Package ‘WaveletComp’

February 19, 2015

Type Package

Title Computational Wavelet Analysis

Version 1.0

Date 2014-12-15

Author Angi Roesch and Harald Schmidbauer

Maintainer Angi Roesch <angi@angi-stat.com>

Description Wavelet analysis and reconstruction of time series, cross-wavelets and phase-difference (with filtering options), significance with simulation algorithms.

Depends R (>= 2.10)

License GPL-2

URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-16 09:26:23

R topics documented:

`WaveletComp-package` *Computational Wavelet Analysis*

Description

Wavelet analysis and reconstruction of time series, cross-wavelets and phase-difference (with filtering options), significance with bootstrap algorithms.

Details

Package:	WaveletComp
Type:	Package
Version:	1.0
Date:	2014-12-15
License:	GPL-2
URL:	http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf

Periodic phenomena of a single time series can be analyzed with function `analyze.wavelet`. Results of the analysis (a time/period image of the wavelet power spectrum, plots of the average power, and phase plots for selected periods and a time/period image of phases) can be accessed through various plot functions (`wt.image`, `wt.avg`, `wt.sel.phases`, `wt.phase.image`). Function `reconstruct` returns the reconstructed time series where reconstruction is according to constraints on significance, period specification, and cone of influence.

The cross-wavelet spectrum and coherency spectrum of two time series can be analyzed with function `analyze.coherency`. Results (a time/period image of cross-wavelet power or coherency, plots of averages, plots of phases and phase-differences for selected periods and the time/period image of phase-differences) can be accessed through corresponding functions (`wc.image`, `wc.avg`, `wc.sel.phases`, `wc.phasediff.image`).

Detrending of the time series at hand is offered as an option. Wavelet transformations are computed using the Morlet wavelet. Smoothing filters are provided in the case of cross-wavelet transformation to compute wavelet coherency.

Significance is assessed with simulation algorithms, a variety of alternative hypotheses to test is available, for which surrogate time series are provided: white noise, shuffling the given time series, time series with a similar spectrum, AR, and ARIMA.

Names and parts of the layout of some routines were inspired by similar functions developed by Huidong Tian and Bernard Cazelles (archived R package `WaveletCo`). The basic concept of the simulation algorithm, and of ridge determination build on ideas developed by these authors. The major part of the code for the computation of the cone of influence, and the code for Fourier-randomized surrogate time series has been adopted from Huidong Tian. The implementation of a choice of filtering windows for the computation of the wavelet coherence was inspired by Luis Aguiar-Conraria and Maria Joana Soares (`GWPackage`).

Cross-wavelet and coherence computation, the simulation algorithm, and ridge determination build heavily on the use of matrices in order to minimize computation time in R.

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian, Bernard Cazelles, Luis Aguiar-Conraria, and Maria Joana Soares.

References

Aguiar-Conraria L., and Soares M.J., 2011. Business cycle synchronization and the Euro: A wavelet analysis. Journal of Macroeconomics 33 (3), 477–489.

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Aguiar-Conraria L., and Soares M.J., 2012. `GWPackage`. Available at `http://sites.google.com/site/aguiarconraria/joanasoares-wavelets`; accessed September 4, 2013.

Carmona R., Hwang W.-L., and Torresani B., 1998. Practical Time Frequency Analysis. Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu P.C., 1994. Wavelet spectrum analysis and ocean wind waves. In: Foufoula-Georgiou E., and Kumar P., (eds.), Wavelets in Geophysics, Academic Press, San Diego, 151–166.

Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. Journal of Atmospheric and Oceanic Technology 24, 2093–2102.

Tian, H., and Cazelles, B., 2012. `WaveletCo`. Available at `http://cran.r-project.org/src/contrib/Archive/WaveletCo/`, archived April 2013; accessed July 26, 2013.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

Veleda D., Montagne R., and Araujo M., 2012. Cross-Wavelet Bias Corrected by Normalizing Scales. Journal of Atmospheric and Oceanic Technology 29, 1401–1408.

`analyze.coherency` *Computation of the cross-wavelet power and wavelet coherence spectrum of two time series*

Description

The two time series are selected from an input data frame by specifying either their names or their column numbers. Optionally, the time series are detrended, using `loess` with parameter `loess.span`. Internally, the series will be standardized before they undergo wavelet transformation.

The cross-wavelet power spectrum is computed applying the Morlet wavelet. P-values to test the null hypothesis that a period (within lowerPeriod and upperPeriod) is irrelevant at a certain time are calculated if desired; this is accomplished with the help of a simulation algorithm. There is a selection of models from which to choose the alternative hypothesis. The selected model will be fitted to the data and simulated according to estimated parameters in order to provide surrogate time series.

For the computation of wavelet coherence, a variety of filtering methods is provided, with flexible window parameters.

Wavelet transformation, as well as p-value computations, are carried out by calling subroutine `wc`.

The name and parts of the layout of subroutine `wc` were inspired by a similar function developed by Huidong Tian and Bernard Cazelles (archived R package `WaveletCo`). The basic concept of the simulation algorithm, and of ridge determination build on ideas developed by these authors. The major part of the code for the computation of the cone of influence, and the code for Fourier-randomized surrogate time series has been adopted from Huidong Tian. The implementation of a choice of filtering windows for the computation of the wavelet coherence was inspired by Luis Aguiar-Conraria and Maria Joana Soares (`GWPackage`).

Cross-wavelet and coherence computation, the simulation algorithm, and ridge determination build heavily on the use of matrices in order to minimize computation time in R.

This function provides a broad variety of final as well as intermediate results which can be further analyzed in detail.

Usage

```
analyze.coherency(my.data, my.pair = c(1, 2), loess.span = 0.75, dt = 1, dj = 1/20,
                  lowerPeriod = 2*dt, upperPeriod = floor(nrow(my.data)/3)*dt,
                  window.type.t = 1, window.type.s = 1,
                  window.size.t = 5, window.size.s = 1/4,
                  make.pval = T, method = "white.noise", params = NULL,
                  n.sim = 100, verbose = T)
```

Arguments

`my.data` data frame of time series (including header, and dates as row names or as separate column named "date" if available)

`my.pair` pair of names or column indices indicating series x and y to be analyzed, e.g. c(1,2), c(2,1), c("dji","ftse").
Default: c(1,2).

`loess.span` parameter alpha in loess controlling the degree of time series smoothing, if the time series is to be detrended; no detrending if loess.span=0.
Default: 0.75.

`dt` time resolution, i.e. sampling resolution on time domain, 1/dt = number of intervals per time step. Default: 1.

`dj` frequency resolution, i.e. sampling resolution on frequency domain, 1/dj = number of suboctaves (voices per octave). Default: 1/20.

`lowerPeriod` lower Fourier period (in time units) for wavelet decomposition. Default: 2*dt.

`upperPeriod` upper Fourier period (in time units) for wavelet decomposition.
Default: (floor of one third of time series length)*dt.

`window.type.t` type of window for smoothing in time direction, select from:

0	("none")	:	no smoothing in time direction
1	("bar")	:	Bartlett

2	("tri")	:	Triangular (Non-Bartlett)
3	("box")	:	Boxcar (Rectangular, Dirichlet)
4	("han")	:	Hanning
5	("ham")	:	Hamming
6	("bla")	:	Blackman

Default: 1 = "bar".

`window.type.s` type of window for smoothing in scale (period) direction, select from:

0	("none")	:	no smoothing in scale (period) direction
1	("bar")	:	Bartlett
2	("tri")	:	Triangular (Non-Bartlett)
3	("box")	:	Boxcar (Rectangular, Dirichlet)
4	("han")	:	Hanning
5	("ham")	:	Hamming
6	("bla")	:	Blackman

Default: 1 = "bar".

`window.size.t` size of the window used for smoothing in time direction in units of 1/dt. Default: 5, which together with dt=1 defines a window of length 5*(1/dt) = 5. Windows of even-numbered sizes are extended by 1.

`window.size.s` size of the window used for smoothing in scale direction in units of 1/dj. Default: 1/4, which together with dj=1/20 defines a window of length (1/4)*(1/dj) = 5. Windows of even-numbered sizes are extended by 1.

`make.pval` Compute p-values? Logical. Default: TRUE.

`method` the method of generating surrogate time series, select from:

"white.noise"	:	white noise
"shuffle"	:	shuffling the given time series
"Fourier.rand"	:	time series with a similar spectrum
"AR"	:	AR(p)
"ARIMA"	:	ARIMA(p,0,q)

Default: "white.noise"

`params` a list of assignments between methods (AR, and ARIMA) and lists of parameter values applying to surrogates. Default: NULL.

Default which includes:

AR: AR = list(p=1), where:

p	:	AR order

ARIMA: ARIMA = list(p=1, q=1, include.mean=T, sd.fac=1, trim = F, trim.prop = 0.01), where:

p	:	AR order
q	:	MA order

include.mean	:	Include a mean/intercept term?
sd.fac	:	magnification factor to boost the residual standard deviation
trim	:	Simulate trimmed data?
trim.prop	:	high/low trimming proportion

`n.sim` number of simulations. Default: 100.

`verbose` Print verbose output on the screen? Logical. Default: TRUE.

Value

A list of class `analyze.coherency` with the following elements:

`series` a data frame with the following columns

date	:	the calendar date (as given in `my.data`)
<x>, <y>	:	the two series which have been analyzed
	:	(detrended, if loess.span != 0; original names retained)
<x>.trend, <y>.trend	:	the two trend series (included if loess.span != 0)

Row names are resumed from `my.data`, and so are dates which were given as rownames.

`loess.span` parameter alpha in loess controlling the degree of time series smoothing if the time series were detrended; no detrending if loess.span=0.

`dt` time resolution, i.e. sampling resolution on time domain, 1/dt = number of intervals per time step.

`dj` frequency resolution, i.e. sampling resolution on frequency domain, 1/dj = number of suboctaves (voices per octave).

`Wave.xy` (complex-valued) cross-wavelet transform (analogous to Fourier cross-frequency spectrum, and to the covariance in statistics)

`Angle` phase difference, i.e. phase lead of <x> over <y> (= phase.x-phase.y)

`sWave.xy` smoothed (complex-valued) cross-wavelet transform

`sAngle` phase difference, i.e. phase lead of <x> over <y>, affected by smoothing

`Power.xy` cross-wavelet power (analogous to Fourier cross-frequency power spectrum)

`Power.xy.avg` average cross-wavelet power in the frequency domain (averages over time)

`Power.xy.pval` p-values of cross-wavelet power

`Power.xy.avg.pval`
p-values of average cross-wavelet power

`Coherency` the (complex-valued) wavelet coherency of series <x> over series <y> in the time/frequency domain, affected by smoothing (analogous to Fourier coherency, and to the coefficient of correlation in statistics)

`Coherence` wavelet coherence (analogous to Fourier coherence, and to the coefficient of determination in statistics (affected by smoothing)

`Coherence.avg` average wavelet coherence in the frequency domain (averages across time)

`Coherence.pval` p-values of wavelet coherence

`Coherence.avg.pval`
p-values of average wavelet coherence

`Wave.x, Wave.y` (complex-valued) wavelet transforms of series <x> and <y>

`Phase.x, Phase.y`
phases of series <x> and <y>

`Ampl.x, Ampl.y` amplitudes of series <x> and <y>

`Power.x, Power.y`
wavelet power of series <x> and <y>

`Power.x.avg, Power.y.avg`
average wavelet power of series <x> and <y>, averages across time

`Power.x.pval, Power.y.pval`
p-values of wavelet power of series <x> and <y>

`Power.x.avg.pval, Power.y.avg.pval`
p-values of average wavelet power of series <x> and <y>

`sPower.x, sPower.y`
smoothed wavelet power of series <x> and <y>

`Ridge.xy` ridge of cross-wavelet power, in the form of a 0-1 matrix: columns correspond to `dt` steps, rows correspond to `dj` steps whose numerical values are given in `Period`

`Ridge.co` ridge of wavelet coherence

`Ridge.x, Ridge.y`
power ridges of series <x> and <y>

`Period` the Fourier periods (in time units)

`Scale` the scales

`nc` number of columns/time steps

`nr` number of rows/scales/Fourier periods

`coi.1, coi.2` borders of the region where the wavelet transforms are not influenced by edge effects (cone of influence)

`axis.1` tick levels corresponding to time steps

`axis.2` tick levels corresponding to Fourier periods (= log2(Period))

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian, Bernard Cazelles, Luis Aguiar-Conraria, and Maria Joana Soares.

