Package 'WaveletComp'

March 18, 2018

Version 1.1

Date 2018-03-18

Title Computational Wavelet Analysis

Author Angi Roesch <angi@angi-stat.com> and Harald Schmidbauer <harald@hs-stat.com>

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

NeedsCompilation no

Repository CRAN

URL Guide booklet at

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

Date/Publication 2018-03-18 14:31:53 UTC

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:	1.1
Date:	2018-03-18
License:	GPL-2
URL:	Guide booklet at 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.

What is new in WaveletComp version 1.1?

Tools for displaying and analyzing periodic phenomena across time have been extended. The main innovations are:

- All functions of family wt.<> (showing results concerning a single time series) can now also be applied to extract univariate outcomes from cross-wavelet and coherence analysis (objects of class "analyze.coherency").
- It is possible to control the color gradation of time-period spectrum plots, and accentuate the contrast, by raising the wavelet power values to any (positive) exponent before plotting.
- Setting a maximum level for the color bar facilitates the visual comparison of time-period spectrum plots. Maximum and minimum plot levels are options for plots of averages too.
- The time and period axes are now easier to individualize by specifying tick marks and labels. Coordinates on the time axis can be conveniently addressed via an index or a "POSIXct" object.
- Graphical parameters of global coverage (cex.axis, font.axis, cex.lab, font.lab, mgp etc., see par) as well as parameters of local coverage (within axis specification options) help fine-tune plots.
- Two more real-world data sets have been included in WaveletComp, namely:
  - Data set "weather.radiation.Mannheim", containing daily weather and ambient radiation readings from Mannheim (Germany).
  - Data set "USelection2016.Instagram", containing hourly numbers of candidate-related media uploads to Instagram right before the 2016 US presidential election.

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 https://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.

Schmidbauer H., Roesch A., Stieler F., 2018. The 2016 US presidential election and media on Instagram: Who was in the lead? Computers in Human Behavior 81, 148–160. doi: 10.1016/j.chb.2017.11.021

Tian, H., and Cazelles, B., 2012. WaveletCo. Available at https://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 = TRUE, method = "white.noise", params = NULL,
                  n.sim = 100,
                  date.format = NULL, date.tz = NULL,
                  verbose = TRUE)
```

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 the series 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 are to be detrended; no detrending if `loess.span = 0`. Default: `0.75`.
`dt`	time resolution, i.e. sampling resolution in the time domain, `1/dt` = number of observations per time unit. For example: a natural choice of `dt` in case of hourly data is `dt = 1/24`, resulting in one time unit equaling one day. This is also the time unit in which periods are measured. If `dt = 1`, the time interval between two consecutive observations will equal one time unit. Default: `1`.
`dj`	frequency resolution, i.e. sampling resolution in the frequency domain, `1/dj` = number of suboctaves (voices per octave). Default: `1/20`.
`lowerPeriod`	lower Fourier period (measured in time units determined by `dt`, see the explanations concerning `dt`) for wavelet decomposition. If `dt = 1`, the minimum admissible value is 2. Default: `2*dt`.
`upperPeriod`	upper Fourier period (measured in time units determined by `dt`, see the explanations concerning `dt`) 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, measured in time units determined by dt, see the explanations concerning dt.

Default: 5, which together with dt = 1 defines a window of length 5*(1/dt) = 5, equaling 5 observations (observation epochs). Windows of even-numbered sizes are extended by 1.

window.size.s size of the window used for smoothing in scale (period) 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 includes two lists named AR and ARIMA:

- AR = list(...), a list containing one single element:

p : AR order.
Default: 1.

- `ARIMA = list(...)`, a list of six elements:

`p`	:	AR order. Default: `1`.
`q`	:	MA order. Default: `1`.
`include.mean`	:	Include a mean/intercept term? Default: `TRUE`.
`sd.fac`	:	magnification factor to boost the residual standard deviation. Default: `1`.
`trim`	:	Simulate trimmed data? Default: `FALSE`.
`trim.prop`	:	high/low trimming proportion. Default: `0.01`.

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

`date.format` optional, and for later reference: the format of calendar date (if available in the input data frame) given as a character string, e.g. `"%Y-%m-%d"`, or equivalently `"%F"`; see `strptime` for a list of implemented date conversion specifications. Explicit information given here will be overwritten by any later specification given in e.g. `wc.image`. If unspecified, date formatting will be attempted according to `as.Date`.
Default: `NULL`.

`date.tz` optional, and for later reference: a character string specifying the time zone of calendar date (if available in the input data frame); see `strptime`. Explicit information given here will be overwritten by any specification given in e.g. `wc.image`. If unspecified, `""` (the local time zone) will be used.
Default: `NULL`.

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

Value

A list of class `"analyze.coherency"` with elements of different dimensions.

The elements of matrix type, namely:

- `Wave.xy`, `Angle`, `sWave.xy`, `sAngle`,
- `Power.xy`, `Power.xy.pval`,
- `Coherency`, `Coherence`, `Coherence.pval`,
- `Wave.x`, `Wave.y`, `Phase.x`, `Phase.y`, `Ampl.x`, `Ampl.y`,
- `Power.x`, `Power.y`, `Power.x.pval`, `Power.y.pval`, `sPower.x`, `sPower.y`,
- `Ridge.xy`, `Ridge.co`, `Ridge.x`, `Ridge.y`,

have the following structure:
columns correspond to observations (observation epochs; "epoch" meaning point in time), rows correspond to scales (Fourier periods) whose values are given in `Scale` (`Period`).

Here is a detailed list of all elements:

`series` a data frame with the following columns:

date	:	the calendar date (if available as column 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 taken over from `my.data`, and so are dates if given as row names.

`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 in the time domain, `1/dt` = number of observations per time unit

`dj` frequency resolution, i.e. sampling resolution in the 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` minus `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 matrix of `0`s and `1`s

`Ridge.co` ridge of wavelet coherence

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

`Period` the Fourier periods (measured in time units determined by `dt`, see the explanations concerning `dt`)

`Scale` the scales (the Fourier periods divided by the Fourier factor)

`nc` number of columns = number of observations = number of observation epochs; "epoch" meaning point in time

`nr` number of rows = number of scales (Fourier periods)

`coi.1, coi.2` borders of the region where the wavelet transforms are not influenced by edge effects (cone of influence). The coordinates of the borders are expressed in terms of internal axes `axis.1` and `axis.2`.

`axis.1` tick levels corresponding to the time steps used for (cross-)wavelet transformation: `1, 1+dt, 1+2dt, ...`. The default time axis in plot functions provided by `WaveletComp` is determined by observation epochs, however; "epoch" meaning point in time.

`axis.2` tick levels corresponding to the log of Fourier periods: `log2(Period)`. This determines the period axis in plot functions provided by `WaveletComp`.

`date.format` the format of calendar date (if available)

`date.tz` the time zone of calendar date (if available)

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 https://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 https://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.image, wc.avg, wc.sel.phases, wc.phasediff.image, wt.image, wt.avg, wt.sel.phases, wt.phase.image, reconstruct

Examples

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

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 + rnorm(series.length)
y <- x1 + x2 + x3 + x4 - 3*x5 + x6 + 3*x7 + x8 + rnorm(series.length)

matplot(data.frame(x, y), type = "l", lty = 1, xaxs = "i", col = 1:2,
 xlab = "index", ylab = "",
 main = "hourly series with periods of 1, 2, 4, 8, 16, 32, 64, 128 days",
 sub = "(out of phase at period 16, different amplitudes at period 64)")
legend("topright", legend = c("x","y"), col = 1:2, lty = 1)

## The following dates refer to the local time zone
```

```
## (possibly allowing for daylight saving time):
my.date <- seq(as.POSIXct("2014-10-14 00:00:00", format = "%F %T"),
               by = "hour",
               length.out = series.length)
my.data <- data.frame(date = my.date, x = x, y = y)

## Computation of cross-wavelet power and wavelet coherence, x over y:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
## There is an option to store the date format and time zone as additional
## parameters within object 'my.wc' for later reference.

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 = TRUE, n.sim = 10,
                           date.format = "%F %T", date.tz = "")
## Note:
## By default, Bartlett windows are used for smoothing in order to obtain
## the wavelet coherence spectrum; window lengths in this example:
## 1*24 + 1 = 25 observations in time direction,
## (1/2)*20 + 1 = 11 units in scale (period) direction.

## Plot of cross-wavelet power
## (with color breakpoints according to quantiles):
wc.image(my.wc, main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels"),
   periodlab = "period (days)")

## The same plot, now with calendar axis
## (according to date format stored in 'my.wc'):
wc.image(my.wc, main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels"),
   periodlab = "period (days)", show.date = TRUE)

## Plot of average cross-wavelet power:
wc.avg(my.wc, siglvl = 0.05, sigcol = 'red',
   periodlab = "period (days)")

## Plot of wavelet coherence
## (with color breakpoints according to quantiles):
wc.image(my.wc, which.image = "wc",  main = "wavelet coherence, x over y",
   legend.params = list(lab = "wavelet coherence levels",
                        lab.line = 3.5, label.digits = 3),
   periodlab = "period (days)")

## plot of average coherence:
wc.avg(my.wc, which.avg = "wc",
   siglvl = 0.05, sigcol = 'red',
   legend.coords = "topleft",
```

```
   periodlab = "period (days)")

## Please see our guide booklet for further examples:
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.


## 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 = TRUE, method = "white.noise", params = NULL,
                n.sim = 100,
                date.format = NULL, date.tz = NULL,
                verbose = TRUE)
```

Arguments

my.data	data frame of time series (including header, and dates as row names 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 detrended; no detrending if loess.span = 0. Default: 0.75.
dt	time resolution, i.e. sampling resolution in the time domain, 1/dt = number of observations per time unit. For example: a natural choice of dt in case of hourly data is dt = 1/24, resulting in one time unit equaling one day. This is also the time unit in which periods are measured. If dt = 1, the time interval between two consecutive observations will equal one time unit. Default: 1.
dj	frequency resolution, i.e. sampling resolution in the frequency domain, 1/dj = number of suboctaves (voices per octave). Default: 1/20.
lowerPeriod	lower Fourier period (measured in time units determined by dt, see the explanations concerning dt) for wavelet decomposition. If dt = 1, the minimum admissible value is 2. Default: 2*dt.
upperPeriod	upper Fourier period (measured in time units determined by dt, see the explanations concerning dt) 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 applying to surrogates. Default: NULL. Default includes two lists named AR and ARIMA: • AR = list(...), a list containing one single element: p : AR order. Default: 1.

- `ARIMA = list(...)`, a list of six elements:

`p`	:	AR order. Default: `1`.
`q`	:	MA order. Default: `1`.
`include.mean`	:	Include a mean/intercept term? Default: `TRUE`.
`sd.fac`	:	magnification factor to boost the residual standard deviation. Default: `1`.
`trim`	:	Simulate trimmed data? Default: `FALSE`.
`trim.prop`	:	high/low trimming proportion. Default: `0.01`.

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

`date.format` optional, and for later reference: the format of calendar date (if available in the input data frame) given as a character string, e.g. `"%Y-%m-%d"`, or equivalently `"%F"`; see `strptime` for a list of implemented date conversion specifications. Explicit information given here will be overwritten by any later specification given in e.g. `wt.image`. If unspecified, date formatting will be attempted according to `as.Date`.
Default: `NULL`.

`date.tz` optional, and for later reference: a character string specifying the time zone of calendar date (if available in the input data frame); see `strptime`. Explicit information given here will be overwritten by any specification given in e.g. `wt.image`. If unspecified, `""` (the local time zone) will be used.
Default: `NULL`.

`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 elements of different dimensions.

