## 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:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>aconv</td>
<td>3</td>
</tr>
<tr>
<td>BlockThresh</td>
<td>4</td>
</tr>
<tr>
<td>block_partition</td>
<td>5</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------</td>
<td>------</td>
</tr>
<tr>
<td>block_partition2d</td>
<td>5</td>
</tr>
<tr>
<td>CircularShift</td>
<td>6</td>
</tr>
<tr>
<td>cubelength</td>
<td>7</td>
</tr>
<tr>
<td>CVlinear</td>
<td>7</td>
</tr>
<tr>
<td>DownDyadHi</td>
<td>8</td>
</tr>
<tr>
<td>DownDyadLo</td>
<td>9</td>
</tr>
<tr>
<td>dyad</td>
<td>9</td>
</tr>
<tr>
<td>dyadlength</td>
<td>10</td>
</tr>
<tr>
<td>FWT2_PO</td>
<td>11</td>
</tr>
<tr>
<td>FWT2_PO</td>
<td>11</td>
</tr>
<tr>
<td>FWT2_TI</td>
<td>12</td>
</tr>
<tr>
<td>FWT3_PO</td>
<td>13</td>
</tr>
<tr>
<td>FWT_PO</td>
<td>14</td>
</tr>
<tr>
<td>FWT_TI</td>
<td>14</td>
</tr>
<tr>
<td>GWN</td>
<td>15</td>
</tr>
<tr>
<td>HardThresh</td>
<td>16</td>
</tr>
<tr>
<td>iconvv</td>
<td>16</td>
</tr>
<tr>
<td>invblock_partition</td>
<td>17</td>
</tr>
<tr>
<td>invblock_partition2d</td>
<td>18</td>
</tr>
<tr>
<td>ITWT2_PO</td>
<td>18</td>
</tr>
<tr>
<td>IWT2_PO</td>
<td>19</td>
</tr>
<tr>
<td>IWT2_TI</td>
<td>20</td>
</tr>
<tr>
<td>IWT3_PO</td>
<td>21</td>
</tr>
<tr>
<td>IWT_PO</td>
<td>22</td>
</tr>
<tr>
<td>IWT_TI</td>
<td>22</td>
</tr>
<tr>
<td>JSThresh</td>
<td>23</td>
</tr>
<tr>
<td>lshift</td>
<td>24</td>
</tr>
<tr>
<td>MAD</td>
<td>24</td>
</tr>
<tr>
<td>MakeONFilter</td>
<td>25</td>
</tr>
<tr>
<td>MakeSignal</td>
<td>26</td>
</tr>
<tr>
<td>MakeSignalNewb</td>
<td>27</td>
</tr>
<tr>
<td>MinMaxThresh</td>
<td>27</td>
</tr>
<tr>
<td>MirrorFilt</td>
<td>28</td>
</tr>
<tr>
<td>MultiMAD</td>
<td>29</td>
</tr>
<tr>
<td>MultiSURE</td>
<td>29</td>
</tr>
<tr>
<td>MultiVisu</td>
<td>30</td>
</tr>
<tr>
<td>packet</td>
<td>30</td>
</tr>
<tr>
<td>PlotSpikes</td>
<td>31</td>
</tr>
<tr>
<td>PlotWaveCoeff</td>
<td>31</td>
</tr>
<tr>
<td>quadlength</td>
<td>32</td>
</tr>
<tr>
<td>RaphNMR</td>
<td>33</td>
</tr>
<tr>
<td>repmat</td>
<td>33</td>
</tr>
<tr>
<td>rshift</td>
<td>34</td>
</tr>
<tr>
<td>ShapeAsRow</td>
<td>34</td>
</tr>
<tr>
<td>SLphantom</td>
<td>35</td>
</tr>
<tr>
<td>SNR</td>
<td>35</td>
</tr>
<tr>
<td>SoftThresh</td>
<td>36</td>
</tr>
<tr>
<td>SUREThresh</td>
<td>36</td>
</tr>
</tbody>
</table>
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

iconv, 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**

```r
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, invbblock_partition`.

**Examples**

```r
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**

```r
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**

```r
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**

```r
block_partition2d(x, L)
```
CircularShift

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.
cubelength

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


---

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, iconv.

