Package ‘biwavelet’

May 20, 2018

Type Package

Title Conduct Univariate and Bivariate Wavelet Analyses

Version 0.20.17

Date 2018-05-19

Author Tarik C. Gouhier, Aslak Grinsted, Viliam Simko

Maintainer Tarik C. Gouhier <tarik.gouhier@gmail.com>

Description This is a port of the WTC MATLAB package written by Aslak Grinsted and the wavelet program written by Christopher Torrence and Gibert P. Compo. This package can be used to perform univariate and bivariate (cross-wavelet, wavelet coherence, wavelet clustering) analyses.

License GPL (>= 2)

URL https://github.com/tgouhier/biwavelet

BugReports https://github.com/tgouhier/biwavelet/issues

LazyData yes

LinkingTo Rcpp

Imports fields, foreach, Rcpp (>= 0.12.2)

Suggests testthat, knitr, rmarkdown, devtools

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-05-19 22:54:19 UTC

R topics documented:

biwavelet-package .................................................. 2
ar1.spectrum .......................................................... 4
ar1_ma0_sim .......................................................... 5
arrow ................................................................. 6
arrow2 ............................................................... 7
biwavelet-package

Conduct Univariate and Bivariate Wavelet Analyses

Description

This is a port of the WTC MATLAB package written by Aslak Grinsted and the wavelet program written by Christopher Torrence and Gibert P. Compo. This package can be used to perform univariate and bivariate (cross-wavelet, wavelet coherence, wavelet clustering) wavelet analyses.

Details

As of biwavelet version 0.14, the bias-corrected wavelet and cross-wavelet spectra are automatically computed and plotted by default using the methods described by Liu et al. (2007) and Veleda et al. (2012). This correction is needed because the traditional approach for computing the power spectrum (e.g., Torrence and Compo 1998) leads to an artificial and systematic reduction in power at lower periods.
Author(s)

Tarik C. Gouhier

Maintainer: Tarik C. Gouhier <tarik.gouhier@gmail.com>

Code based on WTC MATLAB package written by Aslak Grinsted and the wavelet MATLAB program written by Christopher Torrence and Gibert P. Compo.

References


Examples

# As of biwavelet version 0.14, the bias-corrected wavelet and cross-wavelet spectra
# are automatically computed and plotted by default using the methods
# described by Liu et al. (2007) and Veleda et al. (2012). This correction
# is needed because the traditional approach for computing the power spectrum
# (e.g., Torrence and Compo 1998) leads to an artificial and systematic reduction
# in power at low periods.

# EXAMPLE OF BIAS CORRECTION:
require(biwavelet)

# Generate a synthetic time series 's' with the same power at three distinct periods
t1=sin(seq(from=0, to=2*5*pi, length=1000))
t2=sin(seq(from=0, to=2*15*pi, length=1000))
t3=sin(seq(from=0, to=2*40*pi, length=1000))
s=t1+t2+t3

# Compare non-corrected vs. corrected wavelet spectrum
wt1=wt(cbind(1:1000, s))
par(mfrow=c(1,2))
plot(wt1, type="power.corr.norm", main="Bias-corrected")
```r
plot(wt1, type="power.norm", main="Not-corrected")

# ADDITIONAL EXAMPLES
t1 <- cbind(1:100, rnorm(100))
t2 <- cbind(1:100, rnorm(100))

# Continuous wavelet transform
wt.t1 <- wt(t1)

# Plot power
# Make room to the right for the color bar
par(oma = c(0, 0, 0, 1), mar = c(5, 4, 4, 5) + 0.1)
plot(wt.t1, plot.cb=TRUE, plot.phase=FALSE)

# Compute cross-wavelet
xwt.t1t2 <- xwt(t1, t2)

# Plot cross wavelet power and phase difference (arrows)
plot(xwt.t1t2, plot.cb=TRUE)

# Wavelet coherence; nrands should be large (>= 1000)
wtc.t1t2=wtc(t1, t2, nrands=10)
# Plot wavelet coherence and phase difference (arrows)
# Make room to the right for the color bar
par(oma=c(0, 0, 0, 1), mar=c(5, 4, 4, 5) + 0.1)
plot(wtc.t1t2, plot.cb=TRUE)

# Perform wavelet clustering of three time series
t1=cbind(1:100, sin(seq(from=0, to=10*2*pi, length.out=100)))
t2=cbind(1:100, sin(seq(from=0, to=10*2*pi, length.out=100)+0.1*pi))
t3=cbind(1:100, rnorm(100))

# Compute wavelet spectra
wt.t1=wt(t1)
wt.t2=wt(t2)
wt.t3=wt(t3)

# Store all wavelet spectra into array
w.arr=array(NA, dim=c(3, NROW(wt.t1$wave), NCOL(wt.t1$wave)))
w.arr[1, ]=wt.t1$wave
w.arr[2, ]=wt.t2$wave
w.arr[3, ]=wt.t3$wave

# Compute dissimilarity and distance matrices
w.arr.dis <- wclust(w.arr)
plot(hclust(w.arr.dis$dist.mat, method = "ward.D"), sub = "", main = ",
ylab = "Dissimilarity", hang = -1)
```

---

**ar1.spectrum**

Power spectrum of a random red noise process
Description
Generate the power spectrum of a random time series with a specific AR(1) coefficient.

Usage
ar1.spectrum(ar1, periods)

Arguments
- ar1: First order coefficient desired.
- periods: Periods of the time series at which the spectrum should be computed.