The elements of matrix type (namely, `Wave`, `Phase`, `Ampl`, `Power`, `Power.pval`, `Ridge`) have the following structure:
columns correspond to observations (observation epochs; "epoch" meaning point in time), rows correspond to scales (Fourier periods) whose values are given in `Scale` (`Period`).

Here is a detailed list of all elements:

`series` a data frame with the following columns:

date	:	the calendar date (if available as column in `my.data`)
<x>	:	the series which has been analyzed (detrended, if `loess.span != 0`; original name 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` controlling the degree of time series smoothing if the time series was detrended; no detrending if `loess.span = 0`

`dt` time resolution, i.e. sampling resolution in the time domain, `1/dt` = number of observations per time unit

`dj` frequency resolution, i.e. sampling resolution in the 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` wavelet power ridge, in the form of a matrix of `0`s and `1`s

`Period` the Fourier periods (measured in time units determined by `dt`, see the explanations concerning `dt`)

`Scale` the scales (the Fourier periods divided by the Fourier factor)

`nc` number of columns = number of observations = number of observation epochs; "epoch" meaning point in time

`nr` number of rows = number of scales (Fourier periods)

`coi.1, coi.2` borders of the region where the wavelet transforms are not influenced by edge effects (cone of influence). The coordinates of the borders are expressed in terms of internal axes `axis.1` and `axis.2`.

`axis.1` tick levels corresponding to the time steps used for (cross-)wavelet transformation: `1, 1+dt, 1+2dt, ...`. The default time axis in plot functions provided by `WaveletComp` is determined by observation epochs, however; "epoch" meaning point in time.

`axis.2` tick levels corresponding to the log of Fourier periods: `log2(Period)`. This determines the period axis in plot functions provided by `WaveletComp`.

`date.format` the format of calendar date (if available)

`date.tz` the time zone of calendar date (if available)

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 https://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.image`, `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(x, type = "l", xlab = "index", ylab = "", xaxs = "i",
     main = "hourly series with periods of 1, 8, 32, 128 days")

## The following dates refer to the local time zone
## (possibly allowing for daylight saving time):
my.date <- seq(as.POSIXct("2014-10-14 00:00:00", format = "%F %T"),
               by = "hour",
               length.out = series.length)
my.data <- data.frame(date = my.date, x = x)

## Computation of wavelet power:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
```

```
## There is an option to store the date format and time zone as additional
## parameters within object 'my.wt' for later reference.

my.wt <- analyze.wavelet(my.data, "x",
                         loess.span = 0,
                         dt = 1/24, dj = 1/20,
                         lowerPeriod = 1/4,
                         make.pval = TRUE, n.sim = 10,
                         date.format = "%F %T", date.tz = "")

## Plot of wavelet power spectrum (with equidistant color breakpoints):
wt.image(my.wt, color.key = "interval", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels"),
   periodlab = "period (days)")

## Plot of average wavelet power:
wt.avg(my.wt, siglvl = 0.05, sigcol = "red",
   periodlab = "period (days)")

## Please see our guide booklet for further examples:
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.


## 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$date, format = "%F %T", tz = "GMT"),
     FXtrade.transactions$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 = FALSE)
```

Arguments

start.period	period length at start (in steps of time). Default: 100.
end.period	period length at end (in steps of time). Default: 100.
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) is not too small.

Value

the series as vector

Author(s)

Angi Roesch and Harald Schmidbauer

See Also

analyze.wavelet, wt.image, wt.avg, wt.sel.phases, wt.phase.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 period changing linearly from 100 to 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 must be specified.)

Optional: plot of wavelets used for reconstruction, plot of reconstructed series against original (detrended) series. An option is given to individualize the time axis by specifying tick marks and labels.

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 = FALSE,
      only.sig = TRUE, siglvl = 0.05,
      only.ridge = FALSE,
      sel.period = NULL, sel.lower = NULL, sel.upper = NULL,
      rescale = TRUE,
```

```
plot.waves = FALSE, plot.rec = TRUE,
lty = 1, lwd = 1, col = 1:2, ylim = NULL,
show.legend = TRUE,
legend.coords = "topleft", legend.horiz = FALSE, legend.text = NULL,
label.time.axis = TRUE,
show.date = FALSE, date.format = NULL, date.tz = NULL,
timelab = NULL, timetck = 0.02, timetcl = 0.5,
spec.time.axis = list(at = NULL, labels = TRUE,
                      las = 1, hadj = NA, padj = NA),
main.waves = NULL, main.rec = NULL, main = NULL,
lwd.axis = 1,
verbose = TRUE)
```

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 to be applied for the inclusion of reconstruction waves. Default: 0.
only.coi	Exclude borders influenced by edge effects in reconstruction, i.e. include the cone of influence only? Logical. Default: FALSE.
only.sig	Use wavelet power significance in reconstruction? Logical. Default: TRUE.
siglvl	level of wavelet power significance to be applied for the inclusion of reconstruction waves. Default: 0.05.
only.ridge	Select only the wavelet power ridge? 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 = 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 = NULL.) Default: NULL.

`rescale` Shall the reconstructed series finally be rescaled to attain the original (detrended) series' mean and standard deviation? Logical.
Default: `TRUE`.

`plot.waves` Shall reconstruction waves be plotted? Logical.
Default: `FALSE`.

`plot.rec` Shall 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, providing the range of vertical coordinates for the plot.
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 row names or by variable `date` in the data frame which is analyzed.) Logical.
Default: `FALSE`.

`date.format` the format of calendar date given as a character string, e.g. `"%Y-%m-%d"`, or equivalently `"%F"`; see `strptime` for a list of implemented date conversion specifications. Explicit information given here will overturn any specification stored in `WT`. If unspecified, date formatting is attempted according to `as.Date`.
Default: `NULL`.

`date.tz` a character string specifying the time zone of calendar date; see `strptime`. Explicit information given here will overturn any specification stored in `WT`. If unspecified, `""` (the local time zone) is used.
Default: `NULL`.

`timelab` Time axis label.
Default: `"index"`; in case of a calendar axis: `"calendar date"`.

`timetck` length of tick marks on the time axis as a fraction of the smaller of the width or height of the plotting region; see `par`. If `timetck >= 0.5`, `timetck` is interpreted as a fraction of the length of the time axis, so if `timetck = 1` (and `timetcl = NULL`), vertical grid lines will be drawn.
Setting `timetck = NA` is to use `timetcl = -0.5` (which is the R default setting of `tck` and `tcl`).
Default here: `0.02`.

`timetcl` length of tick marks on the time axis as a fraction of the height of a line of text; see `par`. With `timetcl = -0.5` (which is the R default setting of `tcl`), ticks will be drawn outward.
Default here: `0.5`.

`spec.time.axis` a list of tick mark and label specifications for individualized time axis labeling (only effective if `label.time.axis = TRUE`):

- `at`: locations of tick marks (when `NULL`, default plotting will be applied). Valid tick marks can be provided as numerical values or as dates. Dates are used only in the case `show.date = TRUE`, however, and date formats should conform to `as.Date` or the format given in `date.format`.
  Default: `NULL`.
- `labels`: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If `labels` is non-logical, `at` should be of same length.
  Default: `TRUE`.
- `las`: the style of axis labels, see `par`.
  Default: `1` (always horizontal).
- `hadj`: adjustment of labels horizontal to the reading direction, see `axis`.
  Default: `NA` (centering is used).
- `padj`: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see `axis`.
  Default: `NA` (centering is used).

Mismatches will result in a reset to default plotting.

`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`.

`lwd.axis` line width of axes.
Default: `1`.

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

Value

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

`series` a data frame building on `WT$series` with the following columns:

date	:	the calendar date (if available as column in WT$series)
<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$series, and so are dates if given as row names. If `WT` is of class `analyze.coherency`, the second series in the coherency analysis is retained; if `loess.span != 0`, the second series is retained in the detrended version, and the trend is retained as well.

`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` minimum level of wavelet power for waves (wave segments) to be included in the reconstruction

`only.coi` Was the influence of edge effects excluded? I.e. was the cone of influence used only?

`only.sig` Was wavelet power significance used in reconstruction?

`siglvl` level of wavelet power significance

`only.ridge` Was the wavelet power ridge used only?

`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 in the time domain, `1/dt` = number of observations per time unit

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

`Period` the Fourier periods (measured in time units determined by `dt`, see the explanations concerning `dt`)

`Scale` the scales (the Fourier periods divided by the Fourier factor)

`nc` number of columns = number of observations = number of observation epochs; "epoch" meaning point in time

`nr` number of rows = number of scales (Fourier periods)

`axis.1` tick levels corresponding to the time steps used for (cross-)wavelet transformation: `1, 1+dt, 1+2dt, ...`. The default time axis in plot functions provided by `WaveletComp` is determined by observation epochs, however; "epoch" meaning point in time.

axis.2	tick levels corresponding to the log of Fourier periods: log2(Period). This determines the period axis in plot functions provided by WaveletComp.
date.format	the format of calendar date (if available)
date.tz	the time zone of calendar date (if available)

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.image, wt.avg, wt.sel.phases, wt.phase.image, analyze.coherency, wc.image, wc.avg, wc.sel.phases, wc.phasediff.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(x, type = "l", xlab = "index", ylab = "", xaxs = "i",
     main = "hourly series with periods of 1, 8, 32, 128 days")

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

## Computation of wavelet power:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
my.w <- analyze.wavelet(my.data, "x",
                        loess.span = 0,
                        dt = 1/24, dj = 1/20,
                        lowerPeriod = 1/4,
                        make.pval = TRUE, 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"),
   periodlab = "period (days)")

## Reconstruction of the time series,
## including significant components only:
reconstruct(my.w)

## The same reconstruction, but showing wave components first:
reconstruct(my.w, plot.waves = TRUE)

## Reconstruction, including all components whether significant or not:
reconstruct(my.w, only.sig = FALSE)

## Reconstruction, including significant components,
## but selected periods only (e.g. ignoring period 8):
reconstruct(my.w, sel.period = c(1,32,128))

## Reconstruction, including significant components,
## but the ridge only:
reconstruct(my.w, only.ridge = TRUE)

## Alternate styles of the time axis:

## The plot with time elapsed in days, starting from 0 and proceeding
## in steps of 50 days (50*24 hours),
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24

## Insert your specification of time axis:
reconstruct(my.w, only.ridge = TRUE,
   timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## See the periods involved:
my.rec <- reconstruct(my.w, only.ridge = TRUE)
print(my.rec$Period[my.rec$rnum.used])

## The original and reconstructed time series can be retrieved:
plot(my.rec$series$x, type = "l", xlab = "index", ylab = "")
lines(my.rec$series$x.r, col="red")
legend("topleft", legend = c("original","reconstructed"),
       lty = 1, col = c("black","red"))

## Please see also the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.


## 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 = TRUE, sd.fac = 1,
                      trim = FALSE, 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 = TRUE, sd.fac = 1, trim = FALSE, 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 https://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

See Also

analyze.wavelet, analyze.coherency, AR, ARIMA, FourierRand

USelection2016.Instagram

Hourly time series of the number of candidate-related media posted on Instagram during the week before the 2016 US presidential election

---

Description

One week (Sunday, 2016-10-30 23:00:00 EDT through Sunday, 2016-11-06 23:00:00 EST; 170 hours) of hourly readings of the number of media posted on Instagram, supposedly (according to hashtags with which they were annotated) positive/neutral or in opposition towards candidates Trump and Clinton. The 2016 US presidential election took place on Tuesday, 2016-11-08.

