Package ‘rwavelet’

March 14, 2019

Type Package
Title Wavelet Analysis
Version 0.4.0
Date 2019-03-14
Author F. Navarro and C. Chesneau
Maintainer Navarro Fabien <fabien.navarro@ensai.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 http://github.com/fabnavarro/rwavelet
BugReports http://github.com/fabnavarro/rwavelet/issues
License LGPL (>= 2)
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
Imports signal
Suggests knitr, rmarkdown, imager, rgl, misc3d
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2019-03-14 21:40:03 UTC

R topics documented:

   aconv ........................................... 3
   BlockThresh .................................... 3
   block_partition .................................. 4
   block_partition2d ................................ 4
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CircularShift</td>
<td>5</td>
</tr>
<tr>
<td>cubelength</td>
<td>5</td>
</tr>
<tr>
<td>CVlinear</td>
<td>6</td>
</tr>
<tr>
<td>DownDyadHi</td>
<td>7</td>
</tr>
<tr>
<td>DownDyadLo</td>
<td>7</td>
</tr>
<tr>
<td>dyad</td>
<td>8</td>
</tr>
<tr>
<td>dyadlength</td>
<td>9</td>
</tr>
<tr>
<td>FWT2_PO</td>
<td>9</td>
</tr>
<tr>
<td>FWT2_PO</td>
<td>10</td>
</tr>
<tr>
<td>FWT2_TI</td>
<td>11</td>
</tr>
<tr>
<td>FWT3_PO</td>
<td>11</td>
</tr>
<tr>
<td>FWT_PO</td>
<td>12</td>
</tr>
<tr>
<td>FWT_TI</td>
<td>13</td>
</tr>
<tr>
<td>GWN</td>
<td>14</td>
</tr>
<tr>
<td>HardThresh</td>
<td>14</td>
</tr>
<tr>
<td>iconv</td>
<td>15</td>
</tr>
<tr>
<td>invblock_partition</td>
<td>16</td>
</tr>
<tr>
<td>invblock_partition2d</td>
<td>16</td>
</tr>
<tr>
<td>ITWT2_PO</td>
<td>17</td>
</tr>
<tr>
<td>IWT2_PO</td>
<td>17</td>
</tr>
<tr>
<td>IWT2_TI</td>
<td>18</td>
</tr>
<tr>
<td>IWT3_PO</td>
<td>19</td>
</tr>
<tr>
<td>IWT_PO</td>
<td>20</td>
</tr>
<tr>
<td>IWT_TI</td>
<td>20</td>
</tr>
<tr>
<td>lshift</td>
<td>21</td>
</tr>
<tr>
<td>MAD</td>
<td>22</td>
</tr>
<tr>
<td>MakeONFilter</td>
<td>22</td>
</tr>
<tr>
<td>MakeSignal</td>
<td>23</td>
</tr>
<tr>
<td>MakeSignalNewb</td>
<td>24</td>
</tr>
<tr>
<td>MirrorFilt</td>
<td>25</td>
</tr>
<tr>
<td>packet</td>
<td>25</td>
</tr>
<tr>
<td>PlotSpikes</td>
<td>26</td>
</tr>
<tr>
<td>PlotWaveCoeff</td>
<td>27</td>
</tr>
<tr>
<td>quadlength</td>
<td>27</td>
</tr>
<tr>
<td>RaphNMR</td>
<td>28</td>
</tr>
<tr>
<td>repmat</td>
<td>29</td>
</tr>
<tr>
<td>rshift</td>
<td>29</td>
</tr>
<tr>
<td>ShapeAsRow</td>
<td>30</td>
</tr>
<tr>
<td>SLphantom</td>
<td>30</td>
</tr>
<tr>
<td>SNR</td>
<td>31</td>
</tr>
<tr>
<td>SoftThresh</td>
<td>31</td>
</tr>
<tr>
<td>UpDyadHi</td>
<td>32</td>
</tr>
<tr>
<td>UpDyadLo</td>
<td>33</td>
</tr>
<tr>
<td>UpSampleN</td>
<td>33</td>
</tr>
</tbody>
</table>

**Index**

35
**aconv**

*Convolution Tool for Two-Scale Transform.*

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

`y` filtered result.

