Package ‘wsyn’

December 18, 2019

Version 1.0.2
Type Package
Title Wavelet Approaches to Studies of Synchrony in Ecology and Other Fields
License GPL-3
Encoding UTF-8
LazyData true
Imports fields (>= 9.6), graphics (>= 3.4.4), grDevices (>= 3.4.4), MASS (>= 7.3-47), stats (>= 3.4.4)
Suggests knitr, mvtnorm, rmarkdown, testthat, vdiff
VignetteBuilder knitr
RoxygenNote 6.0.1
NeedsCompilation no
Author Daniel C. Reuman [aut, cre],
Thomas L. Anderson [aut],
Jonathan A. Walter [aut],
Lei Zhao [aut],
Lawrence W. Sheppard [aut]
Maintainer Daniel C. Reuman <reuman@ku.edu>
Repository CRAN
Date/Publication 2019-12-18 22:30:03 UTC

R topics documented:

  addranks ................................................................. 3
  addwmfs ................................................................. 3
<table>
<thead>
<tr>
<th>R topics documented</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addwpmfs</td>
<td>4</td>
</tr>
<tr>
<td>bandtest</td>
<td>6</td>
</tr>
<tr>
<td>bctrans</td>
<td>7</td>
</tr>
<tr>
<td>cleandat</td>
<td>8</td>
</tr>
<tr>
<td>cluseigen</td>
<td>10</td>
</tr>
<tr>
<td>clust</td>
<td>11</td>
</tr>
<tr>
<td>clust_methods</td>
<td>14</td>
</tr>
<tr>
<td>coh</td>
<td>16</td>
</tr>
<tr>
<td>coh_methods</td>
<td>19</td>
</tr>
<tr>
<td>errcheck_stdat</td>
<td>21</td>
</tr>
<tr>
<td>errcheck_times</td>
<td>22</td>
</tr>
<tr>
<td>errcheck_tsdat</td>
<td>22</td>
</tr>
<tr>
<td>errcheck_tts</td>
<td>23</td>
</tr>
<tr>
<td>errcheck_wavparam</td>
<td>23</td>
</tr>
<tr>
<td>fastcohtest</td>
<td>24</td>
</tr>
<tr>
<td>fftsurrog</td>
<td>25</td>
</tr>
<tr>
<td>is.connected</td>
<td>26</td>
</tr>
<tr>
<td>makeunweighted</td>
<td>27</td>
</tr>
<tr>
<td>mnphase</td>
<td>28</td>
</tr>
<tr>
<td>modularity</td>
<td>28</td>
</tr>
<tr>
<td>normforcoh</td>
<td>30</td>
</tr>
<tr>
<td>plotmag</td>
<td>30</td>
</tr>
<tr>
<td>plotmap</td>
<td>33</td>
</tr>
<tr>
<td>plotphase</td>
<td>34</td>
</tr>
<tr>
<td>plotrank</td>
<td>36</td>
</tr>
<tr>
<td>power</td>
<td>38</td>
</tr>
<tr>
<td>predsync</td>
<td>39</td>
</tr>
<tr>
<td>print.summary_wsyn</td>
<td>40</td>
</tr>
<tr>
<td>setmints</td>
<td>41</td>
</tr>
<tr>
<td>set_adj</td>
<td>42</td>
</tr>
<tr>
<td>surrog</td>
<td>48</td>
</tr>
<tr>
<td>syncexpl</td>
<td>49</td>
</tr>
<tr>
<td>synmat</td>
<td>51</td>
</tr>
<tr>
<td>tts</td>
<td>53</td>
</tr>
<tr>
<td>tts_methods</td>
<td>54</td>
</tr>
<tr>
<td>warray</td>
<td>55</td>
</tr>
<tr>
<td>wavmatwork</td>
<td>56</td>
</tr>
<tr>
<td>wlm</td>
<td>57</td>
</tr>
<tr>
<td>wlmfit</td>
<td>59</td>
</tr>
<tr>
<td>wlmtest</td>
<td>60</td>
</tr>
<tr>
<td>wlmtest_methods</td>
<td>62</td>
</tr>
<tr>
<td>wlm_methods</td>
<td>64</td>
</tr>
<tr>
<td>wmf</td>
<td>66</td>
</tr>
<tr>
<td>wmf_methods</td>
<td>67</td>
</tr>
<tr>
<td>wpmf</td>
<td>69</td>
</tr>
<tr>
<td>wpmf_methods</td>
<td>71</td>
</tr>
<tr>
<td>wt</td>
<td>72</td>
</tr>
<tr>
<td>wt_methods</td>
<td>74</td>
</tr>
</tbody>
</table>
addranks

Index

addranks

Add rank information to a coh or wlmtest object

Description

When a coh or wlmtest object is created, the ranks slot is NA. This function fills it in.

Usage

addranks(obj)

Arguments

obj An object of class coh or wlmtest

Value

addranks returns another coh or wlmtest object with ranks slot now included. If obj$ranks was not NA, the object is returned as is.

Note

Internal function, no error checking performed

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

See Also

coh, wlmtest, bandtest, browseVignettes("wsyn")

addwmfs

Add wavelet mean field information to a clust object

Description

When a clust object is created, the wmfs slot is NA. This function fills it in.

Usage

addwmfs(obj)
Arguments

obj     An object of class clust

Details

This function uses the values of scale.min, scale.max.input, sigma and f0 stored in obj$methodspecs. It is possible to create a clust object with bad values for these slots. This function throws an error in that case. You can use a correlation-based method for calculating the synchrony matrix and still pass values of scale.min, scale.max.input, sigma and f0 to clust (in fact, this happens by default) - they won't be used by clust, but they will be there for later use by addwmfs and addwpmfs.

Value

addwmfs returns another clust object with wmfs slot now included. If obj$wmfs was not NA, the object is returned as is.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also

clust, addwpmfs, browseVignettes("wsyn")

Examples

```r
sig<-matrix(.8,5,5)
diag(sig)<-1
lents<-50
dat1<-t(mvtnorm::rmvnorm(lents,mean=rep(0,5),sigma=sig))
dat2<-t(mvtnorm::rmvnorm(lents,mean=rep(0,5),sigma=sig))
dat<-rbind(dat1,dat2)
times<-1:lents
dat<-cleandat(dat,times,clev=1)$cdat
coords<-data.frame(Y=rep(0,10),X=1:10)
method<="coh.sig.fast"
clustobj<clust(dat,times,coords,method,nsurrogs = 100)
res<-addwmfs(clustobj)
```
addwpmfs

Usage

addwpmfs(obj, level = 1:length(obj$clusters), sigmethod = "quick", nrand = 1000)

Arguments

obj An object of class clust
level The clustering level(s) to use. 1 corresponds to no clustering. The default is all levels of clustering.
sigmethod Method for significance testing the wpmf, one of quick, fft, aaf (see details of the wpmf function)
nrand The number of randomizations to be used for significance testing

Details

This function uses the values of scale.min, scale.max.input, sigma and f0 stored in obj$methodspecs. It is possible to create a clust object with bad values for these slots. This function throws an error in that case. You can use a correlation-based method for calculating the synchrony matrix and still pass values of scale.min, scale.max.input, sigma and f0 to clust (in fact, this happens by default) - they won't be used by clust, but they will be there for later use by addwmfs and addwpmfs.

Value

addwpmfs returns another clust object with wpmfs slot now included, or more filled in than it was previously. With values of sigmethod other than "quick", this function can be slow, particularly with large nrand. So in that case the user may want to set level equal only to one clustering level of interest. Unlike wmf, old values in obj$wpmfs are overwritten.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also

clust, addwmfs, browseVignettes("wsyn")

Examples

sig<-matrix(.8,5,5)
diag(sig)<-1
lents<-50
dat1<-t(mvtnorm::rmvnorm(lents,mean=rep(0,5),sigma=sig))
dat2<-t(mvtnorm::rmvnorm(lents,mean=rep(0,5),sigma=sig))
dat<-rbind(dat1,dat2)
times<-1:lents
dat<-cleandat(dat,times,clev=1)$cdat
coords<-data.frame(Y=rep(0,10),X=1:10)
method<"coh.sig.fast"
clustobj<-clust(dat,times,coords,method,nsurrogs = 100)
bandtest

Aggregate significance across a timescale band

Description

Computes the aggregate significance of coherence (coh) or of a wavelet linear model test object (wlmtest) across a timescale band, accounting for non-independence of timescales. Also gets the average phase across the band, in the case of coherence.

Usage

bandtest(object, ...)

## Default S3 method:
bandtest(object, ...)

## S3 method for class 'coh'
bandtest(object, band, ...)

## S3 method for class 'wlmtest'
bandtest(object, band, ...)

Arguments

object An object of class coh or wlmtest, must have a non-NA signif slot
...
Passed from the generic to specific methods. Not currently used.
band A length-two numeric vector indicating a timescale band

Value

bandtest returns an object of the same class as its first input but with a bandp slot added. Or if there was already a bandp slot, the output has a bandp slot with an additional row. For a coh object, the bandp slot is a data frame with four columns, the first two indicating the timescale band and the third an associated p-value for the test of coherence over that band. The fourth column is the average phase over the band. For a wlmtest object, the result is only the first three of the above columns.

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>
**References**

Sheppard, L.W., et al. (2016) Changes in large-scale climate alter spatial synchrony of aphid pests. Nature Climate Change. DOI: 10.1038/nclimate2881

**See Also**

coh, wlm, wlmtest, browseVignettes("wsyn")

**Examples**

```r
#Example for a coh object
times<-(-3:100)
ts1<-sin(2*pi*times/10)
ts2<-5*sin(2*pi*times/3)
artsig_x<-matrix(NA,11,length(times)) #the driver
for (counter in 1:11)
{
    artsig_x[counter,]=ts1+ts2+rnorm(length(times),mean=0,sd=1.5)
}
times<-0:100
artsig_y<-matrix(NA,11,length(times)) #the driven
for (counter1 in 1:11)
{
    for (counter2 in 1:101)
    {
        artsig_y[counter1,counter2]<-mean(artsig_x[counter1,counter2:(counter2+2)])
    }
}
artsig_y<-artsig_y+matrix(rnorm(length(times)*11,mean=0,sd=3),11,length(times))
artsig_x<-cleandat(artsig_x,times,1)$cdat
artsig_y<-cleandat(artsig_y,times,1)$cdat
cohobj<-coh(dat1=artsig_x,dat2=artsig_y,times=times,norm="powall",sigmethod="fast",nrand=1000,
f0=0.5,scale.max.input=28)
cohobj<-bandtest(cohobj,c(2,4))

#Example for a wlmtest object - see vignette
```

---

**bctrans**

The one-parameter family of Box-Cox transformations

**Description**

The one-parameter family of Box-Cox transformations

**Usage**

bctrans(y, lambda)
Arguments

y
A numeric, positive values assumed

lambda
The Box-Cox parameter

Details

Internal function. No error checking done. It is assumed the entries of y are positive.

Value

bctrans gives \((y^\text{lambda} - 1) / \text{lambda}\) for \text{lambda} not 0 or \(\ln(y)\) for \text{lambda} equal to 0.

Author(s)

Daniel Reuman, <reuman@ku.edu>

References


See Also

cleandat, browseVignettes("wsyn")

cleandat

Clean (spatio)temporal data matrices to make them ready for analyses using the wsyn package

Description

A data cleaning function for optimal Box-Cox transformation, detrending, standardizing variance, de-meaning

Usage

cleandat(dat, times, clev, lambdas = seq(-10, 10, by = 0.01), mints = NA)

Arguments

dat
A locations x time data matrix, or a time series vector (for 1 location)
times
The times of measurement, spacing 1
clev
The level of cleaning to do, 1 through 5. See details.
lambdas
A vector of lambdas to test for optimal Box-Cox transformation, if Box-Cox is performed. Ignored for clev<4. Defaults to seq(-10,10, by=0.01). See details.
mints

If clev is 4 or 5, then time series are shifted to have this minimum value before Box-Cox transformation. Default NA means use the smallest difference between consecutive, distinct sorted values. NaN means perform no shift.