References

Aguiar-Conraria L., and Soares M.J., 2011. Business cycle synchronization and the Euro: A wavelet analysis. Journal of Macroeconomics 33 (3), 477–489.

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Aguiar-Conraria L., and Soares M.J., 2012. `GWPackage`. Available at http://sites.google.com/site/aguiarconraria/joanasoares-wavelets; accessed September 4, 2013.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu P.C., 1994. Wavelet spectrum analysis and ocean wind waves. In: Foufoula-Georgiou E., and Kumar P., (eds.), Wavelets in Geophysics, Academic Press, San Diego, 151–166.

Tian, H., and Cazelles, B., 2012. `WaveletCo`. Available at http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

Veleda D., Montagne R., and Araujo M., 2012. Cross-Wavelet Bias Corrected by Normalizing Scales. Journal of Atmospheric and Oceanic Technology 29, 1401–1408.

See Also

`wc.avg`, `wc.image`, `wc.sel.phases`, `wc.phasediff.image`

Examples

```
## Not run:
## The following example is adopted from Veleda et al, 2012:

add.noise=TRUE

series.length = 3*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 2*24, length = series.length)
x3 = periodic.series(start.period = 4*24, length = series.length)
x4 = periodic.series(start.period = 8*24, length = series.length)
x5 = periodic.series(start.period = 16*24, length = series.length)
x6 = periodic.series(start.period = 32*24, length = series.length)
x7 = periodic.series(start.period = 64*24, length = series.length)
x8 = periodic.series(start.period = 128*24, length = series.length)

x = x1 + x2 + x3 + x4 + 3*x5 + x6 + x7 + x8
y = x1 + x2 + x3 + x4 + 3*x5 + x6 + 3*x7 + x8

if (add.noise == TRUE){
    x = x + rnorm(length(x))
    y = y + rnorm(length(y))
}

my.data = data.frame(x=x, y=y)
```

```
ts.plot(ts(my.data$x, start=0, frequency=24),
        ts(my.data$y, start=0, frequency=24),
        type="l", col=1:2,
        xlab="time (days)", ylab="hourly data",
       main="a series of hourly data with periods of 1, 2, 4, 8, 16, 32, 64, and 128 days",
        sub="(different amplitudes at periods 16 and 64)")
legend("topright", legend=c("x","y"), col=1:2, lty=1)

## computation of cross-wavelet power and wavelet coherence:
my.wc = analyze.coherency(my.data, c("x","y"), loess.span=0,
                          dt=1/24, dj=1/20,
                          window.size.t=1, window.size.s=1/2,
                          lowerPeriod=1/4,
                          make.pval=T, n.sim=10)

## plot of cross-wavelet power (with color breakpoints according to quantiles):
wc.image(my.wc, timelab="time (days)", periodlab="period (days)",
         main="cross-wavelet power",
         legend.params=list(lab="cross-wavelet power levels"))
## plot of average cross-wavelet power:
wc.avg(my.wc)

## plot of wavelet coherence (with color breakpoints according to quantiles):
wc.image(my.wc, which.image="wc", timelab="time (days)", periodlab="period (days)",
         main="wavelet coherence",
       legend.params=list(lab="wavelet coherence levels", lab.line=3.5, label.digits=3))
## plot of average coherence:
wc.avg(my.wc, which.avg="wc", legend.coords="topleft")

## End(Not run)
```

`analyze.wavelet` *Computation of the wavelet power spectrum of a single time series*

Description

The time series is selected from an input data frame by specifying either its name or its column number. Optionally, the time series is detrended, using `loess` with parameter `loess.span`. Internally, the series will be further standardized before it undergoes wavelet transformation.

The wavelet power spectrum is computed by applying the Morlet wavelet. P-values to test the null hypothesis that a period (within lowerPeriod and upperPeriod) is irrelevant at a certain time are calculated if desired; this is accomplished with the help of a simulation algorithm. There is a selection of models from which to choose the alternative hypothesis. The selected model will be fitted to the data and simulated according to estimated parameters in order to provide surrogate time series.

Wavelet transformation, as well as p-value computations, are carried out by calling subroutine `wt`.

The name and parts of the layout of subroutine `wt` were inspired by a similar function developed by Huidong Tian and Bernard Cazelles (archived R package `WaveletCo`). The basic concept of the simulation algorithm, and of ridge determination build on ideas developed by these authors. The major part of the code for the computation of the cone of influence, and the code for Fourier-randomized surrogate time series has been adopted from Huidong Tian.

Wavelet computation, the simulation algorithm, and ridge determination build heavily on the use of matrices in order to minimize computation time in R.

This function provides a broad variety of final as well as intermediate results which can be further analyzed in detail.

Usage

```
analyze.wavelet(my.data, my.series = 1, loess.span = 0.75, dt = 1, dj = 1/20,
                lowerPeriod = 2*dt, upperPeriod = floor(nrow(my.data)/3)*dt,
                make.pval = T, method = "white.noise", params = NULL,
                n.sim = 100, verbose = T)
```

Arguments

`my.data` data frame of time series (including header, and dates as rownames or as separate column named "date" if available)

`my.series` name or column index indicating the series to be analyzed, e.g. 1, 2, "dji", "ftse". Default: 1

`loess.span` parameter alpha in loess controlling the degree of time series smoothing, if the time series is to be be detrended; no detrending if loess.span=0. Default: 0.75

`dt` time resolution, i.e. sampling resolution on time domain, 1/dt = number of intervals per time unit. Default: 1.

`dj` frequency resolution, i.e. sampling resolution on frequency domain, 1/dj = number of suboctaves (voices per octave). Default: 1/20.

`lowerPeriod` lower Fourier period (in time units) for wavelet decomposition. Default: 2*dt.

`upperPeriod` upper Fourier period (in time units) for wavelet decomposition. Default: (floor of one third of time series length)*dt

`make.pval` Compute p-values? Logical. Default: TRUE

`method` the method of generating surrogate time series, select from:

"white.noise"	:	white noise
"shuffle"	:	shuffling the given time series
"Fourier.rand"	:	time series with a similar spectrum
"AR"	:	AR(p)
"ARIMA"	:	ARIMA(p,0,q)

Default: "white.noise"

`params` a list of assignments between methods (AR, and ARIMA) and lists of parameter values to apply to surrogates. Default: NULL.

Default includes:

AR: AR = list(p=1), where:

p	:	AR order

ARIMA: ARIMA = list(p=1, q=1, include.mean=T, sd.fac=1, trim = F, trim.prop = 0.01), where:

p	:	AR order
q	:	MA order
include.mean	:	Include a mean/intercept term?
sd.fac	:	magnification factor to boost the residual standard deviation
trim	:	Simulate trimmed data?
trim.prop	:	high/low trimming proportion

`n.sim` number of simulations. Default: 100

`verbose` Print verbose output on the screen? Logical. Default: TRUE

Details

Wavelet transformation, as well as p-value computations, are carried out by calling the internal function `wt`.

Value

A list of class `analyze.wavelet` with the following elements:

`series` a data frame with the following columns

date	:	the calendar date (as given in `my.data`)
<x>	:	the series which has been analyzed
	:	(detrended, if loess.span != 0; original names retained)
<x>.trend	:	the trend series (if loess.span != 0)

Row names are taken over from my.data, and so are dates if given as row names.

`loess.span` parameter alpha in loess which controlled the degree of time series smoothing, if the time series was detrended; detrending was omitted if loess.span=0.

`dt` time resolution, i.e. sampling resolution on time domain, 1/dt = number of intervals per time step.

`dj` frequency resolution, i.e. sampling resolution on frequency domain, 1/dj = number of suboctaves (voices per octave).

`Wave` complex wavelet transform of the series

`Phase` phases

`Ampl` amplitudes

`Power` wavelet power in the time/frequency domain

`Power.avg` average wavelet power in the frequency domain (averages over time)

`Power.pval` p-values of wavelet power

`Power.avg.pval` p-values of average wavelet power

Ridge	power ridge, in the form of a 0-1 matrix: columns correspond to dt steps, rows correspond to dj steps whose numerical values are given in Period
Period	the Fourier periods (in time units)
Scale	the scales
nc	number of columns/time steps
nr	number of rows/scales/Fourier periods
coi.1, coi.2	borders of the region where the wavelet transforms are not influenced by edge effects (cone of influence)
axis.1	tick levels corresponding to time steps
axis.2	tick levels corresponding to Fourier periods (= log2(Period))

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian, and Bernard Cazelles.

References

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Carmona R., Hwang W.-L., and Torresani B., 1998. Practical Time Frequency Analysis. Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. Journal of Atmospheric and Oceanic Technology 24, 2093–2102.

Tian, H., and Cazelles, B., 2012. WaveletCo. Available at http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

See Also

wt.avg, wt.image, wt.sel.phases, wt.phase.image, reconstruct

Examples

```
## Not run:
## The following example is adopted from Liu et al, 2007:

series.length = 6*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 8*24, length = series.length)
x3 = periodic.series(start.period = 32*24, length = series.length)
x4 = periodic.series(start.period = 128*24, length = series.length)
x = x1 + x2 + x3 + x4
```

```
plot(ts(x, start=0, frequency=24), type="l",
  xlab="time (days)",
  ylab="hourly data", main="a series of hourly data with periods of 1, 8, 32, and 128 days")

my.data = data.frame(x=x)

my.wt = analyze.wavelet(my.data, "x",
                        loess.span=0,
                        dt=1/24, dj=1/20,
                        lowerPeriod=1/4,
                        make.pval=T, n.sim=10)

## Plot of wavelet power spectrum (with equidistant color breakpoints):
wt.image(my.wt, color.key="interval", legend.params=list(lab="wavelet power levels"))
## Plot of average wavelet power:
wt.avg(my.wt, siglvl=0.05, sigcol="red")

## End(Not run)
```

`FXtrade.transactions` *Series of FX trade transactions*

Description

USD/euro FX (foreign exchange) trade: number of worldwide transactions recorded per 5-minute intervals in July 2010. The data set contains four full weekly cycles (plus three days at the beginning of July 2010), a weekly cycle lasting from Sunday, 21:00, to Friday, 20:55. The number of transactions between Friday, 21:00, and Sunday, 20:55, is 0 or close to 0. For these intervals, variable 'active' is FALSE, otherwise TRUE.

Derived from data delivered by Morning Star.

Usage

```
data("FXtrade.transactions")
```

Format

A data frame of two columns:

date	:	date and GMT time (resolution: 5 minutes), format: "%Y-%m-%d %H:%M:%S" (equivalently, "%F %T")
transactions	:	number of transactions in the 5-minute interval starting with the time indicated
active	:	trade activity indicator

Source

Morning Star, http://www.morningstar.com/

Examples

```
data(FXtrade.transactions)
plot(as.POSIXct(FXtrade.transactions$date, format = "%F %T"),
     FXtrade.transactions$transactions, type = "l",
     xlab = "day", ylab = "transactions in 5-minute intervals")
```

---

marriages.Turkey *Series of monthly marriages in Turkey*

Description

Series of monthly marriages in Turkey (1988-2013), as reported by DIE (Devlet Istatistik Enstitusu) / TUIK (Turkiye Istatistik Kurumu).