**See Also**

`iconv`, `UpDyadHi`, `UpDyadLo`, `DownDyadHi`, `DownDyadLo`.

**Examples**

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

---

**BlockThresh**

*1d Wavelet Block thresholding*

**Description**

This function is used to threshold the coefficients by group (or block).

**Usage**

```plaintext
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.

block_partition

Construct 1d block partition

Description

Construct 1d block partition

Usage

block_partition(x, L)

Arguments

x noisy wc at a given scale.

L block size.

block_partition2d

Construct 2d block partition

Description

Construct 2d block partition

Usage

block_partition2d(x, L)

Arguments

x noisy wc at a given scale.

L block size.
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.

Examples

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

cubelength

Find length and dyadic length of square array.

Description

3-D 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.
CVlinear

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.
w      1-d wavelet coefficients.

Value

CritCV Cross validation criteria.
hat_f_m_2FCV

References


**DownDyadHi**

*Hi-Pass Downsampling operator (periodized)*

**Description**

Hi-Pass Downsampling operator (periodized)

**Usage**

\[
\text{DownDyadHi}(x, \text{qmf})
\]

**Arguments**

- \(x\): 1-d signal at fine scale.
- \(\text{qmf}\): filter.

**Value**

\(y\) 1-d signal at coarse scale.

**See Also**

*DownDyadLo, UpDyadHi, UpDyadLo, FWT_PO, iconv.*

**Examples**

```r
\texttt{qmf} \leftarrow \text{MakeONFilter('Haar')}
\texttt{x} \leftarrow \text{MakeSignal('Heavisine',2^3)}
\text{DownDyadHi}(x, \text{qmf})
```

---

**DownDyadLo**

*Lo-Pass Downsampling operator (periodized)*

**Description**

Lo-Pass Downsampling operator (periodized)

**Usage**

\[
\text{DownDyadLo}(x, \text{qmf})
\]

**Arguments**

- \(x\): 1-d signal at fine scale.
- \(\text{qmf}\): filter.
Value

d 1-d signal at coarse scale.

See Also

DownDyadHi, UpDyadHi, UpDyadLo, FWT_P0, aconv.

Examples

```r
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 coeffs 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_P0 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_P0.

Usage
FTWT2_P0(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)
w <- FTWT2_PO(x, L, qmf)

FWT2_PO
2-d MRA 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)
w <- 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 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

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

Description

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

Usage

```r
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.
**FWT_TI**

*Translation Invariant Forward Wavelet Transform.*

**Examples**

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

**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**

```r
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**

```
gwnHnL sigmaI
```

**Arguments**

- `n` sample size.
- `sigma` standard deviation.

**Value**

epsilon resulting noise.

**Examples**

```
gwnH1PLPN1I
```

---

**HardThresh**

*Apply Hard Threshold.*

**Description**

Apply Hard Threshold.

**Usage**

```
hardthreshHyL tI
```

**Arguments**

- `y` Noisy Data.
- `t` Threshold.

**Value**

x filtered result (y \_\_ly\_\_|y|>t).
iconvv

**Examples**

```r
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)
```

**Description**

Filtering by periodic convolution of \( x \) with \( f \).

**Usage**

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

**Arguments**

- \( f \)  
  filter.
- \( x \)  
  1-d signal.

**Value**

\( y \) filtered result.

**See Also**

- `aconv`, `UpDyadHi`, `UpDyadLo`, `DownDyadHi`, `DownDyadLo`.

**Examples**

```r
qmf <- MakeONFilter('Haar')
x <- MakeSignal('Heavisine', 2^3)
iconvv(qmf, x)
```
**invblock_partition2d**  
*Inversion of the 2d block partition*

---

### Description

Inversion of the 1d block partition

### Usage

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

### Arguments

- `x`: noisy wc at a given scale.
- `n`: scale.
- `L`: block size.

