Package ‘CoFESWave’

January 14, 2020

Type Package

Title This is a Test for CoFESWave package

Version 0.1.0

Author Kim C. Raath, Katherine B. Ensor, Yifan Zhang, Michael Jackson

Maintainer Yifan Zhang <yifanzhang@rice.edu>

Description This is a Test for CoFESWave package.
This is a Test for CoFESWave package.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Imports latex2exp, grDevices, fields

NeedsCompilation no

R topics documented:

`CoFEScoherency` *Wavelet Coherency and Cross Wavelet Transform of two series*

Description

Computes the Wavelet Coherency matrix (wco) of two series x and y, using a Gabor Wavelet Transform. Different windows for smoothing can be used. Also computes the cross-Wavelet Transform (smoothed (scross) or non-smoothed (cross)). It also gives the p-values for the wavelet coherency (pvCo) these are computed with n.sur surrogate series. The surrogates are constructed by fitting an ARMA(p,q) model to our series and building new samples by drawing errors from a Gaussian distribution. The major part of the code has been adopted from L. Aguiar-Conraria and M.J. Soares.

Usage

```
CoFEScoherency(
  x,
  y,
  dt = 1,
  dj = 0.25,
  low.period = 2 * dt,
  up.period = length(x) * dt,
  pad = 0,
  sigma = 1,
  wt.type = 0,
  wt.size = 5,
  ws.type = 0,
  ws.size = 5,
  n.sur = 0,
  p = 0,
  q = 0,
  Phase_diff = FALSE,
  low.fp = 32,
  up.fp = 128,
  date = NULL
)
```

Arguments

`x`	one series
`y`	another serise (same length with x)
`dt`	sampling rate Default: `1`.
`dj`	frequency resolution (i.e. 1/dj = number of voices per octave) Default: `0.25`.
`low.period`	lower period of the decomposition Default: `2*dt`.
`up.period`	upper period of the decomposition Default: `length(x)*dt`.
`pad`	an integer (power of 2) defining the total length of the vectors x and x after zero padding; if pad is not a power of 2, pad with zeros to total length: 2^(next power of 2 + 2)
`sigma`	the sigma parameter for the Gabor wavelet Default: `1 - Morlet Wavelet`.
`wt.type`	type of window for smoothing in time direction

`0`	:	Bartlett
`1`	:	Blackman
`2`	:	Rectangular
`3`	:	Hamming
`4`	:	Hanning
`5`	:	Triangular

Default: `Bartlett`.

`wt.size` (half) size of window in time
Default: `5`.

`ws.type` type of window for smoothing in scale direction

`0`	:	Bartlett
`1`	:	Blackman
`2`	:	Rectangular
`3`	:	Hamming
`4`	:	Hanning
`5`	:	Triangular

Default: `Bartlett`.

`ws.size` (half) size of window in scale
Default: `5`.

`n.sur` integer, number of surrogate series, if we want to compute p-values for the Wavelet Coherency
Default: `0 - no computation`.

`p` non-negative integers, orders of the ARMA(P,Q) model
Default: `0`.

`q` non-negative integers, orders of the ARMA(P,Q) model
Default: `0`.

`Phase_diff` Calculate the phase-difference? Logical.
Default: `TRUE`.

`low.fp` lower periods used in the computation of phase-difference
Default: `32`.

`up.fp` upper periods used in the computation of phase-difference
Default: `128`.

`date` a date series

Details

CoFEScoherency

Value

A list of class `"CoFEScoherency"` with elements of different dimensions.

Here is a detailed list of all elements: #'

`wco` complex wavelet coherency of x and y

`periods`	the vector of Fourier periods (in time units) that correspond to the the scales used
`scales`	the vector of scales, given by s0*2^(j*dj);j=0,...,J1, where J1+1 is number of scales and s0 is minimum scale
`coi`	the "cone-of-influence", which is a vector of n_x=length(x) points that contains the limit of the region where the wavelet transforms are influenced by edge effects
`pvCo`	p-values for wavelet coherency
`cross`	cross-wavelet transform of x and y
`scross`	smoothed cross wavelet transform of x and y
`rwco`	real coherency (absolute value of complex coheency)
`phase.dif`	mean phase difference (in selected periods)
`time.lag`	mean time-lag (in selected periods)
`average.coer`	average coherency (average at all times)
`average.cross`	average cross power (average at all times)
`date`	a date series

Author(s)

CoFES. Credits are also due to L. Aguiar-Conraria and M.J. Soares.

