Package ‘wavelets’

February 17, 2020

Version 0.3-0.2

Date 2020-02-16

Title Functions for Computing Wavelet Filters, Wavelet Transforms and Multiresolution Analyses

Author Eric Aldrich <`ealdrich@gmail.com`>

Maintainer Eric Aldrich <`ealdrich@gmail.com`>

Depends R (>= 2.10), methods

Description Contains functions for computing and plotting discrete wavelet transforms (DWT) and maximal overlap discrete wavelet transforms (MODWT), as well as their inverses. Additionally, it contains functionality for computing and plotting wavelet transform filters that are used in the above decompositions as well as multiresolution analyses.

LazyLoad yes

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-02-17 19:40:02 UTC

R topics documented:

`align` *Align Wavelet Transform Coefficients*

Description

Aligns both wavelet (high pass) and scaling (low pass) coefficients of objects of class `dwt`, `modwt`, `dwpt`, and `modwpt` using phase shift values computed by `wt.filter.shift`.

Usage

```
align(wt, coe=FALSE, inverse=FALSE)
```

Arguments

`wt`	An object of class `dwt`, `modwt`, `dwpt`, or `modwpt`.
`coe`	Logical value indicating whether to use center of energy method in computing phase shifts.
`inverse`	Logical value indicating whether to shift wavelet and scaling coefficients of an aligned object back to their original positions.

Details

An object of class `dwt`, `modwt`, is characterized as ’unaligned’ if the value in the `aligned` slot is `FALSE`. Similarly, these objects are classified as ’aligned’ if the value in the `aligned` slot is `TRUE`. Thus, `align` will only operate on an ’unaligned’ wavelet transform object if `inverse = FALSE` and on an ’aligned’ wavelet transform object if `inverse = TRUE`.

The argument `coe` is passed to the `wt.filter.shift` function to determine what method to use for computing phase shifts (see documentation for `wt.filter.shift`).

Value

Either an aligned or unaligned object of the same class as `wt` (see Details above).

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`wt.filter.shift`, `dwt`, `modwt`.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)

# combine them and compute DWT
newX <- cbind(X1,X2)
wt <- dwt(newX, n.levels=3, boundary="reflection", fast=FALSE)

# align
wt.aligned <- align(wt)
```

`dwt` *Discrete Wavelet Transform*

Description

Computes the discrete wavelet transform coefficients for a univariate or multivariate time series.

Usage

```
dwt(X, filter="la8", n.levels, boundary="periodic", fast=TRUE)
```

Arguments

`X`	A univariate or multivariate time series. Numeric vectors, matrices and data frames are also accepted.
`filter`	Either a `wt.filter` object, a character string indicating which wavelet filter to use in the decomposition, or a numeric vector of wavelet coefficients (not scaling coefficients). See `help(wt.filter)` for acceptable filter names.

`n.levels`	An integer specifying the level of the decomposition. By default this is the value J such that the length of X is at least as great as the length of the level J wavelet filter, but less than the length of the level $J+1$ wavelet filter. Thus, $J \leq \log\left(\frac{N-1}{L-1}+1\right)$, where N is the length of X.
`boundary`	A character string indicating which boundary method to use. `boundary = "periodic"` and `boundary = "reflection"` are the only supported methods at this time.
`fast`	A logical flag which, if true, indicates that the pyramid algorithm is computed with an internal C function. Otherwise, only R code is used in all computations.

Details

The discrete wavelet transform is computed via the pyramid algorithm, using pseudocode written by Percival and Walden (2000), pp. 100-101. When `boundary="periodic"` the resulting wavelet and scaling coefficients are computed without making changes to the original series - the pyramid algorithm treats `X` as if it is circular. However, when `boundary="reflection"` a call is made to `extend.series`, resulting in a new series which is reflected to twice the length of the original series. The wavelet and scaling coefficients are then computed by using a periodic boundary condition on the reflected sereis, resulting in twice as many wavelet and scaling coefficients at each level.

Value

Returns an object of class `dwt`, which is an S4 object with slots

`W`	A list with element i comprised of a matrix containing the ith level wavelet coefficients.
`V`	A list with element i comprised of a matrix containing the ith level scaling coefficients.
`filter`	A `wt.filter` object containing information for the filter used in the decomposition. See `help(wt.filter)` for details.
`level`	An integer value representing the level of wavelet decomposition.
`n.boundary`	A numeric vector indicating the number of boundary coefficients at each level of the decomposition.
`boundary`	A character string indicating the boundary method used in the decomposition. Valid values are "periodic" or "reflection".
`series`	The original time series, `X`, in matrix format.
`class.X`	A character string indicating the class of the input series. Possible values are `"ts"`, `"mts"`, `"numeric"`, `"matrix"`, or `"data.frame"`.
`attr.X`	A list containing the attributes information of the original time series, `X`. This is useful if `X` is an object of class `ts` or `mts` and it is desired to retain relevant time information. If the original time series, `X`, is a matrix or has no attributes, then `attr.X` is an empty list.
`aligned`	A logical value indicating whether the wavelet and scaling coefficients have been phase shifted so as to be aligned with relevant time information from the original series. The value of this slot is initially FALSE and can only be changed to TRUE via the `align` function, with the `dwt` object as input.

`coe` A logical value indicating whether the center of energy method was used in phase alignement of the wavelet and scaling coefficients. By default, this value is FALSE (and will always be FALSE when `aligned` is FALSE) and will be set to true if the `dwt` object is phase shifted via the `align` function and center of energy method.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`modwt`, `wt.filter`.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)

# combine them and compute DWT
newX <- cbind(X1,X2)
wt <- dwt(newX, n.levels=3, boundary="reflection", fast=FALSE)
```

`dwt-class` *Discrete Wavelet Transform Object*

Description

An S4 object containing discrete wavelet transform information.

Slots

W A list with element i comprised of a matrix containing the ith level wavelet coefficients.

V A list with element i comprised of a matrix containing the ith level scaling coefficients.

filter A `wt.filter` object containing information for the filter used in the decomposition. See `help(wt.filter)` for details.

level An integer value representing the level of wavelet decomposition.

n.boundary A numeric vector indicating the number of boundary coefficients at each level of the decomposition.

boundary A character string indicating the boundary method used in the decomposition. Valid values are "periodic" or "reflection".

series The original time series, `X`, in matrix format.

class.X A character string indicating the class of the input series. Possible values are `"ts"`, `"mts"`, `"numeric"`, `"matrix"`, or `"data.frame"`.

attr.X A list containing the attributes information of the original time series, `X`. This is useful if `X` is an object of class `ts` or `mts` and it is desired to retain relevant time information. If the original time series, `X`, is a matrix or has no attributes, then `attr.X` is an empty list.

aligned A logical value indicating whether the wavelet and scaling coefficients have been phase shifted so as to be aligned with relevant time information from the original series. The value of this slot is initially FALSE and can only be changed to TRUE via the `align` function, with the `dwt` object as input.

coe A logical value indicating whether the center of energy method was used in phase alignement of the wavelet and scaling coefficients. By default, this value is FALSE (and will always be FALSE when `aligned` is FALSE) and will be set to true if the `dwt` object is phase shifted via the `align` function and center of energy method.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`dwt`, `modwt`, `modwt-class`, `wt.filter`.