---

### invblock_partition2d

*Inversion of the 2d block partition*

---

### Description

Inversion of the 2d block partition

### Usage

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

### Arguments

- `x`: noisy wc at a given scale.
- `n`: scale.
- `L`: block size.
**ITWT2_PO**

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

**Description**

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

**Usage**

\[
\text{ITWT2_PO}(wc, L, qmf)
\]

**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**

\texttt{FTWT2_PO}, \texttt{MakeONFilter}.

**Examples**

\[
\begin{align*}
qmf & \leftarrow \text{MakeONFilter('Daubechies', 10)} \\
L & \leftarrow 0 \\
x & \leftarrow \text{matrix(rnorm(2^2), ncol=2)} \\
wc & \leftarrow \text{FTWT2_PO}(x, L, qmf) \\
xr & \leftarrow \text{ITWT2_PO}(wc, L, qmf)
\end{align*}
\]

---

**IWT2_PO**

*Inverse 2-d MRA wavelet transform (periodized, orthogonal).*

**Description**

If \(wc\) is the result of a forward 2d wavelet transform, with \(wc \leftarrow \text{FWT2_PO}(x, L, qmf)\). then \(x \leftarrow \text{IWT2_PO}(wc, L, qmf)\) reconstructs \(x\) exactly. \(qmf\) is a nice \(qmf\), e.g. one made by \texttt{MakeONFilter}.

**Usage**

\[
\text{IWT2_PO}(wc, L, qmf)
\]
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

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

Description

Invert 2-d Translation Invariant Wavelet Transform.

Usage

```r
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

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

If \( wc \) is the result of a forward 3d wavelet transform, with \( wc \leftarrow \text{FWT3\_PO}(x, L, \text{qmf}) \), then \( x \leftarrow \text{IWT3\_PO}(wc, L, \text{qmf}) \) reconstructs \( x \) exactly. \text{qmf} is a nice filter, e.g. one made by \text{MakeONFilter}.

Usage

\[
\text{IWT3\_PO}(wc, L, \text{qmf})
\]

Arguments

- \( wc \): 3-d wavelet transform (n by n by n array, n dyadic).
- \( L \): coarse level.
- \( \text{qmf} \): 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

\text{FWT3\_PO}, \text{MakeONFilter}.

Examples

\[
\begin{align*}
\text{qmf} & \leftarrow \text{MakeONFilter}(\text{\textquotesingle Daubechies\textquotesingle}, 10) \\
L & \leftarrow 3 \\
x & \leftarrow \text{array} (\text{rnorm(32^3)}, c(32, 32, 32)) \\
w & \leftarrow \text{FWT3\_PO}(x, L, \text{qmf}) \\
xr & \leftarrow \text{IWT3\_PO}(wc, L, \text{qmf})
\end{align*}
\]
**IWT_PO**

*Inverse Wavelet Transform (periodized, orthogonal).*

**Description**

Suppose $w_c \leftarrow FWT_PO(x, L, qmf)$ where $qmf$ is an orthonormal quad. mirror filter, e.g. one made by `MakeONFilter`. Then $x$ can be reconstructed by $x \leftarrow IWT_PO(w_c, L, qmf)$.

**Usage**

$IWT_PO(w_c, L, qmf)$

**Arguments**

- $w_c$: 1-d wavelet transform: $\text{length}(w_c) = 2^J$.
- $L$: Coarsest scale ($2^{\cdot\text{-}L} = \text{scale of } V_0$); $L \ll J$.
- $qmf$: quadrature mirror filter (orthonormal).

**Value**

$x$: 1-d signal reconstructed from $w_c$.

**See Also**

`FWT_PO`, `MakeONFilter`.

**Examples**

```r
x <- MakeSignal('Ramp', 8)
L <- 0
qmf <- MakeONFilter('Haar')
w_c <- FWT_PO(x, L, qmf)
x_r <- IWT_PO(w_c, 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

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

---

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

Description

Circular left shift of 1-d signal

Usage

```r
lshift(a)
```

Arguments

- `a` 1-d signal.

Value

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

Examples

```r
x <- MakeSignal('HeaviSine', 2^3)
lshift(x)
```
**MAD**  
*Median Absolute Deviation*

**Description**

Compute the median absolute deviation.

**Usage**

```r
MAD(x)
```

**Arguments**

- `x`: 1-d signal.

