaviSine',2^3)
iconvv(qmf,x)
```

---

invblock_partition — *Inversion of the 1d block partition*

Description

Inversion of the 1d block partition

Usage

```
invblock_partition(x, n, L)
```

Arguments

x	partition of coefficients by block.
n	scale.
L	block size.

See Also

block_partition, BlockThresh.

Examples

```
n <- 8
x <- MakeSignal('Ramp', n)
j0 <- 1
qmf <- MakeONFilter('Haar')
wc <- FWT_PO(x, j0, qmf)
L <- 2
wcb <- block_partition(wc, L)
wcib <- invblock_partition(wcb, n, L)
```

invblock_partition2d *Inversion of the 2d block partition*

Description

Inversion of the 2d block partition

Usage

```
invblock_partition2d(x, n, L)
```

Arguments

x	partition of coefficients by block.
n	scale.
L	block size.

Value

out coefficients.

See Also

block_partition2d

Examples

```
n <- 2
x <- matrix(rnorm(n^2), ncol=2)
j0 <- 0
qmf <- MakeONFilter('Haar')
wc <- FWT2_PO(x, j0, qmf)
L <- 2
wcb <- block_partition2d(wc, L)
wcib <- invblock_partition2d(wcb, n, L)
```

ITWT2_PO *Inverse 2-d Tensor Wavelet Transform (periodized, orthogonal)*

Description

If wc is the result of a forward 2d wavelet transform, with wc <- FTWT2_PO(x,L,qmf), then x <- ITWT2_PO(wc,L,qmf) reconstructs x exactly. qmf is a nice qmf, e.g. one made by MakeONFilter.

Usage

```
ITWT2_PO(wc, L, qmf)
```

Arguments

wc	2-d wavelet transform (n by n array, n dyadic).
L	coarse level.
qmf	quadrature mirror filter.

Value

x 2-d signal reconstructed from wc.

See Also

FTWT2_PO, MakeONFilter.

Examples

```
qmf <- MakeONFilter('Daubechies', 10)
L <- 0
x <- matrix(rnorm(2^2), ncol=2)
wc <- FTWT2_PO(x, L, qmf)
xr <- ITWT2_PO(wc,L,qmf)
```

IWT2_PO *Inverse 2-d MRA Wavelet Transform (periodized, orthogonal)*

Description

If `wc` is the result of a forward 2d wavelet transform, with `wc <- FWT2_PO(x,L,qmf)`. then `x <- IWT2_PO(wc,L,qmf)` reconstructs `x` exactly `qmf` is a nice qmf, e.g. one made by `MakeONFilter`.

Usage

```
IWT2_PO(wc, L, qmf)
```

Arguments

wc	2-d wavelet transform (n by n array, n dyadic).
L	coarse level.
qmf	quadrature mirror filter.

Value

x 2-d signal reconstructed from wc.

See Also

FWT2_PO, MakeONFilter.

Examples

```
qmf <- MakeONFilter('Daubechies', 10)
L <- 3
x <- matrix(rnorm(128^2),ncol=128)
wc <- FWT2_PO(x, L, qmf)
xr <- IWT2_PO(wc,L,qmf)
```

IWT2_TI *Invert 2-d Translation Invariant Wavelet Transform*

Description

Invert 2-d Translation Invariant Wavelet Transform

Usage

```
IWT2_TI(tiwt, L, qmf)
```

Arguments

tiwt	translation-invariant wavelet transform table, (3(J-L)+1)n by n.
L	degree of coarsest scale.
qmf	orthonormal quadrature mirror filter.

Value

x 2-d image reconstructed from translation-invariant transform TIWT.

Examples

```
x <- matrix(rnorm(2^2), ncol=2)
L <- 0
qmf <- MakeONFilter('Haar')
TIWT <- FWT2_TI(x, L, qmf)
xr <- IWT2_TI(TIWT,L,qmf)
```

IWT3_PO *Inverse 3-d MRA Wavelet Transform (periodized, orthogonal)*

Description

If `wc` is the result of a forward 3-d wavelet transform, with `wc <- FWT3_PO(x, L, qmf).` then `x <- IWT3_PO(wc, L, qmf)` reconstructs `x` exactly `qmf` is a nice qmf, e.g. one made by `MakeONFilter`.

Usage