`dwt.forward` *Discrete Wavelet Transform and Maximal Overlap Discrete Wavelet Tranform Forward and Backward Pyramid Algorithm*

Description

Implementation of DWT and MODWT forward and backward pyramid algorithms.

Usage

```
dwt.forward(V, filter)
dwt.backward(W, V, filter)
modwt.forward(V, filter, j)
modwt.backward(W, V, filter, j)
```

Arguments

`W`	A vector of wavelet coefficients.
`V`	A vector of scaling coefficients.
`filter`	A `wt.filter` object.
`j`	The level of wavelet and scaling coefficients to compute (for forward algorithm) or the level of wavelet and scaling coefficient inputs (for inverse algorithm).

Details

An implementation of the DWT and MODWT forward and backward pyramid algorithms using pseudocode written by Percival and Walden (2000), pp. 100-101, 177-178. These functions are intended primarily as helper functions for the `dwt`, `modwt`, `idwt` and `imodwt` functions.

Value

`dwt.forward` and `modwt.forward` return a list of two elements containing vectors of wavelet and scaling coefficients for the subsequent level of analysis. `dwt.backward` and `modwt.backward` return a vector of scaling coefficients for the previous level of analysis.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`dwt`, `modwt`, `wt.filter`.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)

# compute the LA8 wavelet filter for both DWT and MODWT
la8.dwt <- wt.filter()
la8.modwt <- wt.filter(modwt=TRUE)

# compute the DWT and MODWT level one wavelet and scaling coefficients
wt.dwt <- dwt.forward(X1, la8.dwt)
wt.modwt <- modwt.forward(X2, la8.modwt, 1)

# compute the original series with the level one coefficients
newX.dwt <- dwt.backward(wt.dwt$W, wt.dwt$V, la8.dwt)
newX.modwt <- modwt.backward(wt.modwt$W, wt.modwt$V, la8.modwt, 1)
```

`ecg` *Electrocardiogram Data*

Description

Electrocardiogram measurements for an individual with arhythmia. Observations were made over an 11.37 second interval at a rate of 180 samples per second. For more detailed information regarding the collection of this data, see Percival and Walden (2000), p. 125.

Usage

```
data(ecg)
```

Format

A time series object containing 2048 observations.

Source

http://www.ms.washington.edu/~s530/data.html

References

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press, sec. 4.10.

`extend.series` *Extend a Time Series*

Description

Extends a univariate or multivariate time series beyond its original length.

Usage

```
extend.series(X, method="reflection", length="double", n, j)
```

Arguments

`X`	A univariate or multivariate time series. Numeric vectors, matrices and data frames are accepted.
`method`	A character string indicating which extension method to use. Possible values are `"periodic"`, `"reflection"`, `"zero"`, `"mean"`, and `"reflection.inverse"`.
`length`	A character string indicating how to determine the length of the extended series. Possible values are `"aribitrary"`, `"powerof2"`, and `"double"`.

`n`	An integer value specifying the length of the extended series. This argument is only relevant when `length = "arbitrary"`.
`j`	An integer value specifying the power of two of which the length of the extended series should be a multiple. This argument is only relevant for `length = "powerof2"`.

Details

The original time series, `X`, is extended to a new length as determined by `length`: if `length = "arbitrary"`, a value for the argument `n` must be specified in order to extend the series to length `n`; if `length = "powerof2"` the series will be extended to the nearst multiple of 2^j, where `j` must be specified as an argument to the function; if `length = "double"` the series will be extended to twice its original length.

Once the length of the extended series is determined, the actual extension is performed using one of five methods: if `method = "periodic"` a periodic extension is made by concatenating the series with itself (without changing the order of the values); if `method = "reflection"` an extension is made by concatenating the series with a reflection (reverse ordering) of itself; if `method = "zero"` the series is padded with zeros; if `method = "mean"` the series is padded with its mean; if `method = "reflection.inverse"` an extension is made by concatenating the series with a reflection (reverse ordering) of itself that is further reflected of the horizontal axis $y = x_N$ where x_N is the last value of the observed series.

In the case that the series is extended beyond twice it's original length, the `"periodic"` and `"reflection"` methods are repetetively applied so that every subseries of length N (the length of the original series) is either a periodic or reflected extension of any other N adjacent values. Similarly, the `"reflection.inverse"` method is repetetively applied so that every $2N$ values are a periodic extension of any adjacent $2N$ values.

Value

The extended time series, in its original class.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`dwt`, `modwt`.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)
```

```
# combine them and extend using reflection method
newX <- cbind(X1,X2)
newX.ext1 <- extend.series(newX, length="arbitrary", n=64)
plot.ts(newX.ext1)

# apply another extension method
newX.ext2 <- extend.series(newX, method="reflection.inverse",
                           length="powerof2", j=6)
plot.ts(newX.ext2)
```

`figure108.wt.filter` *Plot Multiple DWT Wavelet or Scaling Filters*

Description

Plots multiple DWT Wavelet or Scaling Filters similar to Figure 108 in *Wavelet Methods for Time Series Analysis* by Percival and Walden (2000).

Usage

```
figure108.wt.filter(filter.objects, level = 1, l = NULL, wavelet = TRUE)
```

Arguments

`filter.objects`	List containing 'wt.filter' objects, character strings specifying a wavelet filter, or numeric vectors of wavelet coefficients. The list can contain a combination of 'wt.filter' objects, character strings, and numeric vectors. If only one filter is to be plotted, a single 'wt.filter' object, character string, or numeric vector may be supplied. See 'help(wt.filter)' for acceptable filter names.
`level`	If `filter.object` is only a single 'wt.filter' object, character string, or numeric vector, then the level of the filter can be specified in `level`. Defaulted to 1.
`l`	Single integer representing the right hand limit of the horizontal axis. If unspecified, it will default to the length of the filter of greatest length given in `filter.object`. Cannot be less than the length of the filter of greatest length.
`wavelet`	A logical flag indicating whether to plot the wavelet (high pass) or scaling (low pass) filter.

Details

The plotting space available for each filter is dictated by the value of greatest magnitude of all the filters plotted. The vertical plotting space for each level will then be 2 times the absolute value of this magnitude.

The filters are successively plotted in the order given in `filter.object`, where the first filter in `filter.object` is drawn at the top of the plot region, and the successive filters are plotted below.

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`wt.filter`

Examples

```
# Plotting the LA8 Wavelet Filter
filter <- wt.filter()
figure108.wt.filter(filter)

# Alternatively
figure108.wt.filter("la8")

# Plotting the Haar, D4, D6 Wavelet Filters
figure108.wt.filter(list("haar", "d4", "d6"))

# Plotting the Haar, D4, D6 Scaling Filters
figure108.wt.filter(list("haar", "d4", "d6"), wavelet = FALSE)

# Alternatively
haar <- wt.filter("haar")
d6 <- wt.filter("d6")
figure108.wt.filter(list(haar, "d4", d6), wavelet = FALSE)

# Adding an "made up" filter (represented by c(1,-1,1,-1)
figure108.wt.filter(list(haar, "d4", d6, c(1,-1,1,-1)), wavelet = FALSE)
```

`figure98.wt.filter` *Plot a DWT Wavelet or Scaling Filter for Specific Levels.*

Description

Plots a DWT Wavelet or Scaling Filter for specific levels similar to Figure 98 in *Wavelet Methods for Time Series Analysis* by Percival and Walden (2000).