Value
Returns the power spectrum as a vector of real numbers.

Author(s)
Tarik C. Gouhier (tarik.gouhier@gmail.com) Code based on WTC MATLAB package written by Aslak Grinsted.

References

Examples
p <- ar1.spectrum(0.5, 1:25)
Arguments

minroots  Output from get_minroots function.
ar        The ‘ar’ part of AR(1)
n        Length of output series, before un-differencing. A strictly positive integer.

See Also

arima.sim

Description

Helper function for phase.plot (not exported)

Usage

arrow(x, y, l = 0.1, w = 0.3 * l, alpha, col = "black")

Arguments

x        X-coordinate of the arrow.
y        Y-coordinate of the arrow.
l        Length of the arrow.
w        Width of the arrow.
alpha    Angle of the arrow in radians (0 .. 2*pi).
col      Color of the arrow.

Examples

plot.new()
arrow(0,0, alpha = 0)
This is an alternative helper function that plots arrows. It uses `text()` to print a character using a default font. This way, it is possible to render different types of arrows.

Usage

```r
arrow2(x, y, angle, size = 0.1, col = "black", chr = intToUtf8(10139))
```

Arguments

- `x`: X-coordinate of the arrow.
- `y`: Y-coordinate of the arrow.
- `angle`: Angle in radians.
- `size`: Similar to `arrow.len` parameter. Notice that we don’t need the `arrow.lwd` anymore.
- `col`: Color of the arrow.
- `chr`: Character representing the arrow. You should provide the character as escaped UTF-8.

Author(s)

Viliam Simko

Examples

```r
# Not run: arrow2(x[i], y[i], angle = phases[i, j],
# Not run: col = arrow.col, size = arrow.len)
```

Check the format of time series

Usage

```r
check.data(y, x1 = NULL, x2 = NULL)
```
check.datum

Arguments

y  Time series y in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.
x1 Time series x1 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.
x2 Time series x2 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.

Value

Returns a named list containing:

t  Time steps
dt  Size of a time step
n.obs  Number of observations

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


Examples

t1 <- cbind(1:100, rnorm(100))
check.data(y = t1)

Description

Helper function

Usage

check.datum(x)

Arguments

x  matrix
convolve2D

Value

list(t, dt, n.obs)

Note

This function is not exported

---

convolve2D  \hspace{2cm} Fast column-wise convolution of a matrix

Description

Use the Fast Fourier Transform to perform convolutions between a sequence and each column of a matrix.

Usage

convolve2D(x, y, conj = TRUE, type = c("circular", "open"))

Arguments

- **x**: M x n matrix.
- **y**: Numeric sequence of length N.
- **conj**: Logical; if TRUE, take the complex conjugate before back-transforming. TRUE is used for usual convolution.
- **type**: Character; one of circular, open (beginning of word is ok). For circular, the two sequences are treated as circular, i.e., periodic. For open and filter, the sequences are padded with zeros (from left and right) first; filter returns the middle sub-vector of open, namely, the result of running a weighted mean of x with weights y.

Details

This is a corrupted version of convolve made by replacing \texttt{fft} with \texttt{mvfft} in a few places. It would be nice to submit this to the R Developers for inclusion.

Value

M x n matrix

Note

This function was copied from \texttt{waveslim} to limit package dependencies.

Author(s)

Brandon Whitcher
convolve2D_typeopen  Speed-optimized version of convolve2D

Description

Equivalent to convolve2D(x, y, type = "open"). The motivation for this function was that convolution is called many times in a loop from smooth.wavelet, always with the type = "open" parameter.

Usage

convolve2D_typeopen(x, y)

Arguments

x  M x n matrix.
y  Numeric sequence of length N.

Author(s)

Viliam Simko

See Also

convolve2D


enviro.data  Multivariate ENSO (MEI), NPGO, and PDO indices

Description

Monthly indices of ENSO, NPGO, and PDO from 1950 to 2009

Usage

data(enviro.data)

Format

A data frame with 720 observations on the following 6 variables.

month  a numeric vector containing the month
year  a numeric vector containing the year
date  a numeric vector containing the date
mei  a numeric vector containing the MEI index
npgo  a numeric vector containing the NPGO index
pdo  a numeric vector containing the PDO index
get_minroots

Source

MEI: http://www.esrl.noaa.gov/psd/enso/mei
NPGO: http://www.o3d.org/npgo
PDO: http://jisao.washington.edu/pdo

References


Examples

data(enviro.data)
head(enviro.data)

get_minroots  
Helper function (not exported)

Description

Helper function (not exported)

Usage

get_minroots(ar)

Arguments

ar  
The ’ar’ part of AR(1)

Value

double
**phase.plot**

### Supported mother wavelets

**Description**

The list of supported mother wavelets is used in multiple places therefore, we provide it as a lazily evaluated promise.

**Usage**

```
mothers
```

**Format**

An object of class `character` of length 3.

---

**phase.plot**

**Plot phases with arrows**

**Description**

Plot phases with arrows

**Usage**

```
phase.plot(x, y, phases, arrow.len = min(par($)pin[2]/30, par($)pin[1]/40),
           arrow.col = "black", arrow.lwd = arrow.len * 0.3)
```

**Arguments**

- `x` X-coordinates
- `y` Y-coordinates
- `phases` Phases
- `arrow.len` Size of the arrows. Default is based on plotting region.
- `arrow.col` Arrow line color.
- `arrow.lwd` Width/thickness of arrows.