Usage

```
data("USelection2016.Instagram")
```

Format

A data frame of four columns:

date	:	hour of measurement (in EST5EDT, Eastern Time Zone), format: "%F %T"
trump.pos	:	number of media uploads with hashtag suggesting positive/neutral annotation to Trump
clinton.pos	:	number of media uploads with hashtag suggesting positive/neutral annotation to Clinton
trump.neg	:	number of media uploads with hashtag suggesting negative annotation to Trump
clinton.neg	:	number of media uploads with hashtag suggesting negative annotation to Clinton

Details

Media posted on Instagram are usually annotated with hashtags. A hashtag can be used to determine whether a candidate-related posting is positive/neutral towards a candidate (for example, #makeamericagreatagain for Trump, #hillary2016 for Clinton) or opposing a candidate (for example, #dumptrump for Trump, #neverhillary for Clinton). In this way, four hourly time series are obtained: Trump vs. Clinton, supporters vs. opponents. For further details, see the reference below. The date column also contains a label EDT (Eastern Daylight Time) or EST (Eastern Standard Time); daylight saving time ended 2016-11-06 at 1:00 a.m. when clocks were moved back to 1:00 a.m. EST. The time stamp "2016-11-06 02:00:00" therefore occurs twice, once with EDT and once with EST.

Source

Hourly readings of the number of media posted on Instagram were collected using Instagram's built-in API.

References

Schmidbauer H., Roesch A., Stieler F., 2018. The 2016 US presidential election and media on Instagram: Who was in the lead? Computers in Human Behavior 81, 148–160. doi: 10.1016/j.chb.2017.11.021

Examples

```
data(USelection2016.Instagram)
attach(USelection2016.Instagram)

my.date <- as.POSIXct(date, format = "%F %T", tz = "EST5EDT")

plot(my.date, trump.pos, type = "l", col = 1, lwd = 2,
     ylab = "number of media posted on Instagram", ylim = c(0,6e+6),
     xlab = "the week before the Election Day (Tuesday, 2016-11-08)")
lines(my.date, clinton.pos, col = 2, lwd = 2)
lines(my.date, trump.neg, col = 3, lwd = 2)
```

```
lines(my.date, clinton.neg, col = 4, lwd = 2)
legend("topleft", legend=names(USelection2016.Instagram[-1]),
       lty = 1, lwd = 2, col = 1:4)

detach(USelection2016.Instagram)
```

---

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. User-defined minimum and maximum levels can be applied to cross-wavelet power averages, minimum levels can be applied to coherence averages. Also, an option is given to individualize the period axis and/or axis of averages by specifying tick marks and labels.

There is an option to label periods according to significance of averages (if p-values are provided) 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", exponent = 1,
   show.siglvl = TRUE,
   siglvl = c(0.05, 0.1),
   sigcol = c("red", "blue"), sigpch = 20, sigcex = 1,
   minimum.level = NULL, maximum.level = NULL,
   label.avg.axis = TRUE,
   averagelab = NULL, averagetck = 0.02, averagetcl = 0.5,
   spec.avg.axis = list(at = NULL, labels = TRUE,
                        las = 1, hadj = NA, padj = NA),
   label.period.axis = TRUE,
   periodlab = NULL, periodtck = 0.02, periodtcl = 0.5,
   spec.period.axis = list(at = NULL, labels = TRUE,
                           las = 1, hadj = NA, padj = NA),
   show.legend = TRUE, legend.coords = "topright",
   main = NULL,
   lwd = 1, col = 1,
   lwd.axis = 1,
   verbose = FALSE)
```

Arguments

WC	an object of class "analyze.coherency".
which.avg	Which averages should be plotted? "wp" : cross-wavelet power "wc" : wavelet coherence Default: "wp".
exponent	Exponent applied to averages before plotting; the exponent should be positive. Default: 1.
show.siglvl	Label periods according to significance of averages? (Effective only if p-values are provided.) 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 colors 1 : length(siglvl)). Default: c("red","blue").
sigpch	a vector of plotting "characters" (symbols) to use as labels of significance. (It should be of same length as and correspond to siglvl to produce different plotting labels, otherwise the default setting is used. A single input value affects all labels.) Default: 20.
sigcex	a numerical vector working as size of labels of significance. (It should be of same length as and correspond to siglvl to produce different-sized labels, otherwise the default setting is used. A single input value affects all labels. Note that sigcex is affected by cex in par.) Default: 1.
minimum.level	Minimum plot level of cross-wavelet power or wavelet coherence averages considered. Default: NULL (referring to minimum level observed).
maximum.level	Maximum plot level of cross-wavelet power averages considered. Default: NULL (referring to maximum level observed). Wavelet coherence has maximum average level 1 by definition.
label.avg.axis	Label the axis of averages? Logical. Default: TRUE.
averagelab	Label for the axis of averages. Default: "average cross-wavelet power" or "average coherence", depending on which.avg.
averagetck	length of tick marks on the axis of averages as a fraction of the smaller of the width or height of the plotting region; see par. If averagetck >= 0.5, averagetck is interpreted as a fraction of the length of the axis of averages, so if averagetck = 1 (and averagetcl = NULL), vertical grid lines will be drawn.

Setting averagetck = NA is to use averagetcl = -0.5 (which is the R default setting of tck and tcl).

Default here: 0.02.

averagetcl length of tick marks on the axis of averages as a fraction of the height of a line of text; see par. With averagetcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward.

Default here: 0.5.

spec.avg.axis a list of tick mark and label specifications for individualized labeling of the axis of averages (only effective if label.avg.axis = TRUE):

- at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and non-negative values only. Default: NULL.
- labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length. Default: TRUE.
- las: the style of axis labels, see par. Default: 1 (always horizontal).
- hadj: adjustment of labels horizontal to the reading direction, see axis. Default: NA (centering is used).
- padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis. Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

label.period.axis

Label the (Fourier) period axis? Logical.

Default: TRUE.

periodlab (Fourier) period axis label.

Default: "period".

periodtck length of tick marks on the period axis as a fraction of the smaller of the width or height of the plotting region; see par. If periodtck >= 0.5, periodtck is interpreted as a fraction of the length of the period axis, so if periodtck = 1 (and periodtcl = NULL), horizontal grid lines will be drawn.
Setting periodtck = NA is to use periodtcl = -0.5 (which is the R default setting of tck and tcl).

Default here: 0.02.

periodtcl length of tick marks on the period axis as a fraction of the height of a line of text; see par. With periodtcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward.

Default here: 0.5.

spec.period.axis

a list of tick mark and label specifications for individualized period axis labeling (only effective if label.period.axis = TRUE):

at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and positive values only.
Default: NULL.

labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
Default: TRUE.

las: the style of axis labels, see par.
Default: 1 (always horizontal).

hadj: adjustment of labels horizontal to the reading direction, see axis.
Default: NA (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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 — width of line of averages.
Default: 1.

col — color of line of averages.
Default: "black".

lwd.axis — line width of axes.
Default: 1.

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 https://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, wt.image, wt.avg, wt.sel.phases, wt.phase.image, reconstruct

Examples

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

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 + rnorm(series.length)
y <- x1 + x2 + x3 + x4 - 3*x5 + x6 + 3*x7 + x8 + rnorm(series.length)

matplot(data.frame(x, y), type = "l", lty = 1, xaxs = "i", col = 1:2,
 xlab = "index", ylab = "",
 main = "hourly series with periods of 1, 2, 4, 8, 16, 32, 64, 128 days",
 sub = "(out of phase at period 16, different amplitudes at period 64)")
legend("topright", legend = c("x","y"), col = 1:2, lty = 1)

## The following dates refer to the local time zone
## (possibly allowing for daylight saving time):
my.date <- seq(as.POSIXct("2014-10-14 00:00:00", format = "%F %T"),
               by = "hour",
               length.out = series.length)
my.data <- data.frame(date = my.date, x = x, y = y)

## Computation of cross-wavelet power and wavelet coherence, x over y:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
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 = TRUE, n.sim = 10)

## Plot of cross-wavelet power,
## with color breakpoints according to quantiles:
wc.image(my.wc, main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (quantiles)"),
   periodlab = "period (days)")
## Note:
## The default time axis shows an index of given points in time,
## which is the count of hours in our example.
## By default, arrows are plotted which show the phase differences
## of x over y at respective significant periods.
## (Please see our guide booklet for further explanation.)

## With time elapsed in days
## (starting from 0 and proceeding in steps of 50 days)
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24

## Insert your specification of the time axis:
wc.image(my.wc, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (quantiles)"),
   periodlab = "period (days)", timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## Plot of average cross-wavelet power:
wc.avg(my.wc, siglvl = 0.05, sigcol = "red", periodlab = "period (days)")

## The same plot, but with enhanced symbol size, user-defined axes,
## minimum and a maximum plot level of averages:
wc.avg(my.wc, siglvl = 0.05, sigcol = "red", sigcex = 1.5,
   minimum.level = 0, maximum.level = 17,
   periodlab = "period (days)",
   spec.period.axis = list(at = c(1,2,4,8,16,32,64,128)),
   spec.avg.axis = list(at = seq(0,16,2)),
   lwd = 1.5)

## Another style of the plot:
## 'cex.axis' in 'par' controls for the size of axis tick labels,
## while 'cex.lab' controls for the size of axis and legend labels.
## Note that scaling by 'cex' would also affect 'sigcex'.
op <- par(no.readonly = TRUE)
par(cex.lab = 1.3, cex.axis = 1.1)
wc.avg(my.wc, siglvl = 0.05, sigcol = "red", sigcex = 1.5,
   minimum.level = 0, maximum.level = 17,
   periodlab = "period (days)",
   spec.period.axis = list(at = c(1,2,4,8,16,32,64,128)),
   spec.avg.axis = list(at = seq(0,16,2)),
   lwd = 1.5)
par(op)
```

```
## Plot of wavelet coherence
## (with color breakpoints according to quantiles):
wc.image(my.wc, which.image = "wc", main = "wavelet coherence, x over y",
   legend.params = list(label.digits = 3),
   periodlab = "period (days)")

## Plot of average wavelet coherence:
wc.avg(my.wc, which.avg = "wc",
   siglvl = 0.05, sigcol = "red", legend.coords = "topleft",
   periodlab = "period (days)",
   lwd = 1.5)

## The same plot, setting the minimum plot level of averages to 0:
wc.avg(my.wc, which.avg = "wc",
   siglvl = 0.05, sigcol = "red", legend.coords = "topleft",
   minimum.level = 0,
   periodlab = "period (days)",
   lwd = 1.5)

## Please see also the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.


## 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 can be easily transformed into a calendar axis if dates are provided in either row names or a variable named `"date"` in the data frame at hand. Both axes can be relabeled. In particular, an option is given to individualize the period and/or time axis by specifying tick marks and labels.

An option is given to raise cross-wavelet power (or wavelet coherence) values to any (positive) exponent before plotting in order to accentuate the contrast of the image.

The color levels can be defined according to quantiles of values or according to equidistant breakpoints (covering the interval from 0 to maximum level), with the number of levels as a further parameter. A user-defined maximum level can be applied to cross-wavelet power images. 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 coherence, 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 = TRUE,
     plot.contour = TRUE, siglvl.contour = 0.1, col.contour = "white",
     plot.ridge = FALSE, lvl = 0, col.ridge = "black",
     plot.arrow = TRUE, use.sAngle = FALSE,
     p = 1,
     which.arrow.sig = which.image,
     siglvl.arrow = 0.05, col.arrow = "black",
     clear.area = FALSE,
     which.area.sig = which.image, siglvl.area = 0.2,
     color.key = "quantile",
     n.levels = 100,
     color.palette = "rainbow(n.levels, start = 0, end = .7)",
     maximum.level = NULL,
     useRaster = TRUE, max.contour.segments = 250000,
     plot.legend = TRUE,
     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 = TRUE,
     show.date = FALSE, date.format = NULL, date.tz = NULL,
     timelab = NULL, timetck = 0.02, timetcl = 0.5,
     spec.time.axis = list(at = NULL, labels = TRUE,
                           las = 1, hadj = NA, padj = NA),
     label.period.axis = TRUE,
     periodlab = NULL, periodtck = 0.02, periodtcl = 0.5,
     spec.period.axis = list(at = NULL, labels = TRUE,
                             las = 1, hadj = NA, padj = NA),
     main = NULL,
     lwd = 2, lwd.axis = 1,
     graphics.reset = TRUE,
```

```
    verbose = FALSE)
```

