Package ‘wv’

January 16, 2020

Type Package

Title Wavelet Variance

Version 0.1.1

Date 2020-01-16

LazyData true

Maintainer Stéphane Guerrier <stef.guerrier@gmail.com>

Description Provides a series of tools to compute and plot quantities related to classical and robust wavelet variance for time series and regular lattices. More details can be found, for example, in Serroukh, A., Walden, A.T., & Percival, D.B. (2000) <doi:10.2307/2669537> and Guerrier, S. & Molinari, R. (2016) <arXiv:1607.05858>.

Depends R (>= 3.5.0)

License AGPL-3

Imports Rcpp, simts, utils, grDevices, coda, methods, graphics, stats

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.0.2

Encoding UTF-8

Suggests knitr, rmarkdown

URL https://github.com/SMAC-Group/wv

BugReports https://github.com/SMAC-Group/wv/issues

NeedsCompilation yes

Author Stéphane Guerrier [aut, cre],
James Balamuta [aut],
Justin Lee [aut],
Roberto Molinari [aut],
Yuming Zhang [aut],
Nathanael Claussen [ctb],
Haotian Xu [ctb],
Lionel Voirol [ctb]

Repository CRAN

Date/Publication 2020-01-16 18:40:02 UTC

R topics documented:

ACF *Auto-Covariance and Correlation Functions*

Description

The ACF function computes the estimated autocovariance or autocorrelation for both univariate and multivariate cases.

Usage

```
ACF(x, lagmax = 0, cor = TRUE, demean = TRUE)
```

Arguments

x	A `matrix` with dimensions $N \times S$ or N observations and S processes
lagmax	A `integer` indicating the max lag.
cor	A `bool` indicating whether the correlation (`TRUE`) or covariance (`FALSE`) should be computed.
demean	A `bool` indicating whether the data should be detrended (`TRUE`) or not (`FALSE`)

Details

`lagmax` default is $10 * log10(N/m)$ where N is the number of observations and m is the number of series being compared. If `lagmax` supplied is greater than the number of observations, then one less than the total will be taken.

Value

An `array` of dimensions $N \times S \times S$.

Author(s)

Yunxiang Zhang

Examples

```
# Get Autocorrelation
m = ACF(datasets::AirPassengers)

# Get Autocovariance and do not remove trend from signal
m = ACF(datasets::AirPassengers, cor = FALSE, demean = FALSE)
```

---

adis_wv *Wavelet variance of IMU Data from an ADIS 16405 sensor*

Description

This data set contains wavelet variance of gyroscope and accelerometer data from an ADIS 16405 sensor.

Usage

```
adis_wv
```

Format

A list of the following elements:

- "sensor": Name of the sensor.
- "freq": The frequency at which the error signal is measured.
- "n": Sample size of the data.
- "type": The types of sensors considered in the data.
- "axis": The axes of sensors considered in the data.
- "wvar": A list containing the computed wavelet variance based on the data.

Source

The IMU data comes from Department of Geomatics Engineering, University of Calgary.

`ar1_to_wv` *AR(1) process to WV*

Description

This function computes the Haar WV of an AR(1) process

Usage

```
ar1_to_wv(phi, sigma2, tau)
```

Arguments

`phi`	A `double` that is the phi term of the AR(1) process
`sigma2`	A `double` corresponding to variance of AR(1) process
`tau`	A `vec` containing the scales e.g. 2^{τ}

Details

This function is significantly faster than its generalized counter part `arma_to_wv`.

Value

A `vec` containing the wavelet variance of the AR(1) process.

Process Haar Wavelet Variance Formula

The Autoregressive Order 1 (AR(1)) process has a Haar Wavelet Variance given by:

$$\frac{2\sigma^2\left(4\phi^{\frac{\tau_j}{2}+1}-\phi^{\tau_j+1}-\frac{1}{2}\phi^2\tau_j+\frac{\tau_j}{2}-3\phi\right)}{\left(1-\phi\right)^2\left(1-\phi^2\right)\tau_j^2}$$

See Also

arma_to_wv, arma11_to_wv

---

arma11_to_wv *ARMA(1,1) to WV*

Description

This function computes the WV (haar) of an Autoregressive Order 1 - Moving Average Order 1 (ARMA(1,1)) process.