Usage

```
figure98.wt.filter(filter, levels = NULL, wavelet = TRUE, y.normalize =
TRUE)
```

Arguments

`filter`	Either a 'wt.filter' object, a character string specifying a wavelet filter, or a numeric vector of wavelet coefficients. See 'help(wt.filter)' for acceptable filter names.
`levels`	Number or vector indicating levels of filter to plot. See Details.
`wavelet`	A logical flag indicating whether to plot the wavelet (high pass) or scaling (low pass) filter.
`y.normalize`	A logical flag indicating whether to vertically normalize each level of the filter to the plotting space available for each level of the filter.

Details

If a single number is specified for `levels`, then the filter of levels 1 through `levels` will be plotted. Otherwise, a vector will specify which levels of the wavelet coefficients will be plotted. If `levels` is not defined, it will default to the vector 1:7.

The plotting space available for each level of the filter is dictated by the value of greatest magnitude of all the levels of the filter plotted. The vertical plotting space for each level will then be 2 times the absolute value of this magnitude.

The label 'L sub j' on the horizontal axis varies from level to level, but 'L sub j' indicates the length of a filter at level j. The filter vector at a given level j is indexed from 0 to 'L sub j' - 1. 'L sub j' is defined by formula 96a in *Wavelet Methods for Time Series Analysis* by Percival and Walden (2000).

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`wt.filter`

Examples

```
# Plotting LA8 Wavelet Filter Coefficients Levels 1 through 7.
filter <- wt.filter()
figure98.wt.filter(filter)

# Alternatively
figure98.wt.filter("la8")

# Plotting D4 Scaling Filter Coefficients Levels 1, 3, and 5 and not
# vertically normalizing each level to its plotting region.
figure98.wt.filter("d4", levels = c(1,3,5), wavelet = FALSE, y.normalize
= FALSE)
```

idwt — *Inverse Discrete Wavelet Transform*

Description

Computes the inverse discrete wavelet transform for a discrete wavelet transform that was obtained from a univariate or multivariate time series.

Usage

```
idwt(wt, fast=TRUE)
```

Arguments

`wt`	A `dwt` object.
`fast`	A logical flag which, if true, indicates that the inverse pyramid algorithm is computed with an internal C function. Otherwise, only R code is used in all computations.

Details

The inverse discrete wavelet transform is computed via the inverse pyramid algorithm, using pseudocode written by Percival and Walden (2000), p. 101.

Value

An object with class and attributes equivalent to the original series that was used to compute the DWT. In general, the output will be equivalent to the original series (i.e. X = idwt(dwt(X))), however when thresholding or shrinkage methods are used on the `dwt` object, the output of `idwt` may differ from the original series.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`dwt`, `modwt`, `imodwt`.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)

# combine them and compute DWT
newX <- cbind(X1,X2)
wt <- dwt(newX, n.levels=3, boundary="reflection")

# compute the inverse DWT
invX <- idwt(wt)

# compare
newX
invX
```

`imodwt` *Inverse Maximal Overlap Discrete Wavelet Transform*

Description

Computes the inverse maximal overlap discrete wavelet transform for a maximal overlap discrete wavelet transform that was obtained from a univariate or multivariate time series.

Usage

```
imodwt(wt, fast=TRUE)
```

Arguments

`wt`	A `modwt` object.
`fast`	A logical flag which, if true, indicates that the inverse pyramid algorithm is computed with an internal C function. Otherwise, only R code is used in all computations.

Details

The inverse discrete wavelet transform is computed via the inverse pyramid algorithm, using pseudocode written by Percival and Walden (2000), p. 101.

Value

An object with class and attributes equivalent to the original series that was used to compute the MODWT. In general, the output will be equivalent to the original series (i.e. X = imodwt(modwt(X))), however when thresholding or shrinkage methods are used on the `modwt` object, the output of `imodwt` may differ from the original series.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

dwt, modwt, idwt.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)

# combine them and compute MODWT
newX <- cbind(X1,X2)
wt <- modwt(newX, n.levels=3, boundary="reflection")

# compute the inverse MODWT
invX <- imodwt(wt)

# compare
newX
invX
```

---

modwt *Maximal Overlap Discrete Wavelet Transform*

Description

Computes the maximal overlap discrete wavelet transform coefficients for a univariate or multivariate time series.

Usage

```
modwt(X, filter="la8", n.levels, boundary="periodic", fast=TRUE)
```

Arguments

`X`	A univariate or multivariate time series. Numeric vectors, matrices and data frames are also accepted.
`filter`	Either a `wt.filter` object, a character string indicating which wavelet filter to use in the decomposition, or a numeric vector of wavelet coefficients (not scaling coefficients). See `help(wt.filter)` for acceptable filter names.

`n.levels`	An integer specifying the level of the decomposition. By default this is the value J such that the length of X is at least as great as the length of the level J wavelet filter, but less than the length of the level $J+1$ wavelet filter. Thus, $J \leq \log\left(\frac{N-1}{L-1}+1\right)$, where N is the length of X.
`boundary`	A character string indicating which boundary method to use. `boundary = "periodic"` and `boundary = "reflection"` are the only supported methods at this time.
`fast`	A logical flag which, if true, indicates that the pyramid algorithm is computed with an internal C function. Otherwise, only R code is used in all computations.

Details

The maximal overlap discrete wavelet transform is computed via the pyramid algorithm, using pseudocode written by Percival and Walden (2000), p. 178. When `boundary="periodic"` the resulting wavelet and scaling coefficients are computed without making changes to the original series - the pyramid algorithm treats `X` as if it is circular. However, when `boundary="reflection"` a call is made to `extend.series`, resulting in a new series which is reflected to twice the length of the original series. The wavelet and scaling coefficients are then computed by using a periodic boundary condition on the reflected sereis, resulting in twice as many wavelet and scaling coefficients at each level.

Value

Returns an object of class `modwt`, which is an S4 object with slots

`W`	A list with element i comprised of a matrix containing the ith level wavelet coefficients.
`V`	A list with element i comprised of a matrix containing the ith level scaling coefficients.
`filter`	A `wt.filter` object containing information for the filter used in the decomposition. See `help(wt.filter)` for details.
`level`	An integer value representing the level of wavelet decomposition.
`n.boundary`	A numeric vector indicating the number of boundary coefficients at each level of the decomposition.
`boundary`	A character string indicating the boundary method used in the decomposition. Valid values are "periodic" or "reflection".
`series`	The original time series, `X`, in matrix format.
`class.X`	A character string indicating the class of the input series. Possible values are `"ts"`, `"mts"`, `"numeric"`, `"matrix"`, or `"data.frame"`.
`attr.X`	A list containing the attributes information of the original time series, `X`. This is useful if `X` is an object of class `ts` or `mts` and it is desired to retain relevant time information. If the original time series, `X`, is a matrix or has no attributes, then `attr.X` is an empty list.
`aligned`	A logical value indicating whether the wavelet and scaling coefficients have been phase shifted so as to be aligned with relevant time information from the original series. The value of this slot is initially FALSE and can only be changed to TRUE via the `align` function, with the `modwt` object as input.