Usage

```
data("marriages.Turkey")
```

Format

A data frame of three columns:

date	:	end-of-month date format: "%Y-%m-%d"
n.Sun	:	number of Sundays in this month
marriages	:	number of marriages in this month

Source

DIE (Devlet Istatistik Enstitusu) / TUIK (Turkiye Istatistik Kurumu)

Jan 1988 to Dec 2000:
"Evlenme istatistikleri", DIE (Devlet Istatistik Enstitusu, Ankara), ISSN: 1300-1086; several issues.

Jan 2001 to Dec 2013:
http://www.tuik.gov.tr/VeriTabanlari.do?vt_id=21&ust_id=109; accessed Oct 1, 2014.

Examples

```
data(marriages.Turkey)
plot(ts(marriages.Turkey$marriages, frequency=12, start=c(1988,1)), type="l",
     xlab="", ylab="",
     main="monthly marriages in Turkey (1988-2013)")
```

`periodic.series` *Computation of a (deterministic) periodic time series of linearly changing period.*

Description

It computes and returns a sinusoid of a specified length, which has the given initial phase, and linearly changing periods (if requested) starting from a given period length through the given length at the end. There is an option to plot the time series.

Usage

```
periodic.series(start.period = 100, end.period = start.period,
                phase = 0,
                length = 600,
                make.plot = F)
```

Arguments

`start.period`	period length at start (in steps of time). Default: 100
`end.period`	period length at end (in steps of time). Default: 50
`phase`	phase difference (in steps of time), i.e. part of period length which has elapsed relative to the origin. Default: 0
`length`	number of time steps. Default: 600
`make.plot`	Plot time series? Logical. Default: FALSE

Details

This function can be used for illustrating methods and functions. Producing a sinusoid, periodic.series will work best if start.period (and end.period, if different from start.period) are not too small.

Value

the series as vector

Author(s)

Angi Roesch and Harald Schmidbauer

See Also

`analyze.wavelet`, `wt.image`, `reconstruct`

Examples

```
# The following time series involves periods from 100 through 50:
x = periodic.series(start.period = 100, end.period = 50, make.plot = TRUE)
title("time series with linearly changing period from 100 through 50")

# The following three time series involve three different types of period evolution
# starting from period 100:
x1 = 0.8*periodic.series(start.period = 100, end.period = 95, phase = 0, length = 1000)
x2 =     periodic.series(start.period = 100, end.period = 100, phase = 0, length = 1000)
x3 = 1.2*periodic.series(start.period = 100, end.period = 105, phase = 0, length = 1000)

ts.plot(x2, ylim = c(-2, +2), xlab = "time", ylab = "series with variable period")
lines(x1, col = "blue")
lines(x3, col = "red")
legend("topleft", legend = c("speeding up (end period = 95)", "period = 100",
   "slowing down (end period = 105)"),
   lty = 1, col = c("blue", "black", "red"))
```

`reconstruct` *Reconstruction of a (detrended) time series from output provided by an object of class "analyze.wavelet" or "analyze.coherency"*

Description

This function reconstructs a (detrended) time series analyzed by wavelet transformation using either function `analyze.wavelet` or function `analyze.coherency`, subject to optional criteria concerning: minimum wavelet power, significance of wavelet power at a given significance level, specification of (Fourier) periods or period bands, exclusive use of the power ridge and/or the cone of influence. An option is provided to prevent the reconstructed series from final rescaling (applying the original (detrended) series' mean and standard deviation).

(If the object provided as input is of class `analyze.coherency`, then the number or name of the time series can be specified.)

Optional: plot of wavelets used for reconstruction, plot of reconstructed series against original (detrended) series.

Output includes the original (detrended) and the reconstructed time series, along with reconstruction waves and parameters.

Usage

```
reconstruct(WT, my.series = 1, lvl = 0,
      only.coi = F,
      only.sig = T, siglvl = 0.05,
      only.ridge = F,
      sel.period = NULL, sel.lower = NULL, sel.upper = NULL,
      rescale = T,
      plot.waves = F, plot.rec = T,
      lty = 1, lwd = 1, col = 1:2, ylim = NULL,
```

```
show.legend = T,
legend.coords = "topleft", legend.horiz=F, legend.text = NULL,
label.time.axis = T, show.date = F, date.format = NULL, timelab = NULL,
main.waves = NULL, main.rec = NULL, main = NULL, verbose = T)
```

Arguments

`WT`	an object of class `analyze.wavelet` or `analyze.coherency`
`my.series`	In case class(WT) = `analyze.coherency`: number (1 or 2) or name of the series to be analyzed. Default: 1.
`lvl`	minimum level of wavelet power applied for inclusion of reconstruction waves. Default: 0.
`only.coi`	Exclude borders influenced by edge effects, i.e. include the cone of influence only? Logical. Default: FALSE.
`only.sig`	Use wavelet power significance at all? Logical. Default: TRUE.
`siglvl`	level of wavelet power significance applied for inclusion of reconstruction waves. Default: 0.05.
`only.ridge`	Select ridge only? Logical. Default: FALSE.
`sel.period`	a vector of numbers to select Fourier periods (or closest available periods) and corresponding wavelets for the reconstruction. Default: NULL.
`sel.lower`	a number to define a lower Fourier period (or the closest available) for the selection of a band of wavelets for the reconstruction (only effective if sel.period is NULL). Default: NULL.
`sel.upper`	a number to define an upper Fourier period (or the closest available) for the selection of a band of wavelets for the reconstruction (only effective if sel.period is NULL). Default: NULL.
`rescale`	Should the reconstructed series finally be rescaled to attain the original (detrended) series' mean and standard deviation? Logical. Default: TRUE.
`plot.waves`	Should reconstruction waves be plotted? Logical. Default: FALSE.
`plot.rec`	Should the reconstructed series (together with the original (detrended) series) be plotted? Logical. Default: TRUE.
`lty`	parameter for the plot of original vs. reconstructed series: line type, e.g. 1:2. Default: 1.
`lwd`	parameter for the plot of original vs. reconstructed series: line width, e.g. 1:2. Default: 1.
`col`	parameter for the plot of original vs. reconstructed series: color of lines. Default: 1:2.
`ylim`	numeric vector of length 2, giving the time series coordinate range. Default: NULL.
`show.legend`	Include legend into the plot of original vs. reconstructed series? Logical. Default: TRUE.
`legend.coords`	coordinates to position the legend (as in function `legend`). Default: "topleft".
`legend.horiz`	Set the legend horizontally rather than vertically? Logical. Default: FALSE.

`legend.text`	legend text. Default: c("original (detrended)", "reconstructed").
`label.time.axis`	Label the time axis? Logical. Default: TRUE.
`show.date`	Show calendar dates? (Effective only if dates are available as rownames or by variable `date` in the data frame which has been analyzed using `analyze.wavelet`.) Logical. Default: FALSE.
`date.format`	the format of date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see `strptime` for a list of implemented date conversion specifications. (If not specified, as.Date will be applied.) Default: NULL.
`timelab`	Time axis label. Default: "time".
`main.waves`	an overall title for the plot of reconstruction waves. Default: NULL.
`main.rec`	an overall title for the plot of original vs. reconstructed series. Default: NULL.
`main`	an overall title for both plots. Default: NULL.
`verbose`	Print verbose output on the screen? Logical. Default: TRUE.

Value

A list of class `reconstruct` with the following elements:

`series` a data frame with the following columns

date	:	the calendar date (if available as column in my.data)
<x>	:	series <x>, with original name retained
	:	(detrended, if loess.span != 0)
<x>.trend	:	the trend series (if loess.span != 0)
<x>.r	:	the reconstructed (detrended) series

Row names are taken over from WT.

`rec.waves`	data frame of scaled waves used for reconstruction
`loess.span`	parameter alpha in loess controlling the degree of time series smoothing, if the time series was detrended; no detrending if loess.span=0.
`lvl`	level which the wavelet power should have at least for waves (wave segments) to be included in the reconstruction. Default: 0.
`only.coi`	Is the influence of edge effects excluded? I.e. is the cone of influence used only?
`only.sig`	Was wavelet power significance used in reconstruction?
`siglvl`	level of wavelet power significance
`only.ridge`	Select ridge only? Logical.
`rnum.used`	the vector of Fourier period numbers used for reconstruction
`rescale`	Was the reconstructed series rescaled according to the mean and standard deviation taken from the original (detrended) series?
`dt`	time resolution, i.e. sampling resolution on time domain, 1/dt = number of intervals per time step.
`dj`	frequency resolution, i.e. sampling resolution on frequency domain, 1/dj = number of suboctaves (voices per octave).

`Period`	the Fourier periods (in time units)
`Scale`	the scales
`nc`	number of columns/time steps
`nr`	number of rows/scales
`axis.1`	tick levels corresponding to time steps
`axis.2`	tick levels corresponding to Fourier periods = log2(Period)

Author(s)

Angi Roesch and Harald Schmidbauer

References

Carmona R., Hwang W.-L., and Torresani B., 1998. Practical Time Frequency Analysis. Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. Journal of Atmospheric and Oceanic Technology 24, 2093–2102.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

See Also

analyze.wavelet, wt.avg, wt.image, wt.sel.phases, wt.phase.image

Examples

```
## Not run:
## The following example is adopted from Liu et al, 2007:

series.length = 6*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 8*24, length = series.length)
x3 = periodic.series(start.period = 32*24, length = series.length)
x4 = periodic.series(start.period = 128*24, length = series.length)
x = x1 + x2 + x3 + x4

plot(ts(x, start=0, frequency=24), type="l",
 xlab="time (days)",
 ylab="hourly data", main="a series of hourly data with periods of 1, 8, 32, and 128 days")

my.data = data.frame(x=x)

my.w = analyze.wavelet(my.data, "x",
                       loess.span=0,
                       dt=1/24, dj=1/20,
                       lowerPeriod = 1/4,
                       make.pval=T, n.sim=10)

## Plot of wavelet power spectrum (with equidistant color breakpoints):
```

```
wt.image(my.w, color.key="interval", legend.params=list(lab="wavelet power levels"))

## Reconstruction of the time series, including significant components only:
reconstruct(my.w, timelab="time (days)")
## The same reconstruction, but showing wave components first:
reconstruct(my.w, timelab="time (days)", plot.waves=T)

## Reconstruction, including all components whether significant or not:
reconstruct(my.w, timelab="time (days)", only.sig=F)

## Reconstruction, including significant components, but selected periods only:
reconstruct(my.w, timelab="time (days)", sel.period=c(1,8,32,128))

## Reconstruction, including significant components, but the ridge only:
reconstruct(my.w, timelab="time (days)", only.ridge=T)

## See the periods involved:
my.rec = reconstruct(my.w, timelab="time (days)", only.ridge=T)
print(my.rec$Period[my.rec$rnum.used])
## The original and reconstructed time series can be retrieved as well:
plot(my.rec$series$x, type="l", xlab="time (days)", ylab="")
lines(my.rec$series$x.r, col="red")
legend("topleft", legend=c("original","reconstructed"), lty=1, col=c("black","red"))

## End(Not run)
```

SurrogateData *Simulation of surrogates for a given time series x, subject to the specified method and parameters*

Description

It simulates a surrogate for the time series x to be analyzed by wavelet transformation using either function `analyze.wavelet` or function `analyze.coherency`. A set of surrogates is used for significance assessment to test the hypothesis of equal periodic components.