Usage

```
arma11_to_wv(phi, theta, sigma2, tau)
```

Arguments

phi	A double corresponding to the autoregressive term.
theta	A double corresponding to the moving average term.
sigma2	A double the variance of the process.
tau	A vec containing the scales e.g. 2^τ

Details

This function is significantly faster than its generalized counter part arma_to_wv

Value

A vec containing the wavelet variance of the ARMA(1,1) process.

Process Haar Wavelet Variance Formula

The Autoregressive Order 1 and Moving Average Order 1 (ARMA(1,1)) process has a Haar Wavelet Variance given by:

$$\nu_j^2\left(\phi,\theta,\sigma^2\right) = -\frac{2\sigma^2\left(-\frac{1}{2}(\theta+1)^2\left(\phi^2-1\right)\tau_j-(\theta+\phi)(\theta\phi+1)\left(\phi^{\tau_j}-4\phi^{\frac{\tau_j}{2}}+3\right)\right)}{(\phi-1)^3(\phi+1)\tau_j^2}$$

See Also

arma_to_wv

arma_to_wv *ARMA process to WV*

Description

This function computes the Haar Wavelet Variance of an ARMA process

Usage

```
arma_to_wv(ar, ma, sigma2, tau)
```

Arguments

ar	A vec containing the coefficients of the AR process
ma	A vec containing the coefficients of the MA process
sigma2	A double containing the residual variance
tau	A vec containing the scales e.g. 2^τ

Details

The function is a generic implementation that requires a stationary theoretical autocorrelation function (ACF) and the ability to transform an ARMA(p,q) process into an MA(∞) (e.g. infinite MA process).

Value

A vec containing the wavelet variance of the ARMA process.

Process Haar Wavelet Variance Formula

The Autoregressive Order p and Moving Average Order q (ARMA(p,q)) process has a Haar Wavelet Variance given by:

$$\frac{\tau_j \left[1 - \rho\left(\frac{\tau_j}{2}\right)\right] + 2\sum\limits_{i=1}^{\frac{\tau_j}{2}-1} i\left[2\rho\left(\frac{\tau_j}{2} - i\right) - \rho\left(i\right) - \rho\left(\tau_j - i\right)\right]}{\tau_j^2}\sigma_X^2$$

where σ_X^2 is given by the variance of the ARMA process. Furthermore, this assumes that stationarity has been achieved as it directly

See Also

ARMAtoMA_cpp, ARMAacf_cpp, and arma11_to_wv av_ar1 *Calculate Theoretical Allan Variance for Stationary First-Order Autoregressive (AR1) Process*

Description

This function allows us to calculate the theoretical allan variance for stationary first-order autoregressive (AR1) process.

Usage

```
av_ar1(n, phi, sigma2)
```

Arguments

n	An `integer` value for the size of the cluster.
phi	A `double` value for the autocorrection parameter ϕ.
sigma2	A `double` value for the variance parameter σ^2.

Value

A `double` indicating the theoretical allan variance for AR1 process.

Note

This function is based on the calculation of the theoretical allan variance for stationary AR1 process raised in "Allan Variance of Time Series Models for Measurement Data" by Nien Fan Zhang.) This calculation is fundamental and necessary for the study in "A Study of the Allan Variance for Constant-Mean Non-Stationary Processes" by Xu et al. (IEEE Signal Processing Letters, 2017).

Author(s)

Yuming Zhang

Examples

```
av1 = av_ar1(n = 5, phi = 0.9, sigma2 = 1)
av2 = av_ar1(n = 8, phi = 0.5, sigma2 = 2)
```

av_wn — *Calculate Theoretical Allan Variance for Stationary White Noise Process*

Description

This function allows us to calculate the theoretical allan variance for stationary white noise process.

Usage

```
av_wn(sigma2, n)
```

Arguments

sigma2	A `double` value for the variance parameter σ^2.
n	An `integer` value for the size of the cluster.