Details

NAs, Infs, etc. in dat trigger an error. If clev==1, time series are (individually) de-meaned. If clev==2, time series are (individually) linearly detrended and de-meaned. If clev==3, time series are (individually) linearly detrended and de-meaned, and variances are standardized to 1. If clev==4, an optimal Box-Cox normalization procedure is applied jointly to all time series (so the same Box-Cox transformation is applied to all time series after they are individually shifted depending on the value of mints). Transformed time series are then individually linearly detrended, de-meaned, and variances are standardized to 1. If clev==5, an optimal Box-Cox normalization procedure is applied to each time series individually (again after individually shifting according to mints), and transformed time series are then individually linearly detrended, de-meaned, and variances are standardized to 1. Constant time series and perfect linear trends trigger an error for clev>=3. If clev>=4 and the optimal lambda for one or more time series is a boundary case or if there is more than one optimal lambda, it triggers a warning. A wider range of lambda should be considered in the former case.

Value
cleandat returns a list containing the cleaned data, clev, and the optimal lambdas from the Box-Cox procedure (NA for clev<4, see details).

Author(s)
Jonathan Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>; Lei Zhao, <lei.zhao@cau.edu.cn>

References


See Also

wt, wmf, wpmf, coh, wlm, wlmtest, clust, browseVignettes("wsyn")

Examples

times<-1:100
dat<-rnorm(100)
res1<-cleandat(dat,times,1) #this removes the mean
res2<-cleandat(dat,times,2) #detrends and removes the mean
res3<-cleandat(dat,times,3) #variances also standardized
res4<-cleandat(dat,times,4) #also joint Box-Cox applied
res5 <- cleandat(dat, times, 5) # 1-3, also indiv Box-Cox

cludeigen                      Community structure detection in networks

Description
Community structure detection in networks based on the leading eigenvector of the community matrix

Usage
cluseigen(adj)

Arguments
adj
An adjacency matrix. Should be symmetric with diagonal containing zeros.

Details
The difference between this function and the algorithm described by Newman is that this function can be used on an adjacency matrix with negative elements, which is very common for correlation matrices and other measures of pairwise synchrony of time series.

Value
cluseigen returns a list with one element for each of the splits performed by the clustering algorithm. Each element is a vector with entries corresponding to rows and columns of adj and indicating the module membership of the node, following the split. The last element of the list is the final clustering determined by the algorithm when its halting condition is satisfied. The first element is always a vector of all 1s (corresponding to before any splits are performed).

Author(s)
Lei Zhao, <lei.zhao@cau.edu.cn>; Daniel Reuman, <reuman@ku.edu>

References

See Also
clust.modularity, browseVignettes("wsyn")
Examples

```r
adj<-matrix(0, 10, 10) # create a fake adjacency matrix
adj[lower.tri(adj)]<-runif(10*9/2, -1, 1)
adj<-adj+t(adj)
colnames(adj)<-letters[1:10]
z<-cluseigen(adj)
```

---

**clust**

*Detection and description of clusters of synchronous locations*

---

**Description**

Generator function for the `clust` S3 class, which supports tools for detecting clusters (aka, modules, sub-networks, communities, etc.) of especially synchronous locations.

**Usage**

```r
clust(dat, times, coords, method, tsrange = c(0, Inf), nsurrogs = 1000,
      scale.min = 2, scale.max.input = NULL, sigma = 1.05, f0 = 1,
      weighted = TRUE, sigthresh = 0.95)
```

**Arguments**

- `dat`: A locations (rows) x time (columns) matrix of measurements
- `times`: The times at which measurements were made, spacing 1
- `coords`: A data frame containing X,Y coordinates of locations in `data`, with column names either `X` and `Y` or `lon` and `lat` or `longitude` and `latitude`. The data frame may contain other columns with additional metainformation about the sites.
- `tsrange`: A vector containing the min and max of the focal timescale range. Defaults to all timescales that are valid given choices for `scale.min`, `scale.max.input`, `f0`, `sigma`. Only used for wavelet-based methods.
- `nsurrogs`: Number of surrogates for significance test. Defaults to 1000. Only used for surrogate-based methods.
- `scale.min`: The smallest scale of fluctuation that will be examined. At least 2. Used only for wavelet-based methods.
- `scale.max.input`: The largest scale of fluctuation guaranteed to be examined. Only used for wavelet-based methods.
- `sigma`: The ratio of each time scale examined relative to the next timescale. Should be greater than 1. Only used for wavelet-based methods.
- `f0`: The ratio of the period of fluctuation to the width of the envelope. Only used for wavelet-based methods.
weighted If TRUE, create a weighted network. If FALSE, create a binary network using statistical significance. Binary networks are only allowed for networks based on significance.
sigthresh Significance threshold needed, if weighted is false, for a network link to be realized. Typically 0.95, 0.99, or 0.999, etc. Only used if weighted is FALSE.

Details
The following values are valid for method: "pearson", "pearson.sig.std", "pearson.sig.fft", "pearson.sig.aaft", "spearman", "spearman.sig.std", "spearman.sig.fft", "spearman.sig.aaft", "kendall", "kendall.sig.std", "kendall.sig.fft", "kendall.sig.aaft", "ReXWT", "ReXWT.sig.fft", "ReXWT.sig.aaft", "ReXWT.sig.fast", "coh", "coh.sig.fft", "coh.sig.aaft", "coh.sig.fast", "phasecoh", "phasecoh.sig.fft", and "phasecoh.sig.aaft". The first portions of these identifiers correspond to the Pearson, Spearman, and Kendall correlations, the real part of the cross-wavelet transform, the wavelet coherence, and the wavelet phase coherence. The second portions of these identifiers, when present, indicates that significance of the measure specified in the first portion of the identifies is to be used for establishing the synchrony matrix. Otherwise the value itself is used. The third part of the method identifier indicates what type of significance is used. Significance testing is performed using standard approaches (method flag containing std; for correlation coefficients, although these are inappropriate for autocorrelated data), or surrogates generated using the Fourier (method flag containing "fft") or amplitude adjusted Fourier surrogates ("aaft"). For "coh" and "ReXWT", the fast testing algorithm of Sheppard et al. (2017) is also implemented ("fast"). That method uses implicit Fourier surrogates. The choice of wavelet coherence (method flag containing "coh") or the real part of the cross-wavelet transform (method flag containing "ReXWT") depends mainly on treatment of out-of-phase relationships. The "ReXWT" is more akin to a correlation coefficient in that strong in-phase relationships approach 1 and strong antiphase relationships approach -1. Wavelet coherence allows any phase relationship and ranges from 0 to 1. Power normalization is applied for "coh" and for "ReXWT". All significance tests are one-tailed. Synchrony matrices for significance-based methods when weighted is TRUE contain 1 minus the p-values.

Clustering is performed using the the eigenvector-based modularity method of Newman (2006).

Value
clust returns an object of class clust. Slots are:
dat The input
times The input
coords The input
methodspecs A list with elements specifying the method used, and methodological parameters that were in the input.
adj The adjacency matrix that defines the synchrony network
clusters A list with one element for each successive split of the networks into subcomponents carried out by the clustering algorithm. Each element is a vector of length equal to the number of nodes in the original network, giving cluster membership of the nodes. The first element is a vector of all 1s, corresponding to before the first clustering split was performed.
clust

modres  A list of the same length as clusters, with each element containing the results of calling modularity on the network split to that level.

mns  Mean time series for modules. A list of the same length as clusters.

wmfs  Wavelet mean fields for modules. NA when clust is first called, but addwmfs causes this entry to be added. It is a list. See documentation for the method addwmfs.

wpmfs  Wavelet phasor mean fields for modules. NA when clust is first called, but addwpmfs causes this entry to be added. It is a list. See documentation for the method addwpmfs.

Author(s)

Jonathan Walter, <jaw3es@virginia.edu>; Daniel Reuman, <reuman@ku.edu>; Lei Zhao, <lei.zhao@cau.edu.cn>

References


See Also

cludeigen, modularity, addwmfs, addwpmfs, clust_methods, synmat, plotmap, browseVignettes("wsyn")

Examples

```r
sig <- matrix(.8, 5, 5)
diag(sig) <- 1
lents <- 50
dat1 <- t(mvtnorm::rmvnorm(lents, mean = rep(0, 5), sigma = sig))
dat2 <- t(mvtnorm::rmvnorm(lents, mean = rep(0, 5), sigma = sig))
dat <- rbind(dat1, dat2)
times <- 1: lents
dat <- cleandat(dat, times, clev = 1)$cdat
coords <- data.frame(Y = rep(0, 10), X = 1: 10)
method <- "coh.sig.fast"
res <- clust(dat, times, coords, method, nsurrogs = 50)
# nsurrogs should be much higher for a real application
```
clust_methods  Basic methods for the clust class

Description

Set, get, summary, and print methods for the clust class.

Usage

```r
## S3 method for class 'clust'
summary(object, ...)

## S3 method for class 'clust'
print(x, ...)

## S3 method for class 'clust'
set_times(obj, newval)

## S3 method for class 'clust'
set_adj(obj, newval)

## S3 method for class 'clust'
set_clusters(obj, newval)

## S3 method for class 'clust'
set_modres(obj, newval)

## S3 method for class 'clust'
set_mns(obj, newval)

## S3 method for class 'clust'
set_dat(obj, newval)

## S3 method for class 'clust'
set_coords(obj, newval)

## S3 method for class 'clust'
set_methodspecs(obj, newval)

## S3 method for class 'clust'
set_wmfs(obj, newval)

## S3 method for class 'clust'
set_wpmfs(obj, newval)

## S3 method for class 'clust'
get_times(obj)
```
## S3 method for class 'clust'
get_adj(obj)

## S3 method for class 'clust'
get_clusters(obj)

## S3 method for class 'clust'
get_modres(obj)

## S3 method for class 'clust'
get_mns(obj)

## S3 method for class 'clust'
get_dat(obj)

## S3 method for class 'clust'
get_coords(obj)

## S3 method for class 'clust'
get_methodspec(obj)

## S3 method for class 'clust'
get_wmfs(obj)

## S3 method for class 'clust'
get_wpmfs(obj)

### Arguments

- **object, x, obj**  
  An object of class clust

- **...**  
  Not currently used. Included for argument consistency with existing generics.

- **newval**  
  A new value, for the set_* methods

### Value

`summary.clust` produces a summary of a clust object. A `print.clust` method is also available. For clust objects, set_* and get_* methods are available for all slots (see the documentation for clust for a list). The set_* methods just throw an error, to prevent breaking the consistency between the slots of a clust object.

### Author(s)

Daniel Reuman, <reuman@ku.edu>

### See Also

- clust
Examples

```r
sig <- matrix(.8, 5, 5)
diag(sig) <- 1
lents <- 50
dat1 < t(mvtnorm::rmvnorm(lents, mean = rep(0, 5), sigma = sig))
dat2 < t(mvtnorm::rmvnorm(lents, mean = rep(0, 5), sigma = sig))
dat <- rbind(dat1, dat2)
times <- 1:lents
dat <- cleandat(dat, times, clev = 1)$cdat
coods <- data.frame(Y = rep(0, 10), X = 1:10)
method <- "coh.sig.fast"
h <- clust(dat, times, coords, method, nsurrogs = 50)
# nsurrogs should be much higher for a real application
get_times(h)
summary(h)
print(h)
```

---

**coh**

---

**Coherence**

**Description**

Wavelet coherence and wavelet phase coherence, spatial or for single time series. Also the generator function for the `coh` class, which inherits from the `list` class.