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 applied to values before plotting in order to accentuate the contrast of the image; the exponent should be positive. Default: 1.
plot.coi	Plot cone of influence? Logical. Default: TRUE.
plot.contour	Plot contour lines to border the area of cross-wavelet power (or wavelet coherence, depending on which.image) significance? Logical. Default: TRUE.
siglvl.contour	level of cross-wavelet power (or wavelet coherence, depending on which.image) significance to be applied to the plot of contour lines. Default: 0.1.
col.contour	color of contour lines. Default: "white".
plot.ridge	Plot the cross-wavelet power (or wavelet coherence, depending on which.image) ridge? Logical. Default: FALSE.
lvl	minimum level of cross-wavelet power (or wavelet coherence, depending on which.image) for ridge or arrows to be plotted. (Only effective if plot.ridge = TRUE or, when setting p = 0 or p = 2, if plot.arrow = TRUE.) Default: 0.
col.ridge	ridge color. 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.) 0 : Select the area w.r.t. power (or coherence, depending on which.image. The value of lvl is the threshold. Set lvl = 0 to plot arrows over the entire area. 1 : Select the area w.r.t. significance as specified in which.arrow.sig. The value of siglvl.arrow is the threshold.

2 : Intersection of both areas above.

Default: 1.

which.arrow.sig

Which spectrum (and corresponding p-values) should be used to restrict the plot of arrows according to significance?

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

Default: which.image.

siglvl.arrow level of significance for arrows to be plotted.
(Only effective if plot.arrow = TRUE and p = 1 or p = 2.)
Default: 0.05.

col.arrow arrow color. Default: "black".

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

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

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

Default: which.image.

siglvl.area level of significance for the area to be cleared out.
(Only effective 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. (The color palette will be assigned to levels in reverse order!)
Default: "rainbow(n.levels, start = 0, end = .7)".

maximum.level Maximum plot level of cross-wavelet power considered; only effective in case of equidistant breakpoints (color.key equaling "i").

Default: NULL (referring to maximum level observed).
Wavelet coherence has maximum plot level 1 by definition.

useRaster Use a bitmap raster instead of polygons to plot the 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 the 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 row names or by variable date in the data frame which has been analyzed.) Logical.
Default: FALSE.

date.format the format of calendar date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see strptime for a list of implemented date conversion specifications. Explicit information given here will overturn any specification stored in WC. If unspecified, date formatting is attempted according to as.Date.
Default: NULL.

date.tz a character string specifying the time zone of calendar date; see strptime. Explicit information given here will overturn any specification stored in WC. If unspecified, "" (the local time zone) is used.
Default: NULL.

timelab
: Time axis label.
Default: "index"; in case of a calendar axis: "calendar date".

timetck
: length of tick marks on the time axis as a fraction of the smaller of the width or height of the plotting region; see par. If timetck >= 0.5, timetck is interpreted as a fraction of the length of the time axis, so if timetck = 1 (and timetcl = NULL), vertical grid lines will be drawn.
Setting timetck = NA is to use timetcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

timetcl
: length of tick marks on the time axis as a fraction of the height of a line of text; see par. With timetcl = -0.5 (which is the R default setting of tcl), ticks will be drawn outward.
Default here: 0.5.

spec.time.axis
: a list of tick mark and label specifications for individualized time axis labeling (only effective if label.time.axis = TRUE):

  at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical values or as dates. Dates are used only in the case show.date = TRUE, however, and date formats should conform to as.Date or the format given in date.format.
  Default: NULL.

  labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
  Default: TRUE.

  las: the style of axis labels, see par.
  Default: 1 (always horizontal).

  hadj: adjustment of labels horizontal to the reading direction, see axis.
  Default: NA (centering is used).

  padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
  Default: NA (centering is used).

  Mismatches will result in a reset to default plotting.

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

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

periodtck
: length of tick marks on the period axis as a fraction of the smaller of the width or height of the plotting region; see par. If periodtck >= 0.5, periodtck is interpreted as a fraction of the length of the period axis, so if periodtck = 1 (and periodtcl = NULL), horizontal grid lines will be drawn.
Setting periodtck = NA is to use periodtcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

periodtcl — length of tick marks on the period axis as a fraction of the height of a line of text; see par. With periodtcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward.
Default here: 0.5.

spec.period.axis — a list of tick mark and label specifications for individualized period axis labeling (only effective if label.period.axis = TRUE):

- at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and positive values only. Default: NULL.
- labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length. Default: TRUE.
- las: the style of axis labels, see par. Default: 1 (always horizontal).
- hadj: adjustment of labels horizontal to the reading direction, see axis. Default: NA (centering is used).
- padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis. Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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

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

lwd.axis — line width of axes (image and legend bar).
Default: 1.

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 https://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, wt.image, wt.avg, wt.sel.phases, wt.phase.image, reconstruct

Examples

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

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 + rnorm(series.length)
y <- x1 + x2 + x3 + x4 - 3*x5 + x6 + 3*x7 + x8 + rnorm(series.length)

matplot(data.frame(x, y), type = "l", lty = 1, xaxs = "i", col = 1:2,
 xlab = "index", ylab = "",
 main = "hourly series with periods of 1, 2, 4, 8, 16, 32, 64, 128 days",
 sub = "(out of phase at period 16, different amplitudes at period 64)")
legend("topright", legend = c("x","y"), col = 1:2, lty = 1)

## The following dates refer to the local time zone
## (possibly allowing for daylight saving time):
my.date <- seq(as.POSIXct("2014-10-14 00:00:00", format = "%F %T"),
               by = "hour",
```

```
               length.out = series.length)
my.data <- data.frame(date = my.date, x = x, y = y)

## Computation of cross-wavelet power and wavelet coherence, x over y:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
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 = TRUE, n.sim = 10)

## Plot of cross-wavelet power spectrum,
## with color breakpoints according to quantiles:
wc.image(my.wc,
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (quantiles)"),
   periodlab = "period (days)")
## Note:
## The default time axis shows an index of given points in time,
## which is the count of hours in our example.
## By default, arrows are plotted which show the phase differences
## of x over y at respective significant periods.
## (Please see our guide booklet for further explanation.)

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

## The same plot, but adopting a palette of gray colors,
## omitting the arrows:
wc.image(my.wc, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (equidistant)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   plot.arrow = FALSE,
   periodlab = "period (days)")

## The same plot, now with ridge of power:
wc.image(my.wc, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (equidistant)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   plot.arrow = FALSE,
   plot.ridge = TRUE, col.ridge = "red",
   periodlab = "period (days)")

## The plot, turning back to arrows, now in yellow color:
wc.image(my.wc, color.key = "i",
```

```
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (equidistant)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   col.arrow = "yellow",
   periodlab = "period (days)")

## Alternate styles of the time axis:

## The plot with time elapsed in days, starting from 0 and proceeding
## in steps of 50 days (50*24 hours), instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24

## Insert your specification of time axis:
wc.image(my.wc, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (equidistant)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   col.arrow = "yellow",
   periodlab = "period (days)", timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## The plot with (automatically produced) calendar axis:
wc.image(my.wc, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (equidistant)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   col.arrow = "yellow",
   periodlab = "period (days)",
   show.date = TRUE, date.format = "%F %T")

## Individualizing your calendar axis (works with show.date = TRUE)...
## How to obtain, for example, monthly date ticks and labels:

## The sequence of tick positions:
monthly.ticks <- seq(as.POSIXct("2014-11-01 00:00:00", format = "%F %T"),
                     as.POSIXct("2015-11-01 00:00:00", format = "%F %T"),
                     by = "month")
## Observe that the following specification may produce an error:
## 'seq(as.Date("2014-11-01"), as.Date("2015-11-01"), by = "month")'
## Time of the day is missing here!

## The sequence of labels (e.g. information on month and year only):
monthly.labels <- strftime(monthly.ticks, format = "%b %Y")

## Insert your specification of time axis as parameter to wc.image:
wc.image(my.wc, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (equidistant)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   col.arrow = "yellow",
   periodlab = "period (days)",
   show.date = TRUE, date.format = "%F %T",
```

```
   spec.time.axis = list(at = monthly.ticks, labels = monthly.labels,
                         las = 2))
## Note:
## The monthly ticks specify the midpoints of the colored cells and
## match the location of corresponding (default) time index ticks.

## A cross-wavelet power plot with individualized period axis and exponent
## to accentuate contrast in the image:
wc.image(my.wc, exponent = 0.5, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels
   (raised by exponent 0.5, equidistant levels)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   col.arrow = "yellow",
   periodlab = "period (days)",
   spec.period.axis = list(at = c(1,2,4,8,16,32,64,128)))

## An option to switch to the corresponding frequency axis:
my.periods <- c(1,2,4,8,16,32,64,128)
my.frequencies <- paste("1/",my.periods, sep = "")
wc.image(my.wc, exponent = 0.5, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels
   (raised by exponent 0.5, equidistant levels)"),
   color.palette = "gray( (1:n.levels)/n.levels )",
   col.arrow = "yellow",
   periodlab = "frequency (per day)",
   spec.period.axis = list(at = my.periods, labels = my.frequencies))

## Adding, for example, horizontal lines at period ticks...

## There is an option to add further objects to the image plot region,
## by setting 'graphics.reset = FALSE'
## (but recall previous par settings after plotting):

op <- par(no.readonly = TRUE)
wc.image(my.wc, exponent = 0.5, color.key = "i",
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels
   (raised by exponent 0.5, equidistant levels)"),
   color.palette="gray( (1:n.levels)/n.levels )",
   col.arrow = "yellow",
   periodlab = "frequency (per day)",
   spec.period.axis = list(at = my.periods, labels = my.frequencies),
   timelab = "",
   show.date = TRUE, date.format = "%F %T",
   graphics.reset = FALSE)
abline(h = log2(my.periods))
year2015 <- as.POSIXct("2015-01-01 00:00:00", format = "%F %T")
abline(v = year2015)
axis(1, at = year2015, labels = 2015, padj = 1)
par(op)
```

```
## For further axis plotting options:
## Please see the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.

## Plot of wavelet coherence of x over y,
## with color breakpoints according to quantiles:
wc.image(my.wc, which.image = "wc",
   main = "wavelet coherence, x over y",
   legend.params = list(lab = "wavelet coherence levels (quantiles)",
                        lab.line = 3.5, label.digits = 3),
   periodlab = "period (days)")

## Plot of wavelet coherence, but with equidistant color breakpoints:
wc.image(my.wc, which.image = "wc", color.key = "i",
   main = "wavelet coherence, x over y",
   legend.params = list(lab = "wavelet coherence levels (equidistant)"),
   periodlab = "period (days)")


## 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 row names or a variable named `"date"` in the data frame at hand. Both axes can be relabeled. In particular, an option is given to individualize the period and/or time axis by specifying tick marks and labels.