Value

A `double` indicating the theoretical allan variance for the white noise process.

Note

This function is based on the calculation of the theoretical allan variance for stationary white noise process raised in "Allan Variance of Time Series Models for Measurement Data" by Nien Fan Zhang. This calculation is fundamental and necessary for the study in "A Study of the Allan Variance for Constant-Mean Non-Stationary Processes" by Xu et al. (IEEE Signal Processing Letters, 2017).

Author(s)

Yuming Zhang

Examples

```
av1 = av_wn(sigma2 = 1, n = 5)
av2 = av_wn(sigma2 = 2, n = 8)
```

compare_wvar — *Comparison Between Multiple Wavelet Variances*

Description

Displays plots of multiple wavelet variances of different time series accounting for CI values.

Usage

```
compare_wvar(
  ...,
  split = FALSE,
  add_legend = TRUE,
  units = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  col_wv = NULL,
  col_ci = NULL,
  nb_ticks_x = NULL,
  nb_ticks_y = NULL,
  legend_position = NULL,
  ci_wv = NULL,
  point_cex = NULL,
  point_pch = NULL,
  names = NULL,
  cex_labels = 0.8
)
```

Arguments

`...`	One or more time series objects.
`split`	A `boolean` that, if TRUE, arranges the plots into a matrix-like format.
`add_legend`	A `boolean` that, if TRUE, adds a legend to the plot.
`units`	A `string` that specifies the units of time plotted on the x axes. Note: This argument will not be used if xlab is specified.
`xlab`	A `string` that gives a title for the x axes.
`ylab`	A `string` that gives a title for the y axes.
`main`	A `string` that gives an overall title for the plot.
`col_wv`	A `string` that specifies the color of the wavelet variance lines.
`col_ci`	A `string` that specifies the color of the confidence interval shade.
`nb_ticks_x`	An `integer` that specifies the maximum number of ticks for the x-axis.
`nb_ticks_y`	An `integer` that specifies the maximum number of ticks for the y-axis.

`legend_position`
: A `string` that specifies the position of the legend (use `legend_position = NA` to remove legend).

`ci_wv`
: A `boolean` that determines whether confidence interval polygons will be drawn.

`point_cex`
: A `double` that specifies the size of each symbol to be plotted.

`point_pch`
: A `double` that specifies the symbol type to be plotted.

`names`
: A `string` that specifies the name of the WVAR objects.

`cex_labels`
: A `double` that specifies the magnification of the labels (x and y).

Author(s)

Stephane Guerrier and Justin Lee

Examples

```
set.seed(999)
n = 10^4
Xt = arima.sim(n = n, list(ar = 0.10))
Yt = arima.sim(n = n, list(ar = 0.35))
Zt = arima.sim(n = n, list(ar = 0.70))
Wt = arima.sim(n = n, list(ar = 0.95))

wv_Xt = wvar(Xt)
wv_Yt = wvar(Yt)
wv_Zt = wvar(Zt)
wv_Wt = wvar(Wt)

compare_wvar(wv_Xt, wv_Yt, wv_Zt, wv_Wt)
```

`compare_wvar_no_split` *Combined Plot Comparison Between Multiple Wavelet Variances*

Description

This is a helper function for the `compare_var()` function. This method accepts the same set of arguments as `compare_wvar` and returns a single plot that compares multiple wavelet variances of different time series accounting for CI values.

Usage

```
compare_wvar_no_split(graph_details)
```

Arguments

`graph_details`
: List of inputs

Author(s)

Stephane Guerrier, Justin Lee, and Nathanael Claussen

`compare_wvar_split` *Multi-Plot Comparison Between Multiple Wavelet Variances*

Description

This is a helper function for the `compare_var()` function. This method accepts the same set of arguments as `compare_wvar` and returns a comparision of multiple wavelet variances of different time series accounting for CI values as a set of different plots.