**Usage**

```r
coh(dat1, dat2, times, norm, sigmethod = "none", nrand = 1000,
scale.min = 2, scale.max.input = NULL, sigma = 1.05, f0 = 1)
```

**Arguments**

- **dat1**: A locations (rows) x time (columns) matrix (for spatial coherence), or a single time series
- **dat2**: Same format as dat1, same locations and times
- **times**: The times at which measurements were made, spacing 1
- **norm**: The normalization of wavelet transforms to use. Controls the version of the coherence that is performed. One of "none", "phase", "powall", "powind". See details.
- **sigmethod**: The method for significance testing. One of "none", "fftsurrog1", "fftsurrog2", "fftsurrog12", "aaftsurrog1", "aaftsurrog2", "aaftsurrog12", "fast". See details.
- **nrand**: Number of surrogate randomizations to use for significance testing.
- **scale.min**: The smallest scale of fluctuation that will be examined. At least 2.
- **scale.max.input**: The largest scale of fluctuation guaranteed to be examined
The ratio of each time scale examined relative to the next timescale. Should be greater than 1.

\( f_0 \) The ratio of the period of fluctuation to the width of the envelope

**Details**

If the dimensions of \( dat1 \) and \( dat2 \) are \( N \) by \( T \) (\( N \) is 1 for vector \( dat1 \) and \( dat2 \)), and if the wavelet transform of the \( n \)th row of \( dati \) is denoted \( W_{i,n,\sigma}(t) \), then the coherence is the average, over all locations \( n \) and times \( t \) for which wavelet transforms are available, of the quantity \( w_{1,n,\sigma}(t)w_{2,n,\sigma}(t)^* \), where the \( * \) represents complex conjugation and \( w_{i,n,\sigma}(t) \) is a normalization of the wavelet transform. The normalization used depends on \( norm \). If \( norm \) is "none" then raw wavelet transforms are used. If \( norm \) is "phase" then \( w_{i,n,\sigma}(t) = W_{i,n,\sigma}(t)/|W_{i,n,\sigma}(t)| \), which gives the wavelet phase coherence, or the spatial wavelet phase coherence if \( N > 1 \). If \( norm \) is "powall" then the normalization is that described in the "Wavelet mean field" section of the Methods of Sheppard et al. (2016), giving the version of the coherence that was there called simply the wavelet coherence, or the spatial wavelet coherence if \( N > 1 \). If \( norm \) is "powind", then \( w_{i,n,\sigma}(t) \) is obtained by dividing \( W_{i,n,\sigma}(t) \) by the square root of the average of \( W_{i,n,\sigma}(t)W_{i,n,\sigma}(t)^* \) over the times for which it is defined; this is done separately for each \( i \) and \( n \).

The slot \( signif \) is NA if \( sigmethod \) is "none". Otherwise, and if \( sigmethod \) is not "fast", then \( signif$coher \) is the same as \( coher \), and \( signif$scoher \) is a matrix of dimensions \( nrand \) by \( length(coher) \) with rows with magnitudes equal to coherences of surrogate datasets, computed using the normalization specified by \( norm \). The type of surrogate used (Fourier surrogates or amplitude adjusted Fourier surrogates, see \( surrog \)), as well as which of the datasets surrogates are computed on (\( dat1, dat2, \) or both) is determined by \( sigmethod \). The first part of the value of \( sigmethod \) specifies the type of surrogate used, and the numbers in the second part (1, 2, or 12) specify whether surrogates are applied to \( dat1, dat2, \) or both, respectively. Synchrony-preserving surrogates are used. A variety of statements of significance (or lack thereof) can be made by comparing \( signif$coher \) with \( signif$scoher \) (see the \( plotmag, plotrank, \) and \( bandtest \) methods for the \( coh \) class). If \( sigmethod \) is "fast", the fast algorithm of Sheppard et al. (2017) is used. In that case \( signif$coher \) can be compared to \( signif$scoher \) to make significance statements about the coherence in exactly the same way, but \( signif$coher \) will no longer precisely equal \( coher \), and \( coher \) should not be compared directly to \( signif$scoher \). Statements about significance of the coherence should be made using \( signif$coher \) and \( signif$scoher \), whereas \( coher \) should be used whenever the actual value of the coherence is needed. No fast algorithm exists for \( norm \) equal to "phase" (the phase coherence; Sheppard et al, 2017), so if \( norm \) is "phase" and \( sigmethod \) is "fast", the function throws an error.

The slots \( ranks \) and \( bandp \) are empty on an initial call to \( coh \). They are made to compute and hold aggregate significance results over any timescale band of choice. These are filled in when needed by other methods, see \( plotrank \) and \( bandtest \).

Regardless of what the variables represent, the normalized transform of \( dat1 \) is multiplied by the conjugate of the normalized transform of \( dat2 \). Thus, a positive phase of the coherence indicates \( dat1 \) would be leading \( dat2 \).

**Value**

\( coh \) returns an object of class \( coh \). Slots are:

- \( dat1, dat2 \) The input data
times The times associated with the data

sigmethod The method for significance testing, as inputted.

norm The normalization of the wavelet transforms that will be used in computing the coherence. Different values result in different versions of the coherence. One of "none", "phase", "powall", "powind". See details.

wtopt The inputted wavelet transform options scale.min, scale.max.input, sigma, f0 in a list

timescales The timescales associated with the coherence

coh The complex magnitude of this quantity is the coherence, calculated in the usual way (which depends on norm, see details), and with scalloping of the transforms.

signif A list with information from the significance testing. Elements are coher and scoher. See details.

ranks A list with ranking information for signif. NA until plotrank is called, see documentation for plotrank.

bandp A data frame containing results of computing significances of the coherence across timescale bands. Empty on an initial call to coh, filled in by the function bandtest. See details.

Author(s)
Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References
Sheppard, L.W., et al. (2016) Changes in large-scale climate alter spatial synchrony of aphid pests. Nature Climate Change. DOI: 10.1038/nclimate2881

Sheppard, L.W., et al. (2017) Rapid surrogate testing of wavelet coherences. European Physical Journal, Nonlinear and Biomedical Physics, 5, 1. DOI: 10.1051/epjnbp/2017000

See Also
cleandat, coh_methods, bandtest, plotmag, plotphase, plotrank. browseVignettes("wsyn")

Examples
times<-1:100
dat1<-matrix(rnorm(1000),10,100)
dat2<-matrix(rnorm(1000),10,100)
dat1<-cleandat(dat1,times,1)$cdat
dat2<-cleandat(dat2,times,1)$cdat
norm<="powall"
sigmethod<="fast"
nrand<10
res<coh(dat1,dat2,times,norm,sigmethod,nrand)
#for real applications, use a much bigger nrand
### Basic methods for the `coh` class

**Description**

Set, get, summary, and print methods for the `coh` class.

**Usage**

```r
## S3 method for class 'coh'
summary(object, ...)

## S3 method for class 'coh'
print(x, ...)

## S3 method for class 'coh'
set_times(obj, newval)

## S3 method for class 'coh'
set_timescales(obj, newval)

## S3 method for class 'coh'
set_coher(obj, newval)

## S3 method for class 'coh'
set_dat1(obj, newval)

## S3 method for class 'coh'
set_dat2(obj, newval)

## S3 method for class 'coh'
set_wtopt(obj, newval)

## S3 method for class 'coh'
set_norm(obj, newval)

## S3 method for class 'coh'
set_sigmethod(obj, newval)

## S3 method for class 'coh'
set_signif(obj, newval)

## S3 method for class 'coh'
set_ranks(obj, newval)

## S3 method for class 'coh'
set_bandp(obj, newval)
```
## S3 method for class 'coh'
get_times(obj)

## S3 method for class 'coh'
get_timescales(obj)

## S3 method for class 'coh'
get_coher(obj)

## S3 method for class 'coh'
get_dat1(obj)

## S3 method for class 'coh'
get_dat2(obj)

## S3 method for class 'coh'
get_wtopt(obj)

## S3 method for class 'coh'
get_norm(obj)

## S3 method for class 'coh'
get_sigmethod(obj)

## S3 method for class 'coh'
get_signif(obj)

## S3 method for class 'coh'
get_ranks(obj)

## S3 method for class 'coh'
get_bandp(obj)

### Arguments

- `object`, `x`, `obj` An object of class coh
- `...` Not currently used. Included for argument consistency with existing generics.
- `newval` A new value, for the set_* methods

### Value

`summary.coh` produces a summary of a coh object. A print.coh method is also available. For coh objects, set_* and get_* methods are available for all slots (see the documentation for coh for a list). The set_* methods just throw an error, to prevent breaking the consistency between the slots of a coh object.
**errcheck_stdat**

*Error check for appropriate spatio-temporal data*

**Description**

Error checking whether a times vector and a matrix with each row a time series make a legitimate spatio-temporal data set for wavelet analysis

**Usage**

```r
errcheck_stdat(times, dat, callfunc)
```

**Arguments**

- **times**: the times of measurement, spacing 1
- **dat**: each row is a time series - must have at least two rows
- **callfunc**: the function calling this one, for error tracking

**Value**

`errcheck_stdat` returns nothing but throws and error if inputs not appropriate

**Author(s)**

Daniel Reuman, <reuman@ku.edu>
errcheck_times  

**Error check times**

**Description**

Error check whether a vector can represent times at which data suitable for wavelet transforms were measured

**Usage**

```r
errcheck_times(times, callfunc)
```

**Arguments**

- `times`: Tests whether this is a numeric vector with unit-spaced increasing values
- `callfunc`: Function calling this one, for better error messaging

**Value**

`errcheck_times` returns nothing but throws and error if the conditions are not met

**Author(s)**

Daniel Reuman, <reuman@ku.edu>

---

errcheck_tsdat  

**Error check for appropriate temporal data**

**Description**

Error checking whether a times vector and t.series vector make a legitimate time series for wavelet analysis

**Usage**

```r
errcheck_tsdat(times, t.series, callfunc)
```

**Arguments**

- `times`: times of measurement, spacing 1
- `t.series`: the measurements
- `callfunc`: the function from which this one was called, for error tracking

**Value**

`errcheck_tsdat` returns nothing but throws and error if inputs not appropriate
errcheck_tts

Author(s)
Daniel Reuman, <reuman@ku.edu>

Description
Error check whether inputs are suitable for a tts object

Usage
errcheck_tts(times, timescales, values, callfunc)

Arguments
- **times**: times of measurement, spacing 1
- **timescales**: timescales of analysis
- **values**: a times by timescales matrix
- **callfunc**: the function from which this one was called, for error tracking

Value
errcheck_tts returns nothing but throws and error if inputs not appropriate

Author(s)
Daniel Reuman, <reuman@ku.edu>

errcheck_wavparam

Description
Error check wavelet transform parameters

Usage
errcheck_wavparam(scale.min, scale.max.input, sigma, f0, times, callfunc)
Arguments

- **scale.min**: The smallest scale of fluctuation that will be examined. At least 2.
- **scale.max.input**: The largest scale of fluctuation that is guaranteed to be examined
- **sigma**: The ratio of each time scale examined relative to the next timescale. Should be greater than 1.
- **f0**: The ratio of the period of fluctuation to the width of the envelope. Defaults to 1.
- **times**: The times data were measured at, spacing 1
- **callfunc**: Function calling this one, for better error messaging

Value

`errcheck_wavparam` returns nothing but throws an error if the conditions are not met

Author(s)

Daniel Reuman, <reuman@ku.edu>

---

**fastcohtest**  
*Fast algorithm for significance testing coherence using Fourier surrogates*

Description

This is the algorithm of Sheppard et al. (2017) (see references).