**Note**

Arrows pointing to the right mean that x and y are in phase.
Arrows pointing to the left mean that x and y are in anti-phase.
Arrows pointing up mean that y leads x by $\pi/2$.
Arrows pointing down mean that x leads y by $\pi/2$. 
Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Huidong Tian provided a much better implementation of the phase.plot function that allows for more accurate phase arrows.

Original code based on WTC MATLAB package written by Aslak Grinsted.

Examples

# Not run: phase.plot(x, y, phases)

plot.biwavelet  

Plot biwavelet objects

Description

Plot biwavelet objects such as the cwt, cross-wavelet and wavelet coherence.

Usage

## S3 method for class 'biwavelet'
plot(x, ncol = 64, fill.cols = NULL, xlab = "Time", ylab = "Period", tol = 1, plot.cb = FALSE, plot.phase = FALSE, type = "power.corr.norm", plot.coi = TRUE, lwd.coi = 1, col.coi = "white", lty.coi = 1, alpha.coi = 0.5, plot.sig = TRUE, lwd.sig = 4, col.sig = "black", lty.sig = 1, bw = FALSE, legend.loc = NULL, legend.horiz = FALSE, arrow.len = min(par()$pin[2]/30, par()$pin[1]/40), arrow.lwd = arrow.len * 0.3, arrow.cutoff = 0.8, arrow.col = "black", xlim = NULL, ylim = NULL, zlim = NULL, xaxt = "s", yaxt = "s", form = "%y", ...)

Arguments

x  
biwavelet object generated by wt, xwt, or wtc.

ncol  
Number of colors to use.

fill.cols  
Vector of fill colors to be used. Users can specify color vectors using colorRampPalette or brewer.pal from package RColorBrewer. Value NULL generates MATLAB’s jet color palette.

xlab  
X-label of the figure.

ylab  
Y-label of the figure.

tol  
Tolerance level for significance contours. Significance contours will be drawn around all regions of the spectrum where spectrum/percentile >= tol. If strict ith percentile regions are desired, then tol must be set to 1.

plot.cb  
Plot color bar if TRUE.
plot.biwavelet

plot.phase: Plot phases with black arrows.

type: Type of plot to create. Can be power to plot the power, power.corr to plot the bias-corrected power, power.norm to plot the power normalized by the variance, power.corr.norm to plot the bias-corrected power normalized by the variance, wavelet to plot the wavelet coefficients, or phase to plot the phase.

plot.coi: Plot cone of influence (COI) as a semi-transparent polygon if TRUE. Areas that fall within the polygon can be affected by edge effects.

lwd.coi: Line width of COI.

col.coi: Color of COI.

lty.coi: Line type of COI. Value 1 is for solide lines.

alpha.coi: Transparency of COI. Range is 0 (full transparency) to 1 (no transparency).

plot.sig: Plot contours for significance if TRUE.

lwd.sig: Line width of significance contours.

col.sig: Color of significance contours.

lty.sig: Line type of significance contours.

bw: Plot in black and white if TRUE.

legend.loc: Legend location coordinates as defined by image.plot.

legend.horiz: Plot a horizontal legend if TRUE.

arrow.len: Size of the arrows. Default is based on plotting region.

arrow.lwd: Width/thickness of arrows.

arrow.cutoff: Cutoff value for plotting phase arrows. Phase arrows will be be plotted in regions where the significance of the z-values exceeds arrow.cutoff for wt and xwt objects. For pwtc and wtc objects, phase arrows will be plotted in regions where the rsq value exceeds arrow.cutoff. If the object being plotted does not have a significance field, regions whose z-values exceed the arrow.cutoff quantile will be plotted.

arrow.col: Color of arrows.

xlim: The x limits.

ylim: The y limits.

zlim: The z limits.

xaxt: Add x-axis? Use n for none.

yaxt: Add y-axis? Use n for none.

form: Format to use to display dates on the x-axis. See Date for other valid formats.

Details

Arrows pointing to the right mean that x and y are in phase.

Arrows pointing to the left mean that x and y are in anti-phase.

Arrows pointing up mean that x leads y by \(\pi/2\).

Arrows pointing down mean that y leads x by \(\pi/2\).
Author(s)
Tarik C. Gouhier (tarik.gouhier@gmail.com) Code based on WTC MATLAB package written by Aslak Grinsted.

References

See Also
*image.plot*

Examples

```r
library(biwavelet)

t1 <- cbind(1:100, rnorm(100))
t2 <- cbind(1:100, rnorm(100))

# Continuous wavelet transform
wt.t1 <- wt(t1)

# Plot power
# Make room to the right for the color bar
par oma = c(0, 0, 0, 1), mar = c(5, 4, 4, 5) + 0.1
plot(wt.t1, plot.cb = TRUE, plot.phase = FALSE)

# Cross-wavelet transform
xwt.t1t2 <- xwt(t1, t2)

# Plot cross-wavelet
par oma = c(0, 0, 0, 1), mar = c(5, 4, 4, 5) + 0.1
plot(xwt.t1t2, plot.cb = TRUE)

# Example of bias-correction
t1 <- sin(seq(0, 2 * 5 * pi, length.out = 1000))
t2 <- sin(seq(0, 2 * 15 * pi, length.out = 1000))
t3 <- sin(seq(0, 2 * 40 * pi, length.out = 1000))

# This aggregate time series should have the same power
# at three distinct periods
s <- t1 + t2 + t3
```
# Compare plots to see bias-effect on CWT:
# biased power spectrum artificially
# reduces the power of higher-frequency fluctuations.
wt1 <- wt(cbind(1:1000, s))
par(mfrow = c(1,2))
plot(wt1, type = "power.corr.norm", main = "Bias-corrected")
plot(wt1, type = "power.norm", main = "Biased")