Usage

```
compare_wvar_split(graph_details)
```

Arguments

`graph_details`	List of inputs

Author(s)

Stephane Guerrier, Justin Lee, and Nathanael Claussen

`dr_to_wv` *Drift to WV*

Description

This function compute the WV (haar) of a Drift process

Usage

```
dr_to_wv(omega, tau)
```

Arguments

`omega`	A `double` corresponding to the slope of the drift
`tau`	A `vec` containing the scales e.g. 2^τ

Value

A `vec` containing the wavelet variance of the drift.

Process Haar Wavelet Variance Formula

The Drift (DR) process has a Haar Wavelet Variance given by:

$$\nu_j^2\left(\omega\right) = \frac{\tau_j^2 \omega^2}{16}$$

`dwt` *Discrete Wavelet Transform*

Description

Calculation of the coefficients for the discrete wavelet transformation

Usage

```
dwt(x, nlevels = floor(log2(length(x))), filter = "haar")
```

Arguments

`x`	A `vector` with dimensions N x 1.
`nlevels`	A `integer` indicating the J levels of decomposition.
`filter`	A `string` indicating the filter name

Details

Performs a level J decomposition of the time series using the pyramid algorithm. The default J is determined by $floor\left(log_2\left(length\left(x\right)\right)\right)$

Value

A `field<vec>` that contains the wavelet coefficients for each decomposition level

Author(s)

James Balamuta, Justin Lee and Stephane Guerrier

Examples

```
set.seed(999)
x = rnorm(2^8)
ret = dwt(x)

summary(ret)

plot(ret)
```

imar_wv — *Wavelet variance of IMU Data from IMAR Gyroscopes*

Description

This data set contains wavelet variance of IMAR gyroscopes data.

Usage

```
imar_wv
```

Format

A list of the following elements:

- "sensor": Name of the sensor.
- "freq": The frequency at which the error signal is measured.
- "n": Sample size of the data.
- "type": The types of sensors considered in the data.
- "axis": The axes of sensors considered in the data.
- "wvar": A list containing the computed wavelet variance based on the data.

Source

The IMU data comes from Geodetic Engineering Laboratory (TOPO) and Swiss Federal Institute of Technology Lausanne (EPFL).

kvh1750_wv — *Wavelet variance of IMU Data from a KVH1750 IMU sensor*

Description

This data set contains wavelet variance of gyroscope and accelerometer data from an KVH1750 sensor.

Usage

```
kvh1750_wv
```

Format

A list of the following elements:

- "sensor": Name of the sensor.
- "freq": The frequency at which the error signal is measured.
- "n": Sample size of the data.
- "type": The types of sensors considered in the data.
- "axis": The axes of sensors considered in the data.
- "wvar": A list containing the computed wavelet variance based on the data.

Source

The IMU data comes from Department of Geomatics Engineering, University of Calgary.

`ln200_wv` *Wavelet variance of IMU Data from a LN200 sensor*

Description

This data set contains wavelet variance of LN200 gyroscope and accelerometer data.

Usage

```
ln200_wv
```

Format

A list of the following elements:

- "sensor": Name of the sensor.
- "freq": The frequency at which the error signal is measured.
- "n": Sample size of the data.
- "type": The types of sensors considered in the data.
- "axis": The axes of sensors considered in the data.
- "wvar": A list containing the computed wavelet variance based on the data.

Source

The IMU data comes from Geodetic Engineering Laboratory (TOPO) and Swiss Federal Institute of Technology Lausanne (EPFL).

ma1_to_wv — *Moving Average Order 1 (MA(1)) to WV*

Description

This function computes the WV (haar) of a Moving Average order 1 (MA1) process.

Usage

```
ma1_to_wv(theta, sigma2, tau)
```

Arguments

theta	A double corresponding to the moving average term.
sigma2	A double the variance of the process.
tau	A vec containing the scales e.g. 2^{τ}

Details

This function is significantly faster than its generalized counter part arma_to_wv.

Value

A vec containing the wavelet variance of the MA(1) process.