Usage

```r
fastcohtest(dat1, dat2, scale.min, scale.max.input, sigma, f0, nrand, randnums, randbits, norm)
```

Arguments

- **dat1**: A locations (rows) x time (columns) matrix (for spatial coherence), or a single time series
- **dat2**: Same format as dat1, same locations and times
- **scale.min**: The smallest scale of fluctuation that will be examined. At least 2.
- **scale.max.input**: The largest scale of fluctuation guaranteed to be examined
- **sigma**: The ratio of each time scale examined relative to the next timescale. Should be greater than 1.
- **f0**: The ratio of the period of fluctuation to the width of the envelope
- **nrand**: Number of surrogate randomizations to use for significance testing
fftsurrog

randnums A bunch of independent random numbers uniformly distributed on (0,1). There must be \text{nrand} \ast \text{floor}((\text{dim(dat1)}[2]-1)/2) of these.

randbits A bunch of random bits (0 or 1). There must be \text{nrand} of these if time series are of odd length and 2*\text{nrand} if even length. You may pass more than this, so, in particular, you may pass 2*\text{nrand} for even or odd length.

norm The normalization of wavelet transforms to use. Controls the version of the coherence that is performed. One of "none", "powall", "powind". See details in the documentation of \text{coh}.

Value

\text{fastcohtest} returns a list with these elements:

- \text{timescales} The timescales used
- \text{coher} The magnitude of this is the fast-algorithm version of the coherence between the two datasets, for comparison with \text{scoher}
- \text{scoher} A matrix with \text{nrand} rows, the magnitude of each one is the fast-algorithm version of the coherence for a surrogate

Note

Internal function, minimal error checking.

Author(s)

Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References

Sheppard, L.W., et al. (2017) Rapid surrogate testing of wavelet coherences. European Physical Journal, Nonlinear and Biomedical Physics, 5, 1. DOI: 10.1051/epjnbp/2017000

fftsurrog \hspace{1cm} \textit{Surrogate time series using Fourier surrogates}

Description

Creates surrogate time series using Fourier surrogates

Usage

\text{fftsurrog}(\text{dat}, \text{nsurrogs}, \text{syncpres})
is.connected

Arguments

- dat: A locations x time matrix of observations
- nsurrogs: The number of surrogates to produce
- syncpres: Logical. TRUE for "synchrony preserving" surrogates (same phase randomizations used for all time series). FALSE leads to independent phase randomizations for all time series.

Value

fftsurrog returns a list of nsurrogs surrogate datasets

Note

For internal use, no error checking

Author(s)

Jonathan Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References


is.connected

Tests if a graph is connected

Description

Tests if a graph represented by an adjacency matrix is connected.

Usage

is.connected(adj)

Arguments

- adj: An adjacency matrix. Must be a numeric matrix with non-negative entries.

Details

Idea by Ed Scheinerman, circa 2006. Source: http://www.ams.jhu.edu/~ers/matgraph/; routine: matgraph/@graph/isconnected.m
Value

`is.connected` returns TRUE or FALSE depending on whether the graph represented in `adj` is a connected graph.

Author(s)

Lei Zhao, <lei.zhao@cau.edu.cn>

See Also

`cluseigen`, `clust`, `browseVignettes("wsyn")`

Examples

```r
  g1 <- matrix(c(0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0), 4, 4)
  is.connected(g1)
  g2 <- matrix(c(0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0), 4, 4)
  is.connected(g2)
```

Description

Convenience function for converting certain synchrony matrices to unweighted versions

Usage

```r
makeunweighted(mat, sigthresh)
```

Arguments

- `mat`: A synchrony matrix based on significance testing
- `sigthresh`: Significance threshold to use

Value

`makeunweighted` converts to an unweighted version of the input. Entries of `mat` less than `sigthresh` become a 1, other entries become a 0. The diagonal is NA.

Note

Internal function, no error checking

Author(s)

Lei Zhao, <lei.zhao@cau.edu.cn>, Daniel Reuman <reuman@ku.edu>
**mnphase**

*Mean phase of coherence*

**Description**

Gets the mean phase of a bunch of complex numbers

**Usage**

`mnphase(nums)`

**Arguments**

- `nums` A vector of complex numbers

**Value**

`mnphase` returns the mean phase

**Note**

Internal function, no error catching

**Author(s)**

Daniel Reuman, <reuman@ku.edu>

---

**modularity**

*Modularity of a community structure of a graph*

**Description**

Computes the modularity of partitioning of a graph into sub-graphs. Similar to the modularity function in the igraph package, but allows negative edge weights.

**Usage**

`modularity(adj, membership, decomp = FALSE)`

**Arguments**

- `adj` An adjacency matrix, which should be symmetric with zeros on the diagonal.
- `membership` Vector of length equal to the number of graph nodes (columns/rows of `adj`) indicating the cluster/sub-graph each node belongs to.
- `decomp` Logical. If TRUE, calculate the decomposition of modularity by modules and nodes. Default FALSE.
Details

The difference between this function and the function `modularity` in the package `igraph` is that this function can be used with an adjacency matrix with negative elements. This is a common case for matrices arising from a correlation matrix or another synchrony matrix. If the matrix is non-negative, the result of this function should be exactly the same as the result from `modularity` in the `igraph` package.

Value

`modularity` returns a list containing the following:

- `totQ`: The total modularity. This is the only output if `decomp=FALSE`
- `modQ`: The contribution of each module to the total modularity
- `nodeQ`: The contribution of each node to the total modularity

Note

Adapted from code developed by Robert J. Fletcher, Jr.

Author(s)

Jonathan Walter, <jonathan.walter@ku.edu>; Lei Zhao, <lei.zhao@cau.edu.cn>; Daniel Reuman, <reuman@ku.edu>

References


See Also

`clust, cluseigen`, `browseVignettes("wsyn")`

Examples

```r
adj<-matrix(0, 10, 10) # create a fake adjacency matrix
adj[lower.tri(adj)]<-runif(10*9/2, -1, 1)
adj<-adj+t(adj)
colnames(adj)<-letters[1:10]
m<-cluseigen(adj)
z<-modularity(adj, m[[length(m)]], decomp=TRUE)
```
**Description**

A convenience function for performing the normalization step for the coh function.

**Usage**

```
normforcoh(W, norm)
```

**Arguments**

- `W`: An array of wavelet transforms, locations by times by timescales.
- `norm`: The normalization of wavelet transforms to use. Controls the version of the coherence that is performed. One of "none", "phase", "powall", "powind". See details section of the documentation for coh.

**Value**

`normforcoh` returns an array the same dimensions as `W` of normalized transforms.

**Note**

Internal function, no error checking.

**Author(s)**

Daniel Reuman, <reuman@ku.edu>

---

**plotmag**

*For plotting the magnitude of values in tts, coh and wlmtest objects*

**Description**

For plotting the magnitude of values in tts objects (and derived classes) against time and timescale, and coh and wlmtest objects against timescale.
plotmag

Usage

plotmag(object, ...)

## S3 method for class 'tts'
plotmag(object, zlims = NULL, neat = TRUE, colorfill = NULL,
         colorbar = TRUE, title = NULL, filename = NA, ...)

## S3 method for class 'wt'
plotmag(object, zlims = NULL, neat = TRUE, colorfill = NULL,
         colorbar = TRUE, title = NULL, filename = NA, ...)

## S3 method for class 'wmf'
plotmag(object, zlims = NULL, neat = TRUE, colorfill = NULL,
         colorbar = TRUE, title = NULL, filename = NA, ...)

## S3 method for class 'wpmf'
plotmag(object, zlims = NULL, neat = TRUE,
         colorfill = NULL, sigthresh = 0.95, colorbar = TRUE, title = NULL,
         filename = NA, ...)

## S3 method for class 'coh'
plotmag(object, sigthresh = c(0.95, 0.99), bandprows = "all",
         filename = NA, ...)

## S3 method for class 'wlmtest'
plotmag(object, sigthresh = c(0.95, 0.99),
         bandprows = "all", filename = NA, ...)

## Default S3 method:
plotmag(object, ...)

Arguments

object An object of class tts or some class that inherits from tts or of class coh or
        wlmtest
...
... Additional graphics parameters passed to image (graphics package) if colorbar==FALSE,
         or to image.plot (fields package) if colorbar==TRUE (for tts objects)
zlims z axis limits. If specified, must encompass the range of Mod(get_values(object)).
         Default NULL uses this range.
neat Logical. Should timescales with no values be trimmed?
colorfill Color spectrum to use, set through colorRampPalette. Default value NULL pro-
         duces jet colors from Matlab.
colorbar Logical. Should a colorbar legend be plotted?
title Title for the top of the plot.
filename Filename (without extension), for saving as pdf. Default value NA saves no file
         and uses the default graphics device.
sigthresh  Significance threshold(s). Numeric vector with values between 0 and 1. Typically 0.95, 0.99, 0.999, etc. For wpmf objects, contours are plotted at these values; for coh and wlmtest objects the thresholds are plotted on coherence plots.

bandprows  The rows of object$bandp for which to display results in coh plots

Details

For coh (respectively, wlmtest) objects, the modulus of object$coher (respectively, object$wlmobj$coher) is plotted using a solid red line, and the modulus of object$signif$coher is plotted using a dashed red line. The two coherences agree except for sigmethod="fast", for which they are close. The dashed line is what should be compared to the distribution of surrogate coherences (black lines, which only appear for coh objects if signif is not NA). Horizontal axis ticks are labeled as timescales, but are spaced on the axis as log(1/timescale), i.e., log frequencies.

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References

Sheppard, L.W., et al. (2016) Changes in large-scale climate alter spatial synchrony of aphid pests. Nature Climate Change. DOI: 10.1038/nclimate2881

Sheppard, LW et al. (2019) Synchrony is more than its top-down and climatic parts: interacting Moran effects on phytoplankton in British seas. Plos Computational Biology 15, e1006744. doi: 10.1371/journal.pcbi.1006744

See Also

tts, wt, wmf, wpmf, coh, wlmtest, plotphase, bandtest, plotrank, browseVignettes("wsyn")

Examples

# For a wt object
time1<-1:100
time2<-101:200
tsp1<-sin(2*pi*time1/15)
tsp2<-0*time1
tsp1<-0*time2
tsp2<-sin(2*pi*time2/8)
tsl<-tsp1+tsp2
tsl<-tsp2+tsp1
tsl<-c(tsl,ts2)
ra<-rnorm(200,mean=0,sd=0.5)
t.series<-ts+ra
t.series<-t.series-mean(t.series)
times<-c(time1,time2)
res<--wt(t.series, times)
plotmag(res)
#For a wmf object
x1<-0:50
x2<-51:100
x<-c(x1,x2)
ts1<-c(sin(2*pi*x1/10),sin(2*pi*x2/5))+1.1
dat<-matrix(NA,11,length(x))
for (counter in 1:dim(dat)[1])
{
  ts2<-3*sin(2*pi*x/3+2*pi*runif(1))+3.1
  ts3<-rnorm(length(x),0,1.5)
  dat[counter,]<-ts1+ts2+ts3
  dat[counter,]<-dat[counter,]-mean(dat[counter,])
}
times<-x
res<-wmf(dat,times)
plotmag(res)

#similar calls for wpmf, coh, wlm, wlmtest objects
#see documentation

---

**plotmap**

*Map clusters from a clust object*

**Description**

Produces a map of the locations of sampling for a clust object, with colors indicating module (cluster) identity. The sizes of nodes (locations) are scaled according to the strength of membership in its module.

**Usage**

```r
plotmap(inclust, spltlvl = length(inclust$clusters), nodesize = c(1, 3),
filename = NA)
```

**Arguments**

- **inclust**: A clust object, as created with wsyn::clust
- **spltlvl**: The split level in the clustering to use. This is the index of inclust$clusters. Default the final split.
- **nodesize**: A length = 2 vector giving the minimum and maximum node size for plotting. Defaults to c(1,3).
- **filename**: a filename, possibly including path info, but without a file extension. If present, exports the plot as a .pdf using the specified filename. Default NA uses the default plotting device.
**Value**

plotmap produces a map.