References

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

what? "Time-Dependent Spectral Analysis of Epidemiological Time-Series with Wavelets", Journal of the Royal Society Interface, 4, 625.36.

Torrence, C. and Compo, T.C., "A Prectical Guide to Wavelet Analysis" (1998), Bulletin of the American Meteorological Society, 79, 605.618.

Examples

```
## more work
```

`CoFESmpcoherency` *Multiple or Partial Wavelet Coherencies of several series*

Description

Computes multiple(wmco) and/or partial wavelet coherencies (wpco) of columns of matrix X. These are computed using a Gabor Wavelet Transform (different Gabor wavelets can be used). Different windows for smoothing can be used. One can also compute the corresponding p_values (if we take n.sur>0). The surrogates are constructed by fitting an ARMA(p,q) model to our series and building new samples by drawing errors from a Gaussian distribution.

The major part of the code has been adopted from L. Aguiar-Conraria and M.J. Soares.

Usage

```
CoFESmpcoherency(
  X,
  dt = 1,
  dj = 0.25,
  low.period = 2 * dt,
  up.period = nrow(X) * dt,
  pad = 0,
  sigma = 1,
  coher.type = 0,
  index.p = 2,
  wt.type = 0,
  wt.size = 5,
  ws.type = 0,
  ws.size = 5,
  n.sur = 0,
  p = 0,
  q = 0,
  Phase_diff = FALSE,
  low.fp = 32,
  up.fp = 128,
  date = NULL
)
```

Arguments

`X`	matrix with m columns (vectors corresponding to time-series) whose partial or multiple coherencies we want to compute. The first column X[,1] has a special role; e.g. in case of multiple coherency, we want to compte R1.(2...m)
`dt`	sampling rate Default: `1`.
`dj`	frequency resolution (i.e. 1/dj = number of voices per octave) Default: `0.25`.
`low.period`	lower period of the decomposition Default: `2*dt`.
`up.period`	upper period of the decomposition Default: `length(x)*dt`.
`pad`	an integer (power of 2) defining the total length of the vectors x and x after zero padding; if pad is not a power of 2, pad with zeros to total length: 2^(next power of 2 + 2)
`sigma`	the sigma parameter for the Gabor wavelet Default: `1 - Morlet Wavelet`.
`coher.type`	type of coherency we want to compute `0` or "part" : partial (default) 1 or "mult" : multiple
`index.p`	index of series for partial coherency Default: `2`.

wt.type type of window for smoothing in time direction

0	:	Bartlett
1	:	Blackman
2	:	Rectangular
3	:	Hamming
4	:	Hanning
5	:	Triangular

Default: `Bartlett`.

wt.size (half) size of window in time
Default: `5`.

ws.type type of window for smoothing in scale direction

0	:	Bartlett
1	:	Blackman
2	:	Rectangular
3	:	Hamming
4	:	Hanning
5	:	Triangular

Default: `Bartlett`.

ws.size (half) size of window in scale
Default: `5`.

n.sur integer, number of surrogate series, if we want to compute p-values for the Wavelet Coherency
Default: `0 - no computation`.

p non-negative integers, orders of the ARMA(P,Q) model
Default: `0`.

q non-negative integers, orders of the ARMA(P,Q) model
Default: `0`.

Phase_diff Calculate the phase-difference? Logical.
Default: `TRUE`.

low.fp lower periods used in the computation of phase-difference
Default: `32`.

up.fp upper periods used in the computation of phase-difference
Default: `128`.

date a date series

Details

CoFESmpcoherency

Value

A list of class `"CoFESmpcoherency"` with elements of different dimensions.

Here is a detailed list of all elements: #'

wmpco Wavelet Multiple or Partial Coherency Matrix

`periods`	the vector of Fourier periods (in time units) that corresponds to the used scales
`scales`	the vector of scales, given by s0*2^(j*dj);j=0,...,J1, where J1+1 is number of scales and s0 is minimum scale
`coi`	the "cone-of-influence", which is a vector of n_x=length(x) points that contains the limit of the region where the wavelet transforms are influenced by edge effects
`low.fp`	lower periods where phase-diff is computed
`up.fp`	upper periods where phase-diff is computed
`pvMP`	matrix of p-values for the multiple/partial wavelet coherency
`phase.dif`	mean phase difference (in the selected periods)
`time.lag`	time-lag (in the selected periods)
`average.wpco`	mean partial coherency (we take absolute value of partial coherency)
`date`	a date series

Author(s)

CoFES. Credits are also due to L. Aguiar-Conraria and M.J. Soares.