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 = FALSE,
   plot.coi = TRUE,
   plot.contour = TRUE, which.contour = "wp",
   siglvl = 0.1, col.contour = "white",
   n.levels = 100,
   color.palette = "rainbow(n.levels, start = 0, end = .7)",
   useRaster = TRUE, max.contour.segments = 250000,
   plot.legend = TRUE,
   legend.params = list(width = 1.2, shrink = 0.9, mar = 5.1,
                        n.ticks = 6,
                        pi.style = TRUE,
                        label.digits = 1, label.format = "f",
                        lab = NULL, lab.line = 3),
   label.time.axis = TRUE,
   show.date = FALSE, date.format = NULL, date.tz = NULL,
   timelab = NULL, timetck = 0.02, timetcl = 0.5,
   spec.time.axis = list(at = NULL, labels = TRUE,
                         las = 1, hadj = NA, padj = NA),
   label.period.axis = TRUE,
   periodlab = NULL, periodtck = 0.02, periodtcl = 0.5,
   spec.period.axis = list(at = NULL, labels = TRUE,
                           las = 1, hadj = NA, padj = NA),
   main = NULL,
   lwd = 2, lwd.axis = 1,
   graphics.reset = TRUE,
   verbose = FALSE)
```

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 (or wavelet coherence, depending on which.contour) significance? 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 (or wavelet coherence, depending on which.contour) significance to be 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. (The color palette 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 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 the 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`.

`pi.style`: style of labels, logical: if `TRUE`, the symbol `"pi"` is used to label the legend bar; otherwise, labels are numerals.
Default: `TRUE`.

`label.digits`: number of digits of (numerical factors of) labels.
Default: `1` if `pi.style` is `TRUE`, else `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: `3`.

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

`show.date` Show calendar dates? (Effective only if dates are available as row names or by variable `date` in the data frame which is analyzed.) Logical.
Default: `FALSE`.

`date.format` the format of calendar date given as a character string, e.g. `"%Y-%m-%d"`, or equivalently `"%F"`; see `strptime` for a list of implemented date conversion specifications. Explicit information given here will overturn any specification stored in `WC`. If unspecified, date formatting is attempted according to `as.Date`.
Default: `NULL`.

`date.tz` a character string specifying the time zone of calendar date; see `strptime`. Explicit information given here will overturn any specification stored in `WC`. If unspecified, `""` (the local time zone) is used.

Default: `NULL`.

`timelab` Time axis label.

Default: `"index"`; in case of a calendar axis: `"calendar date"`.

`timetck` length of tick marks on the time axis as a fraction of the smaller of the width or height of the plotting region; see `par`. If `timetck >= 0.5`, `timetck` is interpreted as a fraction of the length of the time axis, so if `timetck = 1` (and `timetcl = NULL`), vertical grid lines will be drawn.
Setting `timetck = NA` is to use `timetcl = -0.5` (which is the R default setting of `tck` and `tcl`).

Default here: `0.02`.

`timetcl` length of tick marks on the time axis as a fraction of the height of a line of text; see `par`. With `timetcl = -0.5` (which is the R default setting of `tcl`), ticks will be drawn outward.

Default here: `0.5`.

`spec.time.axis` a list of tick mark and label specifications for individualized time axis labeling (only effective if `label.time.axis = TRUE`):

`at`: locations of tick marks (when `NULL`, default plotting will be applied). Valid tick marks can be provided as numerical values or as dates. Dates are used only in the case `show.date = TRUE`, however, and date formats should conform to `as.Date` or the format given in `date.format`.
Default: `NULL`.

`labels`: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If `labels` is non-logical, `at` should be of same length.
Default: `TRUE`.

`las`: the style of axis labels, see `par`.
Default: `1` (always horizontal).

`hadj`: adjustment of labels horizontal to the reading direction, see `axis`.
Default: `NA` (centering is used).

`padj`: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see `axis`.
Default: `NA` (centering is used).

Mismatches will result in a reset to default plotting.

`label.period.axis`

Label the (Fourier) period axis? Logical.

Default: `TRUE`.

`periodlab` (Fourier) period axis label.

Default: `"period"`.

`periodtck` length of tick marks on the period axis as a fraction of the smaller of the width or height of the plotting region; see `par`. If `periodtck >= 0.5`, `periodtck` is interpreted as a fraction of the length of the period axis, so if `periodtck = 1`

(and periodtcl = NULL), horizontal grid lines will be drawn.
Setting periodtck = NA is to use periodtcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

periodtcl — length of tick marks on the period axis as a fraction of the height of a line of text; see par. With periodtcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward.
Default here: 0.5.

spec.period.axis — a list of tick mark and label specifications for individualized period axis labeling (only effective if label.period.axis = TRUE):

- at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and positive values only.
  Default: NULL.
- labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
  Default: TRUE.
- las: the style of axis labels, see par.
  Default: 1 (always horizontal).
- hadj: adjustment of labels horizontal to the reading direction, see axis.
  Default: NA (centering is used).
- padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
  Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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

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

lwd.axis — line width of axes (image and legend bar).
Default: 1.

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

Examples

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

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)
x3a <- periodic.series(start.period = 4*24, length = series.length)
x3b <- periodic.series(start.period = 4*24, length = series.length,
                       phase = 24)
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 + x3a + x4 + 3*x5 + x6 + x7 + x8 + rnorm(series.length)
y <- x1 + x2 + x3b + x4 - 3*x5 + x6 + 3*x7 + x8 + rnorm(series.length)

matplot(data.frame(x, y), type = "l", lty = 1, xaxs = "i", col = 1:2,
 xlab = "index", ylab = "",
 main = "hourly series with periods of 1, 2, 4, 8, 16, 32, 64, 128 days",
 sub = "(different phases at periods 4 and 16)")
legend("topright", legend = c("x","y"), col = 1:2, lty = 1)
```

```
## The following dates refer to the local time zone
## (possibly allowing for daylight saving time):
my.date <- seq(as.POSIXct("2014-10-14 00:00:00", format = "%F %T"),
               by = "hour",
               length.out = series.length)
my.data <- data.frame(date = my.date, x = x, y = y)

## Computation of cross-wavelet power and wavelet coherence, x over y:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
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 = TRUE, n.sim = 10)

## Plot of cross-wavelet power spectrum,
## with color breakpoints according to quantiles:
wc.image(my.wc,
   main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (quantiles)"),
   periodlab = "period (days)")

## Default plot of phase differences
## (with contour lines referring to cross-wavelet power)
wc.phasediff.image(my.wc, which.contour = "wp",
   main = "image of phase differences, x over y",
   periodlab = "period (days)")

## With time elapsed in days
## (starting from 0 and proceeding in steps of 50 days)
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24
wc.phasediff.image(my.wc, which.contour = "wp",
   main = "image of phase differences, x over y",
   periodlab = "period (days)",
   timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## The same plot, but with (automatically produced) calendar axis:
wc.phasediff.image(my.wc, which.contour = "wp",
   main = "image of phase differences, x over y",
   periodlab = "period (days)",
   show.date = TRUE, date.format = "%F %T")

## For further axis plotting options:
## Please see the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.

## Plot of phase difference with numerals as labels of the color legend bar:
```

```
wc.phasediff.image(my.wc,
   legend.params=list(pi.style = FALSE, label.digits = 2))


## 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`. In particular, an option is given to individualize the phase and/or time axis by specifying tick marks and labels.)

Usage

```
wc.sel.phases(WC, sel.period = NULL, sel.lower = NULL, sel.upper = NULL,
      only.coi = FALSE,
      only.sig = TRUE, which.sig = "wp", siglvl = 0.05,
      phase.cols = c("red", "blue"),
      show.Angle = TRUE, use.sAngle = FALSE, Angle.col = "black",
      show.legend = TRUE, legend.coords = "topleft", legend.horiz = TRUE,
      label.time.axis = TRUE,
      show.date = FALSE, date.format = NULL, date.tz = NULL,
      timelab = NULL, timetck = 0.02, timetcl = 0.5,
      spec.time.axis = list(at = NULL, labels = TRUE,
                            las = 1, hadj = NA, padj = NA),
      label.phase.axis = TRUE,
      phaselab = NULL, phasetck = 0.02, phasetcl = 0.5,
      spec.phase.axis = list(at = NULL, labels = TRUE,
                             las = 1, hadj = NA, padj = NA),
      phaselim = c(-pi,pi+show.legend*ifelse(legend.horiz,0.8,2)),
      main = NULL, sub = NULL,
      lwd = 1, lwd.Angle = 2, lwd.axis = 1,
      verbose = FALSE)
```

Arguments

WC	an object of class "analyze.coherency".
sel.period	a single number which determines the (closest available) Fourier period to be selected. Default: NULL.
sel.lower	a number to define a lower Fourier period (or the closest available) for the selection of a band of periods (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 periods (effective 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 wavelet coherence, depending on which.sig) significance to decide about the inclusion of (parts of) the series of phases? Logical. Default: TRUE.
which.sig	Which spectrum should significance refer to? "wp" : cross-wavelet power "wc" : wavelet coherence "wt" : individual wavelet power Default: "wp"
siglvl	level of cross-wavelet power (or wavelet coherence, depending on which.sig) 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 of the line 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 row names or by variable `date` in the data frame which is analyzed.) Logical.
Default: `FALSE`.

`date.format` the format of calendar date given as a character string, e.g. `"%Y-%m-%d"`, or equivalently `"%F"`; see `strptime` for a list of implemented date conversion specifications. Explicit information given here will overturn any specification stored in `WC`. If unspecified, date formatting is attempted according to `as.Date`.
Default: `NULL`.

`date.tz` a character string specifying the time zone of calendar date; see `strptime`. Explicit information given here will overturn any specification stored in `WC`. If unspecified, `""` (the local time zone) is used.
Default: `NULL`.

`timelab` Time axis label.
Default: `"index"`; in case of a calendar axis: `"calendar date"`.

`timetck` length of tick marks on the time axis as a fraction of the smaller of the width or height of the plotting region; see `par`. If `timetck >= 0.5`, `timetck` is interpreted as a fraction of the length of the time axis, so if `timetck = 1` (and `timetcl = NULL`), vertical grid lines will be drawn.
Setting `timetck = NA` is to use `timetcl = -0.5` (which is the R default setting of `tck` and `tcl`).
Default here: `0.02`.

`timetcl` length of tick marks on the time axis as a fraction of the height of a line of text; see `par`. With `timetcl = -0.5` (which is the R default setting of `tcl`), ticks will be drawn outward.
Default here: `0.5`.

`spec.time.axis` a list of tick mark and label specifications for individualized time axis labeling (only effective if `label.time.axis = TRUE`):

- `at`: locations of tick marks (when `NULL`, default plotting will be applied). Valid tick marks can be provided as numerical values or as dates. Dates are used only in the case `show.date = TRUE`, however, and date formats should conform to `as.Date` or the format given in `date.format`.
  Default: `NULL`.
- `labels`: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If `labels` is non-logical, `at` should be of same length.
  Default: `TRUE`.
- `las`: the style of axis labels, see `par`.
  Default: `1` (always horizontal).
- `hadj`: adjustment of labels horizontal to the reading direction, see `axis`.
  Default: `NA` (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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

phaselab
Phase axis label.
Default: "phase".

phasetck
length of tick marks on the phase axis as a fraction of the smaller of the width or height of the plotting region; see par. If phasetck >= 0.5, phasetck is interpreted as a fraction of the length of the phase axis, so if phasetck = 1 (and phasetcl = NULL), horizontal grid lines will be drawn.
Setting phasetck = NA is to use phasetcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

phasetcl
length of tick marks on the phase axis as a fraction of the height of a line of text; see par. With phasetcl = -0.5 (which is the R default setting of tcl), ticks will be drawn outward.
Default here: 0.5.

spec.phase.axis
a list of tick mark and label specifications for individualized phase axis labeling (only effective if label.phase.axis = TRUE):

at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical values between -pi and pi.
Default: NULL.

labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
Default: TRUE.

las: the style of axis labels, see par.
Default: 1 (always horizontal).

hadj: adjustment of labels horizontal to the reading direction, see axis.
Default: NA (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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.
lwd	width of lines of phases. Default: 1.
lwd.Angle	width of lines of (average) phase differences (the Angles) between the two series (this line will be plotted if show.Angle = TRUE). Default: 2.
lwd.axis	line width of axes. Default: 1.
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 calendar date (if available)
date.format	the format of calendar date as provided
date.tz	the time zone of calendar date as provided
axis.1	tick levels corresponding to the time steps used for (cross-)wavelet transformation: 1, 1+dt, 1+2dt, ...

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

Examples

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

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)
x3a <- periodic.series(start.period = 4*24, length = series.length)
x3b <- periodic.series(start.period = 4*24, length = series.length,
                       phase = 24)
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 + x3a + x4 + 3*x5 + x6 + x7 + x8 + rnorm(series.length)
y <- x1 + x2 + x3b + x4 - 3*x5 + x6 + 3*x7 + x8 + rnorm(series.length)

matplot(data.frame(x, y), type = "l", lty = 1, xaxs = "i", col = 1:2,
 xlab = "index", ylab = "",
 main = "hourly series with periods of 1, 2, 4, 8, 16, 32, 64, 128 days",
 sub = "(different phases at periods 4 and 16)")
legend("topright", legend = c("x","y"), col = 1:2, lty = 1)

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