Process Haar Wavelet Variance Formula

The Moving Average Order 1 (MA(1)) process has a Haar Wavelet Variance given by:

$$\nu_j^2\left(\theta, \sigma^2\right) = \frac{\left(\left(\theta+1\right)^2\tau_j - 6\theta\right)\sigma^2}{\tau_j^2}$$

See Also

arma_to_wv, arma11_to_wv

`modwt` *Maximum Overlap Discrete Wavelet Transform*

Description

Calculates the coefficients for the discrete wavelet transformation

Usage

```
modwt(x, nlevels = floor(log2(length(x) - 1)), filter = "haar")
```

Arguments

`x`	A `vector` with dimensions N x 1.
`nlevels`	A `integer` indicating the J levels of decomposition.
`filter`	A `string` indicating the filter name

Details

Performs a level J decomposition of the time series using the pyramid algorithm. The default J is determined by $floor\left(log_2\left(length\left(x\right)\right)\right)$

Value

A `field<vec>` that contains the wavelet coefficients for each decomposition level

Author(s)

James Balamuta, Justin Lee and Stephane Guerrier

Examples

```
set.seed(999)
x = rnorm(100)
ret = modwt(x)

summary(ret)

plot(ret)
```

navchip_wv — *Wavelet variance of IMU Data from a navchip sensor*

Description

This data set contains wavelet variance of gyroscope and accelerometer data from a navchip sensor.

Usage

```
navchip_wv
```

Format

A list of the following elements:

- "sensor": Name of the sensor.
- "freq": The frequency at which the error signal is measured.
- "n": Sample size of the data.
- "type": The types of sensors considered in the data.
- "axis": The axes of sensors considered in the data.
- "wvar": A list containing the computed wavelet variance based on the data.

Source

The IMU data of the navchip sensor comes from Geodetic Engineering Laboratory (TOPO) and Swiss Federal Institute of Technology Lausanne (EPFL).

qn_to_wv — *Quantisation Noise (QN) to WV*

Description

This function compute the Haar WV of a Quantisation Noise (QN) process

Usage

```
qn_to_wv(q2, tau)
```

Arguments

q2	A double corresponding to variance of drift
tau	A vec containing the scales e.g. 2^{τ}

Value

A vec containing the wavelet variance of the QN.

Process Haar Wavelet Variance Formula

The Quantization Noise (QN) process has a Haar Wavelet Variance given by:

$$\nu_j^2\left(Q^2\right) = \frac{6Q^2}{\tau_j^2}$$

---

robust_eda   *Comparison between classical and robust Wavelet Variances*

Description

Displays a plot of the wavelet variances (classical and robust) for a given time series accounting for CI values.

Usage

```
robust_eda(
  x,
  eff = 0.6,
  units = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  col_wv = NULL,
  col_ci = NULL,
  nb_ticks_x = NULL,
  nb_ticks_y = NULL,
  legend_position = NULL,
  ...
)
```

Arguments

x	A time series objects.
eff	An integer that specifies the efficiency of the robust estimator.
units	A string that specifies the units of time plotted on the x axis.
xlab	A string that gives a title for the x axis.
ylab	A string that gives a title for the y axis.
main	A string that gives an overall title for the plot.
col_wv	A string that specifies the color of the wavelet variance line.
col_ci	A string that specifies the color of the confidence interval shade.

nb_ticks_x	An integer that specifies the maximum number of ticks for the x-axis.
nb_ticks_y	An integer that specifies the maximum number of ticks for the y-axis.
legend_position	A string that specifies the position of the legend (use legend_position = NA to remove legend).
...	Additional arguments affecting the plot.

Value

Plot of wavelet variance and confidence interval for each scale.

Author(s)

Stephane Guerrier, Nathanael Claussen, and Justin Lee

Examples

```
set.seed(999)
n = 10^4
Xt = rnorm(n)
wv = wvar(Xt)

plot(wv)
plot(wv, main = "Simulated white noise", xlab = "Scales")
plot(wv, units = "sec", legend_position = "topright")
plot(wv, col_wv = "darkred", col_ci = "pink")
```

---

rw_to_wv *Random Walk to WV*

Description

This function compute the WV (haar) of a Random Walk process

Usage

```
rw_to_wv(gamma2, tau)
```

Arguments

gamma2	A double corresponding to variance of RW
tau	A vec containing the scales e.g. 2^{τ}

Value

A vec containing the wavelet variance of the random walk.