```
IWT3_PO(wc, L, qmf)
```

Arguments

`wc`	3-d wavelet transform (n by n by n array, n dyadic).
`L`	coarse level.
`qmf`	quadrature mirror filter.

Details

3-d counterpart of Donoho's IWT2_PO, original matlab code by Vicki Yang and Brani Vidakovic.

Value

x 3-d signal reconstructed from wc.

See Also

`FWT3_PO`, `MakeONFilter`.

Examples

```
qmf <- MakeONFilter('Daubechies', 10)
L <- 3
x <- array(rnorm(32^3), c(32, 32, 32))
wc <- FWT3_PO(x, L, qmf)
xr <- IWT3_PO(wc, L, qmf)
```

IWT_PO *Inverse Wavelet Transform (periodized, orthogonal)*

Description

Suppose wc <- FWT_PO(x,L,qmf) where qmf is an orthonormal quad. mirror filter, e.g. one made by MakeONFilter. Then x can be reconstructed by x <- IWT_PO(wc,L,qmf).

Usage

```
IWT_PO(wc, L, qmf)
```

Arguments

wc	1-d wavelet transform: length(wc) = 2^J.
L	Coarsest scale (2^(-L) = scale of V_0); L « J.
qmf	quadrature mirror filter (orthonormal).

Value

x 1-d signal reconstructed from wc.

See Also

FWT_PO, MakeONFilter.

Examples

```
x <- MakeSignal('Ramp', 8)
L <- 0
qmf <- MakeONFilter('Haar')
wc <- FWT_PO(x, L, qmf)
xr <- IWT_PO(wc,L,qmf)
```

IWT_TI *Invert Translation Invariant Wavelet Transform*

Description

Invert Translation Invariant Wavelet Transform

Usage

```
IWT_TI(pkt, qmf)
```

Arguments

`pkt`	translation-invariant wavelet transform table (TIWT).
`qmf`	orthonormal quadrature mirror filter.

Value

`x` 1-d signal reconstructed from translation-invariant transform TIWT.

See Also

`FWT_TI`, `MakeONFilter`.

Examples

```
x <- MakeSignal('Ramp', 8)
L <- 0
qmf <- MakeONFilter('Haar')
TIWT <- FWT_TI(x, L, qmf)
xr <- IWT_TI(TIWT,qmf)
```

`JSThresh` *Apply James-Stein Threshold*

Description

(also called the nonnegative garrote)

Usage

```
JSThresh(y, t)
```

Arguments

`y`	Noisy Data.
`t`	Threshold.

Value

`x` filtered result.

See Also

`HardThresh`, `SoftThresh`

Examples

```
f <- MakeSignal('HeaviSine', 2^3)
qmf <- MakeONFilter('Daubechies', 10)
L <- 0
wc <- FWT_PO(f, L, qmf)
thr <- 2
wct <- JSThresh(wc, thr)
fsoft <- IWT_PO(wct, L, qmf)
```

`lshift` *Circular left shift of 1-d signal*

Description

Circular left shift of 1-d signal

Usage

```
lshift(a)
```

Arguments

`a`	1-d signal.

Value

`l` 1-d signal l(i) = x(i+1) except l(n) = x(1).

Examples

```
x <- MakeSignal('HeaviSine',2^3)
lshift(x)
```

`MAD` *Median Absolute Deviation*

Description

Compute the median absolute deviation.

Usage

```
MAD(x)
```

Arguments

`x`	1-d signal.

Examples

```
x <- c(1, 1, 2, 2, 4, 6, 9)
MAD(x)
```

MakeONFilter *Generate Orthonormal QMF Filter for Wavelet Transform*

Description

The Haar filter (which could be considered a Daubechies-2) was the first wavelet, though not called as such, and is discontinuous.

Usage