`coe`	A logical value indicating whether the center of energy method was used in phase alignement of the wavelet and scaling coefficients. By default, this value is FALSE (and will always be FALSE when `aligned` is FALSE) and will be set to true if the `modwt` object is phase shifted via the `align` function and center of energy method.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`dwt`, `wt.filter`.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)

# combine them and compute DWT
newX <- cbind(X1,X2)
wt <- dwt(newX, n.level=3, boundary="reflection", fast=FALSE)
```

`modwt-class` *Maximal Overlap Discrete Wavelet Transform Object*

Description

An S4 object containing maximal overlap discrete wavelet transform information.

Slots

W A list with element i comprised of a matrix containing the ith level wavelet coefficients.

V A list with element i comprised of a matrix containing the ith level scaling coefficients.

filter A `wt.filter` object containing information for the filter used in the decomposition. See `help(wt.filter)` for details.

level An integer value representing the level of wavelet decomposition.

n.boundary A numeric vector indicating the number of boundary coefficients at each level of the decomposition.

boundary A character string indicating the boundary method used in the decomposition. Valid values are "periodic" or "reflection".

series The original time series, X, in matrix format.

class.X A character string indicating the class of the input series. Possible values are "ts", "mts", "numeric", "matrix", or "data.frame".

attr.X A list containing the attributes information of the original time series, X. This is useful if X is an object of class ts or mts and it is desired to retain relevant time information. If the original time series, X, is a matrix or has no attributes, then attr.X is an empty list.

aligned A logical value indicating whether the wavelet and scaling coefficients have been phase shifted so as to be aligned with relevant time information from the original series. The value of this slot is initially FALSE and can only be changed to TRUE via the align function, with the modwt object as input.

coe A logical value indicating whether the center of energy method was used in phase alignement of the wavelet and scaling coefficients. By default, this value is FALSE (and will always be FALSE when aligned is FALSE) and will be set to true if the modwt object is phase shifted via the align function and center of energy method.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

dwt, dwt-class, modwt, wt.filter.

---

mra *Multiresolution Analysis*

Description

Computes the multiresolution analysis for a univariate or multivariate time series.

Usage

```
mra(X, filter="la8", n.levels, boundary="periodic", fast=TRUE, method="dwt")
```

Arguments

X	A univariate or multivariate time series. Numeric vectors, matrices and data frames are also accepted.
filter	Either a wt.filter object, a character string indicating which wavelet filter to use in the decomposition, or a numeric vector of wavelet coefficients (not scaling coefficients). See help(wt.filter) for acceptable filter names.

`n.levels`	An integer specifying the level of the decomposition. By default this is the value J such that the length of X is at least as great as the length of the level J wavelet filter, but less than the length of the level $J+1$ wavelet filter. Thus, $J \leq \log\left(\frac{N-1}{L-1}+1\right)$, where N is the length of X.
`boundary`	A character string indicating which boundary method to use. `boundary = "periodic"` and `boundary = "reflection"` are the only supported methods at this time.
`fast`	A logical flag which, if true, indicates that the pyramid algorithm is computed with an internal C function. Otherwise, only R code is used in all computations.
`method`	A character string, taking values "dwt" or "modwt", that indicates which type of transform to use when computing the MRA.

Value

Returns an object of class `mra`, which is an S4 object with slots

`D`	A list with element i comprised of a matrix containing the ith level wavelet detail.
`S`	A list with element i comprised of a matrix containing the ith level wavelet smooths.
`filter`	A `wt.filter` object containing information for the filter used in the decomposition. See `help(wt.filter)` for details.
`level`	An integer value representing the level of wavelet decomposition.
`boundary`	A character string indicating the boundary method used in the wavelet decomposition. Valid values are "periodic" or "reflection".
`series`	The original time series, `X`, in matrix format.
`class.X`	A character string indicating the class of the input series. Possible values are `"ts"`, `"mts"`, `"numeric"`, `"matrix"`, or `"data.frame"`.
`attr.X`	A list containing the attributes information of the original time series, `X`. This is useful if `X` is an object of class `ts` or `mts` and it is desired to retain relevant time information. If the original time series, `X`, is a matrix or has no attributes, then `attr.X` is an empty list.
`method`	A character string indicating which type of wavelet decomposition was performed (either "dwt" or "modwt").

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`dwt`, `modwt`, `wt.filter`.

Examples

```
# obtain the two series listed in Percival and Walden (2000), page 42
X1 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,.7,.9,0,.3)
X2 <- c(.2,-.4,-.6,-.5,-.8,-.4,-.9,0,-.2,.1,-.1,.1,-.7,.9,0,.3)

# combine them and compute MRA
newX <- cbind(X1,X2)
mra.out <- mra(newX, n.levels=3, boundary="reflection")
```

`mra-class` *Multiresolution Analysis Object*

Description

An S4 object containing multiresolution analysis information.

Slots

D A list with element i comprised of a matrix containing the ith level wavelet detail.

S A list with element i comprised of a matrix containing the ith level wavelet smooths.

filter A `wt.filter` object containing information for the filter used in the decomposition. See `help(wt.filter)` for details.

level An integer value representing the level of wavelet decomposition.

boundary A character string indicating the boundary method used in the wavelet decomposition. Valid values are "periodic" or "reflection".

series The original time series, `X`, in matrix format.

class.X A character string indicating the class of the input series. Possible values are `"ts"`, `"mts"`, `"numeric"`, `"matrix"`, or `"data.frame"`.

attr.X A list containing the attributes information of the original time series, `X`. This is useful if `X` is an object of class `ts` or `mts` and it is desired to retain relevant time information. If the original time series, `X`, is a matrix or has no attributes, then `attr.X` is an empty list.

method A character string indicating which type of wavelet decomposition was performed (either "dwt" or "modwt").

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`dwt`, `modwt`, `wt.filter`.

`nile` *Yearly Nile River Minima*

Description

Yearly minimum water level of the Nile River starting in 622 A.D. and ending in 1284 A.D. For more information see Percival and Walden (2000), p. 190.

Usage

```
data(nile)
```

Format

A time series object containing 683 observations.

Source

http://www.ms.washington.edu/~s530/data.html

References

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press, sec. 5.9.

`ocean` *Vertical Ocean Sheer Measurements*

Description

Measurements of vertical ocean sheer, obtained by dropping an instrument veritcally into the ocean and recording observations at 0.1 meter intervals. Starting depth for the series is 350.0 meters and ending depth is 1037.4 meters. For more detailed information regarding the collection of this data, see Percival and Walden (2000), p. 193.

Usage

```
data(ocean)
```

Format

A time series object containing 6875 observations.

Source

http://www.ms.washington.edu/~s530/data.html

References

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press, sec. 5.10.

`plot.dwt` *Plot DWT Coefficients*

Description

Plot DWT wavelet and scaling coefficients.

Usage