Process Haar Wavelet Variance Formula

The Random Walk (RW) process has a Haar Wavelet Variance given by:

$$\nu_j^2\left(\gamma^2\right) = \frac{\left(\tau_j^2 + 2\right)\gamma^2}{12\tau_j}$$

`sarma_objdesc` *Create the ts.model obj.desc given split values*

Description

Computes the total phi and total theta vector length.

Usage

```
sarma_objdesc(ar, ma, sar, sma, s, i, si)
```

Arguments

`ar`	A `vec` containing the non-seasonal phi parameters.
`ma`	A `vec` containing the non-seasonal theta parameters.
`sar`	A `vec` containing the seasonal phi parameters.
`sma`	A `vec` containing the seasonal theta parameters.
`s`	An `unsigned integer` containing the frequency of seasonality.
`i`	An `unsigned integer` containing the number of non-seasonal differences.
`si`	An `unsigned integer` containing the number of seasonal differences.

Value

A `vec` with rows:

np Number of Non-Seasonal AR Terms

nq Number of Non-Seasonal MA Terms

nsp Number of Seasonal AR Terms

nsq Number of Seasonal MA Terms

nsigma Number of Variances (always 1)

s Season Value

i Number of non-seasonal differences

si Number of Seasonal Differences

`sp_hfilter` *Haar filter for a spatial case*

Description

Haar filter for a spatial case

Usage

```
sp_hfilter(jscale)
```

Arguments

`jscale`	An `int` of the Number of Scales

`sp_modwt_cpp` *Compute the Spatial Wavelet Coefficients*

Description

Compute the Spatial Wavelet Coefficients

Usage

```
sp_modwt_cpp(X, J1, J2)
```

Arguments

`X`	is a matrix with row, col orientation
`J1, J2`	is the levels of decomposition along the rows, columns

Details

By default this function will return the wavelet coefficient in addition to the wavelet

Value

A `list` of `vectors` containing the wavelet coefficients.

wccv — *Cross Covariance of Matrix*

Description

Calculates the Cross-covariance between multiple wavelet transfomations (dwt or modwt)

Usage

```
wccv(x, decomp = "modwt", filter = "haar", nlevels = NULL)
```

Arguments

`x`	A `vector` with dimensions N x M.
`decomp`	A `string` that indicates whether to use the "dwt" or "modwt" decomposition.
`filter`	A `string` that specifies what wavelet filter to use.
`nlevels`	An `integer` that indicates the level of decomposition. It must be less than or equal to floor(log2(length(x))).

Details

If `nlevels` is not specified, it is set to $\lfloor \log_2 \left(length \left(x \right) \right) \rfloor$

Value

Returns a `matrix` of `lists` of all the possible pair cross-covariance, variance of each wavelet cross-covariance and its 95

Author(s)

Justin Lee

wccv_get_y — *Mapping to log10 scale*

Description

Map x to the value in log10 scale

Usage

```
wccv_get_y(x, tick_y_min, tick_y_step)
```

Arguments

x	A vector with dimensions J x 1.
tick_y_min	A negtive integer the minimum power of 10, which corresponds to the smallest scale on y-axis.
tick_y_step	An integer indicating the increment of the sequence.

Details

tick_y_min is usually chosen as $floor(min(log10(abs(x))))$

Value

A field<vec> that contains values in log10 scale.

Author(s)

James Balamuta and Justin Lee

Examples

```
x = 2^(-1:-9)
y.min = floor(min(log10(abs(x))))
y.step = 2
wccv_get_y(x, y.min, y.step)
```

---

wccv_pair *Cross Covariance of a TS Pair*

Description

Calculates the Cross-covariance between two wavelet transfomations (dwt or modwt)

Usage

```
wccv_pair(x, y, decomp = "modwt", filter = "haar", nlevels = NULL)
```

Arguments

x	A vector with dimensions N x 1.
y	A vector with dimensions N x 1.
decomp	A string that indicates whether to use the "dwt" or "modwt" decomposition.
filter	A string that specifies what wavelet filter to use.
nlevels	An integer that indicates the level of decomposition. It must be less than or equal to floor(log2(length(x))).