# Compare plots to see bias-effect on XWT:
# biased power spectrum artificially
# reduces the power of higher-frequency fluctuations.
x1 <- xwt(cbind(1:1000, s), cbind(1:1000, s))
par(mfrow = c(1,2))
plot(x1, type = "power.corr.norm", main = "Bias-corrected")
plot(x1, type = "power.norm", main = "Biased")

---

**pwtc**

*Compute partial wavelet coherence*

## Description

Compute partial wavelet coherence between y and x1 by partialling out the effect of x2

## Usage

```r
pwtc(y, x1, x2, pad = TRUE, dj = 1/12, s0 = 2 * dt, J1 = NULL,
     max.scale = NULL, mother = "morlet", param = -1, lag1 = NULL,
     sig.level = 0.95, sig.test = 0, nrands = 300, quiet = FALSE)
```

## Arguments

- **y**  
  Time series y in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.

- **x1**  
  Time series x1 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.

- **x2**  
  Time series x2 whose effects should be partialled out in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.

- **pad**  
  Pad the values will with zeros to increase the speed of the transform.

- **dj**  
  Spacing between successive scales.

- **s0**  
  Smallest scale of the wavelet.

- **J1**  
  Number of scales - 1.

- **max.scale**  
  Maximum scale. Computed automatically if left unspecified.

- **mother**  
  Type of mother wavelet function to use. Can be set to *morlet*, *dog*, or *paul*. Significance testing is only available for *morlet* wavelet.
param  Nondimensional parameter specific to the wavelet function.
lag1   Vector containing the AR(1) coefficient of each time series.
sig.level  Significance level.
sig.test  Type of significance test. If set to 0, use a regular $\chi^2$ test. If set to 1, then perform a time-average test. If set to 2, then do a scale-average test.

nrands  Number of Monte Carlo randomizations.
quiet  Do not display progress bar.

Value

Return a biwavelet object containing:

- coi  matrix containing cone of influence
- wave  matrix containing the cross-wavelet transform of y and x1
- rsq  matrix of partial wavelet coherence between y and x1 (with x2 partialled out)
- phase  matrix of phases between y and x1
- period  vector of periods
- scale  vector of scales
- dt  length of a time step
- t  vector of times
- xaxis  vector of values used to plot xaxis
- s0  smallest scale of the wavelet
- dj  spacing between successive scales
- y.sigma  standard deviation of y
- x1.sigma  standard deviation of x1
- mother  mother wavelet used
- type  type of biwavelet object created (pwtc)
- signif  matrix containing sig.level percentiles of wavelet coherence based on the Monte Carlo AR(1) time series

Note

The Monte Carlo randomizations can be extremely slow for large datasets. For instance, 1000 randomizations of a dataset consisting of 1000 samples will take \textasciitilde30 minutes on a 2.66 GHz dual-core Xeon processor.

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com) Code based on WTC MATLAB package written by Aslak Grinsted.
rcpp_row_quantile

References


Examples

library(biwavelet)

y <- cbind(1:100, rnorm(100))
x1 <- cbind(1:100, rnorm(100))
x2 <- cbind(1:100, rnorm(100))

# Partial wavelet coherence of y and x1
pwtc.yx1 <- pwtc(y, x1, x2, nrands = 0)

# Partial wavelet coherence of y and x2
pwtc.yx2 <- pwtc(y, x2, x1, nrands = 0)

# Plot partial wavelet coherence and phase difference (arrows)
# Make room to the right for the color bar
par(mfrow = c(2,1), oma = c(4, 0, 0, 1),
     mar = c(1, 4, 4, 5), mgp = c(1.5, 0.5, 0))

plot(pwtc.yx1, xlab = "", plot.cb = TRUE,
     main = "Partial wavelet coherence of y and x1 | x2")

plot(pwtc.yx2, plot.cb = TRUE,
     main = "Partial wavelet coherence of y and x2 | x1")

rcpp_row_quantile  Row-wise quantile of a matrix

Description

This is a C++ speed-optimized version. It is equivalent to R version quantile(data, q, na.rm = TRUE)
**Usage**

```r
callback_row_quantile(data, q)
```

**Arguments**

- `data`: Numeric matrix whose row quantiles are wanted.
- `q`: Probability with value in [0, 1]

**Value**

A vector of length `nrows(data)`, where each element represents row quantile.

**Author(s)**

Viliam Simko

---

**rcpp_wt_bases_dog**

*Optimized "wt.bases.dog" function.*

**Description**

This is a C++ version optimized for speed. Computes the wavelet as a function of Fourier frequency for "dog" mother wavelet.

**Usage**

```c
rcpp_wt_bases_dog(k, scale, param = 1L)
```

**Arguments**

- `k`: vector of frequencies at which to calculate the wavelet.
- `scale`: the wavelet scale.
- `param`: nondimensional parameter specific to the wavelet function.

**Value**

Returns a list containing:

- `daughter`: wavelet function
- `fourier.factor`: ratio of fourier period to scale
- `coi`: cone of influence
- `dof`: degrees of freedom for each point in wavelet power

**Note**

This C++ implementation is approx. 50
rcpp_wt_bases_morlet

Optimized "wt.bases.morlet" function.