Simulation is subject to model/method specification and parameter setting: Currently, one can choose from a variety of 6 methods (white noise, series shuffling, Fourier randomization, AR, and ARIMA) with respective lists of parameters to set.

The name and layout were inspired by a similar function developed by Huidong Tian (archived R package `WaveletCo`).

Usage

```
SurrogateData(x, method = "white.noise",
              params = list(
                       AR = list(p = 1),
                       ARIMA = list(p = 1, q = 1,
                                    include.mean = T, sd.fac = 1,
                                    trim = F, trim.prop = 0.01))
             )
```

Arguments

`x`	the given time series
`method`	the method of generating surrogate time series, select from: "white.noise" : white noise "shuffle" : shuffling the given time series "Fourier.rand" : time series with a similar spectrum "AR" : AR(p) "ARIMA" : ARIMA(p,0,q) Default: "white.noise"
`params`	a list of assignments between methods (AR, and ARIMA) and lists of parameter values applying to surrogates. Default: NULL. Default includes: AR = list(p=1), where: p : AR order ARIMA = list(p=1, q=1, include.mean=T, sd.fac=1, trim = F, trim.prop = 0.01), where: p : AR order q : MA order include.mean : Include a mean/intercept term? sd.fac : magnification factor to boost the residual standard deviation trim : Simulate trimmed data? trim.prop : high/low trimming proportion

Value

A surrogate series for x is returned which has the same length and properties according to estimates resulting from the model/method specification and parameter setting.

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian.

References

Tian, H., and Cazelles, B., 2012. `WaveletCo`. Available at http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

See Also

`analyze.wavelet`, `analyze.coherency`, `AR`, `ARIMA`, `FourierRand` wc.avg *Plot cross-wavelet power averages and wavelet coherence averages across time of two time series*

Description

This function plots cross-wavelet power averages across time, or alternatively wavelet coherence averages, of two time series, which are provided by an object of class `analyze.coherency`. The vertical axis shows the Fourier periods. The horizontal axis shows the averages.

There is an option to label periods according to significance of averages (if p-values are provided by `analyze.coherency`) at given levels of significance. Labels are point symbols along the line of averages which can be assigned individually.

The idea to show significance levels by colors of plotting characters and its implementation has been adopted from Huidong Tian and Bernard Cazelles (archived R package `WaveletCo`).

Usage

```
wc.avg(WC, which.avg = "wp", show.siglvl = T, siglvl = c(0.05, 0.1),
   sigcol = c("red", "blue"), sigpch = 20,
   label.avg.axis = T, averagelab = NULL,
   label.period.axis = T, periodlab = NULL,
   show.legend = T, legend.coords = "topright",
   main = NULL, lwd = 0.5,
   verbose = F)
```

Arguments

`WC` an object of class `analyze.coherency`.

`which.avg` Which averages should be plotted?

"wp"	:	cross-wavelet power
"wc"	:	wavelet coherence

Default: "wp"

`show.siglvl` Label periods according to significance of averages? (Effective only if p-values are provided by `analyze.coherency`.)
Default: TRUE.

`siglvl` a vector of significance levels (of any length and order). Default: c(0.05, 0.1).

`sigcol` a vector of colors (should be of same length as and correspond to `siglvl`, otherwise 1:length(siglvl)).
Default: c("red","blue").

`sigpch` a vector of plotting characters. (It should be of same length as and correspond to `siglvl` to produce different labels, otherwise the default setting is used. A single input value affects all labels.) Default: 20.

`label.avg.axis`	Label the axis of averages? Logical. Default: TRUE.
`averagelab`	Label for the axis of averages. Default: "average cross-wavelet power" (if which.avg="wp"), "average coherence" (if which.avg="wc").
`label.period.axis`	Label the (Fourier) period axis? Logical. Default: TRUE.
`periodlab`	(Fourier) period axis label. Default: "period".
`show.legend`	Include legend of significance levels into the plot? Logical. Default: TRUE.
`legend.coords`	coordinates to position the legend (as in function `legend`). Default: "topright".
`main`	an overall title for the plot. Default: NULL.
`lwd`	line width. Default: 0.5.
`verbose`	Print verbose output on the screen? Logical. Default: FALSE.

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian and Bernard Cazelles.

References

Aguiar-Conraria L., and Soares M.J., 2011. Business cycle synchronization and the Euro: A wavelet analysis. Journal of Macroeconomics 33 (3), 477–489.

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu P.C., 1994. Wavelet spectrum analysis and ocean wind waves. In: Foufoula-Georgiou E., and Kumar P., (eds.), Wavelets in Geophysics, Academic Press, San Diego, 151–166.

Tian, H., and Cazelles, B., 2012. `WaveletCo`. Available at http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

Veleda D., Montagne R., and Araujo M., 2012. Cross-Wavelet Bias Corrected by Normalizing Scales. Journal of Atmospheric and Oceanic Technology 29, 1401–1408.

See Also

`analyze.coherency`, `wc.image`, `wc.sel.phases`, `wc.phasediff.image`

Examples

```
## Not run:
## The following example is adopted from Veleda et al, 2012:

add.noise=TRUE
```

```
series.length = 3*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 2*24, length = series.length)
x3 = periodic.series(start.period = 4*24, length = series.length)
x4 = periodic.series(start.period = 8*24, length = series.length)
x5 = periodic.series(start.period = 16*24, length = series.length)
x6 = periodic.series(start.period = 32*24, length = series.length)
x7 = periodic.series(start.period = 64*24, length = series.length)
x8 = periodic.series(start.period = 128*24, length = series.length)

x = x1 + x2 + x3 + x4 + 3*x5 + x6 + x7 + x8
y = x1 + x2 + x3 + x4 + 3*x5 + x6 + 3*x7 + x8

if (add.noise == TRUE){
    x = x + rnorm(length(x))
    y = y + rnorm(length(y))
}

my.data = data.frame(x=x, y=y)

ts.plot(ts(my.data$x, start=0, frequency=24),
     ts(my.data$y, start=0, frequency=24),
     type="l", col=1:2,
     xlab="time (days)", ylab="hourly data",
     main="a series of hourly data with periods of 1, 2, 4, 8, 16, 32, 64, and 128 days",
     sub="(different amplitudes at periods 16 and 64)")
legend("topright", legend=c("x","y"), col=1:2, lty=1)

## computation of cross-wavelet power and wavelet coherence:
my.wc = analyze.coherency(my.data, c("x","y"), loess.span=0,
                          dt=1/24, dj=1/20,
                          window.size.t=1, window.size.s=1/2,
                          lowerPeriod=1/4,
                          make.pval=T, n.sim=10)

## plot of cross-wavelet power (with color breakpoints according to quantiles):
wc.image(my.wc, timelab="time (days)", periodlab="period (days)",
         main="cross-wavelet power")
## plot of average cross-wavelet power:
wc.avg(my.wc, siglvl=0.05, sigcol="red")

## plot of wavelet coherence (with color breakpoints according to quantiles):
wc.image(my.wc, which.image="wc", timelab="time (days)", periodlab="period (days)",
         main="wavelet coherence",
         legend.params=list(label.digits=3))
## plot of average wavelet coherence:
wc.avg(my.wc, which.avg="wc", siglvl=0.05, sigcol="red", legend.coords="topleft")

## End(Not run)
```

`wc.image` *Image plot of the cross-wavelet power spectrum and wavelet coherence spectrum of two time series*

Description

This function plots the cross-wavelet power image, or alternatively the wavelet coherence image, of two time series, which are provided by an object of class `analyze.coherency`. The vertical axis shows the Fourier periods. The horizontal axis shows time step counts, but it can be easily transformed into a calendar axis if dates are provided in either rownames or a variable named date in the data frame at hand. Both axes can be relabeled.

An option is given to raise values by any exponent before plotting.

The color levels can be defined according to quantiles of values or according to equidistant breakpoints (covering the interval from 0 to maximum value), with the number of levels as a further parameter. In addition, there is an option to adopt an individual color palette.

Further plot design options concern: plot of the cone of influence, plot of contour lines to border areas of significance, plot of the ridge, and plot of arrows (optional: "smoothed" arrows computed from smoothing filters as defined in `analyze.coherency`) to reflect phase differences.

For that matter, the significance level of contour lines can be defined separately. The plot of the ridge can be restricted to a high-level region ("high" according to a given level of plotted values). In particular, the area to be filled with arrows can be determined in several ways: to reflect significance (at a given level) with respect to cross-wavelet power, wavelet coherency, or individual wavelet power, and/or to flag a high-value region. Furthermore, there is an option to clear out the area where the p-values of cross-wavelet power (coherence, respectively) exceed a given level.

Finally, there is an option to format and insert a color legend (a right-hand vertical color bar) and to set the plot title. For further processing of the plot, graphical parameters of plot regions are provided as output.

The name and parts of the layout were inspired by a similar function developed by Huidong Tian and Bernard Cazelles (archived R package `WaveletCo`). The code for the arrow design to reflect phase differences has been adopted from Huidong Tian.

Usage

```
wc.image(WC,
     which.image = "wp", exponent = 1,
     plot.coi = T,
     plot.contour = T, siglvl.contour = 0.1, col.contour = "white",
     plot.ridge = F, lvl = 0, col.ridge = "black",
     plot.arrow = T, use.sAngle = F,
     p = 1,
     which.arrow.sig = which.image, siglvl.arrow = 0.05, col.arrow = "black",
     clear.area = F,
     which.area.sig = which.image, siglvl.area = 0.2,
     color.key = "quantile",
     n.levels = 100, color.palette = "rainbow(n.levels, start=0, end=.7)",
     useRaster = T, max.contour.segments = 250000,
```

```
plot.legend = T,
legend.params = list(width=1.2, shrink=0.9, mar=5.1,
                     n.ticks=6, label.digits=1, label.format="f",
                     lab=NULL, lab.line=2.5),
label.time.axis = T, show.date = F, date.format = NULL, timelab = NULL,
label.period.axis = T, periodlab = NULL,
main = NULL,
lwd = 2,
graphics.reset = T,
verbose = F)
```

Arguments

`WC` an object of class `analyze.coherency`

`which.image` Which image is to be plotted?

"wp"	:	cross-wavelet power
"wc"	:	wavelet coherence

Default: "wp"

`exponent` Exponent of values to be plotted. Default: 1.

`plot.coi` Plot cone of influence? Logical. Default: TRUE

`plot.contour` Plot contour lines to border the area of cross-wavelet power, respectively wavelet coherence significance at level `siglvl.contour`? Logical. Default: TRUE.

`siglvl.contour` level of cross-wavelet power, respectively wavelet coherence significance applied to the plot of contour lines. Default: 0.1.

`col.contour` Color of contour lines. Default: "white".

`plot.ridge` Plot the cross-wavelet, resp. wavelet coherence power ridge? Logical. Default: FALSE.

`lvl` minimum level of cross-wavelet power (or wavelet coherence) for the ridge to be plotted, or alternatively, within the area of arrows (if p=0 or 2). Default: 0.

`col.ridge` color of the cross-wavelet power, resp. wavelet coherence ridge. Default: "black".

`plot.arrow` Plot arrows depicting the phase difference? Logical. Default: TRUE.

`use.sAngle` Use smoothed version of phase difference? Logical. Default: FALSE.

`p` Which area should be filled with arrows displaying phase differences? (Only effective if `plot.arrow`=TRUE.)

p=0	:	area with high values of `which.lvl` only (cf. `lvl`)
p=1	:	area of significance of `which.sig` only (cf. `siglvl`)
p=2	:	area with both high values and significance (combining p=0 and p=2)

Default: 1

`which.arrow.sig`
: Which spectrum and corresponding p-values should be used to restrict the area of arrows according to significance?