Examples

```r
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**

```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.
dyadlength

**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**

\[
x <- \text{MakeSignal('Ramp', 8)}
dyadlength(x)
\]
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 Forward 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

```r
qmf <- MakeONFilter('Daubechies', 10)
L <- 3
x <- matrix(rnorm(128^2), ncol=128)
w <- FWT2_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

`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

```r
x <- matrix(rnorm(2^2), ncol=2)
L <- 0
qmf <- MakeONFilter('Haar')
TIWT <- FWT2_TI(x, L, qmf)
```
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_P0  
*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_P0`.

**Usage**

```r
FWT_P0(x, L, qmf)
```

**Arguments**

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

**Value**

`wc` 1-d wavelet transform of `x`.

**See Also**

`IWT_P0, MakeONFilter`.

**Examples**

```r
x <- MakeSignal("Ramp", 8)
L <- 0
qmf <- MakeONFilter("Haar")
wc <- FWT_P0(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**

```r
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**

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

**Arguments**

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

**Value**

epsilon: resulting noise.

**Examples**

```r
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)
invblock_partition

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)
iconv(qmf, x)
```

Inversion of the 1d block partition

Description

Inversion of the 1d block partition

Usage

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

Arguments

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

See Also

- `block_partition`, `BlockThresh`.

Examples

```r
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  \textit{Inversion of the 2d block partition}

**Description**

Inversion of the 2d block partition

**Usage**

\[
\text{invblock\_partition2d}(x, n, L)
\]

**Arguments**

- \(x\)  
  partition of coefficients by block.
- \(n\)  
  scale.
- \(L\)  
  block size.

**Value**

out coefficients.

**See Also**

- \text{block\_partition2d}

**Examples**

\[
\begin{align*}
  &n \leftarrow 2 \\
  &x \leftarrow \text{matrix}(\text{rnorm}(n^2), \text{ncol}=2) \\
  &j0 \leftarrow 0 \\
  &qmf \leftarrow \text{MakeONFilter}(\text{\'Haar\')} \\
  &wc \leftarrow \text{FWT2\_PO}(x, j0, qmf) \\
  &L \leftarrow 2 \\
  &wcb \leftarrow \text{block\_partition2d}(wc, L) \\
  &wcib \leftarrow \text{invblock\_partition2d}(wcb, n, L)
\end{align*}
\]

ITWT2\_PO  \textit{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 \text{MakeONFilter}.

**Usage**

\[
\text{ITWT2\_PO}(wc, L, qmf)
\]
**IWT2_PO**

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

**Description**

If \(wc\) is the result of a forward 2d wavelet transform, with \(wc \leftarrow FWT2_PO(x, L, qmf)\). then \(x \leftarrow IWT2_PO(wc, L, qmf)\) 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**

`FTWT2_PO`, `MakeONFilter`
Examples

```r
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)
```

---

**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 3-d wavelet transform, with \( wc \leftarrow FWT3\_PO(x, L, qmf) \). then \( x \leftarrow IWT3\_PO(wc, L, qmf) \) reconstructs \( x \) exactly. \( qmf \) is a nice qmf, e.g. one made by \texttt{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 \texttt{IWT2\_PO}, original matlab code by Vicki Yang and Brani Vidakovic.

Value

\( x \) 3-d signal reconstructed from \( wc \).

See Also

\texttt{FWT3\_PO, MakeONFilter}.

Examples

\begin{verbatim}
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)
\end{verbatim}
**IWT_PO**

*Inverse Wavelet Transform (periodized, orthogonal)*

**Description**
Suppose \( wc \leftarrow \text{FWT\_PO}(x, L, \text{qmf}) \) where \text{qmf} is an orthonormal quad. mirror filter, e.g. one made by \text{MakeONFilter}. Then \( x \) can be reconstructed by \( x \leftarrow \text{IWT\_PO}(wc, L, \text{qmf}) \).

**Usage**

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

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

**Value**
\( x \): 1-d signal reconstructed from \( wc \).

**See Also**
- \text{FWT\_PO}, \text{MakeONFilter}.

**Examples**

\[
\begin{align*}
  x &\leftarrow \text{MakeSignal}(\text{'Ramp'}, 8) \\
  L &\leftarrow 0 \\
  \text{qmf} &\leftarrow \text{MakeONFilter}(\text{'Haar'}) \\
  wc &\leftarrow \text{FWT\_PO}(x, L, \text{qmf}) \\
  xr &\leftarrow \text{IWT\_PO}(wc,L,\text{qmf})
\end{align*}
\]

**IWT_TI**

*Invert Translation Invariant Wavelet Transform*

**Description**
Invert Translation Invariant Wavelet Transform

**Usage**

\[
\text{IWT\_TI}(\text{pkt}, \text{qmf})
\]
**JSThresh**

*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)
```

---

**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

```r
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**
```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, 1, 2, 2, 4, 6, 9)
MAD(x)
```

---

**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**

- **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)

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
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


---

**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

\[
x <- \text{MakeSignal}('\text{HeaviSine}', 2^3) \\
h <- \text{MirrorFilt}(x)
\]
MultiMAD

Apply Shrinkage with level-dependent Noise level estimation

Description
Apply Shrinkage with level-dependent Noise level estimation

Usage
MultiMAD(wc, L)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wc</td>
<td>Wavelet Transform of noisy sequence.</td>
</tr>
<tr>
<td>L</td>
<td>low-resolution cutoff for Wavelet Transform.</td>
</tr>
</tbody>
</table>

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

MultiSURE

Apply Shrinkage to Wavelet Coefficients

Description
SURE refers to Stein’s Unbiased Risk Estimate.