Description

This is a C++ version optimized for speed. Computes the wavelet as a function of Fourier frequency for "morlet" mother wavelet.

Usage

rcpp_wt_bases_morlet(k, scale, param = -1L)

Arguments

- k: vector of frequencies at which to calculate the wavelet.
- scale: the wavelet scale.
- param: nondimensional parameter specific to the wavelet function.

Value

Returns a list containing:

- daughter: wavelet function
- fourier.factor: ratio of fourier period to scale
- coi: cone of influence
- dof: degrees of freedom for each point in wavelet power

Note

This C++ implementation is approx. 60

Author(s)

Viliam Simko
rcpp_wt_bases_paul

Optimized "wt.bases.paul" function.

Description

This is a C++ version optimized for speed. Computes the wavelet as a function of Fourier frequency for "paul" mother wavelet.

Usage

rcpp_wt_bases_paul(k, scale, param = -1L)

Arguments

k vector of frequencies at which to calculate the wavelet.
scale the wavelet scale.
param nondimensional parameter specific to the wavelet function.

Value

Returns a list containing:

daughter wavelet function
fourier.factor ratio of fourier period to scale
coi cone of influence
dof degrees of freedom for each point in wavelet power

Note

This c++ implementation is approx. 59

Author(s)

Viliam Simko
smooth.wavelet  

Smooth wavelet in both the time and scale domains

Description

The time smoothing uses a filter given by the absolute value of the wavelet function at each scale, normalized to have a total weight of unity, which is a Gaussian function for the Morlet wavelet. The scale smoothing is done with a boxcar function of width 0.6, which corresponds to the decorrelation scale of the Morlet wavelet.

Usage

smooth.wavelet(wave, dt, dj, scale)

Arguments

- wave: wavelet coefficients
- dt: size of time steps
- dj: number of octaves per scale
- scale: wavelet scales

Value

Returns the smoothed wavelet.

Note

This function is used internally for computing wavelet coherence. It is only appropriate for the morlet wavelet.

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Code based on WTC MATLAB package written by Aslak Grinsted.

References


Examples

# Not run: smooth.wtl <- smooth.wavelet(wave, dt, dj, scale)
wclust

Compute dissimilarity between multiple wavelet spectra

Description

Compute dissimilarity between multiple wavelet spectra

Usage

wclust(w.arr, quiet = FALSE)

Arguments

w.arr N x p x t array of wavelet spectra where N is the number of wavelet spectra to be compared, p is the number of periods in each wavelet spectrum and t is the number of time steps in each wavelet spectrum.

quiet Do not display progress bar.

Value

Returns a list containing:

diss.mat square dissimilarity matrix
dist.mat (lower triangular) distance matrix

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


Examples

library(biwavelet)

t1 <- cbind(1:100, sin(seq(0, 10 * 2 * pi, length.out = 100)))
t2 <- cbind(1:100, sin(seq(0, 10 * 2 * pi, length.out = 100) + 0.1 * pi))
t3 <- cbind(1:100, rnorm(100)) # white noise

## Compute wavelet spectra
wt.t1 <- wt(t1)
wt.t2 <- wt(t2)
```r
wt.t3 <- wt(t3)

## Store all wavelet spectra into array
w.arr <- array(dim = c(3, NROW(wt.t1$wave), NCOL(wt.t1$wave)))
w.arr[1, , ] <- wt.t1$wave
w.arr[2, , ] <- wt.t2$wave
w.arr[3, , ] <- wt.t3$wave

## Compute dissimilarity and distance matrices
w.arr.dis <- wclust(w.arr)
plot(hclust(w.arr.dis$dist.mat, method = "ward.D"),
     sub = "", main = ",", ylab = "Dissimilarity", hang = -1)
```

---

**Function: wdist**

*Compute dissimilarity between two wavelet spectra*

**Description**

Compute dissimilarity between two wavelet spectra

**Usage**

```
wdist(wt1, wt2, cutoff = 0.99)
```

**Arguments**

- **wt1**: power, wave or rsq matrix from biwavelet object generated by `wt`, `xwt`, or `wtc`.
- **wt2**: power, wave or rsq matrix from biwavelet object generated by `wt`, `xwt`, or `wtc`.
- **cutoff**: Cutoff value used to compute dissimilarity. Only orthogonal axes that contribute more than `1 - cutoff` to the total covariance between the two wavelet spectra will be used to compute their dissimilarity.

**Value**

Returns wavelet dissimilarity.

**Author(s)**

Tarik C. Gouhier (tarik.gouhier@gmail.com)

**References**


Examples

library(biwavelet)

t1 <- cbind(1:100, sin(seq(0, 10 * 2 * pi, length.out = 100)))
t2 <- cbind(1:100, sin(seq(0, 10 * 2 * pi, length.out = 100) + 0.1 * pi))

# Compute wavelet spectra
wt.t1 <- wt(t1)
wt.t2 <- wt(t2)

# Compute dissimilarity
wdist(wt.t1$wave, wt.t2$wave)

Compute wavelet transform

Description

Compute wavelet transform

Usage

wt(d, pad = TRUE, dt = NULL, dj = 1/12, s0 = 2 * dt, J1 = NULL,
  max.scale = NULL, mother = "morlet", param = -1, lag1 = NULL,
  sig.level = 0.95, sig.test = 0, do.sig = TRUE, arima.method = "CSS")