```
## S3 method for class 'dwt'
plot(x, levels = NULL, draw.boundary = FALSE, type = "stack",
col.plot = "black", col.boundary = "red", X.xtick.at = NULL, X.ytick.at
= NULL, Stack.xtick.at = NULL, Stack.ytick.at = NULL, X.xlab = "t",
y.rlabs = TRUE, plot.X = TRUE, plot.W = TRUE, plot.V = TRUE, ...)
```

Arguments

`x`	An object of class `dwt`.
`levels`	Number, vector, or list of two vectors indicating range of levels to plot. See details.
`draw.boundary`	Logical value indicating whether to draw boundary coefficients.
`type`	Type of plot to draw. Currently only "stack" is implemented.
`col.plot`	Color of wavelet and scaling coefficients.
`col.boundary`	Color of boundary coefficient lines.
`X.xtick.at`	Vector specifying the extreme tick mark locations and the number of intervals between those extreme tick marks on the horizontal axis of the plot of the original times series. This vector takes the form similar to par("xaxp"), and is defaulted to par("xaxp") when `X.xtick.at` is not specified.
`X.ytick.at`	Vector specifying the extreme tick mark locations and the number of intervals between those extreme tick marks on the vertical axis of the plot of the original times series. This vector takes the form similar to par("yaxp"), and is defaulted to par("yaxp") when `X.ytick.at` is not specified.
`Stack.xtick.at`	Vector of form similar to `X.xtick.at` specifying the tick mark locations on the horizontal axis of the stacked plot. This is applicable only if `plot.X` is FALSE.
`Stack.ytick.at`	Vector of form similar to `X.ytick.at` specifying the tick mark locations on the vertical axis of the stacked plot. This is applicable only if `plot.X` is FALSE.
`X.xlab`	String specifying the label of the horizontal axis of the plot of the original time series.

`y.rlabs`	Logical flag indicating whether to draw the vertical labels on the right vertical axis of the stacked plot. These labels indicate the number of boundary coefficients to the right of the right boundary line, and the level of decomposition of the wavelet (or scaling) coefficients.
`plot.X`	Logical flag indicating whether to draw original time series.
`plot.W`	Logical flag indicating whether to draw the wavelet coefficients in the stacked plot.
`plot.V`	Logical flag indicating whether to draw the scaling coefficients in the stacked plot.
`...`	Additional paramters that are acceptable arguments to the generic `plot` function

Details

`plot.dwt` plots the DWT wavelet and scaling coefficients. The `type` parameter species the type of plot, which is currently only stack. If the DWT object is defined for multiple time series, only the data pertaining to the first time series of the DWT object is plotted. Thus, only the wavelet coefficients and scaling coefficients of the first time series of the DWT object will be plotted.

If a single number is specified for `levels`, then the wavelet coefficients of levels 1 through `levels` will be plotted. Otherwise, a vector or the first element of a list will specify which levels of the wavelet coefficients will be plotted. Unless specified in the second element of a list, only one level of scaling coefficients will be plotted and this level is equal to the highest level of the wavelet coefficients plotted.

For each respective axis, the distance between a tick mark on the plot of the original time series is equivalent to the distance between a tick mark on the stacked plot. Thus, when altering the relative spacing of the tick marks on the plot of the original time series using `X.xtick.at` or `X.ytick.at`, the tick marks of the stacked plot are automatically adjusted. If the plot of the original time series is not drawn, then the user can alter the spacing of the tick marks using `Stack.xtick.at` and `Stack.ytick.at` for the horizontal and vertical axes, respectively.

One of `plot.W` or `plot.V` must be TRUE.

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`stackplot`

Examples

```
X <- rnorm(2048)
dwtobj <- dwt(X)
```

```
# Plotting wavelet coefficients of levels 1 through 6 and scaling
# coefficients of level 6.
plot.dwt(dwtobj, levels = 6)

# Plotting wavelet coefficients of levels 1, 3, 5, and scaling
# coefficients of levels 4 and 5, and green boundary coefficient lines.
plot.dwt(dwtobj, levels = list(c(1,3,5),c(4,5)), draw.boundary = TRUE,
col.boundary = "green")

# Plotting wavelet coefficients of level 1 through 6 and not plotting
# any scaling coefficients.
plot.dwt(dwtobj, levels = 6, plot.V = FALSE)
```

`plot.dwt.multiple` *Plot Multiple DWT Objects*

Description

Plot wavelet and scaling coefficients of multiple DWT objects.

Usage

```
## S3 method for class 'multiple'
plot.dwt(x, levels = NULL, ylim = NULL, draw.dashed.lines =
TRUE, draw.level.labels = TRUE, col = c("red","blue"), ...)
```

Arguments

`x`	A list of `dwt` objects.
`levels`	Number, vector, or list of two vectors indicating range of levels to plot. See details.
`ylim`	Vector specifying the lower and upper limits of the vertical y-axis of each plot.
`draw.dashed.lines`	Boolean indicating whether dashed lines should be drawn between levels of wavelet and scaling coefficients.
`draw.level.labels`	Boolean indicating whether the labels for the levels of wavelet and scaling coefficients should be drawn.
`col`	Vector of length 2 for the alternating colors of the coefficients to be drawn. The colors alternate by level of the wavelet or scaling coefficients drawn. The first element of the vector is the color of the coefficients of the first level drawn.
`...`	Additional paramters that are acceptable arguments to the generic `plot` function

Details

If a single number is specified for `levels`, then the wavelet coefficients of levels 1 through `levels` will be plotted. Otherwise, a vector or the first element of a list will specify which levels of the wavelet coefficients will be plotted. Unless specified in the second element of a list, only one level of scaling coefficients will be plotted and this level is equal to the highest level of the wavelet coefficients plotted. If a DWT object is defined for multiple time series, only the data pertaining to the first time series of the DWT object is plotted. Thus, only the wavelet coefficients and scaling coefficients of the first time series of the DWT objects will be plotted.

For each `dwtobject` in the list of `x`, `plot.dwt.multiple` takes the coefficients of the `dwt` object and concatenates wavelet coefficients and scaling coefficients by levels specified in `levels`. The wavelet coefficients will always be plotted preceding the scaling coefficients.

This function allows users to visually examine differences in the DWT transform of a time series using different filters (different `dwt` objects).

For an example, see Figure 126 of *Wavelet Methods for Time Series Analysis* by Percival and Walden (2000).

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`plot.dwt`

Examples

```
X <- rnorm(2048)
# Create DWT Object of X with the "la8" filter.
dwtobj1 <- dwt(X, filter = "la8")
# Create DWT Object of X with the "d4" filter.
dwtobj2 <- dwt(X, filter = "d4")
# Create DWT Object of X with the "haar" filter
dwtobj3 <- dwt(X, filter = "haar")
# Create DWT Object of X with the "c6" filter
dwtobj4 <- dwt(X, filter = "c6")

#Create list of dwt objects
dwtlist <- list(dwtobj1, dwtobj2, dwtobj3, dwtobj4)

# Plot the dwt objects and the wavelet coefficients of level 1 through 6
# and the scaling coefficients of level 6.  The first level drawn will
# be purple and the next level drawn will be gold.
plot.dwt.multiple(dwtlist, levels = 6, col = c("purple", "gold"))
```

```
# Plot the dwt objects and the wavelet coefficients of level 1, 3, and 5
# and scaling coefficients of level 2, and 4.
plot.dwt.multiple(dwtlist, levels = list(c(1,3,5), c(2,4)))
```

`plot.modwt` *Plot MODWT Coefficients*

Description

Plot MODWT wavelet and scaling coefficients.

Usage