```
MakeONFilter(Type, Par)
```

Arguments

`Type`	string, 'Haar', 'Beylkin', 'Coiflet', 'Daubechies' 'Symmlet', 'Vaidyanathan','Battle'.
`Par`	integer, it is a parameter related to the support and vanishing moments of the wavelets, explained below for each wavelet.

Details

The Beylkin filter places roots for the frequency response function close to the Nyquist frequency on the real axis.

The Coiflet filters are designed to give both the mother and father wavelets 2*Par vanishing moments; here Par may be one of 1,2,3,4 or 5.

The Daubechies filters are minimal phase filters that generate wavelets which have a minimal support for a given number of vanishing moments. They are indexed by their length, Par, which may be one of 4,6,8,10,12,14,16,18 or 20. The number of vanishing moments is par/2.

Symmlets are also wavelets within a minimum size support for a given number of vanishing moments, but they are as symmetrical as possible, as opposed to the Daubechies filters which are highly asymmetrical. They are indexed by Par, which specifies the number of vanishing moments and is equal to half the size of the support. It ranges from 4 to 10.

The Vaidyanathan filter gives an exact reconstruction, but does not satisfy any moment condition. The filter has been optimized for speech coding.

The Battle-Lemarie filter generate spline orthogonal wavelet basis. The parameter Par gives the degree of the spline. The number of vanishing moments is Par+1.

Value

`qmf` quadrature mirror filter.

See Also

FWT_PO, IWT_PO, FWT2_PO, IWT2_PO.

Examples

```
Type <- 'Coiflet'
Par <- 1
qmf <- MakeONFilter(Type, Par)
```

MakeSignal — *Make artificial signal*

Description

Make artificial signal

Usage

```
MakeSignal(name, n)
```

Arguments

name	string, 'HeaviSine', 'Bumps', 'Blocks', 'Doppler', 'Ramp','Cusp', 'Sing', 'HiSine', 'LoSine', 'LinChirp', 'TwoChirp', 'QuadChirp', 'MishMash', 'WernerSorrows' (Heisenberg), 'Leopold' (Kronecker), 'Riemann', 'HypChirps', 'LinChirps', 'Chirps', 'Gabor', 'sineoneoverx', 'Cusp2', 'SmoothCusp', 'Piece-Regular' (PieceWise Smooth), 'Piece-Polynomial' (Piece-Wise 3rd degree polynomial).
n	desired signal length.

Value

sig 1-d signal.

See Also

FWT_PO, IWT_PO, FWT2_PO, IWT2_PO.

Examples

```
name <- 'Cusp'
n <- 2^5
sig <- MakeSignal(name,n)
```

`MakeSignalNewb` *Make artificial 1-d signal*

Description

Make artificial 1-d signal

Usage

```
MakeSignalNewb(name, n)
```

Arguments

`name`	string, 'Cusp','Step','Wave','Blip','Blocks', 'Bumps','HeaviSine','Doppler','Angles', 'Parabolas','Time Shifted Sine','Spikes','Corner'
`n`	desired signal length.

Value

`sig` 1-d signal.

See Also

FWT_PO, IWT_PO, FWT2_PO, IWT2_PO.

Examples

```
name <- 'Cusp'
n <- 2^5
sig <- MakeSignalNewb(name,n)
```

`MinMaxThresh` *Minimax Thresholding*

Description

Minimax Thresholding

Usage

```
MinMaxThresh(y)
```

Arguments

`y`	signal upon which to perform thresholding.

Value

`x` result.

References

D.L. Donoho and I.M. Johnstone (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3), 425–455.

`MirrorFilt` *Apply (-1)^t modulation*

Description

h(t) = (-1)^(t-1) * x(t), 1 <= t <= length(x)

Usage

```
MirrorFilt(x)
```

Arguments

`x`	1-d signal.

Value

`h` 1-d signal with DC frequency content shifted to Nyquist frequency

See Also

`DownDyadHi`.

Examples

```
x <- MakeSignal('HeaviSine',2^3)
h <- MirrorFilt(x)
```

MultiMAD — *Apply Shrinkage with level-dependent Noise level estimation*

Description

Apply Shrinkage with level-dependent Noise level estimation

Usage

```
MultiMAD(wc, L)
```

Arguments

wc	Wavelet Transform of noisy sequence.
L	low-resolution cutoff for Wavelet Transform.

Value

ws result of applying VisuThresh to each wavelet level, after scaling so MAD of coefficienst at each level = .6745

MultiSURE — *Apply Shrinkage to Wavelet Coefficients*

Description

SURE referes to Stein's Unbiased Risk Estimate.

Usage