Arguments

d Time series in matrix format (n rows x 2 columns). The first column should
  contain the time steps and the second column should contain the values.
pad Pad the values will with zeros to increase the speed of the transform.
dt Length of a time step.
dj Spacing between successive scales.
s0 Smallest scale of the wavelet.
J1 Number of scales - 1.
max.scale Maximum scale. Computed automatically if left unspecified.
mother Type of mother wavelet function to use. Can be set to morlet, dog, or paul.
param Nondimensional parameter specific to the wavelet function.
lag1 AR(1) coefficient of time series used to test for significant patterns.
sig.level Significance level.
sig.test Type of significance test. If set to 0, use a regular \(\chi^2\) test. If set to 1, then
  perform a time-average test. If set to 2, then do a scale-average test.
do.sig Perform significance testing if TRUE.
arima.method Fitting method. This parameter is passed as the method Parameter to the arima
  function.
Value

Returns a biwavelet object containing:

- coi: matrix containing cone of influence
- wave: matrix containing the wavelet transform
- power: matrix of power
- power.corr: matrix of bias-corrected power using the method described by Liu et al. (2007)
- phase: matrix of phases
- period: vector of periods
- scale: vector of scales
- dt: length of a time step
- t: vector of times
- xaxis: vector of values used to plot xaxis
- s0: smallest scale of the wavelet
- dj: spacing between successive scales
- sigma2: variance of time series
- mother: mother wavelet used
- type: type of biwavelet object created (wt)
- signif: matrix containing significance levels

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Code based on wavelet MATLAB program written by Christopher Torrence and Gibert P. Compo.

References


Examples

t1 <- cbind(1:100, rnorm(100))

## Continuous wavelet transform
wt.t1 <- wt(t1)

## Plot power
## Make room to the right for the color bar
par oma = c(0, 0, 0, 1), mar = c(5, 4, 4, 5) + 0.1)
plot(wt.t1, plot.cb = TRUE, plot.phase = FALSE)
wt.bases

Compute wavelet

Description
Computes the wavelet as a function of Fourier frequency.

Usage
wt.bases(mother = "morlet", ...)

Arguments
- **mother**: Type of mother wavelet function to use. Can be set to morlet, dog, or paul.
- **...**: See parameters k, scale and param in functions: `wt.bases.morlet`, `wt.bases.paul` and `wt.bases.dog`

Value
Returns a list containing:
- **daughter**: wavelet function
- **fourier.factor**: ratio of fourier period to scale
- **coi**: cone of influence
- **dof**: degrees of freedom for each point in wavelet power

Author(s)
Tarik C. Gouhier (tarik.gouhier@gmail.com)

Code based on wavelet MATLAB program written by Christopher Torrence and Gibert P. Compo.

References

Examples
```r
# Not run: wb <- wt.bases(mother, k, scale[a1], param)
```
wt.bases.dog

Helper method (not exported)

Description

Helper method (not exported)

Usage

wt.bases.dog(k, scale, param = -1)

Arguments

- **k**: Vector of frequencies at which to calculate the wavelet.
- **scale**: The wavelet scale.
- **param**: Nondimensional parameter specific to the wavelet function.

Value

Returns a list containing:

- **daughter**: wavelet function
- **fourier.factor**: ratio of fourier period to scale
- **coi**: cone of influence
- **dof**: degrees of freedom for each point in wavelet power

wt.bases.morlet

Helper method (not exported)

Description

Helper method (not exported)

Usage

wt.bases.morlet(k, scale, param = -1)

Arguments

- **k**: Vector of frequencies at which to calculate the wavelet.
- **scale**: The wavelet scale.
- **param**: Nondimensional parameter specific to the wavelet function.
Value

Returns a list containing:

- `daughter`: wavelet function
- `fourier.factor`: ratio of Fourier period to scale
- `coi`: cone of influence
- `dof`: degrees of freedom for each point in wavelet power

Description

Helper method (not exported)

Usage

```r
wt.bases.paul(k, scale, param = -1)
```

Arguments

- `k`: Vector of frequencies at which to calculate the wavelet.
- `scale`: The wavelet scale.
- `param`: Nondimensional parameter specific to the wavelet function.

Value

Returns a list containing:

- `daughter`: wavelet function
- `fourier.factor`: ratio of Fourier period to scale
- `coi`: cone of influence
- `dof`: degrees of freedom for each point in wavelet power
**wt.sig**  
*Determine significance of wavelet transform*

**Description**
Determine significance of wavelet transform

**Usage**
```r
wt.sig(d, dt, scale, sig.test = 0, sig.level = 0.95, dof = 2,
   lag1 = NULL, mother = "morlet", param = -1, sigma2 = NULL,
   arima.method = "CSS")
```

**Arguments**
- `d`: Time series in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.
- `dt`: Length of a time step.
- `scale`: The wavelet scale.
- `sig.test`: Type of significance test. If set to 0, use a regular $\chi^2$ test. If set to 1, then perform a time-average test. If set to 2, then do a scale-average test.
- `sig.level`: Significance level.
- `dof`: Degrees of freedom for each point in wavelet power.
- `lag1`: AR(1) coefficient of time series used to test for significant patterns.
- `mother`: Type of mother wavelet function to use. Can be set to `morlet`, `dog`, or `paul`.
- `param`: Nondimensional parameter specific to the wavelet function.
- `sigma2`: Variance of time series
- `arima.method`: Fitting method. This parameter is passed as the method Parameter to the `arima` function.