**Author(s)**

Jonathan Walter, <jaw3es@virginia.edu>

**References**


**See Also**

`clust`, `browseVignettes("wsyn")`

**Examples**

```r
Tmax<-500
tim<-1:Tmax
ts1<-sin(2*pi*tim/5)
ts1s<-sin(2*pi*tim/5+pi/2)
ts2<-sin(2*pi*tim/12)
ts2s<-sin(2*pi*tim/12+pi/2)
gp1A<-1:2
gp1B<-3:4
gp2A<-5:6
gp2B<-7:8
d<-matrix(NA,Tmax,8)
d[,c(gp1A,gp1B)]<-ts1
d[,c(gp2A,gp2B)]<-ts1s
d[,c(gp1A,gp2A)]<-d[,c(gp1A,gp2A)]+matrix(ts2,Tmax,4)
d[,c(gp1B,gp2B)]<-d[,c(gp1B,gp2B)]+matrix(ts2s,Tmax,4)
d<-d+matrix(rnorm(Tmax*8,0,2),Tmax,8)
d<-t(d)
d<-cleandat(d,1:Tmax,1)$cdat
coords<-data.frame(X=c(rep(1,4),rep(2,4)),Y=rep(c(1:2,4:5),times=2))
cl5<-clust(dat=d,times=1:Tmax,coords=coords,method="ReXWT",tsrange=c(4,6))
plotmap(cl5)
cl12<-clust(dat=d,times=1:Tmax,coords=coords,method="ReXWT",tsrange=c(11,13))
plotmap(cl12)
```

---

**plotphase**

For plotting the phases of values in `tts` and `coh` objects

**Description**

For plotting the phases of values in `tts` objects (and derived classes) against time and timescale, and `coh` objects against timescale.
Usage

plotphase(object, ...)  

## S3 method for class 'tts'
plotphase(object, filename = NA, ...)

## S3 method for class 'wt'
plotphase(object, filename = NA, ...)

## S3 method for class 'wmf'
plotphase(object, filename = NA, ...)

## S3 method for class 'wpmf'
plotphase(object, filename = NA, ...)

## S3 method for class 'coh'
plotphase(object, bandprows = "all", filename = NA, ...)

## Default S3 method:
plotphase(object, ...)

Arguments

object A coh object.

... Passed from the generic to specific methods. The plotphase.tss method passes it to fields::image.plot.

filename Filename (without extension), for saving as pdf. Default value NA saves no file and uses the default graphics device.

bandprows The rows of object$bandp for which to display p-value results in the plot

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References

Sheppard, L.W., et al. (2016) Changes in large-scale climate alter spatial synchrony of aphid pests. Nature Climate Change. DOI: 10.1038/nclimate2881

See Also

tts, wt, wmf, wpmf, coh, plotmag, plotrank, browseVignettes("wsyn")

Examples

# For a tts object
times<-1:100
timescales<-1:100
cplx<-complex(modulus=1,argument=seq(from=-pi,to=pi,length.out=100))
values1<-matrix(cplx,length(times),length(timescales))
tts1<-tts(times,timescales,values1)
plotphase(tts1)

#For a coh object
times<-(-3:100)
ts1<-sin(2*pi*times/10)
ts2<-5*sin(2*pi*times/3)
artsig_x<-matrix(NA,11,length(times)) #the driver
for (counter in 1:11)
{
  artsig_x[counter,]=ts1+ts2+rnorm(length(times),mean=0, sd=1.5)
}
times<-0:100
artsig_y<-matrix(NA,11,length(times)) #the driven
for (counter1 in 1:11)
{
  for (counter2 in 1:101)
  {
    artsig_y[counter1,counter2]<-mean(artsig_x[counter1,counter2:(counter2+2)])
  }
}
artsig_y<-artsig_y+matrix(rnorm(length(times)*11,mean=0, sd=3),11,length(times))
artsig_x<-artsig_x[4:104]
artsig_x<-cleandat(artsig_x,times,1)$cdat
artsig_y<-cleandat(artsig_y,times,1)$cdat
res<-coh(dat1=artsig_x,dat2=artsig_y,times=times,norm="powall", sigmethod="fast", nrand=50, f0=0.5, scale.max.input=28)
res<-bandtest(res,c(2,4))
res<-bandtest(res,c(4,30))
res<-bandtest(res,c(8,12))
plotphase(res)

---

plotrank  

Plots ranks slot for coh and wlmtest objects

Description

Plots the ranks slot for coh and wlmtest objects to help identify statistical significance of coherence

Usage

plotrank(object, ...)

## S3 method for class 'coh'
plotrank(object, sigthresh = 0.95, bandprows = "all", filename = NA, ...)
## S3 method for class 'wlmtest'
plotrank(object, sigthresh = 0.95, bandprows = "all",
         filename = NA, ...)

## Default S3 method:
plotrank(object, ...)

**Arguments**

- **object**: A coh or wlmtest object. Must have a non-NA signif slot.
- **...**: Passed from the generic to specific methods. Not currently used.
- **sigthresh**: Significance threshold(s). Numeric vector with values between 0 and 1. Typically 0.95, 0.99, 0.999, etc. The threshold(s) are plotted on the rank plot as dashed horizontal line(s).
- **bandprows**: The rows of object$bandp for which to display p-value results in the plot.
- **filename**: Filename (without extension), for saving as pdf. Default value NA saves no file and uses the default graphics device.

**Details**

The plot shows the modulus of object$ranks$coher versus log(1/object$timescales). Horizontal axis ticks are labeled as timescales, but are spaced on the axis as log(1/timescale), i.e., log frequencies. p-values from object$bandp are displayed above the rank plot.

**Author(s)**

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

**References**

Sheppard, L.W., et al. (2016) Changes in large-scale climate alter spatial synchrony of aphid pests. Nature Climate Change. DOI: 10.1038/nclimate2881

Sheppard, LW et al. (2019) Synchrony is more than its top-down and climatic parts: interacting Moran effects on phytoplankton in British seas. Plos Computational Biology 15, e1006744. doi: 10.1371/journal.pcbi.1006744

**See Also**

coh, wlmtest, bandtest, plotphase, plotmag, browseVignettes("wsyn")

**Examples**

```r
# For a coh object
times<-(-3:100)
ts1<-sin(2*pi*times/10)
ts2<-5*sin(2*pi*times/3)
artsig_x<-matrix(NA,11,length(times)) # the driver
```
for (counter in 1:11) {
    artsig_x[counter,]=ts1+ts2+rnorm(length(times),mean=0,sd=1.5)
}
times<-0:100
artsig_y<-matrix(NA,11,length(times)) # the driven
for (counter1 in 1:11) {
    for (counter2 in 1:101) {
        artsig_y[counter1,counter2]<-mean(artsig_x[counter1,counter2:(counter2+2)])
    }
}
artsig_y<-artsig_y+matrix(rnorm(length(times)*11,mean=0,sd=3),11,length(times))
artsig_x<-cleandat(artsig_x,times,1)$cdat
artsig_y<-cleandat(artsig_y,times,1)$cdat
res<-coh(dat1=artsig_x,dat2=artsig_y,times=times,norm="powall",sigmethod="fast",
         nrand=100,f0=0.5,scale.max.input=28)
# use larger nrand for a real application
res<-bandtest(res,c(2,4))
res<-bandtest(res,c(8,12))
plotrank(res)

# For a wlmtest object, see vignette

---

**power**

*Power of a tts object*

**Description**

Returns the power of a `tts` object, i.e., the mean over time of the squared magnitude (which is a function of timescale)

**Usage**

```r
class2power(object)
```

```r
# S3 method for class 'tts'
class2power(object)
```

**Arguments**

- `object`: A `tts` object

**Value**

`power` returns a data frame with columns `timescales` and `power`


**predsync**

**Author(s)**

Daniel Reuman, <reuman@ku.edu>

**See Also**

tts, wt, wmf, wpmf, browseVignettes("wsyn")

**Examples**

times<-1:10
timescales<-1:10
values<-matrix(rep(complex(modulus=1,argument=2*pi*c(0:9)/10),times=10),10,10)
ttsobj<-tts(times,timescales,values)
res<-power(ttsobj)

---

**predsync**

Predicted synchrony of a wavelet linear model

**Description**

Predicted synchrony of a wlm object. This is described in the first paragraph of Appendix S15 of Sheppard et al (2019).

**Usage**

predsync(wlmobj)

## S3 method for class 'wlm'
predsync(wlmobj)

**Arguments**

wlmobj A wlm object

**Value**

predsync returns a tts object. Plotting the magnitude (see plotmag) displays a picture of predicted synchrony versus time and timescale that is comparable with the wavelet mean field (see wmf) of the response variable of the model. Calling the power function on that tts object should give the same results as one of the columns of output of syncexpl. Only norm="powall" implemented so far.

**Author(s)**

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>
References

Sheppard, LW et al. (2019) Synchrony is more than its top-down and climatic parts: interacting Moran effects on phytoplankton in British seas. Plos Computational Biology 15, e1006744. doi: 10.1371/journal.pcbi.1006744

See Also

wlm, tts, plotmag, wmf, power, syncexpl, browseVignettes("wsyn")

Examples

times<-(-3:100)
ts1<-sin(2*pi*times/10)
ts2<-5*sin(2*pi*times/3)
artsig_x<-matrix(NA,11,length(times)) #the driver
for (counter in 1:11)
{
  artsig_x[counter]<-ts1+ts2+rnorm(length(times),mean=0,sd=.5)
}
times<-0:100
artsig_y<-matrix(NA,11,length(times)) #the driven
for (counter1 in 1:11)
{
  for (counter2 in 1:101)
  {
    artsig_y[counter1,counter2]<-mean(artsig_x[counter1,counter2:(counter2+2)])
  }
}artsig_y<-artsig_y+matrix(rnorm(length(times)*11,mean=0, sd=1),11,length(times))
artsig_x<-cleandat(artsig_x,times,1)$cdat
artsig_y<-cleandat(artsig_y,times,1)$cdat
artsig_i<-cleandat(artsig_i,times,1)$cdat
dat<-list(driven=artsig_y,driver=artsig_x,irrelevant=artsig_i)
resp<1
pred<2:3
norm<-“powall”
wmobj<-wlm(dat,times,resp,pred,norm)

res<-predsync(wmobj)
setmints

Usage

## S3 method for class 'summary_wsyn'
print(x, ...)

Arguments

x           A summary_wsyn object
...

Value

print.summary_wsyn is called for its effect of printing to the screen.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also

tts_methods, wt_methods, wmf_methods, wpmf_methods, coh_methods, wlm_methods, wlmtest_methods, clust_methods, browseVignettes("wsyn")

Examples

times<-1:10
timescales<-1/c(1:10)
values<-matrix(1,length(times),length(timescales))
h<-tts(times,timescales,values)
print(summary(h))

setmints

Shifts a vector according to the argument mints

Description

Shifts a vector according to the argument mints

Usage

setmints(ts, mints)

Arguments

ts           A vector of numeric values representing a time series
mints        The time series is shifted to have this minimum value. Default NA means use the smallest difference between consecutive, distinct sorted values of the time series. NaN means perform no shift.
Value

\texttt{setmints} returns the shifted vector.

Daniel Reuman, \texttt{<reuman@ku.edu>}

Note

This is an internal function, and no error checking is done.

\begin{itemize}
\item \texttt{set_adj}
\end{itemize}

\textit{Set and get methods for classes in the \texttt{wsyn} package}

\textbf{Description}

Set and get methods for classes in the \texttt{wsyn} package. There are methods for each slot of each class, named \texttt{set_*} and \texttt{get_*} for * the slot name. Below are listed function specs for the generics and the default methods.