```
## Computation of cross-wavelet power and wavelet coherency of x over y:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
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 = TRUE, n.sim = 10)

## Plot of cross-wavelet power spectrum,
## with color breakpoints according to quantiles:
wc.image(my.wc, main = "cross-wavelet power spectrum, x over y",
   legend.params = list(lab = "cross-wavelet power levels (quantiles)"),
   periodlab = "period (days)")

## 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, sel.period = 64, show.Angle = TRUE)

## With time elapsed in days
## (starting from 0 and proceeding in steps of 50 days)
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24
wc.sel.phases(my.wc, sel.period = 64, show.Angle = TRUE,
   timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## The same plot, but with (automatically produced) calendar axis:
wc.sel.phases(my.wc, sel.period = 64, show.Angle = TRUE,
   show.date = TRUE, date.format = "%F %T")

## For further axis plotting options:
## Please see the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.

## Now, select period 16...
## and observe that corresponding components are out of phase:
wc.sel.phases(my.wc, sel.period = 16, show.Angle = TRUE,
   show.date = TRUE, date.format = "%F %T")
## ... compare to period 4...
wc.sel.phases(my.wc, sel.period = 4, show.Angle = TRUE,
   show.date = TRUE, date.format = "%F %T")

## In the following, no periods are selected.
## In this case, instead of individual phases, the plot shows
## average phases for each series:
wc.sel.phases(my.wc)


## End(Not run)
```

`weather.radiation.Mannheim`

Series of weather data and ambient gamma dose rate readings

Description

Ten years (from January 2005 through December 2014; 3652 values) of daily mean temperature, relative humidity, and ambient gamma dose rate (German abbreviation: ODL) readings from Mannheim-Rheinau (Germany).

ODL data retrieved in 2015 from ODL-INFO http://odlinfo.bfs.de/download.php of the German Federal Office for Radiation Protection (in German: Bundesamt fuer Strahlenschutz, BfS), weather data from the German Meteorological Office (in German: Deutscher Wetterdienst, DWD) ftp://ftp-cdc.dwd.de/pub/CDC/observations_germany/climate/daily/kl/historical/. — We owe this example to our former student Nadiya Appelhans.

Usage

```
data("weather.radiation.Mannheim")
```

Format

A data frame of four columns:

date	:	day of measurement (in UTC, Coordinated Universal Time), format: "%Y-%m-%d"
temperature	:	mean daily temperature, in degrees Celsius
humidity	:	mean daily relative humidity, in percent
radiation	:	mean daily ambient gamma dose rate (ODL), in microsieverts per hour

Details

The ambient gamma dose rate is an equivalent dose representing the stochastic health effects of low levels of ionizing radiation on the human body. According to BfS, radioactivity is to be found everywhere in the environment. It may be of natural or artificial origin. — We drew Mannheim randomly from a set of places for which weather and radiation data were available; there is no conspicuous radiation in Mannheim.

Source

The Federal Office for Radiation Protection (BfS) Germany, http://www.bfs.de/EN/home/home_node.html, and in particular http://www.bfs.de/EN/topics/ion/environment/environment_node.html

ODL-INFO http://odlinfo.bfs.de/DE/service/downloadbereich.html

DWD (Deutscher Wetterdienst) https://www.dwd.de/EN/Home/home_node.html

Examples

```
data(weather.radiation.Mannheim)

plot(as.Date(weather.radiation.Mannheim$date, tz = "UTC"),
     weather.radiation.Mannheim$radiation, type = "l",
     xlab = "day",
     ylab = "mean daily ambient gamma dose rate (ODL) in microsieverts per hour")
```

`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 must be specified.) The vertical axis shows the Fourier periods. The horizontal axis shows the averages. User-defined minimum and maximum average levels can be applied. Also, an option is given to individualize the period axis and/or axis of averages by specifying tick marks and labels.

There is an option to label periods according to significance of averages (if p-values are provided) 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, exponent = 1,
  show.siglvl = TRUE, siglvl = c(0.05, 0.1),
  sigcol = c("red", "blue"), sigpch = 20, sigcex = 1,
  minimum.level = NULL, maximum.level = NULL,
  label.avg.axis = TRUE,
  averagelab = NULL, averagetck = 0.02, averagetcl = 0.5,
  spec.avg.axis = list(at = NULL, labels = TRUE,
                       las = 1, hadj = NA, padj = NA),
  label.period.axis = TRUE,
  periodlab = NULL, periodtck = 0.02, periodtcl = 0.5,
  spec.period.axis = list(at = NULL, labels = TRUE,
                          las = 1, hadj = NA, padj = NA),
  show.legend = TRUE, legend.coords = "topright",
  main = NULL,
  lwd = 1, col = 1,
  lwd.axis = 1,
  verbose = FALSE)
```

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.
exponent	Exponent applied to averages before plotting; the exponent should be positive. Default: 1.
show.siglvl	Label periods according to significance of averages? (Effective only if p-values are provided.) 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 colors 1 : length(siglvl)). Default: c("red","blue").
sigpch	a vector of plotting "characters" (symbols) to use as labels of significance. (It should be of same length as and correspond to siglvl to produce different plotting labels, otherwise the default setting is used. A single input value affects all labels.) Default: 20.
sigcex	a numerical vector working as size of labels of significance. (It should be of same length as and correspond to siglvl to produce different-sized labels, otherwise the default setting is used. A single input value affects all labels. Note that sigcex is affected by cex in par.) Default: 1.
minimum.level	Minimum plot level of wavelet power averages considered. Default: NULL (referring to minimum level observed).
maximum.level	Maximum plot level of wavelet power averages considered. Default: NULL (referring to maximum level observed).
label.avg.axis	Label the axis of averages? Logical. Default: TRUE.
averagelab	Label for the axis of averages. Default: "average wavelet power".
averagetck	length of tick marks on the axis of averages as a fraction of the smaller of the width or height of the plotting region; see par. If averagetck >= 0.5, averagetck is interpreted as a fraction of the length of the axis of averages, so if averagetck = 1 (and averagetcl = NULL), vertical grid lines will be drawn. Setting averagetck = NA is to use averagetcl = -0.5 (which is the R default setting of tck and tcl). Default here: 0.02.
averagetcl	length of tick marks on the axis of averages as a fraction of the height of a line of text; see par. With averagetcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward. Default here: 0.5.

spec.avg.axis a list of tick mark and label specifications for individualized labeling of the axis of averages (only effective if label.avg.axis = TRUE):

- at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and non-negative values only. Default: NULL.
- labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length. Default: TRUE.
- las: the style of axis labels, see par. Default: 1 (always horizontal).
- hadj: adjustment of labels horizontal to the reading direction, see axis. Default: NA (centering is used).
- padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis. Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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

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

periodtck length of tick marks on the period axis as a fraction of the smaller of the width or height of the plotting region; see par. If periodtck >= 0.5, periodtck is interpreted as a fraction of the length of the period axis, so if periodtck = 1 (and periodtcl = NULL), horizontal grid lines will be drawn.
Setting periodtck = NA is to use periodtcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

periodtcl length of tick marks on the period axis as a fraction of the height of a line of text; see par. With periodtcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward.
Default here: 0.5.

spec.period.axis
a list of tick mark and label specifications for individualized period axis labeling (only effective if label.period.axis = TRUE):

- at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and positive values only. Default: NULL.
- labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length. Default: TRUE.
- las: the style of axis labels, see par. Default: 1 (always horizontal).

hadj: adjustment of labels horizontal to the reading direction, see axis.
Default: NA (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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 — width of line of averages.
Default: 1.

col — color of line of averages.
Default: "black".

lwd.axis — line width of axes.
Default: 1.

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 https://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(x, type = "l", xlab = "index", ylab = "", xaxs = "i",
     main = "hourly series with periods of 1, 8, 32, 128 days")

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

## Computation of wavelet power:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
my.wt <- analyze.wavelet(my.data, "x", loess.span = 0,
                         dt = 1/24, dj = 1/20,
                         lowerPeriod = 1/4,
                         make.pval = TRUE, n.sim = 10)

## Plot of wavelet power spectrum (with equidistant color breakpoints):
wt.image(my.wt, color.key = "i", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (equidistant levels)"),
   periodlab = "period (days)")
## Note:
## The default time axis shows an index of given points in time,
## which is the count of hours in our example.

## With time elapsed in days
## (starting from 0 and proceeding in steps of 50 days)
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24

## Insert your specification of time axis:
wt.image(my.wt, color.key = "i", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (equidistant levels)"),
   periodlab = "period (days)", timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## Plot of average wavelet power:
wt.avg(my.wt, siglvl = 0.05, sigcol = "red", periodlab = "period (days)")

## The same plot, but with enhanced symbol size, user-defined period axis,
## and horizontal grid:
wt.avg(my.wt, siglvl = 0.05, sigcol = "red", sigcex = 1.3,
```

```
   periodlab = "period (days)",
   spec.period.axis = list(at = c(1,8,32,128)),
   periodtck = 1, periodtcl = NULL,
   lwd = 1.5, lwd.axis = 0.25)

## Another style of the plot:
## With user-defined period axis and axis of averages,
## minimum and maximum plot levels of averages:

op <- par(no.readonly = TRUE)
par(cex.lab = 1.3, cex.axis = 1.1)
wt.avg(my.wt, siglvl = 0.05, sigcol = "red", sigcex = 1.3,
   minimum.level = 0, maximum.level = 11,
   periodlab = "period (days)",
   spec.period.axis = list(at = c(1,8,32,128)),
   spec.avg.axis = list(at = 0:10),
   lwd = 1.5)
par(op)
## Note:
## 'cex.axis' in 'par' controls for the size of axis tick labels,
## while 'cex.lab' controls for the size of axis and legend labels.
## Scaling by 'cex' would also affect 'sigcex'.


## Please see also the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.


## 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 must 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 row names or as a variable named `"date"` in the data frame at hand. Both axes can be relabeled. In particular, an option is given to individualize the period and/or time axis by specifying tick marks and labels.