**Value**
Returns a list containing:
- `signif`: vector containing significance level for each scale
- `signif`: vector of red-noise spectrum for each period

**Author(s)**
Tarik C. Gouhier (tarik.gouhier@gmail.com)
Code based on wavelet MATLAB program written by Christopher Torrence and Gibert P. Compo.
References


Examples

```r
# Not run: wt.sig(d, dt, scale, sig.test, sig.level, lag1,
#    dof = -1, mother = "morlet", sigma2 = 1)
```

### Description

Compute wavelet coherence

### Usage

```r
wtc(d1, d2, pad = TRUE, dj = 1/12, s0 = 2 * dt, J1 = NULL,
    max.scale = NULL, mother = "morlet", param = -1, lag1 = NULL,
    sig.level = 0.95, sig.test = 0, nrands = 300, quiet = FALSE)
```

### Arguments

- **d1**: Time series 1 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.
- **d2**: Time series 2 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.
- **pad**: Pad the values will with zeros to increase the speed of the transform.
- **dj**: Spacing between successive scales.
- **s0**: Smallest scale of the wavelet.
- **J1**: Number of scales - 1.
- **max.scale**: Maximum scale. Computed automatically if left unspecified.
- **mother**: Type of mother wavelet function to use. Can be set to *morlet*, *dog*, or *paul*.
- **param**: Nondimensional parameter specific to the wavelet function.
- **lag1**: Vector containing the AR(1) coefficient of each time series.
- **sig.level**: Significance level.
- **sig.test**: Type of significance test. If set to 0, use a regular $\chi^2$ test. If set to 1, then perform a time-average test. If set to 2, then do a scale-average test.
- **nrands**: Number of Monte Carlo randomizations.
- **quiet**: Do not display progress bar.
Value

Return a biwavelet object containing:

- `coi` matrix containing cone of influence
- `wave` matrix containing the cross-wavelet transform
- `wave.corr` matrix containing the bias-corrected cross-wavelet transform using the method described by Veleda et al. (2012)
- `power` matrix of power
- `power.corr` matrix of bias-corrected cross-wavelet power using the method described by Veleda et al. (2012)
- `rsq` matrix of wavelet coherence
- `phase` matrix of phases
- `period` vector of periods
- `scale` vector of scales
- `dt` length of a time step
- `t` vector of times
- `xaxis` vector of values used to plot xaxis
- `s0` smallest scale of the wavelet
- `dj` spacing between successive scales
- `d1.sigma` standard deviation of time series 1
- `d2.sigma` standard deviation of time series 2
- `mother` mother wavelet used
- `type` type of biwavelet object created (wtc)
- `signif` matrix containing sig.level percentiles of wavelet coherence based on the Monte Carlo AR(1) time series

Note

The Monte Carlo randomizations can be extremely slow for large datasets. For instance, 1000 randomizations of a dataset consisting of 1000 samples will take ~30 minutes on a 2.66 GHz dual-core Xeon processor.

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Code based on WTC MATLAB package written by Aslak Grinsted.
References


Examples

t1 <- cbind(1:100, rnorm(100))
t2 <- cbind(1:100, rnorm(100))

## Wavelet coherence
wtc.tlt2 <- wtc(t1, t2, nrands = 10)

## Plot wavelet coherence and phase difference (arrows)
## Make room to the right for the color bar
par(ma = c(0, 0, 0, 1), mar = c(5, 4, 4, 5) + 0.1)
plot(wtc.tlt2, plot.cb = TRUE, plot.phase = TRUE)

wtc.sig

Determine significance of wavelet coherence

Description

Determine significance of wavelet coherence

Usage

wtc.sig(nrands = 300, lag1, dt, ntimesteps, pad = TRUE, dj = 1/12, s0, J1, max.scale = NULL, mother = "morlet", sig.level = 0.95, quiet = FALSE)

Arguments

nrands Number of Monte Carlo randomizations.
lag1 Vector containing the AR(1) coefficient of each time series.
dt Length of a time step.
ntimesteps Number of time steps in time series.
pad Pad the values will with zeros to increase the speed of the transform.
Spacing between successive scales.

Smallest scale of the wavelet.

Number of scales - 1.

Maximum scale.

Type of mother wavelet function to use. Can be set to morlet, dog, or paul. Significance testing is only available for morlet wavelet.

Significance level to compute.

Do not display progress bar.

Returns significance matrix containing the sig.level percentile of wavelet coherence at each time step and scale.

The Monte Carlo randomizations can be extremely slow for large datasets. For instance, 1000 randomizations of a dataset consisting of 1000 samples will take ~30 minutes on a 2.66 GHz dual-core Xeon processor.

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Code based on WTC MATLAB package written by Aslak Grinsted.


# Not run: wtc.sig <- wtc.sig(nrands, lag1 = c(d1.ar1, d2.ar1), dt, pad, dj, j1, s0, mother = "morlet")
Description

Parallelized Monte Carlo simulation equivalent to `wtc.sig`.