"wp"	:	cross-wavelet power
"wc"	:	wavelet coherence
"wt"	:	individual wavelet power

Default: `which.image`

`siglvl.arrow`
: level of significance referring to `which.arrow.sig` (if plot.arrow=TRUE and p=1 or 2).
Default: 0.05

`col.arrow`
: color for the plot of arrows. Default: "black".

`clear.area`
: Clear out an area where p-values are above a certain level? Logical.
(p-values will refer to the spectrum defined by which.area.sig and significance level siglvl.area.)
Default: FALSE

`which.area.sig`
: Which power spectrum and corresponding p-values should be used to clear the outer area? (if clear.area=TRUE)

"wp"	:	cross-wavelet power
"wc"	:	wavelet coherence
"wt"	:	individual wavelet power

Default: `which.image`

`siglvl.area`
: level of significance referring to which.area.sig (if clear.area=TRUE)
Default: 0.2

`color.key`
: How to assign colors to power and coherence levels? Two options:

"interval" or "i"	:	equidistant breakpoints (from 0 through maximum value)
"quantile" or "q"	:	quantiles

Default: "quantile"

`n.levels`
: Number of color levels. Default: 100.

`color.palette`
: Definition of color levels. (It will be assigned to levels in reverted order!) Default: "rainbow(n.levels, start=0, end=.7)".

`useRaster`
: Use a bitmap raster instead of polygons to plot the wavelet power image? Logical. Default: TRUE.

`max.contour.segments`
: limit on the number of segments in a single contour line, positive integer. Default: 250000 (`options(...)` default settings: 25000)

`plot.legend`
: Plot color legend (a vertical bar of colors and breakpoints)? Logical. Default: TRUE

`legend.params`	a list of parameters for the plot of color legend, parameter values can be set selectively (style in parts adopted from image.plot in the R package "fields" by Douglas Nychka):

width	:	width of legend bar. Default: 1.2.
shrink	:	a vertical shrinkage factor. Default: 0.9.
mar	:	right margin of legend bar. Default: 5.1.
n.ticks	:	number of ticks for labels. Default: 6.
label.digits	:	digits of labels. Default: 1.
label.format	:	format of labels. Default: "f".
lab	:	axis label. Default: NULL.
lab.line	:	line (in user coordinate units) where to put the axis label. Default: 2.5.

`label.time.axis`	Label the time axis? Logical. Default: TRUE.
`show.date`	Show calendar dates? (Effective only if dates are available as rownames or by variable `date` in the data frame which has been analyzed using `analyze.coherency`.) Logical. Default: FALSE.
`date.format`	the format of date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see `strptime` for a list of implemented date conversion specifications. (If not specified, as.Date will be applied.) Default: NULL.
`timelab`	Time axis label. Default: "time".
`label.period.axis`	Label the (Fourier) period axis? Logical. Default: TRUE.
`periodlab`	(Fourier) period axis label. Default: "period".
`main`	an overall title for the plot. Default: NULL.
`lwd`	line width of contour lines and ridge. Default: 2.
`graphics.reset`	Reset graphical parameters? Logical. Default: TRUE
`verbose`	Print verbose output on the screen? Logical. Default: FALSE.

Value

A list of class `graphical parameters` with the following elements:

`op`	original graphical parameters
`image.plt`	image plot region
`legend.plt`	legend plot region

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian, and Bernard Cazelles.

References

Aguiar-Conraria L., and Soares M.J., 2011. Business cycle synchronization and the Euro: A wavelet analysis. Journal of Macroeconomics 33 (3), 477–489.

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu P.C., 1994. Wavelet spectrum analysis and ocean wind waves. In: Foufoula-Georgiou E., and Kumar P., (eds.), Wavelets in Geophysics, Academic Press, San Diego, 151–166.

Tian, H., and Cazelles, B., 2012. `WaveletCo`. Available at http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

Veleda D., Montagne R., and Araujo M., 2012. Cross-Wavelet Bias Corrected by Normalizing Scales. Journal of Atmospheric and Oceanic Technology 29, 1401–1408.

See Also

`analyze.coherency`, `wc.avg`, `wc.sel.phases`, `wc.phasediff.image`

Examples

```
## Not run:
## The following example is adopted from Veleda et al, 2012:

add.noise=TRUE

series.length = 3*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 2*24, length = series.length)
x3 = periodic.series(start.period = 4*24, length = series.length)
x4 = periodic.series(start.period = 8*24, length = series.length)
x5 = periodic.series(start.period = 16*24, length = series.length)
x6 = periodic.series(start.period = 32*24, length = series.length)
x7 = periodic.series(start.period = 64*24, length = series.length)
x8 = periodic.series(start.period = 128*24, length = series.length)

x = x1 + x2 + x3 + x4 + 3*x5 + x6 + x7 + x8
y = x1 + x2 + x3 + x4 + 3*x5 + x6 + 3*x7 + x8

if (add.noise == TRUE){
    x = x + rnorm(length(x))
    y = y + rnorm(length(y))
}
```

```
my.date = seq(as.POSIXct("2014-10-14 00:00:00","%F %T"), by="hour",
              length.out=series.length)
my.data = data.frame(date=my.date, x=x, y=y)

ts.plot(ts(my.data$x, start=0, frequency=24),
     ts(my.data$y, start=0, frequency=24),
     type="l", col=1:2,
     xlab="time (days)", ylab="hourly data",
     main="a series of hourly data with periods of 1, 2, 4, 8, 16, 32, 64, and 128 days",
     sub="(different amplitudes at periods 16 and 64)")
legend("topright", legend=c("x","y"), col=1:2, lty=1)

## computation of cross-wavelet power and wavelet coherence:
my.wc = analyze.coherency(my.data, c("x","y"), loess.span=0,
                          dt=1/24, dj=1/20,
                          window.size.t=1, window.size.s=1/2,
                          lowerPeriod=1/4,
                          make.pval=T, n.sim=10)

## plot of cross-wavelet power, with color breakpoints according to quantiles:
wc.image(my.wc, timelab="time (days)", periodlab="period (days)",
         main="cross-wavelet power",
         legend.params=list(lab="cross-wavelet power levels (quantiles)"))

## The same plot, but with equidistant color breakpoints:
wc.image(my.wc, color.key="i", timelab="time (days)", periodlab="period (days)",
         main="cross-wavelet power",
         legend.params=list(lab="cross-wavelet power levels (equidistant levels)"))

## The same plot, but adopting a palette of gray colors:
wc.image(my.wc, color.key="i", timelab="time (days)", periodlab="period (days)",
         main="cross-wavelet power",
         legend.params=list(lab="cross-wavelet power levels (equidistant levels)"),
         color.palette="gray( (1:n.levels)/n.levels )", plot.arrow=F)

## The same plot, but with yellow arrows and calendar axis:
wc.image(my.wc, color.key="i", timelab="", periodlab="period (days)",
         main="cross-wavelet power",
         legend.params=list(lab="cross-wavelet power levels (equidistant levels)"),
         color.palette="gray( (1:n.levels)/n.levels )",
         col.arrow="yellow",
         show.date=T)

## With additional ridge:
wc.image(my.wc, color.key="i", timelab="", periodlab="period (days)",
         main="cross-wavelet power",
         legend.params=list(lab="cross-wavelet power levels (equidistant levels)"),
         color.palette="gray( (1:n.levels)/n.levels )",
         col.arrow="yellow",
         show.date=T,
         plot.ridge=T, col.ridge="red")
```

```
## The same plot, but with yellow arrows and individualized calendar axis:
my.plot = wc.image(my.wc, color.key="i", timelab="", periodlab="period (days)",
             main="cross-wavelet power",
             legend.params=list(lab="cross-wavelet power levels (equidistant levels)"),
             color.palette="gray( (1:n.levels)/n.levels )",
             col.arrow="yellow",
             label.time.axis =F)
## recover plot region:
par(new=T, plt=my.plot$image.plt)
## empty plot
plot(my.date, rep(1,series.length), type="n",
     xaxs = "i", yaxs ="i", xaxt="n", yaxt="n",
     xlab="", ylab="")
## individualized calendar axis:
axis.POSIXct(1, at=
 seq(as.POSIXct("2014-11-01 00:00:00", "%F %T"), my.date[length(my.date)], by="month"),
 format="%b %Y", las=2)
## return to default plot region:
par(my.plot$op)

## plot of wavelet coherence, with color breakpoints according to quantiles:
wc.image(my.wc, which.image="wc",
  timelab="time (days)", periodlab="period (days)",
  main="wavelet coherence",
  legend.params=list(lab="wavelet coherence levels (quantiles)", lab.line=3.5,
                     label.digits=3))
## plot of wavelet coherence, but with equidistant color breakpoints:
wc.image(my.wc, which.image="wc", color.key="i",
         timelab="time (days)", periodlab="period (days)",
         main="wavelet coherence",
         legend.params=list(lab="wavelet coherence levels (equidistant levels)"))


## End(Not run)
```

`wc.phasediff.image`	*Image plot of phase differences of periodic components for two time series*

Description

This function plots the phase difference image of two time series, which is provided by an object of class `analyze.coherency`. The vertical axis shows the Fourier periods. The horizontal axis shows time step counts, but can be easily transformed into a calendar axis if dates are provided in either rownames or a variable named date in the data frame at hand. Both axes can be relabeled.

The color levels are defined according to equidistant breakpoints (covering the interval from -pi to +pi), with the number of levels as a further parameter. In addition, there is an option to adopt an individual color palette.

If the default palette is retained, colors indicate the following. Green: phase differences close to zero, which means that the two time series are in phase at the respective period. Yellowgreen: in phase, series 1 leading. Turquoise: in phase, series 2 leading. Red: phase differences are close to +pi, out of phase, series 2 leading. Blue: phase differences are close to -pi, out of phase, series 1 leading.

Further plot design options concern: plot of the cone of influence, plot of contour lines to border areas of significance with respect to cross-wavelet power or wavelet coherence at a given significance level.

Finally, there is an option to insert and format a color legend (a right-hand vertical color bar) and to set the plot title. For further processing of the plot, graphical parameters of plot regions are provided as output.

Usage

```
wc.phasediff.image(WC, use.sAngle = F,
   plot.coi = T,
   plot.contour = T, which.contour = "wp", siglvl = 0.1, col.contour = "white",
   n.levels = 100, color.palette = "rainbow(n.levels, start=0, end=.7)",
   useRaster = T, max.contour.segments = 250000,
   plot.legend = T,
   legend.params = list(width=1.2, shrink=0.9, mar=5.1,
                        n.ticks=6, label.digits=2, label.format="f",
                        lab=NULL, lab.line=2.5),
   label.time.axis = T, show.date = F, date.format = NULL, timelab = NULL,
   label.period.axis = T, periodlab = NULL,
   main = NULL,
   graphics.reset = T,
   verbose = F)
```

Arguments

`WC` an object of class `analyze.coherency`

`use.sAngle` Use smoothed version of phase difference? Logical. Default: FALSE.

`plot.coi` Plot cone of influence? Logical. Default: TRUE

`plot.contour` Plot contour lines to border the area of cross-wavelet power, respectively wavelet coherence significance at level siglvl.contour? Logical. Default: TRUE.

`which.contour` Contour lines of which spectrum should be plotted?

"wp"	:	cross-wavelet power
"wc"	:	wavelet coherence

Default: "wp"

`siglvl` level of cross-wavelet power, respectively wavelet coherence significance applied to the plot of contour lines. Default: 0.1.

`col.contour` Color of contour lines. Default: "white".

