

Package ‘dwt’

October 5, 2011

Version 0.7

Date 2011-10-05

Title R routines for carrying out the Discrete Wavelet Transforms (DWT) and Maximum Overlap Discrete Wavelet Transforms (MODWT) for univariate time series.

Author Peter F. Craigmile

Maintainer Peter F. Craigmile <pfc@stat.osu.edu>

Depends R (>= 2.8)

Description DWT R code

License file LICENSE

URL <http://www.r-project.org>, <http://www.stat.osu.edu/~pfc/>

R topics documented:

center.of.energy	2
circularly.filter	2
common.scaling.filters	3
downsample.by.two	3
dwt	4
dwt.all.filters	4
dwt.as.vector	5
dwt.cascade.next	5
dwt.cascade.previous	6
dwt.detail	6
dwt.filter	7
dwt.filter.aligning	7
dwt.freqs	8
dwt.map.vector	8
dwt.matrix	9
dwt.matrix.elems	9
dwt.matrix.row.subset	10
dwt.next.filters	10
dwt.scaling.to.wavelet	11
dwt.smooth	11
dwt.sqgain	11

dwt.times	12
dwt.wavelet.to.scaling	12
dwt.wavelets.zero	13
idwt	13
imodwt	13
modwt	14
modwt.cascade.next	14
modwt.cascade.previous	15
modwt.detail	15
modwt.freqs	16
modwt.smooth	16
modwt.times	17
modwt.wavelets.zero	17
periodize.filter	18
plot.dwt	18
plot.modwt	18
rotate.vector	19
stackplot	19
upsample.by	20
upsample.by.two	20

Index	21
--------------	-----------

<code>center.of.energy</code>	<i>Calculate the center of energy of a filter</i>
-------------------------------	---

Description

Calculate the center of energy of the filter 'f'

References: Wickerhauser (1994, p.171 and p.341) Percival and Walden (2000, p.118)

Usage

```
center.of.energy(f)
```

Arguments

f

<code>circularly.filter</code>	<i>Circularly filter some data given a supplied filter</i>
--------------------------------	--

Description

Circularly filter the sequence 'x' using the filter 'f'.

Usage

```
circularly.filter(x, f, L = length(f))
```

Arguments

x

f

L

common.scaling.filters

A definition of some commonly used scaling filters

Usage

```
data(common.scaling.filters)
```

Format

The format is:

downsample.by.two *Downsample a vector by two, leaving only the even or odd indexes*

Description

Return the even indexes of 'x' (if 'even' is TRUE) or the odd indexes of 'x' (if 'even' is FALSE)

Usage

```
downsample.by.two(x, even = TRUE)
```

Arguments

x

even

dwt

Calculate the discrete wavelet transform (DWT)

Description

Perform a DWT decomposition of the data 'x' to level 'nlevels' using a given 'wavelet'.

Assumes : Can compose to 'nlevels' levels; i.e., length of 'x' is divisible by $2^{nlevels}$.

Usage

```
dwt(x, wavelet = "LA8", nlevels = floor(log2(length(x))), use.C = TRUE)
```

Arguments

x

wavelet

nlevels

use.C

dwt.all.filters

Calculate all the wavelet and scaling filters up to a given level

Description

Using the dwt.filter object 'filter', calculate the all the wavelet and scaling filters up to level 'J'.

Assumes : 'J' is an integer > 0.

Usage

```
dwt.all.filters(filter, J)
```

Arguments

filter

J

dwt.as.vector	<i>Combine the list of wavelet and scaling coefficients into a single vector</i>
---------------	--

Description

Take the wavelet and scaling coefficients in dwt object 'dx', and put them in a vector ordered as:
 $(W_1, W_2, \dots, W_{nlevels}, V_{nlevels})^T$

Usage

```
dwt.as.vector(dx)
```

Arguments

dx

dwt.cascade.next	<i>Calculate the next level of DWT scaling and wavelet coefficients given the current scaling coefficients.</i>
------------------	---

Description

Given the scaling coefficients 'V' on level j-1, calculate the scaling '\$V\$' and wavelet coefficients '\$W\$' on the level j using the dwt filter object 'dwt.filter.obj'.

Usage

```
dwt.cascade.next(V, dwt.filter.obj, use.C = TRUE)
```

Arguments

V

dwt.filter.obj

use.C

```
dwt.cascade.previous
```

Calculate the previous level of DWT scaling coefficients given the current scaling and wavelet coefficients.

Description

Given the scaling coefficients 'V' and wavelet coefficients 'W' on scale j+1, calculate the level j scaling coefficients using the dwt filter object 'dwt.filter.obj'.