Usage

```r
wtc.sig_parallel(nrands = 300, lag1, dt, ntimesteps, pad = TRUE,
                  dj = 1/12, s0, J1, max.scale = NULL, mother = "morlet",
                  sig.level = 0.95, quiet = TRUE)
```

Arguments

- `nrands` Number of Monte Carlo randomizations.
- `lag1` Vector containing the AR(1) coefficient of each time series.
- `dt` Length of a time step.
- `ntimesteps` Number of time steps in time series.
- `pad` Pad the values will with zeros to increase the speed of the transform.
- `dj` Spacing between successive scales.
- `s0` Smallest scale of the wavelet.
- `J1` Number of scales - 1.
- `max.scale` Maximum scale.
- `mother` Type of mother wavelet function to use. Can be set to `morlet`, `dog`, or `paul`. Significance testing is only available for `morlet` wavelet.
- `sig.level` Significance level to compute.
- `quiet` Do not display progress bar.

See Also

`foreach`

`wtc.sig`

Examples

```r
# Not run: library(foreach)
# library(doParallel)
# cl <- makeCluster(4, outfile="") # number of cores. Notice 'outfile'
# registerDoParallel(cl)
# wtc.sig_parallel( your parameters go here )
# stopCluster(cl)
```
xwt

Compute cross-wavelet

Description

Compute cross-wavelet

Usage

```r
xwt(d1, d2, pad = TRUE, dj = 1/12, s0 = 2 * dt, J1 = NULL,
    max.scale = NULL, mother = "morlet", param = -1, lag1 = NULL,
    sig.level = 0.95, sig.test = 0, arima.method = "CSS")
```

Arguments

- **d1**: Time series 1 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.
- **d2**: Time series 2 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values.
- **pad**: Pad the values will with zeros to increase the speed of the transform.
- **dj**: Spacing between successive scales.
- **s0**: Smallest scale of the wavelet.
- **J1**: Number of scales - 1.
- **max.scale**: Maximum scale. Computed automatically if left unspecified.
- **mother**: Type of mother wavelet function to use. Can be set to morlet, dog, or paul. Significance testing is only available for morlet wavelet.
- **param**: Nondimensional parameter specific to the wavelet function.
- **lag1**: Vector containing the AR(1) coefficient of each time series.
- **sig.level**: Significance level.
- **sig.test**: Type of significance test. If set to 0, use a regular $\chi^2$ test. If set to 1, then perform a time-average test. If set to 2, then do a scale-average test.
- **arima.method**: Fitting method. This parameter is passed as the method parameter to the `arima` function.

Value

Returns a biwavelet object containing:

- **coi**: matrix containing cone of influence
- **wave**: matrix containing the cross-wavelet transform
- **wave.corr**: matrix containing the bias-corrected cross-wavelet transform using the method described by Veleda et al. (2012)
- **power**: matrix of power
power.corr matrix of bias-corrected cross-wavelet power using the method described by Veleda et al. (2012)

phase matrix of phases
period vector of periods
scale vector of scales
dt length of a time step
t vector of times
xaxis vector of values used to plot xaxis
s0 smallest scale of the wavelet
dj spacing between successive scales
d1.sigma standard deviation of time series 1
d2.sigma standard deviation of time series 2
mother mother wavelet used
type type of biwavelet object created (xwt)
signif matrix containing significance levels

Author(s)
Tarik C. Gouhier (tarik.gouhier@gmail.com) Code based on WTC MATLAB package written by Aslak Grinsted.

References


Examples

```r
library(biwavelet)

t1 <- cbind(1:100, rnorm(100))
t2 <- cbind(1:100, rnorm(100))

# Compute Cross-wavelet
xwt.tltz <- xwt(t1, t2)
plot(xwt.tltz, plot.cb = TRUE, plot.phase = TRUE, 
    main = "Plot cross-wavelet and phase difference (arrows)"
)
```
# Real data
data(enviro.data)

# Cross-wavelet of MEI and NPGO
xwt.mei.npgo <- xwt(subset(enviro.data, select = c("date", "mei")),
                    subset(enviro.data, select = c("date", "npgo")))

# Make room to the right for the color bar
par(oma = c(0, 0, 0, 1), mar = c(5, 4, 4, 5) + 0.1)
plot(xwt.mei.npgo, plot.cb = TRUE, plot.phase = TRUE)
Index

*Topic **coherence**
  biwavelet-package, 2
*Topic **cross-wavelet**
  biwavelet-package, 2
*Topic **datasets**
  MOTHERS, 12
*Topic **dataset**
  enviro.data, 10
*Topic **wavelet**
  biwavelet-package, 2

ar1.spectrum, 4
ar1.ma0_sim, 5
arima, 25, 30, 36
arima_sim, 5, 6
arrow, 6
arrow2, 7

biwavelet (biwavelet-package), 2
biwavelet-package, 2
brewer_pal, 13

check.data, 7
check.datum, 8
colorRampPalette, 13
convolve2D, 9, 10
convolve2D_typeopen, 10

Date, 14

enviro.data, 10

fft, 9
foreach, 35

get_minroots, 6, 11

image.plot, 14, 15

MOTHERS, 12
mvfft, 9

phase.plot, 6, 12
plot.biwavelet, 13
pwtc, 14, 16, 17

RColorBrewer, 13
rcpp_row_quantile, 18
rcpp_wt_bases_dog, 19
rcpp_wt_bases_morlet, 20
rcpp_wt_bases_paul, 21
smooth.wavelet, 10, 22
text, 7

wclust, 23
wdist, 24
wt, 13, 14, 24, 25, 26
wt.bases, 27
wt.bases.dog, 27, 28
wt.bases.morlet, 27, 28
wt.bases.paul, 27, 29
wt.sig, 30
wtc, 13, 14, 24, 31, 32
wtc.sig, 33, 35
wtc_sig_parallel, 35
xwt, 13, 14, 24, 36, 37