**Examples**

```r
x <- c(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**

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

**Arguments**

- `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_RO, IWT_RO, FWT2_RO, IWT2_RO.

Examples

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

Arguments


d: desired signal length.

Value

sig: 1-d signal.

See Also

fwt_po, iwt_po, fwtR_po, iwtR_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


d: desired signal length.

Value

sig: 1-d signal.

See Also

fwt_po, iwt_po, fwtR_po, iwtR_po.
MirrorFilt

Examples

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

---

MirrorFilt

Apply \((-1)^t\) modulation

Description

\[ h(t) = (-1)^{(t-1)} \times x(t), 1 \leq t \leq \text{length}(x) \]

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

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

---

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)
```

---

**PlotSpike**

*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**
\[ h(t) = (-1)^{(t-1) \mod (2^L)} * x(t), 1 \leq t \leq \text{length}(x) \]

**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. Adrian 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**

Replicate and tile an array.

**Usage**

```matlab
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

---

**rshift**

*Circular right shift of 1-d signal*

---

**Description**

Circular right shift of 1-d signal

**Usage**

```matlab
rshift(a)
```

**Arguments**

- `a`: 1-d signal.

**Value**

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

**Examples**

```matlab
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**

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

---

**SLphantom**  
*3d 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

**Source**

http://tomography.o-x-t.com/2008/04/13/3d-shepp-logan-phantom/
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 (\( |y| > t \)).
Examples

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

---

**UpDyadHi**

*Hi-Pass Upsampling operator; periodized*

**Description**

Hi-Pass Upsampling operator; periodized

**Usage**

```r
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**

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

*Lo-Pass Upsampling operator; periodized*

**Description**
Lo-Pass Upsampling operator; periodized

**Usage**

\[ \text{UpDyadLo}(x, \text{qmf}) \]

**Arguments**

- \( x \): 1-d signal at coarser scale.
- \( \text{qmf} \): filter.

**Value**

\( y \): 1-d signal at finer scale.

**See Also**
- `DownDyadLo`
- `DownDyadHi`
- `UpDyadHi`
- `IWT_PO`
- `iconvv`

**Examples**

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

---

**UpSampleN**

*Upsampling operator*

**Description**

Upsampling operator

**Usage**

\[ \text{UpSampleN}(x, s) \]

**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
## Index

**Topic datasets**
- RaphNMR, 28
- SLphantom, 30

- aconv, 3, 8, 15, 32

- block_partition, 4
- block_partition2d, 4
- BlockThresh, 3

- CircularShift, 5
- cubelength, 5
- CVlinear, 6

- DownDyadHi, 3, 7, 8, 15, 25, 32, 33
- DownDyadLo, 3, 7, 7, 15, 32, 33
- dyad, 8, 9
- dyadlength, 9

- FWT2 PO, 9, 17
- FWT2 PO, 10, 18, 23, 24
- FWT TI, 11
- FWT3 PO, 11, 19
- FWT PO, 7, 8, 12, 20, 23, 24, 27
- FWT TI, 13, 21

- GWN, 14

- HardThresh, 14

- iconv, 3, 7, 15, 33
- invblock_partition, 16
- invblock_partition2d, 16
- ITWT2 PO, 9, 10, 17
- IWT2 PO, 10, 17, 23, 24
- IWT2 TI, 18
- IWT3 PO, 11, 12, 19
- IWT PO, 12, 20, 23, 24, 27, 32, 33
- IWT TI, 11, 13, 20

- lshift, 21

- MAD, 22
- MakeONFilter, 9–13, 17–21, 22
- MakeSignal, 23
- MakeSignalNewb, 24
- MirrorFilt, 25

- packet, 25
- PlotSpikes, 26, 27
- PlotWaveCoeff, 26, 27

- quadlength, 9, 27

- RaphNMR, 28
- repmat, 29
- rshift, 29

- ShapeAsRow, 30
- SLphantom, 30
- SNR, 31
- SoftThresh, 31

- UpDyadHi, 3, 7, 8, 15, 32, 33
- UpDyadLo, 3, 7, 8, 15, 32, 33
- UpsampleN, 33