Usage

```
dwt.cascade.previous(V, W, dwt.filter.obj, use.C = TRUE)
```

Arguments

V

W

dwt.filter.obj

use.C

```
dwt.detail
```

Calculate the DWT detail coefficients at a given level

Description

Calculates the level 'j' detail coefficients of the DWT object 'dx'.

Assumes: $1 \leq j \leq dx\$nlevels$.

Usage

```
dwt.detail(dx, j, use.C = TRUE)
```

Arguments

dx

j

use.C

dwt.filter	Create a DWT filter object given a filter name or set of scaling coefficients
------------	---

Description

If the argument 'scaling' is missing, creates a 'dwt.filter' object containing the scaling and wavelet filters of a given name 'name', if it exists in the list of 'common.scaling.filters'. If the vector defining the scaling filter, 'scaling', is provided then define a 'dwt.filter' using that vector instead.

Usage

```
dwt.filter(name, scaling)
```

Arguments

name
scaling

dwt.filter.aligning	Calculate the aligning value at a given DWT wavelet or scaling level, given a DWT filter object
---------------------	---

Description

calculate the aligning factor for a level 'j' wavelet (is.wavelet==TRUE) or scaling (is.wavelet==FALSE) coefficient for a specific wavelet 'filter'.

Usage

```
dwt.filter.aligning(filter, j, is.wavelet = TRUE)
```

Arguments

filter
j
is.wavelet

dwt.freqs	<i>Calculate the approximate bandpass frequencies for a DWT transform at a specified set of levels</i>
-----------	--

Description

Returns the approximate bandpass frequencies for the level 'js' wavelet coefficients of the dwt class 'dx'.

Usage

```
dwt.freqs(dx, js)
```

Arguments

dx

js

dwt.map.vector	<i>The inverse of dwt.as.vector</i>
----------------	-------------------------------------

Description

Takes a vector of the form

$(W_1, W_2, \dots, W_{nlevels}, V_{nlevels})^T$

and places these wavelet and scaling coefficients back in the dwt object 'dx'.

Usage

```
dwt.map.vector(dx, w)
```

Arguments

dx

w

dwt.matrix	<i>Calculate the DWT matrix</i>
------------	---------------------------------

Description

Calculates the $dx \times dx$ orthonormal dwt matrix using dwt class 'dx'.

ADVICE: This matrix is sparse – do not use for computer intensive calculations.

Usage

```
dwt.matrix(dx, use.C = TRUE)
```

Arguments

dx

use.C

dwt.matrix.elems	<i>Calculate the unique DWT matrix rows (modulo circular shifts) - need to explain this better</i>
------------------	--

Description

Calculate a subset of wavelet transform matrix for the dwt object 'dx'. Only include the rows corresponding to the first wavelet coefficient per level, and the first level dx levels scaling coefficient.

Usage

```
dwt.matrix.elems(dx, use.C = TRUE)
```

Arguments

dx

use.C

```
dwt.matrix.row.subset
```

Calculate a subset of the rows of the DWT matrix

Description

Calculates the $\text{length}(\text{rows}) \times dx$ orthonormal dwt matrix using dwt class 'dx' at a subset of rows as given by 'rows'.

ADVICE: This matrix is sparse – do not use for computer intensive calculations.

Usage

```
dwt.matrix.row.subset(dx, rows, use.C = TRUE)
```

Arguments

dx

rows

use.C

```
dwt.next.filters
```

Calculate the next set of DWT wavelet and scaling filters, given the previous set

Description

Given the wavelet and scaling filters at level 'j-1' 'prev.filters' calculate the level 'j' filters using the dwt.filter object 'filter'.

Assumes : 'j' is an integer > 1.

Usage

```
dwt.next.filters(prev.filters, filter, j)
```

Arguments

prev.filters

filter

j

dwt.scaling.to.wavelet

Convert a scaling filter to a wavelet filter

Description

Convert the scaling filter 'scaling' to a wavelet filter. Assumes : 'scaling' is a vector of even length.

Usage

```
dwt.scaling.to.wavelet(scaling)
```

Arguments

scaling

dwt.smooth

Calculate the DWT smooth coefficients at a given level

Description

Calculates the level 'j' smooth coefficients of the DWT object 'dx'.

Assumes: $1 \leq j \leq dx\$nlevels$.

Usage

```
dwt.smooth(dx, j, use.C = TRUE)
```

Arguments

dx

j

use.C

dwt.sqgain

Calculate the square gain function for a wavelet or scaling filter at a given level, at the specified frequencies

Description

Calculates the square gain function for the level 'level' wavelet (if wavelet=T) or scaling (if wavelet=F) filter at frequencies 'freqs', using the dwt filter object, 'filter'.