Details

If `nlevels` is not specified, it is set to $\lfloor \log_2 \left(length \left(x \right) \right) \rfloor$

Value

Returns a `list` of a `matrix` containing cross-covariance, variance of each wavelet cross-covariance and its 95

Author(s)

Justin Lee

`wn_to_wv` *Gaussian White Noise to WV*

Description

This function compute the Haar WV of a Gaussian White Noise process

Usage

```
wn_to_wv(sigma2, tau)
```

Arguments

`sigma2`	A `double` corresponding to variance of WN
`tau`	A `vec` containing the scales e.g. 2^τ

Value

A `vec` containing the wavelet variance of the white noise.

Process Haar Wavelet Variance Formula

The Gaussian White Noise (WN) process has a Haar Wavelet Variance given by:

$$\nu_j^2 \left(\sigma^2 \right) = \frac{\sigma^2}{\tau_j^2}$$

wvar *Wavelet Variance*

Description

Calculates the (MO)DWT wavelet variance

Usage

```
wvar(x, ...)

## S3 method for class 'lts'
wvar(
  x,
  decomp = "modwt",
  filter = "haar",
  nlevels = NULL,
  alpha = 0.05,
  robust = FALSE,
  eff = 0.6,
  to.unit = NULL,
  ...
)

## S3 method for class 'gts'
wvar(
  x,
  decomp = "modwt",
  filter = "haar",
  nlevels = NULL,
  alpha = 0.05,
  robust = FALSE,
  eff = 0.6,
  to.unit = NULL,
  ...
)

## S3 method for class 'ts'
wvar(
  x,
  decomp = "modwt",
  filter = "haar",
  nlevels = NULL,
  alpha = 0.05,
  robust = FALSE,
  eff = 0.6,
  to.unit = NULL,
```

```
  ...
)

## S3 method for class 'imu'
wvar(
  x,
  decomp = "modwt",
  filter = "haar",
  nlevels = NULL,
  alpha = 0.05,
  robust = FALSE,
  eff = 0.6,
  to.unit = NULL,
  ...
)

## Default S3 method:
wvar(
  x,
  decomp = "modwt",
  filter = "haar",
  nlevels = NULL,
  alpha = 0.05,
  robust = FALSE,
  eff = 0.6,
  freq = 1,
  from.unit = NULL,
  to.unit = NULL,
  ...
)
```

Arguments

`x`	A `vector` with dimensions N x 1.
`...`	Further arguments passed to or from other methods.
`decomp`	A `string` that indicates whether to use a "dwt" or "modwt" decomposition.
`filter`	A `string` that specifies which wavelet filter to use.
`nlevels`	An `integer` that indicates the level of decomposition. It must be less than or equal to floor(log2(length(x))).
`alpha`	A `double` that specifies the significance level which in turn specifies the $1-\alpha$ confidence level.
`robust`	A `boolean` that triggers the use of the robust estimate.
`eff`	A `double` that indicates the efficiency as it relates to an MLE.
`to.unit`	A `string` indicating the unit to which the data is converted.
`freq`	A `numeric` that provides the rate of samples.
`from.unit`	A `string` indicating the unit from which the data is converted.

Details

The default value of `nlevels` will be set to $\lfloor \log_2 \left(length\left(x \right) \right) \rfloor$, unless otherwise specified.

Value

A `list` with the structure:

- "variance": Wavelet Variance
- "ci_low": Lower CI
- "ci_high": Upper CI
- "robust": Robust active
- "eff": Efficiency level for Robust calculation
- "alpha": p value used for CI
- "unit": String representation of the unit

Author(s)

James Balamuta, Justin Lee and Stephane Guerrier

Examples

```
set.seed(999)
x = rnorm(100)

# Default
wvar(x)

# Robust
wvar(x, robust = TRUE, eff=0.3)

# Classical
wvar(x, robust = FALSE, eff=0.3)

# 90% Confidence Interval
wvar(x, alpha = 0.10)
```

Index