An option is given to raise wavelet power values to any (positive) exponent before plotting in order to accentuate the contrast of the image.

The color levels can be defined according to quantiles of values or according to equidistant breakpoints (covering the interval from 0 to maximum level), with the number of levels as a further parameter. A user-defined maximum level can be applied. 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, exponent = 1,
     plot.coi = TRUE,
     plot.contour = TRUE, siglvl = 0.1, col.contour = "white",
     plot.ridge = TRUE, lvl = 0, col.ridge = "black",
     color.key = "quantile",
     n.levels = 100,
     color.palette = "rainbow(n.levels, start = 0, end = .7)",
     maximum.level = NULL,
     useRaster = TRUE, max.contour.segments = 250000,
     plot.legend = TRUE,
     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 = TRUE,
     show.date = FALSE, date.format = NULL, date.tz = NULL,
     timelab = NULL, timetck = 0.02, timetcl = 0.5,
     spec.time.axis = list(at = NULL, labels = TRUE,
                           las = 1, hadj = NA, padj = NA),
     label.period.axis = TRUE,
     periodlab = NULL, periodtck = 0.02, periodtcl = 0.5,
     spec.period.axis = list(at = NULL, labels = TRUE,
                             las = 1, hadj = NA, padj = NA),
     main = NULL,
     lwd = 2, lwd.axis = 1,
     graphics.reset = TRUE,
     verbose = FALSE)
```

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.
`exponent`	Exponent applied to values before plotting in order to accentuate the contrast of the image; the exponent should be positive. Default: 1.

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

`plot.contour` Plot contour lines to border the area of wavelet power significance? Logical.
Default: `TRUE`.

`siglvl` level of wavelet power significance to be 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 ridge to be plotted.
Default: `0`.

`col.ridge` ridge color. Default: `"black"`.

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

`maximum.level` Maximum plot level of wavelet power considered; only effective in case of equidistant breakpoints (`color.key` equaling `"i"`).
Default: `NULL` (referring to maximum level observed).

`useRaster` Use a bitmap raster instead of polygons to plot the 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 the 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 row names or by variable date in the data frame which is analyzed.) Logical.
Default: FALSE.

date.format the format of calendar date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see strptime for a list of implemented date conversion specifications. Explicit information given here will overturn any specification stored in WT. If unspecified, date formatting is attempted according to as.Date.
Default: NULL.

date.tz a character string specifying the time zone of calendar date; see strptime. Explicit information given here will overturn any specification stored in WT. If unspecified, "" (the local time zone) is used.
Default: NULL.

timelab Time axis label.
Default: "index"; in case of a calendar axis: "calendar date".

timetck length of tick marks on the time axis as a fraction of the smaller of the width or height of the plotting region; see par. If timetck >= 0.5, timetck is interpreted as a fraction of the length of the time axis, so if timetck = 1 (and timetcl = NULL), vertical grid lines will be drawn.
Setting timetck = NA is to use timetcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

timetcl length of tick marks on the time axis as a fraction of the height of a line of text; see par. With timetcl = -0.5 (which is the R default setting of tcl), ticks will be drawn outward.
Default here: 0.5.

spec.time.axis a list of tick mark and label specifications for individualized time axis labeling (only effective if label.time.axis = TRUE):

at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical values or as dates. Dates are used only in the case show.date = TRUE, however, and date formats should conform to as.Date or the format given in date.format.
Default: NULL.

labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
Default: TRUE.

las: the style of axis labels, see par.
Default: 1 (always horizontal).

hadj: adjustment of labels horizontal to the reading direction, see axis.
Default: NA (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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

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

periodtck
: length of tick marks on the period axis as a fraction of the smaller of the width or height of the plotting region; see par. If periodtck >= 0.5, periodtck is interpreted as a fraction of the length of the period axis, so if periodtck = 1 (and periodtcl = NULL), horizontal grid lines will be drawn.
Setting periodtck = NA is to use periodtcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

periodtcl
: length of tick marks on the period axis as a fraction of the height of a line of text; see par. With periodtcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward.
Default here: 0.5.

spec.period.axis
: a list of tick mark and label specifications for individualized period axis labeling (only effective if label.period.axis = TRUE):

at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and positive values only.
Default: NULL.

labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
Default: TRUE.

las: the style of axis labels, see par.
Default: 1 (always horizontal).

hadj: adjustment of labels horizontal to the reading direction, see axis.
Default: NA (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

	Mismatches will result in a reset to default plotting.
`main`	an overall title for the plot. Default: `NULL`.
`lwd`	line width of contour lines and ridge. Default: `2`.
`lwd.axis`	line width of axes (image and legend bar). Default: `1`.
`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 https://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(x, type = "l", xlab = "index", ylab = "", xaxs = "i",
     main = "hourly series with periods of 1, 8, 32, 128 days")

## The following dates refer to the local time zone
## (possibly allowing for daylight saving time):
my.date <- seq(as.POSIXct("2014-10-14 00:00:00", format = "%F %T"),
               by = "hour",
               length.out = series.length)
my.data <- data.frame(date = my.date, x = x)

## Computation of wavelet power:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
my.wt <- analyze.wavelet(my.data, "x",
                         loess.span = 0,
                         dt = 1/24, dj = 1/20,
                         lowerPeriod = 1/4,
                         make.pval = TRUE, n.sim = 10)

## Plot of wavelet power spectrum
## with color breakpoints referring to quantiles:
wt.image(my.wt, main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (quantiles)",
                         lab.line = 3.5,
                         label.digits = 2),
   periodlab = "period (days)")
## Note:
## The default time axis shows an index of given points in time,
## which is the count of hours in our example.

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

## Alternative styles of the time axis:

## The plot with time elapsed in days, starting from 0 and proceeding
## in steps of 50 days (50*24 hours),
```

```
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24
## Insert your specification of the time axis:
wt.image(my.wt, color.key = "i", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (equidistant)"),
   periodlab = "period (days)", timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## The plot with (automatically produced) calendar axis:
wt.image(my.wt, color.key = "i", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (equidistant)"),
   periodlab = "period (days)",
   show.date = TRUE, date.format = "%F %T")

## Individualizing your calendar axis (works with 'show.date = TRUE')...
## How to obtain, for example, monthly date ticks and labels:

## The sequence of tick positions:
monthly.ticks <- seq(as.POSIXct("2014-11-01 00:00:00", format = "%F %T"),
                     as.POSIXct("2016-11-01 00:00:00", format = "%F %T"),
                     by = "month")
## Observe that the following specification may produce an error:
## 'seq(as.Date("2014-11-01"), as.Date("2016-11-01"), by = "month")'
## Time of the day is missing here!

## The sequence of labels (e.g. information on month and year only):
monthly.labels <- strftime(monthly.ticks, format = "%b %Y")

## Insert your specification of the time axis:
wt.image(my.wt, color.key = "i", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (equidistant)"),
   periodlab = "period (days)",
   show.date = TRUE, date.format = "%F %T",
   spec.time.axis = list(at = monthly.ticks, labels = monthly.labels,
                         las = 2))
## Note:
## The monthly ticks specify the midpoints of the colored cells and match
## the location of corresponding (default) time index ticks.

## Furthermore, the plot with an individualized period axis:
wt.image(my.wt, color.key = "i", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (equidistant)"),
   periodlab = "period (days)",
   show.date = TRUE, date.format = "%F %T",
   spec.time.axis = list(at = monthly.ticks, labels = monthly.labels,
                         las = 2),
   spec.period.axis = list(at = c(1,8,32,128)))

## Switching the time axis from index to time elapsed in hours
## (starting from 0, and proceeding in steps of 500 hours),
## and the period axis from days to hours:
index.ticks  <- seq(1, series.length, by = 500)
```

```
index.labels <- index.ticks - 1
wt.image(my.wt, color.key = "i", main = "wavelet power spectrum",
   legend.params = list(lab = "wavelet power levels (equidistant)"),
   timelab = "time elapsed (hours)", periodlab = "period (hours)",
   spec.time.axis = list(at = index.ticks, labels = index.labels),
   spec.period.axis = list(at = c(1,8,32,128), labels = c(1,8,32,128)*24))

## A plot with different colors:
wt.image(my.wt, main = "wavelet power spectrum",
   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",
   periodlab = "period (days)")

## In the case of monthly (or quarterly) data, the time axis should be
## labeled at equally spaced time points. An example:

monthyear <- seq(as.Date("2014-01-01"), as.Date("2018-01-01"),
                 by = "month")
monthyear <- strftime(monthyear, format = "%b %Y")

xx <- periodic.series(start.period = 6, length = length(monthyear))
xx <- xx + 0.2*rnorm(length(monthyear))

plot(xx, type = "l", xlab = "index", ylab = "", xaxs = "i",
     main = "monthly series with period of 6 months")

monthly.data <- data.frame(date = monthyear, xx = xx)

my.wt <- analyze.wavelet(monthly.data, "xx", loess.span = 0,
                         dt = 1, dj = 1/250,
                         make.pval = TRUE, n.sim = 250)
## Note:
## The natural choice of 'dt' in this example is 'dt = 1',
## resulting in periods measured in months.
## (Setting 'dt = 1/12' would result in periods measured in years.)

## The default wavelet power plot then shows the monthly:
wt.image(my.wt, main = "wavelet power spectrum",
   periodlab = "period (months)")

## The following plot shows the elapsed time, measured in months:
wt.image(my.wt, main = "wavelet power spectrum",
   periodlab = "period (months)", timelab = "time elapsed (months)",
   spec.time.axis = list(at = 1:length(monthyear),
                         labels = (1:length(monthyear))-1))

## In case you prefer the monthyear labels themselves:
wt.image(my.wt,  main = "wavelet power spectrum",
   periodlab = "period (months)", timelab = "month and year",
   spec.time.axis = list(at = 1:length(monthyear), labels = monthyear))

## You may sometimes wish to enhance your plot with additional information.
```

```
## There is an option to add further objects to the image plot region,
## by setting 'graphics.reset = FALSE'
## (but recall previous par settings after plotting):

op <- par(no.readonly = TRUE)
wt.image(my.wt, main = "wavelet power spectrum",
   periodlab = "period (months)",
   spec.period.axis = list(at = c(2,4,6,8,12)),
   spec.time.axis = list(at = 1:length(monthyear),
                         labels = substr(monthyear,1,3)),
   graphics.reset = FALSE)
abline(h = log2(6), lty = 3)
abline(v = seq(1, length(monthyear), by = 12), lty = 3)
mtext(2014:2018, side = 1,
      at = seq(1, length(monthyear), by = 12), line = 2)
par(op)

## For further axis plotting options:
## Please see the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.


## 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 must 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 row names or as a variable named `"date"` in the data frame at hand. Both axes can be relabeled. In particular, an option is given to individualize the period and/or time axis by specifying tick marks and labels.

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 = TRUE,
       plot.contour = TRUE,
       siglvl = 0.1, col.contour = "white",
       plot.ridge = TRUE, col.ridge = "black",
       n.levels = 100,
       color.palette = "rainbow(n.levels, start = 0, end = .7)",
       useRaster = TRUE, max.contour.segments = 250000,
       plot.legend = TRUE,
       legend.params = list(width = 1.2, shrink = 0.9, mar = 5.1,
                            n.ticks = 6,
                            pi.style = TRUE,
                            label.digits = 1, label.format = "f",
                            lab = NULL, lab.line = 3),
       label.time.axis = TRUE,
       show.date = FALSE, date.format = NULL, date.tz = NULL,
       timelab = NULL, timetck = 0.02, timetcl = 0.5,
       spec.time.axis = list(at = NULL, labels = TRUE,
                             las = 1, hadj = NA, padj = NA),
       label.period.axis = TRUE,
       periodlab = NULL, periodtck = 0.02, periodtcl = 0.5,
       spec.period.axis = list(at = NULL, labels = TRUE,
                               las = 1, hadj = NA, padj = NA),
       main = NULL,
       lwd = 2, lwd.axis = 1,
       graphics.reset = TRUE,
       verbose = FALSE)
```

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? Logical. Default: TRUE.
siglvl	level of wavelet power significance to be 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	ridge color. Default: "black".
n.levels	Number of color levels. Default: 100.

color.palette Definition of color levels. (The color palette 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 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 the 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.

pi.style: style of labels, logical: if TRUE, the symbol "pi" is used to label the legend bar; otherwise, labels are numerals.
Default: TRUE.

label.digits: number of digits of (numerical factors of) labels.
Default: 1 if pi.style is TRUE, else 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: 3.

label.time.axis

Label the time axis? Logical.