```
## S3 method for class 'modwt'
plot(x, levels = NULL, draw.boundary = FALSE, type = "stack",
col.plot = "black", col.boundary = "red", X.xtick.at = NULL, X.ytick.at
= NULL, Stack.xtick.at = NULL, Stack.ytick.at = NULL, X.xlab = "t",
y.rlabs = TRUE, plot.X = TRUE, plot.W = TRUE, plot.V = TRUE, ...)
```

Arguments

`x`	An object of class `dwt`.
`levels`	Number, vector, or list of two vectors indicating range of levels to plot. See details.
`draw.boundary`	Logical value indicating whether to draw boundary coefficients.
`type`	Type of plot to draw. Currently only "stack" is implemented.
`col.plot`	Color of wavelet and scaling coefficients.
`col.boundary`	Color of boundary coefficient lines.
`X.xtick.at`	Vector specifying the extreme tick mark locations and the number of intervals between those extreme tick marks on the horizontal axis of the plot of the original times series. This vector takes the form similar to par("xaxp"), and is defaulted to par("xaxp") when `X.xtick.at` is not specified.
`X.ytick.at`	Vector specifying the extreme tick mark locations and the number of intervals between those extreme tick marks on the vertical axis of the plot of the original times series. This vector takes the form similar to par("yaxp"), and is defaulted to par("yaxp") when `X.ytick.at` is not specified.
`Stack.xtick.at`	Vector of form similar to `X.xtick.at` specifying the tick mark locations on the horizontal axis of the stacked plot. This is applicable only if `plot.X` is FALSE.
`Stack.ytick.at`	Vector of form similar to `X.ytick.at` specifying the tick mark locations on the vertical axis of the stacked plot. This is applicable only if `plot.X` is FALSE.
`X.xlab`	String specifying the label of the horizontal axis of the plot of the original time series.

`y.rlabs`	Logical flag indicating whether to draw the vertical labels on the right vertical axis of the stacked plot. These labels indicate the number of boundary coefficients to the right of the right boundary line, and the level of decomposition of the wavelet (or scaling) coefficients.
`plot.X`	Logical flag indicating whether to draw original time series.
`plot.W`	Logical flag indicating whether to draw the wavelet coefficients in the stacked plot.
`plot.V`	Logical flag indicating whether to draw the scaling coefficients in the stacked plot.
`...`	Additional paramters that are acceptable arguments to the generic `plot` function

Details

`plot.modwt` plots the MODWT wavelet and scaling coefficients. The `type` parameter species the type of plot, which is currently only stack. If the MODWT object is defined for multiple time series, only the data pertaining to the first time series of the MODWT object is plotted. Thus, only the wavelet coefficients and scaling coefficients of the first time series of the MODWT object will be plotted. If a single number is specified for `levels`, then the wavelet coefficients of levels 1 through `levels` will be plotted. Otherwise, a vector or the first element of a list will specify which levels of the wavelet coefficients will be plotted. Unless specified in the second element of a list, only one level of scaling coefficients will be plotted and this level is equal to the highest level of the wavelet coefficients plotted.

For each respective axis, the distance between a tick mark on the plot of the original time series is equivalent to the distance between a tick mark on the stacked plot. Thus, when altering the relative spacing of the tick marks on the plot of the original time series using `X.xtick.at` or `X.ytick.at`, the tick marks of the stacked plot are automatically adjusted. If the plot of the original time series is not drawn, then the user can alter the spacing of the tick marks using `Stack.xtick.at` and `Stack.ytick.at` for the horizontal and vertical axes, respectively.

One of `plot.W` or `plot.V` must be TRUE.

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`stackplot`

Examples

```
X <- rnorm(2048)
modwtobj <- modwt(X)
```

```
# Plotting wavelet coefficients at levels 1 through 6 and scaling
# coefficients at level 6.
plot.modwt(modwtobj, levels = 6)

# Plotting wavelet coefficients of levels 1, 3, 5, and scaling
# coefficients of levels 4 and 5, and green boundary coefficients
# lines.
plot.modwt(modwtobj, levels = list(c(1,3,5),c(4,5)), draw.boundary =
TRUE, col.boundary = "green")

# Plotting wavelet coefficients at level 1 through 6 and not plotting
# any scaling coefficients.
plot.modwt(modwtobj, levels = 6, plot.V = FALSE)
```

`scalingshift.dwt` *Find Circular Shift of DWT Scaling Coefficients*

Description

Finds circular shift of DWT scaling coefficients at a specified level, for a specified filter length.

Usage

```
scalingshift.dwt(L, j, N = NULL)
```

Arguments

`L`	Length of wavelet transform filter used.
`j`	Level of DWT scaling coefficients to be shifted.
`N`	Length of original time series sample.

Details

This function computes the circular shift associated with a wavelet transform filter of length `L` and level `j`. `L` must be of even length.

Each coefficient's index will be circularly shifted forward by the value outputted. For example, if we have a vector of DWT scaling coefficients (with NAs inserted) with indices 1,2,3,4,5,6,7,8 and the value outputted from the function `scalingshift.dwt` is 2, the new order of the vector of aligned DWT wavelet coefficients would be 7,8,1,2,3,4,5,6.

If `N` is specified, the function will output the "shift" Modulo N. Because the shift is circular, the value outputted when N is specified is equivalent to the case when N is not specified.

This function calculates the absolute value of vjH in formulas (114c) in *Wavelet Methods for Time Series Analysis* by Percival and Walden (2000). (And optionally vjH Modulo N)

Value

`shift`	Circular shift for specified level of DWT scaling coefficients for a specified filter length.

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`waveletshift.dwt` and `plot.dwt`

Examples

```
# Finding the circular shift for a wavelet transform filter of length 8,
# pertaining to DWT Scaling Coefficients of level 5.
scalingshift.dwt(8, 5)

# If the sample size of the original time series is of length 1024.
scalingshift.dwt(8, 5, N = 1024)
```

`squaredgain.wt.filter` *Plot Squared Gain Function for Wavelet Filter*

Description

Plots squared gain function for a wavelet filter.

Usage

```
squaredgain.wt.filter(filter, level = 1, N = NULL, draw.bands = TRUE,
wavelet = TRUE)
```

Arguments

`filter`	Either a 'wt.filter' object, a character string specifying a wavelet filter, or a numeric vector of wavelet coefficients. See 'help(wt.filter)' for acceptable filter names.
`level`	Level of wavelet filter. Applicable only if a character string is supplied in `filter`
`N`	Length of filter vector after padding the wavelet filter with zeroes. The zeroes are inserted after the filter values.
`draw.bands`	Draws dashed lines marking the lower and upper ends of the nominal pass-band for the wavelet filter.
`wavelet`	A logical flag indicating whether to plot the squared gain function for the wavelet (high pass) or scaling (low pass) filter.

Details

If `N` is not specified, and the filter supplied is of length less than 1024, then the filter vector will be padded to a length of 1024. Otherwise, if the filter supplied is of length greater than 1024, then the filter vector will be padded to the first 'power of 2' that is greater than the length of the filter supplied.

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`wt.filter`

Examples

```
# Plotting the squared gain function of LA8 Wavelet Filter Coefficients.
filter <- wt.filter("la8")
squaredgain.wt.filter(filter)

# Plotting the squared gain function of LA8 Scaling Filter Coefficients.
squaredgain.wt.filter(filter, wavelet = FALSE)

# Plotting the squared gain function of Haar Wavelet Filter Coefficients
# without supplying a filter object.
squaredgain.wt.filter("haar")
```

`stackplot` *Stacked Plotting*

Description

Plot an object as a stacked set of plots, plotting each column using the same x-axis and plotting the y-values at the same scale with an offset equal to the magnitude of each column vector.

Usage