`n.levels` Number of color levels. Default: 100.

`color.palette` Definition of color levels. (It will be assigned to levels in reverse order!) Default: "rainbow(n.levels, start=0, end=.7)".

`useRaster` Use a bitmap raster instead of polygons to plot the wavelet power image? Logical. Default: TRUE.

`max.contour.segments`
limit on the number of segments in a single contour line, positive integer. Default: 250000 (`options(...)` default settings: 25000)

`plot.legend` Plot color legend (a vertical bar of colors and breakpoints)? Logical. Default: TRUE

`legend.params` a list of parameters for the plot of color legend, parameter values can be set selectively (style in parts adopted from image.plot in the R package fields by Douglas Nychka):

width	:	width of legend bar. Default: 1.2.
shrink	:	a vertical shrinkage factor. Default: 0.9.
mar	:	right margin of legend bar. Default: 5.1.
n.ticks	:	number of ticks for labels. Default: 6.
label.digits	:	digits of labels. Default: 2.
label.format	:	format of labels. Default: "f".
lab	:	axis label. Default: NULL.
lab.line	:	line (in user coordinate units) where to put the axis label. Default: 2.5.

`label.time.axis`
Label the time axis? Logical. Default: TRUE.

`show.date` Show calendar dates? (Effective only if dates are available as rownames or by variable `date` in the data frame which has been analyzed using `analyze.coherency`.) Logical. Default: FALSE.

`date.format` the format of date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see `strptime` for a list of implemented date conversion specifications. (If not specified, as.Date will be applied.) Default: NULL.

`timelab` Time axis label. Default: "time".

`label.period.axis`
Label the (Fourier) period axis? Logical. Default: TRUE.

`periodlab` (Fourier) period axis label. Default: "period".

`main` an overall title for the plot. Default: NULL.

`graphics.reset` Reset graphical parameters? Logical. Default: TRUE

`verbose` Print verbose output on the screen? Logical. Default: FALSE.

Value

A list of class `graphical parameters` with the following elements:

`op`	original graphical parameters
`image.plt`	image plot region
`legend.plt`	legend plot region

Author(s)

Angi Roesch and Harald Schmidbauer

References

Aguiar-Conraria L., and Soares M.J., 2011. Business cycle synchronization and the Euro: A wavelet analysis. Journal of Macroeconomics 33 (3), 477–489.

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu P.C., 1994. Wavelet spectrum analysis and ocean wind waves. In: Foufoula-Georgiou E., and Kumar P., (eds.), Wavelets in Geophysics, Academic Press, San Diego, 151–166.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

Veleda D., Montagne R., and Araujo M., 2012. Cross-Wavelet Bias Corrected by Normalizing Scales. Journal of Atmospheric and Oceanic Technology 29, 1401–1408.

See Also

`analyze.coherency`, `wc.avg`, `wc.image`, `wc.sel.phases`

Examples

```
## Not run:
## The following example is adopted from Veleda et al, 2012:

add.noise=TRUE

series.length = 3*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 2*24, length = series.length)
x3 = periodic.series(start.period = 4*24, length = series.length)
x4 = periodic.series(start.period = 8*24, length = series.length)
x5 = periodic.series(start.period = 16*24, length = series.length)
x6 = periodic.series(start.period = 32*24, length = series.length)
x7 = periodic.series(start.period = 64*24, length = series.length)
x8 = periodic.series(start.period = 128*24, length = series.length)

x = x1 + x2 + x3 + x4 + 3*x5 + x6 + x7 + x8
```

```
y = x1 + x2 + x3 + x4 + 3*x5 + x6 + 3*x7 + x8

if (add.noise == TRUE){
    x = x + rnorm(length(x))
    y = y + rnorm(length(y))
}

my.data = data.frame(x=x, y=y)

ts.plot(ts(my.data$x, start=0, frequency=24),
     ts(my.data$y, start=0, frequency=24),
     type="l", col=1:2,
     xlab="time (days)", ylab="hourly data",
     main="a series of hourly data with periods of 1, 2, 4, 8, 16, 32, 64, and 128 days",
     sub="(different amplitudes at periods 16 and 64)")
legend("topright", legend=c("x","y"), col=1:2, lty=1)

## computation of cross-wavelet power and wavelet coherence:
my.wc = analyze.coherency(my.data, c("x","y"),
                          loess.span=0,
                          dt=1/24, dj=1/20,
                          window.size.t=1, window.size.s=1/2,
                          lowerPeriod=1/4,
                          make.pval=T, n.sim=10)

## plot of phase differences (with contour lines referring to cross-wavelet power)
wc.phasediff.image(my.wc, which.contour="wp", timelab="time (days)",
                   main="image of phase differences")

## End(Not run)
```

`wc.sel.phases` *Comparison plot of phases for selected periodic components of two time series*

Description

This function plots the phases for periodic components of two time series, which are provided by an object of class `analyze.coherency`.

Periodic components can be selected by specification of a single Fourier period or of a period band. In the latter case, and in the default case (no specification at all), phases are averaged across periods for each time series. Other options: restriction to the cone of influence, restriction to an area of significance (with respect to cross-wavelet power, wavelet coherence or individual wavelet power). Phase differences (i.e. angles, smoothed or not smoothed) can be added to the plot.

(The time axis can be altered to display dates, see e.g. `wt.image`.)

Usage

```
wc.sel.phases(WC, sel.period = NULL, sel.lower = NULL, sel.upper = NULL,
      only.coi = F,
      only.sig = T, which.sig = "wp", siglvl=0.05,
      phase.cols = c("red", "blue"),
      show.Angle = T, use.sAngle = F, Angle.col = "black",
      show.legend = T, legend.coords = "topleft", legend.horiz = T,
      label.time.axis = T, show.date = F, date.format = NULL, timelab = NULL,
      label.phase.axis = T, phaselab = NULL,
      phaselim = c(-pi,pi+show.legend*ifelse(legend.horiz,0.8,2)),
      main = NULL, sub = NULL,
      verbose = F)
```

Arguments

`WC` an object of class `analyze.wavelet`.

`sel.period` a single number which determines the (closest available) Fourier period to be selected. Default: NULL.

`sel.lower` a lower number which determines the lower (closest available) Fourier period to be selected if `sel.period` is NULL. Default: NULL.

`sel.upper` an upper number which determines the upper (closest available) Fourier period to be selected if `sel.period` is NULL. Default: NULL.

`only.coi` Exclude borders influenced by edge effects, i.e. include the cone of influence only? Logical. Default: FALSE.

`only.sig` Use cross-wavelet power or coherence significance to decide about the inclusion of (parts of) the phases' series? Logical. Default: TRUE.

`which.sig` Which spectrum should significance refer to?

"wp"	:	cross-wavelet power (default)
"wc"	:	wavelet coherence
"wt"	:	individual wavelet power

Default: "wp"

`siglvl` level of significance. Default: 0.05.

`phase.cols` a vector of two colors for the plot of (average) phases referring to the two time series. Default: c("red","blue").

`show.Angle` Show the (average) phase difference (the Angle) between the two series? Logical. Default: TRUE.

`use.sAngle` Use smoothed version of phase difference? Logical. Default: FALSE.

`Angle.col` Color for the plot of Angles. Default: "black".

`show.legend` Include legend? Logical. Default: TRUE.

`legend.coords` Coordinates to position the legend (with the same options as given in function `legend`). Default: "topleft".

`legend.horiz` Set the legend horizontally rather than vertically? Logical. Default: TRUE.

`label.time.axis`
: Label the time axis? Logical. Default: TRUE.

`show.date`
: Show calendar dates? (Effective only if dates are available as rownames or as variable `date` in the data frame analyzed using `analyze.coherency`.) Logical. Default: FALSE.

`date.format`
: the format of date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see `strptime` for a list of implemented date conversion specifications. (If not specified, as.Date will be applied.) Default: NULL.

`timelab`
: Time axis label. Default: "time".

`label.phase.axis`
: Label the phase axis? Logical. Default: TRUE.

`phaselab`
: Phase axis label. Default: "phase".

`phaselim`
: numeric vector of length 2, giving the phase coordinate range. Default: c(-pi,pi+0.8) (+0.8 in order to accomodate the horizontal legend, +2 in case of a vertical legend).

`main`
: an overall title for the plot. Default: NULL.

`sub`
: a subtitle for the plot. Default: NULL. In this case, the selected period range will be given in the subtitle.

`verbose`
: Print verbose output on the screen? Logical. Default: FALSE.

Value

A list of class `sel.phases` with the following elements:

`Period`
: the selected period (or period band)

`Phase.x`
: time series of (average) phases at the selected period (or period band), case of series x

`Phase.y`
: time series of (average) phases at the selected period (or period band), case of series y

`Angle`
: time series of (average) phase differences (non-smoothed version) at the selected period (or period band)

`sAngle`
: time series of (average) smoothed phase differences at the selected periods

`only.coi`
: Is the influence of edge effects excluded? I.e. is the cone of influence used only?

`only.sig`
: Was significance used in selection of phases?

`which.sig`
: Which spectrum was used to refer to significance?

"wp"	:	cross-wavelet power
"wc"	:	wavelet coherence
"wt"	:	individual wavelet power

`siglvl`
: level of significance

`date`
: time series of dates (if available)

`time.axis`
: tick levels corresponding to the time steps used for wavelet transformation

Author(s)

Angi Roesch and Harald Schmidbauer

References

Aguiar-Conraria L., and Soares M.J., 2011. Business cycle synchronization and the Euro: A wavelet analysis. Journal of Macroeconomics 33 (3), 477–489.

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu P.C., 1994. Wavelet spectrum analysis and ocean wind waves. In: Foufoula-Georgiou E., and Kumar P., (eds.), Wavelets in Geophysics, Academic Press, San Diego, 151–166.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

Veleda D., Montagne R., and Araujo M., 2012. Cross-Wavelet Bias Corrected by Normalizing Scales. Journal of Atmospheric and Oceanic Technology 29, 1401–1408.

See Also

`analyze.wavelet`, `wt.image`, `wt.avg`, `wt.phase.image`, `reconstruct`

Examples

```
## Not run:
## The following example is adopted from Veleda et al, 2012

add.noise=TRUE

series.length = 3*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 2*24, length = series.length)
x3 = periodic.series(start.period = 4*24, length = series.length)
x4 = periodic.series(start.period = 8*24, length = series.length)
x5 = periodic.series(start.period = 16*24, length = series.length)
x6 = periodic.series(start.period = 32*24, length = series.length)
x7 = periodic.series(start.period = 64*24, length = series.length)
x8 = periodic.series(start.period = 128*24, length = series.length)

x = x1 + x2 + x3 + x4 + 3*x5 + x6 + x7 + x8
y = x1 + x2 + x3 + x4 + 3*x5 + x6 + 3*x7 + x8

if (add.noise == TRUE){
    x = x + rnorm(length(x))
    y = y + rnorm(length(y))
}

my.data = data.frame(x=x, y=y)
ts.plot(ts(my.data$x, start=0, frequency=24),
```

```
     ts(my.data$y, start=0, frequency=24), type="l", col=1:2,
     xlab="time (days)", ylab="hourly data",
     main="a series of hourly data with periods of 1, 2, 4, 8, 16, 32, 64, and 128 days",
     sub="(different amplitudes at periods 16 and 64)")
legend("topright", legend=c("x","y"), col=1:2, lty=1)

## computation of cross-wavelet power and wavelet coherency
my.wc = analyze.coherency(my.data, c("x","y"), loess.span=0,
                          dt=1/24, dj=1/20,
                          window.size.t=1, window.size.s=1/2,
                          lowerPeriod=1/4,
                          make.pval=T, n.sim=10)

## plot of cross-wavelet power
wc.image(my.wc, timelab="time (days)", periodlab="period (days)",
         main="cross-wavelet power")

## Select period 64 and compare plots of corresponding phases, including the
## phase differences (angles) in their non-smoothed (default) version:
wc.sel.phases(my.wc, timelab="time (days)", sel.period=64, show.Angle=T)

## In the following, no periods are selected. In this case, instead of individual phases
## the plot shows average phases:
wc.sel.phases(my.wc, timelab="time (days)")

## End(Not run)
```

---

wt.avg *Plot of wavelet power averages across time of a single time series*

Description

This function plots wavelet power averages across time of a single time series, which are provided by an object of class `analyze.wavelet`, or alternatively of class `analyze.coherency`. (In the latter case, the series number or name can be specified.) The vertical axis shows the Fourier periods. The horizontal axis shows the averages.