\textbf{Usage}

\begin{verbatim}
set_adj(obj, newval)
## Default S3 method:
set_adj(obj, newval)

dset_clusters(obj, newval)
## Default S3 method:
dset_clusters(obj, newval)

dset_modres(obj, newval)
## Default S3 method:
dset_modres(obj, newval)

dset_mns(obj, newval)
## Default S3 method:
dset_mns(obj, newval)

dset_coords(obj, newval)
## Default S3 method:
dset_coords(obj, newval)

dset_methodspecs(obj, newval)
\end{verbatim}
set_adj

## Default S3 method:
set_methodspecs(obj, newval)

set_wmfs(obj, newval)

## Default S3 method:
set_wmfs(obj, newval)

set_wpmfs(obj, newval)

## Default S3 method:
set_wpmfs(obj, newval)

get_adj(obj)

## Default S3 method:
get_adj(obj)

get_clusters(obj)

## Default S3 method:
get_clusters(obj)

get_modres(obj)

## Default S3 method:
get_modres(obj)

get_mns(obj)

## Default S3 method:
get_mns(obj)

get_coords(obj)

## Default S3 method:
get_coords(obj)

get_methodspec(obj)

## Default S3 method:
get_methodspec(obj)

get_wmfs(obj)

## Default S3 method:
get_wmfs(obj)
get_wpmfs(obj)

## Default S3 method:
get_wpmfs(obj)

set_coher(obj, newval)

## Default S3 method:
set_coher(obj, newval)

set_dat1(obj, newval)

## Default S3 method:
set_dat1(obj, newval)

set_dat2(obj, newval)

## Default S3 method:
set_dat2(obj, newval)

set_norm(obj, newval)

## Default S3 method:
set_norm(obj, newval)

set_sigmethod(obj, newval)

## Default S3 method:
set_sigmethod(obj, newval)

set_ranks(obj, newval)

## Default S3 method:
set_ranks(obj, newval)

set_bandp(obj, newval)

## Default S3 method:
set_bandp(obj, newval)

get_coher(obj)

## Default S3 method:
get_coher(obj)

get_dat1(obj)

## Default S3 method:
get_dat1(obj)
get_dat2(obj)

## Default S3 method:
get_dat2(obj)
get_norm(obj)

## Default S3 method:
get_norm(obj)
get_sigmethod(obj)

## Default S3 method:
get_sigmethod(obj)
get_ranks(obj)

## Default S3 method:
get_ranks(obj)
get_bandp(obj)

## Default S3 method:
get_bandp(obj)
set_times(obj, newval)

## Default S3 method:
set_times(obj, newval)
set_timescales(obj, newval)

## Default S3 method:
set_timescales(obj, newval)
set_values(obj, newval)

## Default S3 method:
set_values(obj, newval)
get_times(obj)

## Default S3 method:
get_times(obj)
get_timescales(obj)
## Default S3 method:
get_timescales(obj)

get_values(obj)

## Default S3 method:
get_values(obj)

set_coefs(obj, newval)

## Default S3 method:
set_coefs(obj, newval)

set_modval(obj, newval)

## Default S3 method:
set_modval(obj, newval)

set_wts(obj, newval)

## Default S3 method:
set_wts(obj, newval)

get_coefs(obj)

## Default S3 method:
get_coefs(obj)

get_modval(obj)

## Default S3 method:
get_modval(obj)

get_wts(obj)

## Default S3 method:
get_wts(obj)

set_wlmobj(obj, newval)

## Default S3 method:
set_wlmobj(obj, newval)

set_drop(obj, newval)

## Default S3 method:
set_drop(obj, newval)
get_wlmbj(obj)

## Default S3 method:
get_wlmbj(obj)

get_drop(obj)

## Default S3 method:
get_drop(obj)

set_signif(obj, newval)

## Default S3 method:
set_signif(obj, newval)

get_signif(obj)

## Default S3 method:
get_signif(obj)

set_dat(obj, newval)

## Default S3 method:
set_dat(obj, newval)

set_wtopt(obj, newval)

## Default S3 method:
set_wtopt(obj, newval)

get_dat(obj)

## Default S3 method:
get_dat(obj)

get_wtopt(obj)

## Default S3 method:
get_wtopt(obj)

### Arguments

**obj**  
An object of one of the classes defined in the package

**newval**  
A new value of the slot in question, for the `set_*` methods
Details
There are methods for the tts, wt, wmf, wpmf, coh, wlm, wlmtest, and clust classes. See documentation for the generator functions for these classes (which in all cases have the same name as the class) for lists of slots for each class.

Value

set_* methods throw an error - setting of individual slots is not allowed, as it breaks consistency with the other slots. get_* just returns the value in question.

Author(s)
Daniel Reuman, <reuman@ku.edu>

Examples

times<-1:10
timescales<-1/c(1:10)
values<-matrix(1,length(times),length(timescales))
h<-tts(times,timescales,values)
get_times(h)

---

surrog

Creates surrogate time series, either Fourier surrogates or amplitude adjusted Fourier surrogates

Description
For significance testing wavelet coherence and other purposes

Usage

surrog(dat, nsurrogs, surrtype, syncpres)

Arguments

dat A locations x time matrix of observations (for multiple-time series input), or a single vector
nsurrogs The number of surrogates to produce
surrtype Either "fft" (for Fourier surrogates) or "aafft" (for amplitude adjusted Fourier surrogates). Fourier surrogates are appropriate for time series with normal marginals; otherwise consider aafft surrogates.
syncpres Logical. TRUE for "synchrony preserving" surrogates (same phase randomizations used for all time series). FALSE leads to independent phase randomizations for all time series.
Details

Fourier surrogates are somewhat faster than aaf\textsuperscript{t} surrogates, and may be much faster when some of the time series in the data have ties. Prenormalization (e.g., using cleandat) can make it possible to use fft surrogates.

Value

surrog returns a list of nsurrogs surrogate datasets

Author(s)

Jonathan Walter, \texttt{<jaw3es@virginia.edu>}; Lawrence Sheppard, \texttt{<lwsheppard@ku.edu>}; Daniel Reuman, \texttt{<reuman@ku.edu>}

References


See Also

\texttt{wpmf, coh, wlmtest, synmat}, \texttt{browseVignettes("wsyn")}

Examples

\begin{verbatim}
times<-1:100
dat<-sin(2*pi*times/10)
nsurrogs<-10
surrtype="fft"
syncpres<-TRUE
res<-surrog(dat,nsurrogs,surrtype,syncpres)
\end{verbatim}
Usage

syncexpl(object)

## S3 method for class 'wlm'
syncexpl(object)

Arguments

object A wlm object

Details

This function only works for norm="powall" at present. See Sheppard et al (2018) for details of the meaning and computation of the columns.

Value

syncexpl returns a data frame with columns for timescales, sync (the time-averaged square magnitude of the wavelet mean field of the response transforms), syncexpl (synchrony explained by the model predictors), columns named for each predictor (synchrony explained by that predictor), interactions (synchrony explained by all interaction effects), columns named for each pair of predictors (synchrony explained by individual pairwise interactions). There are also columns for crossterms and resid (residuals). The cross terms must be small for a given timescale band for the other results to be meaningful. All columns are functions of timescales.

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References

Sheppard, LW et al. (2019) Synchrony is more than its top-down and climatic parts: interacting Moran effects on phytoplankton in British seas. Plos Computational Biology 15, e1006744. doi: 10.1371/journal.pcbi.1006744

See Also

wlm, predsync, wlmtest, browseVignettes("wsyn")

Examples

times<-(-3:100)
ts1<-sin(2*pi*times/10)
ts2<-5*sin(2*pi*times/3)
artsig_x<-matrix(NA,11,length(times)) #the driver
for (counter in 1:11)
{
  artsig_x[counter,]=ts1+ts2+rnorm(length(times),mean=0,sd=1.5)
}
times<-0:100  
artsig_y<-matrix(NA,11,length(times)) #the driven  
for (counter1 in 1:11)  
{  
    for (counter2 in 1:101)  
    {  
        artsig_y[counter1,counter2]<-mean(artsig_x[counter1,counter2:(counter2+2)])  
    }  
}  
artsig_y<-artsig_y+matrix(rnorm(length(times)*11,mean=0,sd=3),11,length(times))  
artsig_x<-artsig_x[,4:104]  
artsig_i<-matrix(rnorm(11*length(times)),11,length(times)) #the irrelevant  
artsig_x<-cleandat(artsig_x,times,1)$cdat  
artsig_y<-cleandat(artsig_y,times,1)$cdat  
artsig_i<-cleandat(artsig_i,times,1)$cdat  

dat<-list(driven=artsig_y,driver=artsig_x,irrelevant=artsig_i)  
resp<-1  
pred<-2:3  
norm<"powall"  
wlmobj<-wlm(dat,times,resp,pred,norm)  
res<-syncexpl(wlmobj)

---

**synmat**

_Synchrony matrices_

**Description**

Calculate synchrony matrices using a variety of methods

**Usage**

`synmat(dat, times, method, tsrange = c(0, Inf), nsurrogs = 1000, scale.min = 2, scale.max.input = NULL, sigma = 1.05, f0 = 1, weighted = TRUE, sigthresh = 0.95)`

**Arguments**

- **dat**  
  A locations (rows) x time (columns) matrix of measurements
- **times**  
  The times at which measurements were made, spacing 1
- **method**  
  Method for synchrony calculation. See details.
- **tsrange**  
  A vector containing the min and max of the focal timescale range. Defaults to all timescales that are valid given choices for scale.min, scale.max.input, f0, sigma. Only used for wavelet-based methods.
- **nsurrogs**  
  Number of surrogates for significance test. Defaults to 1000. Only used for surrogate-based methods.
scale.min  
The smallest scale of fluctuation that will be examined. At least 2. Used only for wavelet-based methods.

scale.max.input  
The largest scale of fluctuation guaranteed to be examined. Only used for wavelet-based methods.

sigma  
The ratio of each time scale examined relative to the next timescale. Should be greater than 1. Only used for wavelet-based methods.

f0  
The ratio of the period of fluctuation to the width of the envelope. Only used for wavelet-based methods.

weighted  
If TRUE, create a weighted network. If FALSE, create a binary network using statistical significance. Binary networks are only allowed for networks based on significance.

sigthresh  
Significance threshold needed, if weighted is false, for a network link to be realized. Typically 0.95, 0.99, or 0.999, etc. Only used if weighted is FALSE.

Details

The following values are valid for method: "pearson", "pearson.sig.std", "pearson.sig.fft", "pearson.sig.a aft", "spearman", "spearman.sig.std", "spearman.sig.fft", "spearman.sig.a aft", "kendall", "kendall.sig.std", "kendall.sig.fft", "kendall.sig.a aft", "ReXWT", "ReXWT.sig.fft", "ReXWT.sig.a aft", "ReXWT.sig.fast", "coh", "coh.sig.fft", "coh.sig.a aft", "coh.sig.fast", "phasecoh", "phasecoh.sig.fft", and "phasecoh.sig.a aft". The first portions of these identifiers correspond to the Pearson, Spearman, and Kendall correlations, the real part of the cross-wavelet transform, the wavelet coherence, and the wavelet phase coherence. The second portions of these identifiers, when present, indicates that significance of the measure specified in the first portion of the identifiers is to be used for establishing the synchrony matrix. Otherwise the value itself is used. The third part of the method identifier indicates what type of significance is used. Significance testing is performed using standard approaches (method flag containing std; for correlation coefficients, although these are inappropriate for autocorrelated data), or surrogates generated using the Fourier (method flag containing "fft") or amplitude adjusted Fourier surrogates ("a aft"). For "coh" and "ReXWT", the fast testing algorithm of Sheppard et al. (2017) is also implemented ("fast"). That method uses implicit Fourier surrogates. The choice of wavelet coherence (method flag containing "coh") or the real part of the cross-wavelet transform (method flag containing "ReXWT") depends mainly on treatment of out-of-phase relationships. The "ReXWT" is more akin to a correlation coefficient in that strong in-phase relationships approach 1 and strong antiphase relationships approach -1. Wavelet coherence allows any phase relationship and ranges from 0 to 1. Power normalization is applied for "coh" and for "ReXWT". All significance tests are one-tailed. Synchrony matrices for significance-based methods when weighted is TRUE contain 1 minus the p-values.

Value

synmat returns a synchrony matrix, of type depending on the method argument. See details. Diagonal entries are left as NA.