```
MultiSURE(wc, L)
```

Arguments

wc	Wavelet Transform of noisy sequence with N(0,1) noise.
L	low-frequency cutoff for Wavelet Transform.

Value

ws result of applying SUREThresh to each dyadic block.

MultiVisu — *Apply Universal Thresholding to Wavelet Coefficients*

Description

Apply Universal Thresholding to Wavelet Coefficients

Usage

```
MultiVisu(wc, L)
```

Arguments

`wc`	Wavelet Transform of noisy sequence with N(0,1) noise.
`L`	low-frequency cutoff for Wavelet Transform

Value

`x` result of applying VisuThresh to each High Frequency Dyadic Block.

packet — *Packet table indexing*

Description

Packet table indexing

Usage

```
packet(d, b, n)
```

Arguments

`d`	depth of splitting in packet decomposition.
`b`	block index among 2^d possibilities at depth d.
`n`	length of signal.

Value

`p` linear indices of all coeff's in that block.

Examples

```
packet(1, 1, 8)
```

PlotSpikes *Plot 1-d signal as baseline with series of spikes*

Description

Plot 1-d signal as baseline with series of spikes

Usage

```
PlotSpikes(base, t, x, L, J)
```

Arguments

base	number, baseline level.
t	ordinate values.
x	1-d signal, specifies spike deflections from baseline.
L	level of coarsest scale.
J	least power of two greater than n.

Value

A plot of spikes on a baseline.

See Also

PlotWaveCoeff.

Examples

```
## Not run:
PlotSpikes(base, t, x, L, J)

## End(Not run)
```

PlotWaveCoeff *Spike-plot display of wavelet coefficients*

Description

Spike-plot display of wavelet coefficients

Usage

```
PlotWaveCoeff(wc, L, scal)
```

Arguments

`wc`	1-d wavelet transform.
`L`	level of coarsest scale.
`scal`	scale factor (0 ==> autoscale).

Value

A display of wavelet coefficients (coarsest level NOT included) by level and position.

See Also

`FWT_PO`, `IWT_PO`, `PlotSpikes`.

Examples

```
x <- MakeSignal('Ramp', 128)
qmf <- MakeONFilter('Daubechies', 10)
L <- 3
scal <- 1
wc <- FWT_PO(x, L, qmf)
PlotWaveCoeff(wc,L,scal)
```

`quadlength` *Find length and dyadic length of square matrix*

Description

h(t) = (-1)^(t-1) * x(t), 1 <= t <= length(x)

Usage

```
quadlength(x)
```

Arguments

`x`	2-d image; dim(n,n), n = 2^J (hopefully).

Value

`n` length(x).

`J` least power of two greater than n.

Examples

```
quadlength(matrix(1:16,ncol=4))
```

`RaphNMR` *Nuclear magnetic resonance (NMR) signal*

Description

A dataset containing a NMR signal.

Usage

```
data(RaphNMR)
```

Format

A numeric vector of length 1024.

Source

MRS Unit, VA Medical Center, San Francisco. Adrain Maudsley, Ph.D., Professor of Radiology. This NMR signal was obtained from Chris Raphael, then a postdoctoral fellow in the Department of Statistics at Stanford University who was working on Hidden Markov Models for restoring NMR Spectra.

`repmat` *Replicate and tile an array*

Description

Repeat copies of array (equivalent of the repmat matlab function).

Usage

```
repmat(a, n, m)
```

Arguments

`a`	input array (scalar, vector, matrix).
`n`	number of time to repeat input array in row and column dimensions.
`m`	repetition factor.

Examples

```
repmat(10,3,2)
```

rshift — *Circular right shift of 1-d signal*

Description

Circular right shift of 1-d signal

Usage

```
rshift(a)
```

Arguments

`a`	1-d signal.

Value

`r` 1-d signal r(i) = x(i-1) except r(1) = x(n).

Examples

```
x <- MakeSignal('HeaviSine', 2^3)
rshift(x)
```

ShapeAsRow — *Make signal a row vector*

Description

Make signal a row vector

Usage