There is an option to label periods according to significance of averages (if p-values are provided by `analyze.wavelet`) at given levels of significance. Labels are point symbols along the line of averages which can be assigned individually.

The idea to show significance levels by colors of plotting characters and its implementation has been adopted from Huidong Tian and Bernard Cazelles (archived R package `WaveletCo`).

Usage

```
wt.avg(WT, my.series = 1,
  show.siglvl = T, siglvl = c(0.05, 0.1), sigcol = c("red", "blue"), sigpch = 20,
  label.avg.axis = T, averagelab = NULL,
  label.period.axis = T, periodlab = NULL,
```

```
show.legend = T, legend.coords = "topright",
main = NULL, lwd = 0.5,
verbose = F)
```

Arguments

WT	an object of class analyze.wavelet or analyze.coherency
my.series	In case class(WT) = analyze.coherency: number (1 or 2) or name of the series to be analyzed. Default: 1.
show.siglvl	Label periods according to significance of averages? (Effective only if p-values are provided by analyze.coherency.) Default: TRUE.
siglvl	a vector of significance levels (of any length and order). Default: c(0.05, 0.1).
sigcol	a vector of colors (should be of same length as and correspond to siglvl, otherwise 1:length(siglvl)). Default: c("red","blue").
sigpch	a vector of plotting characters. (It should be of same length as and correspond to siglvl to produce different labels, otherwise the default setting is used. A single input value affects all labels.) Default: 20.
label.avg.axis	Label the axis of averages? Logical. Default: TRUE.
averagelab	Label for the axis of averages. Default: "average wavelet power".
label.period.axis	Label the (Fourier) period axis? Logical. Default: TRUE.
periodlab	(Fourier) period axis label. Default: "period".
show.legend	Include legend of significance levels into the plot? Logical. Default: TRUE.
legend.coords	coordinates to position the legend (as in function legend). Default: "topright".
main	an overall title for the plot. Default: NULL.
lwd	line width. Default: 0.5.
verbose	Print verbose output on the screen? Logical. Default: FALSE.

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian and Bernard Cazelles

References

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Carmona R., Hwang W.-L., and Torresani B., 1998. Practical Time Frequency Analysis. Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. Journal of Atmospheric and Oceanic Technology 24, 2093–2102.

Tian, H., and Cazelles, B., 2012. WaveletCo. Available at http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

See Also

analyze.wavelet, wt.image, wt.sel.phases, wt.phase.image, reconstruct

Examples

```
## Not run:
## The following example is adopted from Liu et al, 2007:

series.length = 6*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 8*24, length = series.length)
x3 = periodic.series(start.period = 32*24, length = series.length)
x4 = periodic.series(start.period = 128*24, length = series.length)
x = x1 + x2 + x3 + x4
plot(ts(x, start=0, frequency=24), type="l",
        xlab="time (days)", ylab="hourly data",
        main="a series of hourly data with periods of 1, 8, 32, and 128 days")

my.data = data.frame(x=x)

my.w = analyze.wavelet(my.data, "x", loess.span=0, dt=1/24, dj=1/20,
                       lowerPeriod=1/4, make.pval=T, n.sim=10)

## Plot of wavelet power spectrum (with equidistant color breakpoints):
wt.image(my.w, color.key="i",
         legend.params=list(lab="wavelet power levels (equidistant levels)"))
## Plot of average wavelet power:
wt.avg(my.w, siglvl=0.05, sigcol="red")

## End(Not run)
```

---

wt.image *Image plot of the wavelet power spectrum of a single time series*

Description

This function plots the wavelet power spectrum of a single time series, which is provided by an object of class analyze.wavelet, or alternatively of class analyze.coherency. (In the latter case, the series number or name can be specified.) The vertical axis shows the Fourier periods. The horizontal axis shows time step counts, but can be easily transformed into a calendar axis if dates are provided in either rownames or as a variable named date in the data frame at hand. Both axes can be relabeled.

The color levels can be defined according to quantiles of power or according to equidistant breakpoints (covering the interval from 0 to maximum power), with the number of levels as a further parameter. In addition, there is an option to adopt an individual color palette.

Further plot design options concern: plot of the cone of influence, plot of wavelet power contour lines at a specified level of significance, plot of power ridges.

Finally, there is an option to insert and format a color legend (a right-hand vertical color bar) and to set the plot title. For further processing of the plot, graphical parameters of plot regions are provided as output.

The name and parts of the layout were inspired by a similar function developed by Huidong Tian and Bernard Cazelles (archived R package `WaveletCo`).

Usage

```
wt.image(WT, my.series = 1,
     plot.coi = T,
     plot.contour = T, siglvl = 0.1, col.contour = "white",
     plot.ridge = T, lvl = 0, col.ridge = "black",
     color.key = "quantile",
     n.levels = 100, color.palette = "rainbow(n.levels, start=0, end=.7)",
     useRaster = T, max.contour.segments = 250000,
     plot.legend = T,
     legend.params = list(width = 1.2, shrink = 0.9, mar = 5.1,
                          n.ticks = 6, label.digits = 1, label.format = "f",
                          lab = NULL, lab.line = 2.5),
     label.time.axis = T, show.date = F, date.format = NULL, timelab = NULL,
     label.period.axis = T, periodlab = NULL,
     main = NULL,
     lwd = 2,
     graphics.reset = T,
     verbose = F)
```

Arguments

`WT`	an object of class `analyze.wavelet` or `analyze.coherency`
`my.series`	In case class(WT) = `analyze.coherency`: number (1 or 2) or name of the series to be analyzed. Default: 1.
`plot.coi`	Plot cone of influence? Logical. Default: TRUE
`plot.contour`	Plot contour lines to border the area of wavelet power significance at level `siglvl`? Logical. Default: TRUE.
`siglvl`	level of wavelet power significance applied to the plot of contour lines. Default: 0.1.
`col.contour`	Color of contour lines. Default: "white".
`plot.ridge`	Plot the wavelet power ridge? Logical. Default: TRUE.
`lvl`	minimum level of wavelet power for the ridge to be plotted. Default: 0.

`col.ridge` color of the power ridge. Default: "black".

`color.key` How to assign colors to power levels? Two options:

"interval" or "i"	:	equidistant breakpoints (from 0 through maximum value)
"quantile" or "q"	:	quantiles

Default: "quantile"

`n.levels` Number of color levels. Default: 100.

`color.palette` Definition of color levels. (It will be assigned to levels in reverse order!) Default: "rainbow(n.levels, start=0, end=.7)".

`useRaster` Use a bitmap raster instead of polygons to plot the wavelet power image? Logical. Default: TRUE.

`max.contour.segments`
limit on the number of segments in a single contour line, positive integer. Default: 250000 (`options(...)` default settings: 25000)

`plot.legend` Plot color legend (a vertical bar of colors and breakpoints)? Logical. Default: TRUE

`legend.params` a list of parameters for the plot of color legend, parameter values can be set selectively (style in parts adopted from image.plot in the R package "fields" by Douglas Nychka):

width	:	width of legend bar. Default: 1.2.
shrink	:	a vertical shrinkage factor. Default: 0.9.
mar	:	right margin of legend bar. Default: 5.1.
n.ticks	:	number of ticks for labels. Default: 6.
label.digits	:	digits of labels. Default: 1.
label.format	:	format of labels. Default: "f".
lab	:	axis label. Default: NULL.
lab.line	:	line (in user coordinate units) where to put the axis label. Default: 2.5.

`label.time.axis`
Label the time axis? Logical. Default: TRUE.

`show.date` Show calendar dates? (Effective only if dates are available as rownames or as variable `date` in the data frame analyzed using `analyze.wavelet`.) Logical. Default: FALSE.

`date.format` the format of date given as a character string, e.g. "%Y-%m-%d", or equivalently

"%F"; see `strptime` for a list of implemented date conversion specifications. (If not specified, as.Date will be applied.) Default: NULL.

`timelab` Time axis label. Default: "time".

`label.period.axis` Label the (Fourier) period axis? Logical. Default: TRUE.

`periodlab` (Fourier) period axis label. Default: "period".

`main` an overall title for the plot. Default: NULL.

`lwd` line width of contour lines and ridge. Default: 2.

`graphics.reset` Reset graphical parameters? Logical. Default: TRUE

`verbose` Print verbose output on the screen? Logical. Default: FALSE.

Value

A list of class `graphical parameters` with the following elements:

`op` original graphical parameters

`image.plt` image plot region

`legend.plt` legend plot region

Author(s)

Angi Roesch and Harald Schmidbauer; credits are also due to Huidong Tian and Bernard Cazelles

References

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Carmona R., Hwang W.-L., and Torresani B., 1998. Practical Time Frequency Analysis. Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. Journal of Atmospheric and Oceanic Technology 24, 2093–2102.

Tian, H., and Cazelles, B., 2012. `WaveletCo`. Available at http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

See Also

`analyze.wavelet`, `wt.avg`, `wt.sel.phases`, `wt.phase.image`, `reconstruct`

Examples

```
## Not run:
## The following example is adopted from Liu et al, 2007:

series.length = 6*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 8*24, length = series.length)
x3 = periodic.series(start.period = 32*24, length = series.length)
x4 = periodic.series(start.period = 128*24, length = series.length)
x = x1 + x2 + x3 + x4

plot(ts(x, start=0, frequency=24), type="l",
 xlab="time (days)", ylab="hourly data",
 main="a series of hourly data with periods of 1, 8, 32, and 128 days")

my.date =  seq(as.POSIXct("2014-10-14 00:00:00","%F %T"), by="hour",
               length.out=series.length)
my.data = data.frame(date=my.date, x=x)

my.w = analyze.wavelet(my.data, "x",
                       loess.span=0,
                       dt=1/24, dj=1/20,
                       lowerPeriod=1/4,
                       make.pval=T, n.sim=10)

## Plot of wavelet power spectrum with breakpoints referring to quantiles:
wt.image(my.w,
   legend.params=list(lab="wavelet power levels (quantiles)", lab.line=3.5,
                      label.digits=2))

## The same plot, but with equidistant color breakpoints:
wt.image(my.w, color.key="i",
         legend.params=list(lab="wavelet power levels (equidistant levels)"))

## The plot with calendar axis:
wt.image(my.w, color.key="i",
         legend.params=list(lab="wavelet power levels (equidistant levels)"),
         show.date=T, date.format="%F %T", timelab="")

## The same plot, but now with individualized calendar axis:
my.plot = wt.image(my.w, color.key="i",
                   legend.params=list(lab="wavelet power levels (equidistant levels)"),
                   label.time.axis=F)
## recover plot region:
par(new=T, plt=my.plot$image.plt)
## empty plot
plot(my.date, rep(1,series.length), type="n",
     xaxs = "i", yaxs="i", xaxt="n", yaxt="n",
     xlab="", ylab="")
## individualized calendar axis:
axis.POSIXct(1, at=
 seq(as.POSIXct("2014-11-01 00:00:00", "%F %T"), my.date[length(my.date)], by="month"),
```

```
 format="%b %Y", las=2)
## return to default plot region:
par(my.plot$op)

## A plot with different colors:
wt.image(my.w,
   legend.params=list(lab="wavelet power levels (quantiles)", lab.line=3.5,
                      label.digits=2),
   color.palette="gray((1:n.levels)/n.levels)", col.ridge="yellow")


## End(Not run)
```

---

wt.phase.image — *Image plot of the phases of periodic components for a single time series*

Description

This function plots the wavelet phase image for a time series, which is provided by an object of class `analyze.wavelet`, or alternatively of class `analyze.coherency`. (In the latter case, the series number or name can be specified.) The vertical axis shows the Fourier periods. The horizontal axis shows time step counts, but can be easily transformed into a calendar axis if dates are provided in either rownames or as a variable named date in the data frame at hand. Both axes can be relabeled.