Author(s)

Jonathan Walter, <jaw3es@virginia.edu>; Daniel Reuman, <reuman@ku.edu>; Lei Zhao, <lei.zhao@cau.edu.cn>
tts

References


See Also

clust, coh, surrog, browseVignettes("wsyn")

Examples

sig<-matrix(.9,5,5)
diag(sig)<-1
dat1<-t(mvtnorm::rmvnorm(30,mean=rep(0,5),sigma=sig))
dat2<-t(mvtnorm::rmvnorm(30,mean=rep(0,5),sigma=sig))
dat<-rbind(dat1,dat2)
times<-1:30
dat<-cleandat(dat,times,clev=2)$cdat
method<="pearson.sig.fft"
res<-synmat(dat,times,method,nsurrogs=100,weighted=FALSE,
sigthresh=0.95)

---

tts

Creator function for the tts class

Description

The tts (time/ timescale) class is for matrices for which the rows correspond to times and the columns correspond to timescales. This is a general class from which other classes inherit (e.g., wt, wmf, wpmf). tts inherits from the list class.

Usage

tts(times, timescales, values)

Arguments

times A numeric vector of increasing real values, spacing 1
timescales A numeric vector with positive entries
values A complex or numeric matrix of dimensions length(times) by length(timescales)

Value

tts returns an object of class tts. Slots are:
times a numeric vector of evenly spaced times
timescales a numeric vector of positive timescales
values a complex or numeric matrix of dimensions length(times) by length(timescales)
Author(s)
Daniel Reuman, <reuman@ku.edu>

See Also
tts_methods, wt, wmf, wpmf, browseVignettes("wsyn")

Examples
```r
times<-1:10
timescales<-1/c(1:10)
values<-matrix(1,length(times),length(timescales))
h<-tts(times,timescales,values)
```

---

**tts_methods**

*Basic methods for the tts class*

Description
Set, get, summary, and print methods for the tts class.

Usage
```r
## S3 method for class 'tts'
summary(object, ...)

## S3 method for class 'tts'
print(x, ...)

## S3 method for class 'tts'
set_times(obj, newval)

## S3 method for class 'tts'
set_timescales(obj, newval)

## S3 method for class 'tts'
set_values(obj, newval)

## S3 method for class 'tts'
get_times(obj)

## S3 method for class 'tts'
get_timescales(obj)

## S3 method for class 'tts'
get_values(obj)
```
warray

Arguments

object, x, obj  An object of class tts
...  Not currently used. Included for argument consistency with existing generics.
newval  A new value, for the set_* methods

Value

summary.tts produces a summary of a tts object. A print.tts method is also available. For tts objects, set_* and get_* methods are available for all slots, i.e., * equal to times, timescales, and values. The set_* methods just throw an error. Although class tts is flexible enough that setting of individual slots could have been allowed, because wt and other classes are based on it and because individual slots of those classes should not be changed, for consistency the same is forced for the tts class.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also
tts

Examples

times<-1:10
timescales<-1/c(1:10)
values<-matrix(1,length(times),length(timescales))
h<-tts(times,timescales,values)
get_times(h)
summary(h)
print(h)

warray  Creates an array of wavelet transforms from input timeseries

Description

Creates an array of wavelet transforms from input timeseries

Usage

warray(dat, times, scale.min = 2, scale.max.input = NULL, sigma = 1.05, f0 = 1)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A locations (rows) x time (columns) matrix</td>
</tr>
<tr>
<td>times</td>
<td>A vector of timestep values (e.g. years), spacing 1</td>
</tr>
<tr>
<td>scale.min</td>
<td>The smallest scale of fluctuation that will be examined. At least 2.</td>
</tr>
<tr>
<td>scale.max.input</td>
<td>The largest scale of fluctuation that will be examined. Note that if this is set too high relative to the length of the timeseries it will be truncated.</td>
</tr>
<tr>
<td>sigma</td>
<td>The ratio of each time scale examined relative to the next timescale. Greater than 1.</td>
</tr>
<tr>
<td>f0</td>
<td>The ratio of the period of fluctuation to the width of the envelope</td>
</tr>
</tbody>
</table>

Value

warray returns a list containing:

- wavarray: locations x time x timescales array of wavelet transforms
- times: the time steps specified (e.g., years)
- timescales: the timescales (1/frequency) computed for the wavelet transforms

Note

Important for interpreting the phase: the phases grow through time, i.e., they turn anti-clockwise.
This function is internal, no error checking.

Author(s)

Lauren Hallett, <hallett@uoregon.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

Description

Facilitates the computations in synmat for coherence and ReXWT methods

Usage

wavmatwork(dat, times, scale.min, scale.max.input, sigma, f0, norm, treatment)
Arguments

dat  A locations (rows) x time (columns) matrix of measurements

times  The times at which measurements were made, spacing 1

scale.min  The smallest scale of fluctuation that will be examined. At least 2. Used only
            for wavelet-based methods.

scale.max.input  The largest scale of fluctuation guaranteed to be examined. Only used for
                 wavelet-based methods.

sigma  The ratio of each time scale examined relative to the next timescale. Should be
        greater than 1. Only used for wavelet-based methods.

f0  The ratio of the period of fluctuation to the width of the envelope. Only used for
     wavelet-based methods.

norm  The normalization of wavelet transforms to be used. One of "none", "phase",
      "powind".

treatment  Either "Mod" or "Re"

Value

wavmatwork returns a list consisting of:

  timescales  The timescales of analysis

  wavarray  An array, locations by locations by timescales, containing either the coherences
             (for treatment="Mod") or the real parts of the cross-wavelet transforms (for
             treatment="Re") between locations.

Note

Internal function, no error checking done.

Author(s)

Daniel Reuman, <reuman@ku.edu>

wlm  Wavelet linear models

Description

Fits wavelet linear models. Also the generator function of the wlm class, which inherits from the
list class.

Usage

wlm(dat, times, resp, pred, norm, scale.min = 2, scale.max.input = NULL,
    sigma = 1.05, f0 = 1)
Arguments

dat  A list of matrices representing the data (or in the case of one location, a list of vectors). All the same dimensions (respectively, lengths)
times  The times at which measurements were made, spacing 1
resp  Index in dat for the response variable of the model
pred  Vector of indices in dat for the predictor variables of the model; must differ from resp
norm  The normalization of wavelet transforms to use. One of "none", "powall", "powind". See details.
scale.min  The smallest scale of fluctuation that will be examined. At least 2.
scale.max.input  The largest scale of fluctuation that will be examined. Note that if this is set too high relative to the length of the timeseries it will be truncated.
sigma  The ratio of each time scale examined relative to the next timescale. Greater than 1.
f0  The ratio of the period of fluctuation to the width of the envelope

Details

Normalization is as specified in the documentation for coh, HOWEVER, only the "powall" option is currently implemented, other choices throw an error. Details are specified in appendices S7 and S9 of Sheppard et al, 2018. The output modval is v in appendix S7, and coefs are the betas in equation 12 in that appendix.

Value

wlm returns an object of class wlm. Slots are:

dat  The input data list, but reordered and subsetted so the response is first and only used predictors are included
times  The times associated with the data
norm  The input
wtopt  The inputted wavelet transform options scale.min, scale.max.input, sigma, f0 in a list
wts  List of transforms, normalized as specified in norm. Same length as the output dat, each entry a locations x time x timescales array of transforms.
timescales  The timescales associated with the wavelet transforms of the data
coeffs  A list (data frame, actually) of complex vectors, each of length the same as timescales. These are the model coefficients (which depend on timescale), and correspond to the wts.
modval  The model values.
coher  Appropriately normalized version of coherence of the model and response transforms. See details.
wlmfit

Fits a wavelet linear model

Description
Stripped down internal function for doing the fitting

Usage
wlmfit(wts, norm)

Arguments
wts List of normalized transforms, normalized as specified in norm. Each entry a locations x time x timescales array of transforms. The first is the response variable, others are the predictors.

norm The normalization that was used. One of "none", "powall", "powind". See details.

Details
Only norm="powall" works now, other options throw an error.
Value

`wlmfit` returns a list with these elements:

- **coefs**: Model coefficients
- **modval**: The right hand side of the model
- **coher**: Appropriately normalized coherence of the model and response variable

Note

Internal function, no error checking done.

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References

Sheppard, LW et al. (2019) Synchrony is more than its top-down and climatic parts: interacting Moran effects on phytoplankton in British seas. Plos Computational Biology 15, e1006744. doi: 10.1371/journal.pcbi.1006744

---

**wlmtest**  
*Statistical comparison of wavelet linear models*

Description

Compared a wavelet linear model with a nested model. Also the generator function for the `wlmtest` class.

Usage

```r
wlmtest(wlmobj, drop, sigmethod, nrand = 1000)
```

Arguments

- **wlmobj**: A `wlm` object
- **drop**: Either names or indices of variables in `wlmobj$dat` that are being dropped to form the simpler, nested model. The first variable in `wlmobj$dat`, which is the response, is not allowed here.
- **sigmethod**: Method for significance testing. One of "fft", "aft", "fast". See details.
- **nrand**: The number of randomizations to do for significance
Details

The slot `signif` provides the core information on significance. If `sigmethod` is not "fast", then `signif$coher` is the same as `wlmobj$coher`, and `signif$scoher` is a matrix of dimensions `nrand` by `length(signif$coher)` with rows equal to coherences between refitted models and the response-variable transforms, for datasets where the variables specified in `drop` have been replaced by surrogates. Normalization as specified in `norm` is used. The type of surrogate used (Fourier surrogates or amplitude adjusted Fourier surrogates, see `surrog`) is determined by `sigmethod` ("fft" or "aaft"). Synchrony-preserving surrogates are used. A variety of statements of significance (or lack thereof) can be made by comparing `signif$coher` with `signif$scoher` (see the `plotmag`, `plotrank`, and `bandtest` methods for the `wlmtest` class). If `sigmethod` is "fast", a fast algorithm of Lawrence Sheppard is used which is a generalization to wavelet linear models of the fast algorithm for coherence described in Sheppard et al (2017). In that case `signif$coher` can be compared to `signif$scoher` to make significance statements about the coherence in exactly the same way, but `signif$coher` will no longer precisely equal `wlmobj$coher`, and `wlmobj$coher` should not be compared directly to `signif$scoher`. Statements about significance of the coherence should be made using `signif$coher` and `signif$scoher`, whereas `wlmobj$coher` should be used whenever the actual value of the coherence is needed.

The slots `ranks` and `bandp` are empty on an initial call to `wlmtest`. They are made to compute and hold aggregate significance results over any timescale band of choice. These are filled in when needed by other methods, see `plotrank` and `bandtest`.

Value

`wlmtest` returns an object of class `wlmtest`. Slots are:

- `wlmobj` The input
- `drop` The input
- `signif` A list with information from the significance testing. Elements are `sigmethod` (the input), `coher` and `scoher`. See details.
- `ranks` A list with ranking information for `signif`. NA until `plotrank` or `bandtest` is called.
- `bandp` A data frame containing results of computing significances across timescale bands. Empty on an initial call to `wlmtest`, filled in by the function `bandtest`. See details.

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References

Sheppard, L.W., et al. (2016) Changes in large-scale climate alter spatial synchrony of aphid pests. Nature Climate Change. DOI: 10.1038/nclimate2881

Sheppard, L.W., et al. (2017) Rapid surrogate testing of wavelet coherences. European Physical Journal, Nonlinear and Biomedical Physics, 5, 1. DOI: 10.1051/epjnbp/2017000
Sheppard, LW et al. (2019) Synchrony is more than its top-down and climatic parts: interacting Moran effects on phytoplankton in British seas. Plos Computational Biology 15, e1006744. doi: 10.1371/journal.pcbi.1006744

See Also
wlm, plotrank, bandtest, coh, wlmtest_methods, browseVignettes("wsyn")

Examples
times<-1:30
dat<-list(v1=matrix(rnorm(300),10,30), v2=matrix(rnorm(300),10,30), v3=matrix(rnorm(300),10,30), v4=matrix(rnorm(300),10,30), v5=matrix(rnorm(300),10,30))
dat<-lapply(FUN=function(x){cleandat(x,times,1)$cdat},X=dat)
resp<-1
pred<-2:3
norm<="powall"
wlmobj<-wlm(dat,times,resp,pred,norm)
drop<3
sigmethod<="fft"
res<-wlmtest(wlmobj,drop,sigmethod,nrand=10)

wlmtest_methods

Basic methods for the wlmtest class

Description
Set, get, summary, and print methods for the wlmtest class.