```
ShapeAsRow(sig)
```

Arguments

`sig`	a row or column vector.

Value

`row` a row vector.

Examples

```
sig <- matrix(1:4)
row <- ShapeAsRow(sig)
```

SLphantom — *3-d Shepp-Logan phantom*

Description

A dataset containing a 3d head phantom that can be used to test 3-d reconstruction algorithms. Shepp-Logan phantom is well-known imitation of human cerebral.

Usage

```
data(SLphantom)
```

Format

A numeric array of size 64x64x64.

SNR — *Signal/Noise ratio*

Description

Signal/Noise ratio

Usage

```
SNR(x, y)
```

Arguments

x	Original reference signal.
y	Restored or noisy signal.

Value

Signal/Noise ratio.

Examples

```
n <- 2^4
x <- MakeSignal('HeaviSine', n)
y <- x + rnorm(n, mean=0, sd=1)
SNR(x, y)
```

SoftThresh *Apply Soft Threshold*

Description

Apply Soft Threshold

Usage

```
SoftThresh(y, t)
```

Arguments

y	Noisy Data.
t	Threshold.

Value

x filtered result (y 1_|y|>t).

See Also

HardThresh

Examples

```
f <- MakeSignal('HeaviSine', 2^3)
qmf <- MakeONFilter('Daubechies', 10)
L <- 0
wc <- FWT_PO(f, L, qmf)
thr <- 2
wct <- SoftThresh(wc, thr)
fsoft <- IWT_PO(wct, L, qmf)
```

SUREThresh *Adaptive Threshold Selection Using Principle of SURE*

Description

SURE referes to Stein's Unbiased Risk Estimate.

Usage

```
SUREThresh(y)
```

Arguments

`y`	Noisy Data with Std. Deviation = 1.

Value

`x` Estimate of mean vector

`thresh` Threshold used.

`UpDyadHi` *Hi-Pass Upsampling operator; periodized*

Description

Hi-Pass Upsampling operator; periodized

Usage

```
UpDyadHi(x, qmf)
```

Arguments

`x`	1-d signal at coarser scale.
`qmf`	filter.

Value

`u` 1-d signal at finer scale.

See Also

`DownDyadLo`, `DownDyadHi`, `UpDyadLo`, `IWT_PO`, `aconv`.

Examples

```
qmf <- MakeONFilter('Haar')
x <- MakeSignal('HeaviSine',2^3)
UpDyadHi(x,qmf)
```

UpDyadLo — *Lo-Pass Upsampling operator; periodized*

Description

Lo-Pass Upsampling operator; periodized

Usage

```
UpDyadLo(x, qmf)
```

Arguments

`x`	1-d signal at coarser scale.
`qmf`	filter.

Value

y 1-d signal at finer scale.

See Also

DownDyadLo, DownDyadHi, UpDyadHi, IWT_PO, iconvv.

Examples

```
qmf <- MakeONFilter('Haar')
x <- MakeSignal('HeaviSine',2^3)
UpDyadLo(x,qmf)
```

UpSampleN — *Upsampling operator*

Description

Upsampling operator

Usage

```
UpSampleN(x, s)
```

Arguments

`x`	1-d signal, of length n.
`s`	upsampling scale, default = 2.

Value

y 1-d signal, of length s*n with zeros interpolating alternate samples y(s*i-1) = x(i), i=1,...,n

---

ValSUREThresh — *Adaptive Threshold Selection Using Principle of SURE*

Description

SURE referes to Stein's Unbiased Risk Estimate.

Usage

```
ValSUREThresh(x)
```

Arguments

`x`	Noisy Data with Std. Deviation = 1.

Value

`thresh` Value of Threshold.

VisuThresh — *Visually calibrated Adaptive Smoothing*

Description

Visually calibrated Adaptive Smoothing

Usage

```
VisuThresh(y, thresh = "soft")
```

Arguments

`y`	Signal upon which to perform visually calibrated Adaptive Smoothing.
`thresh`	'hard' or 'soft'.

Value

`x` result of applying VisuThresh.

References

D.L. Donoho and I.M. Johnstone (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3), 425–455.

Index