Usage
MultiSURE(wc, L)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wc</td>
<td>Wavelet Transform of noisy sequence with N(0,1) noise.</td>
</tr>
<tr>
<td>L</td>
<td>low-frequency cutoff for Wavelet Transform.</td>
</tr>
</tbody>
</table>

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**

```matlab
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**

```matlab
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**

```matlab
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
```r
## 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.

\( \text{scal} \)  
scaling factor (0 == autoscale).

Value

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

See Also

FWT_P0, IWT_P0, PlotSpikes.

Examples

\[
\begin{align*}
\text{x} & \leftarrow \text{MakeSignal('Ramp', 128)} \\
\text{qmf} & \leftarrow \text{MakeONFilter('Daubechies', 10)} \\
\text{L} & \leftarrow 3 \\
\text{scal} & \leftarrow 1 \\
\text{wc} & \leftarrow \text{FWT_P0(x, L, qmf)} \\
\text{PlotWaveCoeff(wc, L, scal)}
\end{align*}
\]

\[
\text{quadlength} \quad \text{Find length and dyadic length of square matrix}
\]

Description

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

Usage

\[
\text{quadlength(x)}
\]

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

\[
\text{quadlength(matrix(1:16,ncol=4))}
\]
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

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**
```r
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** 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**
```r
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 ($y \mid |y| > t$).

**See Also**

- `HardThresh`

**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)
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)`
UpDyadHi

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**

```r
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, iconv

**Examples**

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

---

**UpSampleN**  
*Upsampling operator*

**Description**

Upsampling operator

**Usage**

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

**Arguments**

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

**Value**

\[ y \text{ 1-d signal, of length } s \times n \text{ with zeros interpolating alternate samples } y(s \times i - 1) = x(i), i = 1, \ldots, n \]

---

**Description**

SURE refers to Stein’s Unbiased Risk Estimate.

**Usage**

\[ \text{ValSUREThresh}(x) \]

**Arguments**

- \( x \) Noisy Data with Std. Deviation = 1.

**Value**

- \( \text{thresh} \) Value of Threshold.

---

**VisuThresh**

*Visually calibrated Adaptive Smoothing*

**Description**

Visually calibrated Adaptive Smoothing

**Usage**

\[ \text{VisuThresh}(y, \text{thresh} = "soft") \]

**Arguments**

- \( y \) Signal upon which to perform visually calibrated Adaptive Smoothing.
- \( \text{thresh} \) 'hard' or 'soft'.

**Value**

- \( x \) result of applying VisuThresh.

**References**

Index

* datasets
  - RaphNMR, 33
  - SLphantom, 35

- aconv, 3, 9, 17, 37
- block_partition, 5, 17
- block_partition2d, 5, 18
- BlockThresh, 4, 5, 17
- CircularShift, 6
- cubelength, 7
- CVlinear, 7

- DownDyadHi, 3, 8, 9, 17, 28, 37, 38
- DownDyadLo, 3, 8, 9, 17, 37, 38
- dyad, 9, 10
- dyadlength, 10
- FWT2_PO, 11, 19
- FWT2_PO, 11, 19, 26, 27
- FWT2_TI, 6, 12
- FWT3_PO, 7, 13, 21
- FWT_PO, 8, 9, 14, 22, 26, 27, 32
- FWT_TI, 14, 23
- GWN, 15

- HardThresh, 16, 23, 36
- iconv, 3, 8, 16, 38
- invblock_partition, 4, 5, 17
- invblock_partition2d, 6, 18
- ITWT2_PO, 11, 18
- IWT2_PO, 11, 12, 19, 26, 27
- IWT2_TI, 6, 20
- IWT3_PO, 7, 13, 21
- IWT_PO, 14, 22, 26, 27, 32, 37, 38
- IWT_TI, 12, 14, 15, 22
- JSThresh, 23

- lshift, 24
- MAD, 24
- MakeONFilter, 11–15, 18, 19, 21–23, 25
- MakeSignal, 26
- MakeSignalNewb, 27
- MinMaxThresh, 27
- MirrorFilt, 28
- MultiMAD, 29
- MultiSURE, 29
- MultiVisu, 30
- packet, 30
- PlotSpikes, 31, 32
- PlotWaveCoeff, 31, 31
- quadlength, 10, 32
- RaphNMR, 33
- repmat, 33
- rshift, 34
- ShapeAsRow, 34
- SLphantom, 35
- SNR, 35
- SoftThresh, 16, 23, 36
- SUREThresh, 36
- UpDyadHi, 3, 8, 9, 17, 37, 38
- UpDyadLo, 3, 8, 9, 17, 37, 38
- UpSampleN, 38
- ValSUREThresh, 39
- VisuThresh, 39