```
stackplot(x, y = NULL, type = "h", axes.labels = FALSE,
xlab = "", ylab = "", y.llabs = NULL, y.rlabs = NULL, draw.divides =
TRUE, xtick.at = NULL, ytick.at = NULL, col = "black", main = "")
```

Arguments

`x`	Specifies the coordinates of the points to be plotted. Alternatively, `x` can be a list containing matrices representing the values to be plotted, or a time series object. The x-values will be specified by the rownames of each matrix. The y-values, or offsets of each level, will be represented by the values of the elements in the columns of each matrix.
`y`	If `x` specifies the coordinates of the points to be plotted, then `y` can be a list containing matrices that represent the values to be plotted. The y-values, or offsets of each level will be represented by the values of the elements in the columns of each matrix.
`type`	Type of plots to be drawn. Currently only `h`, or histogram, and `l`, or lines, are implemented.
`axes.labels`	Boolean representing whether or not to display the numeric labels of the horizontal and vertical axes.
`xlab`	String indicating the label for the horizontal x-axis.
`ylab`	String indicating the label for the vertical y-axis.
`y.llabs`	Vector or list of labels to be placed on left side of plot.
`y.rlabs`	Vector or list of labels to be placed on right side of plot.
`draw.divides`	Logical value indicating whether to draw lines to separate each subplot in the stackplot.
`xtick.at`	Vector specifying the tick mark locations on the horizontal axis of the plot. Defaulted to par("xaxp").
`ytick.at`	Vector specifying the tick mark locations on the vertical axis of the plot. Defaulted to par("yaxp").
`col`	String representing color of plot.
`main`	String representing the title of the plot.

Details

Each object specified by `x` or `y` will be coerced into a list, and each object of the list will be coerced into a matrix using the function `as.matrix`. Each column of a matrix will represent the vertical y-values for each individual plot in the stacked plot. Horizontal x-values are discussed below.

`stackplot` will plot a matrix of values as a series of `stacked` plots, where the individual columns are plotted using the same `x` and `y` scales, but offset by the magnitude (= abs(max-min)) of the column.

If the object to be plotted is a Time Series object, the `start` and `end` values of the object will represent the first and last horizontal x-values of the points to be plotted. The Time Series will be coerced into matrix.

The limits of the horizontal x-axis and vertical y-axis will be dictated by the data in the first object of the list.

If the object to be plotted is a list of matrices, and `x` is not specified, the horizontal x-values of the points to be plotted for a given matrix will default to the rownames of that matrix. The rownames can be either `numeric` or `character`. If the first matrix of the list does not have rownames, then the horizontal x-values for each matrix (and its corresponding plot) will default to range from 1 to the length of the columns of the respective matrix.

If the first matrix of the list does have specified rownames, then the horizontal x-values assume the values of the rownames of the respective matrix. However, if another matrix of the list does not have specified rownames, then its horizontal x-values will default to range from 1 to the length of the columns of that respective matrix.

The user can specify the type of plot (currently only "h" and "l") to be drawn for every element of the list in `x` or `y` using the vector `type`. If the Nth element (N>1) in `type` is NA and the Nth element of the list in `x` or `y` exists, then the type of plot for the Nth element of the list will be equal to the type of plot for the N-1th element of the list. Similarly for `col`, which specifies the colors for each element of the list in `x` or `y`.

The labels `y.llabs` and `y.rlabs` will be placed at the "zero" lines of each individual plot starting from the bottom working up to the top.

Author(s)

Kelvin Ma, kkym@u.washington.edu

See Also

`plot.dwt` and `plot.modwt`

Examples

```
A <- array(c(1,2,3,4), c(3,4))

# Plotting the vectors c(1,2,3), c(4,1,2), c(3,4,1), and c(2,3,4) in a
# stacked plot with 4 separate plots).  The x-values will be 1:3.
stackplot(A)

#Plotting A with numeric labels on the axis and making the plot "red".
#Also labeling the x-axis with "X-Label", and the y-axis with "Y-Label".
stackplot(A, axes.labels = TRUE, xlab = "X-Label", ylab = "Y-Label")

timeSeries <- ts(A)
# Plotting the Time Series created by coercing the array A into a Time
# Series.  Observe that this plot similar to the plot in the first example.
stackplot(timeSeries)

lfig <- 2:11
hfig <- 1:10
# Plotting a stacked plot of two types: histogram and lines.  Where the
# line joins the points (1,2) and (10,11), and the histogram is similar to
# the histogram formed by entering plot(1:10, type = "h").  The line will
# be red and the histogram will be yellow.
stackplot(list(lfig, hfig), type = c("l", "h"), col = c("red", "yellow"))
```

subtidal *Subtidal Sea Level Measurements*

Description

Wwater level measurements from a stilling well on the open coast in Crescent Bay, CA. Observations were recorded every six minutes, from the beginning of 1980 to the end of 1991. For more detailed information regarding the collection of this data, see Percival and Walden (2000), p. 185.

Usage

```
data(subtidal)
```

Format

A time series object containing 8746 observations.

Source

http://www.ms.washington.edu/~s530/data.html

References

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press, sec. 5.8.

waveletshift.dwt *Find Circular Shift of DWT Wavelet Coefficients*

Description

Finds circular shift of DWT wavelet coefficients at a specified level, for a specified filter length.

Usage

```
waveletshift.dwt(L, j, N = NULL)
```

Arguments

`L`	Length of wavelet transform filter used.
`j`	Level of DWT wavelet coefficients to be shifted.
`N`	Length of original time series sample.

Details

This function computes the circular shift associated with a wavelet transform filter of length `L` and level `j`. `L` must be of even length.

Each coefficient's index will be circularly shifted forward by the value outputted. For example, if we have a vector of DWT wavelet coefficients (with NAs inserted) with indices 1,2,3,4,5,6,7,8 and the value outputted from the function `waveletshift.dwt` is 2, the new order of the vector of aligned DWT wavelet coefficients would be 7,8,1,2,3,4,5,6.

If `N` is specified, the function will output the "shift" Modulo N. Because the shift is circular, the value outputted when N is specified is equivalent to the case when N is not specified.

This function calculates the absolute value of vjG in formulas (114c) in *Wavelet Methods for Time Series Analysis* by Percival and Walden (2000). (And optionally vjG Modulo N)

Value

`shift`	Circular shift for specified level of DWT wavelet coefficients for a specified filter length.

Author(s)

Kelvin Ma, kkym@u.washington.edu

References

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

See Also

`scalingshift.dwt` and `plot.dwt`

Examples

```
# Finding the circular shift for a wavelet transform filter of length 8,
# pertaining to DWT Wavelet Coefficients of level 5.
waveletshift.dwt(8, 5)

# If the sample size of the original time series is of length 1024.
waveletshift.dwt(8, 5, N = 1024)
```

`wt.filter` *Wavelet Transform Filter*

Description

Generates a wavelet transform filter.

Usage