The color levels are defined according to equidistant breakpoints (covering the interval from -pi to +pi), with the number of levels as a further parameter. In addition, there is an option to adopt an individual color palette.

If the default palette is retained, colors indicate the following. Green: Phases close to zero. Red: phases close to +pi. Blue: phases close to -pi.

Further plot design options concern: plot of the cone of influence, plot of contour lines to border areas of significance with respect to cross-wavelet power or wavelet coherency at a given significance level, plot of power ridges.

Finally, there is an option to insert and format a color legend (a right-hand vertical color bar) and to set the plot title. For further processing of the plot, graphical parameters of plot regions are provided as output.

Usage

```
wt.phase.image(WT, my.series = 1,
       plot.coi = T, plot.contour = T, siglvl = 0.1, col.contour = "white",
       plot.ridge = T, col.ridge = "black",
       n.levels = 100, color.palette = "rainbow(n.levels, start=0, end=.7)",
       useRaster = T, max.contour.segments = 250000,
       plot.legend = T,
       legend.params = list(width=1.2, shrink=0.9, mar=5.1,
                            n.ticks=6, label.digits=2, label.format="f",
                            lab=NULL, lab.line=2.5),
```

```
label.time.axis = T, show.date = F, date.format = NULL, timelab = NULL,
label.period.axis = T, periodlab = NULL,
main = NULL,
graphics.reset = T,
verbose = F)
```

Arguments

`WT`	an object of class `analyze.wavelet` or `analyze.coherency`
`my.series`	In case class(WT) = `analyze.coherency`: number (1 or 2) or name of the series to be analyzed. Default: 1.
`plot.coi`	Plot cone of influence? Logical. Default: TRUE
`plot.contour`	Plot contour lines to border the area of wavelet power significance at level siglvl? Logical. Default: TRUE.
`siglvl`	level of wavelet power significance applied to the plot of contour lines. Default: 0.1.
`col.contour`	Color of contour lines. Default: "white".
`plot.ridge`	Plot the wavelet power ridge? Logical. Default: TRUE.
`col.ridge`	Color of the power ridge. Default: "black".
`n.levels`	Number of color levels. Default: 100.
`color.palette`	Definition of color levels. (It will be assigned to levels in reverse order!) Default: "rainbow(n.levels, start=0, end=.7)".
`useRaster`	Use a bitmap raster instead of polygons to plot the wavelet power image? Logical. Default: TRUE.
`max.contour.segments`	limit on the number of segments in a single contour line, positive integer. Default: 250000 (`options(...)` default settings: 25000)
`plot.legend`	Plot color legend (a vertical bar of colors and breakpoints)? Logical. Default: TRUE
`legend.params`	a list of parameters for the plot of color legend, parameter values can be set selectively (style in parts adopted from image.plot in the R package "fields" by Douglas Nychka):

width	:	width of legend bar. Default: 1.2.
shrink	:	a vertical shrinkage factor. Default: 0.9.
mar	:	right margin of legend bar. Default: 5.1.
n.ticks	:	number of ticks for labels. Default: 6.
label.digits	:	digits of labels. Default: 2.
label.format	:	format of labels. Default: "f".

lab	:	axis label. Default: NULL.
lab.line	:	line (in user coordinate units) where to put the axis label. Default: 2.5.

`label.time.axis`
: Label the time axis? Logical. Default: TRUE.

`show.date`
: Show calendar dates? (Effective only if dates are available as rownames or as variable `date` in the data frame analyzed using `analyze.wavelet`.) Logical. Default: FALSE.

`date.format`
: the format of date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see `strptime` for a list of implemented date conversion specifications. (If not specified, as.Date will be applied.) Default: NULL.

`timelab`
: Time axis label. Default: "time".

`label.period.axis`
: Label the (Fourier) period axis? Logical. Default: TRUE.

`periodlab`
: (Fourier) period axis label. Default: "period".

`main`
: an overall title for the plot. Default: NULL.

`graphics.reset`
: Reset graphical parameters? Logical. Default: TRUE

`verbose`
: Print verbose output on the screen? Logical. Default: FALSE.

Value

A list of class `graphical parameters` with the following elements:

`op`
: original graphical parameters

`image.plt`
: image plot region

`legend.plt`
: legend plot region

Author(s)

Angi Roesch and Harald Schmidbauer

References

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Carmona R., Hwang W.-L., and Torresani B., 1998. Practical Time Frequency Analysis. Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. Journal of Atmospheric and Oceanic Technology 24, 2093–2102.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

See Also

analyze.wavelet, wt.image, wt.avg, wt.sel.phases, reconstruct

Examples

```
## Not run:
## The following example is adopted from Liu et al, 2007:

series.length = 6*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 8*24, length = series.length)
x3 = periodic.series(start.period = 32*24, length = series.length)
x4 = periodic.series(start.period = 128*24, length = series.length)
x = x1 + x2 + x3 + x4

plot(ts(x, start=0, frequency=24), type="l",
  xlab="time (days)", ylab="hourly data",
  main="a series of hourly data with periods of 1, 8, 32, and 128 days")

my.data = data.frame(x=x)

my.w = analyze.wavelet(my.data, "x",
                       loess.span=0,
                       dt=1/24, dj=1/20,
                       lowerPeriod=1/4,
                       make.pval=T, n.sim=10)

## Plot of wavelet power spectrum with equidistant color breakpoints:
wt.image(my.w, color.key="i",
   legend.params=list(lab="wavelet power levels (equidistant levels)"))

## Image of phases:
wt.phase.image(my.w, timelab="time (days)", main = "image of phases")

## End(Not run)
```

---

wt.sel.phases *Plot phases for selected periodic components of a single time series*

Description

This function plots the phases for selected periodic components of a time series, which are provided by an object of class analyze.wavelet.

Periodic components can be selected by specification of a single Fourier period or of a period band. In the latter case, and in the default case (no specification at all), there is an option to average the phases across periods. Other options: restriction to the cone of influence, restriction to an area of significance (with respect to wavelet power).

(The time axis can be altered to give dates, see e.g. wt.image.)

Usage

```
wt.sel.phases(WT, sel.period = NULL, sel.lower = NULL, sel.upper = NULL,
       only.coi = F,
       only.sig = T, siglvl=0.05,
       show.avg.phase = F, phase.avg.col = "black",
       label.time.axis = T, show.date = F, date.format = NULL, timelab = NULL,
      label.phase.axis = T, phaselab = NULL, main = NULL, sub = NULL, verbose = F)
```

Arguments

`WT`	an object of class `analyze.wavelet`.
`sel.period`	a single number which determines the (closest available) Fourier period to be selected. Default: NULL.
`sel.lower`	a lower number which determines the lower (closest available) Fourier period to be selected if `sel.period` is NULL. Default: NULL.
`sel.upper`	an upper number which determines the upper (closest available) Fourier period to be selected if `sel.period` is NULL. Default: NULL.
`only.coi`	Exclude borders influenced by edge effects, i.e. include the cone of influence only? Logical. Default: FALSE.
`only.sig`	Use wavelet power significance to decide about the inclusion of (parts of) the phases' series? Logical. Default: TRUE.
`siglvl`	level of wavelet power significance. Default: 0.05.
`show.avg.phase`	Average phases over selected periods? (Effective only if a band of periods is selected.) Logical. Default: FALSE.
`phase.avg.col`	Color for the plot of phase averages. Default: "black".
`label.time.axis`	Label the time axis? Logical. Default: TRUE.
`show.date`	Show calendar dates? (Effective only if dates are available as rownames or by variable `date` in the data frame which has been analyzed using `analyze.wavelet`.) Logical. Default: FALSE.
`date.format`	the format of date given as a character string, e.g. "%Y-%m-%d %H:%M:%S", or equivalently "%F %T"; see `strptime` for a list of implemented date conversion specifications. (If not specified, as.Date will be applied.) Default: NULL.
`timelab`	Time axis label. Default: "time".
`label.phase.axis`	Label the phase axis? Logical. Default: TRUE.
`phaselab`	Phase axis label. Default: "phase".
`main`	an overall title for the plot. Default: NULL.
`sub`	a subtitle for the plot. Default: NULL. In this case, the selected period range will be given in the subtitle.
`verbose`	Print verbose output on the screen? Logical. Default: FALSE.

Value

A list of class "sel.phases" with the following elements:

Period	the selected period (or period band)
Phase	time series of (average) phases at the selected period (or period band)
only.coi	Is the influence of edge effects excluded? I.e. is the cone of influence used only?
only.sig	Was wavelet power significance used in selection of phases?
siglvl	level of wavelet power significance
date	time series of dates (if available)
time.axis	tick levels corresponding to the time steps used for wavelet transformation

Author(s)

Angi Roesch and Harald Schmidbauer

References

Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

Carmona R., Hwang W.-L., and Torresani B., 1998. Practical Time Frequency Analysis. Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

Cazelles B., Chavez M., Berteaux, D., Menard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. Oecologia 156, 287–304.

Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. Journal of Atmospheric and Oceanic Technology 24, 2093–2102.

Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society 79 (1), 61–78.

See Also

analyze.wavelet, wt.image, wt.avg, wt.phase.image, reconstruct

Examples

```
## Not run:
## The following example is adopted from Liu et al, 2007:

series.length = 6*128*24
x1 = periodic.series(start.period = 1*24, length = series.length)
x2 = periodic.series(start.period = 8*24, length = series.length)
x3 = periodic.series(start.period = 32*24, length = series.length)
x4 = periodic.series(start.period = 128*24, length = series.length)
x = x1 + x2 + x3 + x4

plot(ts(x, start=0, frequency=24), type="l",
 xlab="time (days)",
 ylab="hourly data", main="a series of hourly data with periods of 1, 8, 32, and 128 days")
```

```
my.date = seq(as.POSIXct("2014-10-14 00:00:00","%F %T"), by="hour",
              length.out=series.length)
my.data = data.frame(date=my.date, x=x)

my.w = analyze.wavelet(my.data, "x", loess.span=0, dt=1/24, dj=1/20,
                       lowerPeriod=1/4, make.pval=T, n.sim=10)

## Plot of wavelet power spectrum (with equidistant color breakpoints):
wt.image(my.w, color.key="i", timelab="time (days)",
   legend.params=list(lab="wavelet power levels (equidistant levels)"))

## Select period 16 and plot corresponding phases across time:
wt.sel.phases(my.w, timelab="time (days)", sel.period=8)

## The same plot, but with calendar axis:
wt.sel.phases(my.w, timelab="", sel.period=8,
              show.date=T, date.format="%F %T")

## In the following, no periods are selected;
## the plot shows average phases instead of individual phases:
wt.sel.phases(my.w, timelab="time (days)", show.avg.phase=T)


## End(Not run)
```

Index