References

Aguiar-Conraria, L. and Soares, M.J. (2010)

Soares, M.J. (2010), "Multiple and Partial Wavelet Coherencies" (private notes) available at http:

Torrence, C. and Compo, T.C., "A Prectical Guide to Wavelet Analysis" (1998), Bulletin of the American Meteorological Society, 79, 605.618.

Examples

```
## more work
```

`CoFESWave` *WaveL2E: A package for ...*

Description

The WaveL2E package provides ? categories of important functions: WaveL2E, plot.CoFESWaveWCO, etc.

CoFESWave functions

WaveL2E

CoFESWave.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

```
CoFESWave.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`.

Details

CoFESWave.image

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)

CoFES. Credits are also due toAngi Roesch, Harald Schmidbauer, 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

`CoFESWave.Transform`, `CoFESWave.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 = "\
               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 = "\

## 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 = "\
                     as.POSIXct("2016-11-01 00:00:00", format = "\
                     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 = "\

## 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 = "\
        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 = "\
         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 = "\

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)
```

`CoFESWave.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

```
CoFESWave.reconstruct(
  WT,
  my.series = 1,
  lvl = 0,
  only.ridge = FALSE,
  only.sig = TRUE,
  siglvl = 0.05,
  only.coi = 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.ridge`	Select only the wavelet power ridge? 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.coi
: Exclude borders influenced by edge effects in reconstruction, i.e. include the cone of influence only? Logical.
Default: FALSE.

sel.period
: a vector of numbers to select Fourier periods (or closest available periods) and corresponding wavelets for the reconstruction.
Default: NULL.

sel.lower
: a number to define a lower Fourier period (or the closest available) for the selection of a band of wavelets for the reconstruction.
(Only effective if sel.period = 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`.

Details

CoFESWave.reconstruct

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)

CoFES. Credits are also due toAngi 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

`CoFESWave.Transform`

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)
```

`CoFESWave.Transform` *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

```
CoFESWave.Transform(
  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 columnnamed `"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

CoFESWave.Transform

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)

CoFES. Credits are also due toAngi Roesch, Harald Schmidbauer, 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

`CoFESWave.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 = "\
               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 = "\
```

```
## 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)
```

plot_CoFESWaveWCO *plot_CoFESWaveWCO*

Description

To plot (absolute value) of wavelet coherency or multiple or partial wavelet coherencies of several series, coi (and levels of sigificance)

Usage

```
plot_CoFESWaveWCO(
  Coherency.Object,
  pE = 5,
  horizons.label = TRUE,
  low.FP = 32,
  up.FP = 128
)
```

Arguments

Coherency.Object	output of function CoFEScoherency or CoFESmpcoherency
pE	a constant to enhance the quality of picture
horizons.label	Label the horizontal axis with labels (instead of numbers)? Logical. Default: TRUE.
low.FP	lower periods used in the frequency band
up.FP	upper periods used in the frequency band

Author(s)

CoFES. Credits are also due to L. Aguiar-Conraria and M.J. Soares.

Examples

```
## more work
```

WaveL2E — *WaveL2E is a function*

Description

WaveL2E is a function

Usage

```
WaveL2E(
  x,
  date = NULL,
  block = 1,
  base_plot = TRUE,
  L2E = TRUE,
  Chi_square = TRUE
)
```

Arguments

`x`	a series
`date`	a date series
`block`	number of observations within a block Default: `1`.
`base_plot`	Plot wavelet power spectrum of x? Logical Default: `TRUE`.
`L2E`	Apply L2E thresholding method? Logical Default: `TRUE`.
`Chi_square`	Apply L2E_Chi_square thresholding method? Logical Default: `TRUE`.

Value

A list of class `"WaveL2E"` with elements of different dimensions.

Here is a detailed list of all elements: #'

`sig`	tmp
`w`	tmp
`dis0`	tmp
`thresh`	tmp
`qthresh`	tmp
`original`	tmp
`Ana_Wave`	tmp
`Emp_WaveL2E`	tmp
`Emp_WaveL2E_MAD`	tmp

`recon_L2E`	tmp
`PTV_L2E`	tmp
`PSL_L2E`	tmp
`recon_Chi_square`	tmp
`PTV_Chi_square`	tmp
`PSL_Chi_square`	tmp
`date`	a date series

Author(s)

CoFES.

Examples

```
## WaveL2E(x)
```

Index