Note : This is a brute force method – does not use the FFT of the filter since we can potentially evaluate on any set of frequencies.

Usage

```
dwt.sqgain(freqs, filter, level, wavelet = TRUE)
```

Arguments

```
freqs
filter
level
wavelet
```

dwt.times	<i>Calculate the time points for a given DWT wavelet level or scaling level</i>
-----------	---

Description

Returns a vector of the times associated with the scaling (is.wavelet=FALSE) or wavelet (is.wavelet=TRUE) coefficients for level 'j' of the DWT object 'dx'.

Usage

```
dwt.times(dx, j, is.wavelet = TRUE)
```

Arguments

```
dx
j
is.wavelet
```

dwt.wavelet.to.scaling	<i>Convert a wavelet filter to a scaling filter</i>
------------------------	---

Description

Creates a scaling filter from a given 'wavelet' filter
Assumes : 'wavelet' is a vector of even length.

Usage

```
dwt.wavelet.to.scaling(wavelet)
```

Arguments

```
wavelet
```

`dwt.wavelets.zero` *Zero out the DWT wavelet coefficients on a given set of levels*

Description

Make the wavelet coefficients on level 'levels' of the dwt class 'dx' zero. If ignore.boundary is TRUE, do not zero out the boundary coefficients

Usage

```
dwt.wavelets.zero(dx, levels, ignore.boundary = FALSE)
```

Arguments

```
dx
levels
ignore.boundary
```

`idwt` *Calculate the inverse discrete wavelet transform (DWT)*

Description

Performs an inverse DWT of the dwt class 'dx'.

Usage

```
idwt(dx, use.C = TRUE)
```

Arguments

```
dx
use.C
```

`imodwt` *Calculate the inverse maximum overlap discrete wavelet transform (MODWT)*

Description

Performs an inverse MODWT of the modwt class 'dx'.

Usage

```
imodwt(dx, use.C = TRUE)
```

Arguments

```
dx
use.C
```

modwt

Calculate the maximum overlap discrete wavelet transform (MODWT)

Description

Perform a MODWT decomposition of the data 'x' to level 'nlevels' using a given 'wavelet'.

Usage

```
modwt(x, wavelet = "LA8", nlevels = floor(log2(length(x))), use.C = TRUE)
```

Arguments

```
x
wavelet
nlevels
use.C
```

modwt.cascade.next

Calculate the next level of MODWT scaling and wavelet coefficients given the current scaling coefficients.

Description

Given the MODWT scaling coefficients '\$V\$' on level '\$j\$'-1, calculate the MODWT scaling '\$V\$' and wavelet coefficients '\$W\$' on the level '\$j\$' using the dwt filter object 'dwt.filter.obj'.

Usage

```
modwt.cascade.next(V, j, dwt.filter.obj, use.C = TRUE)
```

Arguments

```
V
j
dwt.filter.obj

use.C
```

```
modwt.cascade.previous
```

Calculate the previous level of MODWT scaling coefficients given the current scaling and wavelet coefficients.

Description

Given the scaling coefficients 'V' and wavelet coefficients 'W' on scale j+1, calculate the level j scaling coefficients using the modwt filter object 'dwt.filter.obj'.

Usage

```
modwt.cascade.previous(V, W, j, dwt.filter.obj, use.C)
```

Arguments

```
V
W
j
dwt.filter.obj

use.C
```

```
modwt.detail
```

Calculate the MODWT detail coefficients at a given level

Description

Calculates the level 'j' detail coefficients of the MODWT object 'dx'.

Assumes: $1 \leq j \leq dx\$nlevels$.

Usage

```
modwt.detail(dx, j, use.C = TRUE)
```

Arguments

```
dx
j
use.C
```

<code>modwt.freqs</code>	<i>Calculate the approximate bandpass frequencies for a MODWT transform at a specified set of levels</i>
--------------------------	--

Description

Returns the approximate bandpass frequencies for the level 'js' wavelet coefficients of the modwt class 'dx'.

Usage

```
modwt.freqs(dx, js)
```

Arguments

`dx`

`js`

<code>modwt.smooth</code>	<i>Calculate the MODWT smooth coefficients at a given level</i>
---------------------------	---

Description

Calculates the level 'j' smooth coefficients of the MODWT object 'dx'.

Assumes: $1 \leq j \leq dx\$nlevels$.