Default: TRUE.

show.date Show calendar dates? (Effective only if dates are available as row names or by variable date in the data frame which is analyzed.) Logical.

Default: FALSE.

date.format the format of calendar date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see strptime for a list of implemented date conversion specifications. Explicit information given here will overturn any specification stored in WT. If unspecified, date formatting is attempted according to as.Date.

Default: NULL.

`date.tz` a character string specifying the time zone of calendar date; see `strptime`. Explicit information given here will overturn any specification stored in `WT`. If unspecified, `""` (the local time zone) is used.

Default: `NULL`.

`timelab` Time axis label.

Default: `"index"`; in case of a calendar axis: `"calendar date"`.

`timetck` length of tick marks on the time axis as a fraction of the smaller of the width or height of the plotting region; see `par`. If `timetck >= 0.5`, `timetck` is interpreted as a fraction of the length of the time axis, so if `timetck = 1` (and `timetcl = NULL`), vertical grid lines will be drawn.
Setting `timetck = NA` is to use `timetcl = -0.5` (which is the R default setting of `tck` and `tcl`).

Default here: `0.02`.

`timetcl` length of tick marks on the time axis as a fraction of the height of a line of text; see `par`. With `timetcl = -0.5` (which is the R default setting of `tcl`), ticks will be drawn outward.

Default here: `0.5`.

`spec.time.axis` a list of tick mark and label specifications for individualized time axis labeling (only effective if `label.time.axis = TRUE`):

`at`: locations of tick marks (when `NULL`, default plotting will be applied). Valid tick marks can be provided as numerical values or as dates. Dates are used only in the case `show.date = TRUE`, however, and date formats should conform to `as.Date` or the format given in `date.format`.
Default: `NULL`.

`labels`: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If `labels` is non-logical, `at` should be of same length.
Default: `TRUE`.

`las`: the style of axis labels, see `par`.
Default: `1` (always horizontal).

`hadj`: adjustment of labels horizontal to the reading direction, see `axis`.
Default: `NA` (centering is used).

`padj`: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see `axis`.
Default: `NA` (centering is used).

Mismatches will result in a reset to default plotting.

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

Default: `TRUE`.

`periodlab` (Fourier) period axis label.

Default: `"period"`.

`periodtck` length of tick marks on the period axis as a fraction of the smaller of the width or height of the plotting region; see `par`. If `periodtck >= 0.5`, `periodtck` is interpreted as a fraction of the length of the period axis, so if `periodtck = 1`

(and periodtcl = NULL), horizontal grid lines will be drawn.
Setting periodtck = NA is to use periodtcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

periodtcl — length of tick marks on the period axis as a fraction of the height of a line of text; see par. With periodtcl = -0.5 (which is the R default setting of tcl) ticks will be drawn outward.
Default here: 0.5.

spec.period.axis — a list of tick mark and label specifications for individualized period axis labeling (only effective if label.period.axis = TRUE):

- at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical and positive values only.
  Default: NULL.
- labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
  Default: TRUE.
- las: the style of axis labels, see par.
  Default: 1 (always horizontal).
- hadj: adjustment of labels horizontal to the reading direction, see axis.
  Default: NA (centering is used).
- padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
  Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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

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

lwd.axis — line width of axes (image and legend bar).
Default: 1.

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(x, type = "l", xlab = "index", ylab = "", xaxs = "i",
     main = "hourly series with periods of 1, 8, 32, 128 days")

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

## Computation of wavelet power:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
my.wt <- analyze.wavelet(my.data, "x",
                         loess.span = 0,
                         dt = 1/24, dj = 1/20,
                         lowerPeriod = 1/4,
```

```
                        make.pval = TRUE, n.sim = 10)

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

## Default image of phases:
wt.phase.image(my.wt,
   main = "image of phases",
   periodlab = "period (days)")

## With time elapsed in days
## (starting from 0 and proceeding in steps of 50 days)
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24
wt.phase.image(my.wt,
   main = "image of phases",
   periodlab = "period (days)",
   timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## The same plot, but with (automatically produced) calendar axis:
wt.phase.image(my.wt,
   main = "image of phases", periodlab = "period (days)",
   show.date = TRUE, date.format = "%F %T")

## For further axis plotting options:
## Please see the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.

## Image plot of phases with numerals as labels of the color legend bar:
wt.phase.image(my.wt,
   legend.params=list(pi.style = FALSE, label.digits = 2))


## 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"`, or alternatively of class `"analyze.coherency"`. (In the latter case, the series number or name must be specified.)

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`. In particular, an option is given to individualize the phase and/or time axis by specifying tick marks and labels.)

Usage

```
wt.sel.phases(WT, my.series = 1,
      sel.period = NULL, sel.lower = NULL, sel.upper = NULL,
      only.coi = FALSE,
      only.sig = TRUE, siglvl = 0.05,
      show.avg.phase = FALSE, phase.avg.col = "black",
      label.time.axis = TRUE,
      show.date = FALSE, date.format = NULL, date.tz = NULL,
      timelab = NULL, timetck = 0.02, timetcl = 0.5,
      spec.time.axis = list(at = NULL, labels = TRUE,
                            las = 1, hadj = NA, padj = NA),
      label.phase.axis = TRUE,
      phaselab = NULL, phasetck = 0.02, phasetcl = 0.5,
      spec.phase.axis = list(at = NULL, labels = TRUE,
                             las = 1, hadj = NA, padj = NA),
      main = NULL, sub = NULL,
      lwd = 1, lwd.axis = 1,
      verbose = FALSE)
```

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.
`sel.period`	a single number which determines the (closest available) Fourier period to be selected. Default: `NULL`.
`sel.lower`	a number to define a lower Fourier period (or the closest available) for the selection of a band of periods (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 periods (effective 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 Show average phases over selected periods? (Effective only if a band of periods is selected.) Logical.
Default: FALSE.

phase.avg.col color of line 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 row names or by variable date in the data frame which is analyzed.) Logical.
Default: FALSE.

date.format the format of calendar date given as a character string, e.g. "%Y-%m-%d", or equivalently "%F"; see strptime for a list of implemented date conversion specifications. Explicit information given here will overturn any specification stored in WT. If unspecified, date formatting is attempted according to as.Date.
Default: NULL.

date.tz a character string specifying the time zone of calendar date; see strptime. Explicit information given here will overturn any specification stored in WT. If unspecified, "" (the local time zone) is used.
Default: NULL.

timelab Time axis label.
Default: "index"; in case of a calendar axis: "calendar date".

timetck length of tick marks on the time axis as a fraction of the smaller of the width or height of the plotting region; see par. If timetck >= 0.5, timetck is interpreted as a fraction of the length of the time axis, so if timetck = 1 (and timetcl = NULL), vertical grid lines will be drawn.
Setting timetck = NA is to use timetcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

timetcl length of tick marks on the time axis as a fraction of the height of a line of text; see par. With timetcl = -0.5 (which is the R default setting of tcl), ticks will be drawn outward.
Default here: 0.5.

spec.time.axis a list of tick mark and label specifications for individualized time axis labeling (only effective if label.time.axis = TRUE):

- at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical values or as dates. Dates are used only in the case show.date = TRUE, however, and date formats should conform to as.Date or the format given in date.format.
  Default: NULL.
- labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
  Default: TRUE.

las: the style of axis labels, see par.
Default: 1 (always horizontal).

hadj: adjustment of labels horizontal to the reading direction, see axis.
Default: NA (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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

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

phasetck
: length of tick marks on the phase axis as a fraction of the smaller of the width or height of the plotting region; see par. If phasetck >= 0.5, phasetck is interpreted as a fraction of the length of the phase axis, so if phasetck = 1 (and phasetcl = NULL), horizontal grid lines will be drawn.
Setting phasetck = NA is to use phasetcl = -0.5 (which is the R default setting of tck and tcl).
Default here: 0.02.

phasetcl
: length of tick marks on the phase axis as a fraction of the height of a line of text; see par. With phasetcl = -0.5 (which is the R default setting of tcl), ticks will be drawn outward.
Default here: 0.5.

spec.phase.axis
: a list of tick mark and label specifications for individualized phase axis labeling (only effective if label.phase.axis = TRUE):

at: locations of tick marks (when NULL, default plotting will be applied). Valid tick marks can be provided as numerical values between -pi and pi.
Default: NULL.

labels: either a logical value specifying whether annotations at the tick marks are the tick marks themselves, or any vector of labels. If labels is non-logical, at should be of same length.
Default: TRUE.

las: the style of axis labels, see par.
Default: 1 (always horizontal).

hadj: adjustment of labels horizontal to the reading direction, see axis.
Default: NA (centering is used).

padj: adjustment of labels perpendicular to the reading direction (this can be a vector of adjustments for each label), see axis.
Default: NA (centering is used).

Mismatches will result in a reset to default plotting.

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.
lwd	line width of phases. Default: 1.
lwd.axis	line width of axes. Default: 1.
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 calendar date (if available)
date.format	the format of calendar date (if available)
date.tz	the time zone of calendar date (if available)
axis.1	tick levels corresponding to the time steps used for (cross-)wavelet transformation: 1, 1+dt, 1+2dt, ... The default time axis in plot functions provided by WaveletComp is determined by observation epochs, however; "epoch" meaning point in time.

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(x, type = "l", xlab = "index", ylab = "", xaxs = "i",
     main = "hourly series with periods of 1, 8, 32, 128 days")

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

## Computation of wavelet power:
## a natural choice of 'dt' in the case of hourly data is 'dt = 1/24',
## resulting in one time unit equaling one day.
## This is also the time unit in which periods are measured.
my.wt <- analyze.wavelet(my.data, "x",
                         loess.span = 0,
                         dt = 1/24, dj = 1/20,
                         lowerPeriod = 1/4,
                         make.pval = TRUE, n.sim = 10)

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

## Select period 8 and plot corresponding phases across time:
wt.sel.phases(my.wt, sel.period = 8)

## With time elapsed in days
## (starting from 0 and proceeding in steps of 50 days)
## instead of the (default) time index:
index.ticks  <- seq(1, series.length, by = 50*24)
index.labels <- (index.ticks-1)/24
wt.sel.phases(my.wt, sel.period = 8,
   timelab = "time elapsed (days)",
   spec.time.axis = list(at = index.ticks, labels = index.labels))

## The same plot, but with (automatically produced) calendar axis:
```

```
wt.sel.phases(my.wt, sel.period = 8,
   show.date = TRUE, date.format = "%F %T")

## For further axis plotting options:
## Please see the examples in our guide booklet,
## URL http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf.

## In the following, no period is selected.
## By setting 'show.avg.phase = TRUE', the plot shows average phases
## instead of individual phases:
wt.sel.phases(my.wt, show.avg.phase = TRUE)


## End(Not run)
```

Index