```
wt.filter(filter="la8", modwt=FALSE, level=1)
```

Arguments

`filter`	A character string indicating which wavelet transform filter to compute or a numeric vector of wavelet (high pass) filter coefficients (not scaling (low pass) coefficients). If a numeric vector is supplied, the length must be even.
`modwt`	A logical value indicating whether to compute the maximal overlap discrete wavelet transform filter.
`level`	An integer value indicating the level of the wavelet filter to compute.

Details

The character strings currently supported are derived from one of four classes of wavelet transform filters: Daubechies, Least Asymetric, Best Localized and Coiflet. The prefixes for filters of these classes are `d`, `la`, `bl` and `c`, respectively. Following the prefix, the filter name consists of an integer indicating length. Supported lengths are as follows:

Daubechies 2,4,6,8,10,12,14,16,18,20.

Least Asymetric 8,10,12,14,16,18,20.

Best Localized 14,18,20.

Coiflet 6,12,18,24,30.

Thus, to obtain the Daubechies wavelet transform filter of length 4, the character string `"d4"` can be passed to `wt.filter`.

This naming convention has one exception: the Daubechies wavelet transform filter of length 2 is denoted by `haar` instead of `d2`.

Value

Returns an object of class `wt.filter`, which is an S4 object with slots

`L`	An integer representing the length of the wavelet and scaling filters.
`h`	A numeric vector of wavelet filter coefficients.
`g`	A numeric vector of scaling filter coefficients.
`wt.class`	A character string indicating the class of the wavelet transform filter. Possible values are `"Daubechies"`, `"Least Asymetric"`, `"Best Localized"`, and `"Coiflet"`. If the `wt.filter` object is generated from a numeric vector of wavelet coefficients, `wt.class` is `"none"`.
`wt.name`	A character string indicating the name of the wavlet filter as listed in the Details section, above. If the `wt.filter` object is generated from a numeric vector of wavelet coefficients, `wt.name` is `"none"`.
`transform`	A character string indicating whether the resulting wavelet transform object contains DWT or MODWT coefficients. Possible values are `"dwt"` and `"modwt"`.

Note

The notation h and g for wavelet and scaling coefficients, respectively, follows Percival and Walden (2000). In other texts and articles the reverse notation is often adopted.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`wt.filter.qmf`, `dwt`, `modwt`.

Examples

```
wt.filter("la14")

wt.filter(1:10, modwt=TRUE)
```

`wt.filter-class` *Wavelet Transform Filter Object*

Description

An S4 object containing wavelet transform filter information.

Slots

L An integer representing the length of the wavelet and scaling filters.

level An integer representing the level of the wavelet filter.

h A numeric vector of wavelet filter coefficients.

g A numeric vector of scaling filter coefficients.

wt.class A character string indicating the class of the wavelet transform filter. Possible values are `"Daubechies"`, `"Least Asymetric"`, `"Best Localized"`, and `"Coiflet"`. If the `wt.filter` object is generated from a numeric vector of wavelet coefficients, `wt.class` is `"none"`.

wt.name A character string indicating the name of the wavlet filter as listed in the Details section, above. If the `wt.filter` object is generated from a numeric vector of wavelet coefficients, `wt.name` is `"none"`.

transform A character string indicating whether the resulting wavelet transform object contains DWT or MODWT coefficients. Possible values are `"dwt"` and `"modwt"`.

Note

The notation h and g for wavelet and scaling coefficients, respectively, follows Percival and Walden (2000). In other texts and articles the reverse notation is often adopted.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`wt.filter.qmf`, `dwt`, `modwt`.

`wt.filter.equivalent` *Equivalent Wavelet Transform Filter*

Description

Generates an equivalent filter for a specified wavelet transform filter and level..

Usage

```
wt.filter.equivalent(wt.filter, J)
```

Arguments

`wt.filter`	A `wt.filter` object.
`J`	Level of equivalent filter to compute.

Details

A wavelet transform of level j can be computed by iterating the pyramid algorithm j times with a level 1 wavelet filter, or by simply implementing the pyramid once with the level j equivalent filter. The equivalent filter is obtained by iteratively filtering the level 1 wavelet filter and scaling filters with upsampled versions of themselves. For details regarding the computation, see Percival and Walden (2000), eqs. 95e and 96e.

Value

Returns the original `wt.filter` object with modifications made to the wavelet and scaling filter coefficients, filter length and filter level.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`wt.filter`.

Examples

```
wf <- wt.filter("la14")
wt.filter.equivalent(wf, 3)
```

`wt.filter.qmf` *Quadrature Mirror Filter*

Description

Computes the quadrature mirror filter of a series of even length.

Usage

```
wt.filter.qmf(x, inverse=FALSE)
```

Arguments

`x`	A numeric vector of even length
`inverse`	A logical flag indicating whether to compute the inverse quadrature mirror filter.

Details

The quadrature mirror filter is computed as outlined in Percival and Walden, page 75. Specifically, the forward QMF makes use of a wavelet filter to obtain a scaling filter and the inverse QMF does the opposite.

Value

A numeric vector which contains the values of `x` in reverse order, with the values of the resulting vector at even indices being negated in the case that `inverse=FALSE` and the values at odd indices being negated in the case that `inverse=TRUE`.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`wt.filter.qmf`, `dwt`, `modwt`.

Examples

```
# obtain both the la8 wavelet and scaling filters
la8 <- wt.filter("la8")

# now compare output of qmf with values in wt.filter object
wt.filter.qmf(la8@h)
la8@g
```

---

wt.filter.shift *Wavelet Transform Filter Phase Shift*

Description

Computes phase shifts for wavelet transform coefficients corresponding to a particular filter.

Usage

```
wt.filter.shift(filter, J, wavelet=TRUE, coe=FALSE, modwt=FALSE)
```

Arguments

`filter`	A `wt.filter` object, a character string indicating which wavelet transform filter to compute, or a numeric vector of wavelet (high pass) filter coefficients (not scaling (low pass) coefficients). If a numeric vector is supplied, the length must be even.
`J`	A vector of positive integers indicating levels for which to compute phase shifts.
`wavelet`	A logical flag indicating whether to compute the wavelet (high pass) or scaling (low pass) phase shift(s).
`coe`	A logical value indicating whether to use the center of energy method (see Percival and Walden 2000, page 118) to compute the phase shift(s).
`modwt`	A logical value indicating whether to compute MODWT phase shift(s).

Details

For wavelet filters of class ’Least Assymetric’ or ’Coiflet’, the default method for calculating phase shifts is outlined on pages 112-114 and page 124 of Percival and Walden 2000. For wavelet filters of class ’Best Localized’, the default shifts are given on page 119 of Percival and Walden 2000. For the Haar filter, both the level j wavelet and scaling phase shifts are half the length of the level j wavelet and scaling filters and the phase shifts for the D(4) filter are determined by specifying $\nu = -1$ in equations (114a) and (114b) of Percival and Walden 2000.

For all other filters, the center of energy method is the default method for computing phase shifts (thus rendering the `coe` argument irrelevant). If `coe = TRUE`, then the center of energy method is used regardless of filter class.

By default the DWT phase shifts are computed, using the MODWT phase shifts and the methods outlined on pages 115-116 of Percival and Walden 2000. If `modwt = TRUE`, then only the MODWT phase shifts are computed.

Value

A vector of phase shifts, expressed in absolute value. Each element corresponds to the equivalent element in `J`.

Author(s)

Eric Aldrich. ealdrich@gmail.com.

References

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

See Also

`wt.filter`, `align`,

Examples

```
wt.filter.shift("la14", J=1:6)
```

Index