Usage

```
modwt.smooth(dx, j, use.C = TRUE)
```

Arguments

`dx`

`j`

`use.C`

modwt.times	<i>Calculate the time points for a given MODWT wavelet level or scaling level</i>
-------------	---

Description

Returns a vector of the times associated with the scaling (is.wavelet=F) or wavelet (is.wavelet=T) coefficients for level 'j' of the MODWT object 'dx'.

Usage

```
modwt.times(dx, j, is.wavelet = TRUE)
```

Arguments

dx

j

is.wavelet

modwt.wavelets.zero	<i>Zero out the MODWT wavelet coefficients on a given set of levels</i>
---------------------	---

Description

Make the wavelet coefficients on level 'levels' of the modwt class 'dx' zero.

Usage

```
modwt.wavelets.zero(dx, levels)
```

Arguments

dx

levels

<code>periodize.filter</code>	<i>Periodize a filter to a specified length</i>
-------------------------------	---

Description

periodize the filter 'f' of length 'L' into a vector of length 'n'.

Usage

```
periodize.filter(f, n, L = length(filter))
```

Arguments

f
n
L

<code>plot.dwt</code>	<i>Display the DWT decomposition graphically</i>
-----------------------	--

Usage

```
plot.dwt(x, xlab = "time", col = "black", bg = "white", bdc col = "black", ...)
```

Arguments

x
xlab
col
bg
bdc col
...

<code>plot.modwt</code>	<i>Display the MODWT decomposition graphically</i>
-------------------------	--

Usage

```
plot.modwt(x, xlab = "time", col = "black", bg = "white", bdc ol = "blue", ...)
```

Arguments

x
xlab
col
bg
bdc ol
...

rotate.vector	<i>Rotate a vector by a given number of shifts to the right</i>
---------------	---

Description

Rotate the vector 'x' 'delta' units to the right.

Usage

```
rotate.vector(x, delta)
```

Arguments

x
delta

stackplot	<i>Produce a stackplot</i>
-----------	----------------------------

Usage

```
stackplot(xs, ys, ts, xlabels, types, vlines = list(),  
          lowers = sapply(ys, min), uppers = sapply(ys, max),  
          xlab = "time", col = "black", bg = "white", bdc col = "blue", ...)
```

Arguments

xs
ys
ts
xlabels
types
vlines
lowers
uppers
xlab
col
bg
bdc col
...

upsample.by	<i>Upsample a vector by infilling with zeros</i>
-------------	--

Description

Interleave 'n' zeroes between the values of 'x'. If before is TRUE, each value of 'x' occurs before the zeroes.

Usage

```
upsample.by(x, n, before = TRUE)
```

Arguments

x
n
before

upsample.by.two	<i>upsample a vector by adding zero after or below each element</i>
-----------------	---

Description

Interleave one zero between the values of 'x'. If before is TRUE, each value of 'x' occurs before the zero.

Usage

```
upsample.by.two(x, before = FALSE)
```

Arguments

x
before

Index

*Topic **datasets**

- `common.scaling.filters`, 3
- `center.of.energy`, 2
- `circularly.filter`, 2
- `common.scaling.filters`, 3
-
- `downsample.by.two`, 3
- `dwt`, 4
 - `dwt.all.filters`, 4
 - `dwt.as.vector`, 5
 - `dwt.cascade.next`, 5
 - `dwt.cascade.previous`, 6
 - `dwt.detail`, 6
 - `dwt.filter`, 7
 - `dwt.filter.aligning`, 7
 - `dwt.freqs`, 8
 - `dwt.map.vector`, 8
 - `dwt.matrix`, 9
 - `dwt.matrix.elems`, 9
 - `dwt.matrix.row.subset`, 10
 - `dwt.next.filters`, 10
 - `dwt.scaling.to.wavelet`, 11
 - `dwt.smooth`, 11
 - `dwt.sqgain`, 11
 - `dwt.times`, 12
 - `dwt.wavelet.to.scaling`, 12
 - `dwt.wavelets.zero`, 13
-
- `idwt`, 13
- `imodwt`, 13
-
- `modwt`, 14
 - `modwt.cascade.next`, 14
 - `modwt.cascade.previous`, 15
 - `modwt.detail`, 15
 - `modwt.freqs`, 16
 - `modwt.smooth`, 16
 - `modwt.times`, 17
 - `modwt.wavelets.zero`, 17
-
- `periodize.filter`, 18
- `plot.dwt`, 18
- `plot.modwt`, 18
-
- `rotate.vector`, 19
-
- `stackplot`, 19
-
- `upsample.by`, 20
- `upsample.by.two`, 20