Usage
## S3 method for class 'wlmtest'
summary(object, ...)

## S3 method for class 'wlmtest'
print(x, ...)

## S3 method for class 'wlmtest'
set_wlmobj(obj, newval)

## S3 method for class 'wlmtest'
set_drop(obj, newval)

## S3 method for class 'wlmtest'
set_signif(obj, newval)

## S3 method for class 'wlmtest'
set_ranks(obj, newval)
## S3 method for class 'wlmtest'
set_bandp(obj, newval)
## S3 method for class 'wlmtest'
get_wlmobj(obj)
## S3 method for class 'wlmtest'
get_drop(obj)
## S3 method for class 'wlmtest'
get_signif(obj)
## S3 method for class 'wlmtest'
get_ranks(obj)
## S3 method for class 'wlmtest'
get_bandp(obj)

Arguments

object, x, obj An object of class wlmtest
... Not currently used. Included for argument consistency with existing generics.
newval A new value, for the set_* methods

Value

summary.wlmtest produces a summary of a wlmtest object. A print.wlmtest method is also available. For wlmtest objects, set_* and get_* methods are available for all slots (see the documentation for wlmtest for a list). The set_* methods just throw an error, to prevent breaking the consistency between the slots of a wlmtest object.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also

wlmtest

Examples

times<-1:30
data<-list(v1=matrix(rnorm(300),10,30),v2=matrix(rnorm(300),10,30),v3=matrix(rnorm(300),10,30),
v4=matrix(rnorm(300),10,30),v5=matrix(rnorm(300),10,30))
data<-lapply(FUN=function(x){cleandat(x,times,1)$cdat},X=data)
resp<-1
pred<-2:3
norm<="powall"
wlmobj<-wlm(dat,times,resp,pred,norm)
drop<-3
sigmethod<="fft"
h<-wlmt(test(wlmobj.drop,sigmethod,nrand=10))
get_times(get_wlmobj(h))
summary(h)
print(h)

---

**wlm_methods**  
*Basic methods for the wlm class*

**Description**

Set, get, summary, and print methods for the wlm class.

**Usage**

```r
## S3 method for class 'wlm'
summary(object, ...)

## S3 method for class 'wlm'
print(x, ...)

## S3 method for class 'wlm'
set_times(obj, newval)

## S3 method for class 'wlm'
set_timescales(obj, newval)

## S3 method for class 'wlm'
set_coefs(obj, newval)

## S3 method for class 'wlm'
set_modval(obj, newval)

## S3 method for class 'wlm'
set_coher(obj, newval)

## S3 method for class 'wlm'
set_dat(obj, newval)

## S3 method for class 'wlm'
set_wtopt(obj, newval)

## S3 method for class 'wlm'
set_norm(obj, newval)
```
## S3 method for class 'wlm'
set_wts(obj, newval)

## S3 method for class 'wlm'
get_times(obj)

## S3 method for class 'wlm'
get_timescales(obj)

## S3 method for class 'wlm'
get_coefs(obj)

## S3 method for class 'wlm'
get_modval(obj)

## S3 method for class 'wlm'
get_coher(obj)

## S3 method for class 'wlm'
get_dat(obj)

## S3 method for class 'wlm'
get_wtopt(obj)

## S3 method for class 'wlm'
get_norm(obj)

## S3 method for class 'wlm'
get_wts(obj)

**Arguments**

object, x, obj  An object of class wlm

...  Not currently used. Included for argument consistency with existing generics.

newval  A new value, for the set_* methods

**Value**

`summary.wlm` produces a summary of a wlm object. A `print.wlm` method is also available. For `wlm` objects, `set_*` and `get_*` methods are available for all slots (see the documentation for `wlm` for a list). The `set_*` methods just throw an error, to prevent breaking the consistency between the slots of a `wlm` object.

**Author(s)**

Daniel Reuman, <reuman@ku.edu>
wmf

Computes the wavelet mean field from a matrix of spatiotemporal data. Also the creator function for the \texttt{wmf} class.

Description

Computes the wavelet mean field from a matrix of spatiotemporal data. Also the creator function for the \texttt{wmf} class. The \texttt{wmf} class inherits from the \texttt{tts} class, which inherits from the \texttt{list} class.

Usage

\begin{verbatim}
wmf(dat, times, scale.min = 2, scale.max.input = NULL, sigma = 1.05, f0 = 1)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{dat} \hfill A locations (rows) x time (columns) matrix
\item \texttt{times} \hfill A vector of time step values (e.g., years), spacing 1
\item \texttt{scale.min} \hfill The smallest scale of fluctuation that will be examined. At least 2.
\item \texttt{scale.max.input} \hfill The largest scale of fluctuation that will be examined. Note that if this is set too high relative to the length of the timeseries it will be truncated.
\item \texttt{sigma} \hfill The ratio of each time scale examined relative to the next timescale. Greater than 1.
\item \texttt{f0} \hfill The ratio of the period of fluctuation to the width of the envelope
\end{itemize}
Value

wmf returns an object of class wmf. Slots are:

values  A matrix of complex numbers containing the wavelet mean field, of dimensions
length(times) by the number of timescales. Entries not considered reliable
(longer timescales, near the edges of the time span) are set to NA.
times  The time steps specified (e.g., years)
timescales  The timescales (1/frequency) computed for the wavelet transforms
dat  The data matrix (locations by time) from which the wmf was computed
wtopt  The inputted wavelet transform options scale.min, scale.max.input, sigma, f0 in
a list

Author(s)

Jonathan Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel
Reuman, <reuman@ku.edu>

References

Nature Climate Change. DOI: 10.1038/nclimate2881

See Also

wmf_methods, tts, wpmf, plotmag, browseVignettes("wsyn")

Examples

times<-1:30 #generate time steps
#generate fake count data for 20 locations
dat<-matrix(rpois(20*length(times),20),nrow=20,ncol=length(times))
dat<-cleandat(dat=dat,times=times,clev=2)$cdat #detrend and demean
wmf<-wmf(dat,times)
Usage

```r
## S3 method for class 'wmf'
summary(object, ...)

## S3 method for class 'wmf'
print(x, ...)

## S3 method for class 'wmf'
set_times(obj, newval)

## S3 method for class 'wmf'
set_timescales(obj, newval)

## S3 method for class 'wmf'
set_values(obj, newval)

## S3 method for class 'wmf'
set_dat(obj, newval)

## S3 method for class 'wmf'
set_wtopt(obj, newval)

## S3 method for class 'wmf'
get_times(obj)

## S3 method for class 'wmf'
get_timescales(obj)

## S3 method for class 'wmf'
get_values(obj)

## S3 method for class 'wmf'
get_dat(obj)

## S3 method for class 'wmf'
get_wtopt(obj)
```

Arguments

- `object, x, obj` An object of class `wmf`
- `...` Not currently used. Included for argument consistency with existing generics.
- `newval` A new value, for the `set_*` methods

Value

`summary.wmf` produces a summary of a `wmf` object. A `print.wmf` method is also available. For `wmf` objects, `set_*` and `get_*` methods are available for all slots, i.e., * equal to `times, timescales,`
wtopt, values, and dat. The set_* methods just throw an error, to prevent breaking the consistency between the slots of a wmf object.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also

wmf

Examples

times<-1:30 #generate time steps
#generate fake count data for 20 locations
dat<-matrix(rpois(20*length(times),20),nrow=20,ncol=length(times))
dat<-cleandat(dat=dat,times=times,clev=2)$cdat #detrend and demean
h<-wmf(dat,times)
get_times(h)
summary(h)
print(h)

wmf  Wavelet phasor mean field

Description

Computes the wavelet phasor mean field from a matrix of spatiotemporal data. Also the creator function for the wpmf class. The wpmf class inherits from the tts class, which inherits from the list class.

Usage

wmf(dat, times, scale.min = 2, scale.max.input = NULL, sigma = 1.05, f0 = 1, sigmethod = "none", nrand = 1000)

Arguments

dat        A locations (rows) x time (columns) matrix
times      A vector of time step values, spacing 1
scale.min  The smallest scale of fluctuation that will be examined. At least 2.
scale.max.input  The largest scale of fluctuation guaranteed to be examined
sigma      The ratio of each time scale examined relative to the next timescale. Should be greater than 1.
f0         The ratio of the period of fluctuation to the width of the envelop
sigmethod  Method for significance testing the wmpf, one of quick, fft, aaft (see details)
nrand      The number of randomizations to be used for significance testing
Details

For sigmethod equal to \texttt{quick}, the empirical \texttt{wpmf} is compared to a distribution of magnitudes of sums of random phasors, using the same number of phasors as there are time series. The \texttt{signif} output is a list with first element "quick" and second element a vector of \texttt{nrand} magnitudes of sums of random phasors. For sigmethod equal to \texttt{fft}, the empirical \texttt{wpmf} is compared to \texttt{wmpfs} of Fourier surrogate datasets. The \texttt{signif} output is a list with first element "fft", second element equal to \texttt{nrand}, and third element the fraction of surrogate-based \texttt{wpmf} magnitudes that the empirical \texttt{wpmf} magnitude is greater than (times by timescales matrix). For sigmethod equal to \texttt{aaff}, \texttt{aaft} surrogates are used instead. Output has similar format to the \texttt{fft} case. Values other than \texttt{quick}, \texttt{fft}, and \texttt{aaff} for \texttt{sigmethod} result in no significance testing.

Value

\texttt{wpmf} returns an object of class \texttt{wpmf}. Slots are:

- \texttt{values}: A matrix of complex numbers containing the wavelet phasor mean field, of dimensions \(\text{length(times)}\) by the number of timescales. Entries not considered reliable (longer timescales, near the edges of the time span) are set to NA.
- \texttt{times}: The times associated with the data and the \texttt{wpmf}.
- \texttt{timescales}: The timescales associated with the \texttt{wpmf}.
- \texttt{signif}: A list with information from the significance testing. Format depends on sigmethod (see details).
- \texttt{dat}: The data matrix (locations by time) from which the \texttt{wpmf} was computed.
- \texttt{wtopt}: The inputted wavelet transform options \texttt{scale.min, scale.max.input, sigma, f0} in a list.

Author(s)

Thomas Anderson, <anderstl@gmail.com>, Jon Walter, <jaw3es@virginia.edu>; Lawrence Sheppard, <lwsheppard@ku.edu>; Daniel Reuman, <reuman@ku.edu>

References

Sheppard, L.W., et al. (2016) Changes in large-scale climate alter spatial synchrony of aphid pests. Nature Climate Change. DOI: 10.1038/nclimate2881

See Also

\texttt{wpmf_methods.wmf}, \texttt{tts}, \texttt{plotmag}, \texttt{browseVignettes("wsyn")}

Examples

\begin{verbatim}
times<-1:30 #generate time steps
generate fake count data for 20 locations
dat<-matrix(rpois(20*length(times),20),nrow=20,ncol=length(times))
dat<-cleandat(dat=dat,times=times,clev=2)$cdat #detrend and demean
res<-wpmf(dat,times)
\end{verbatim}
Description

Set, get, summary, and print methods for the wpmf class.

Usage

```r
## S3 method for class 'wpmf'
summary(object, ...)

## S3 method for class 'wpmf'
print(x, ...)

## S3 method for class 'wpmf'
set_times(obj, newval)

## S3 method for class 'wpmf'
set_timescales(obj, newval)

## S3 method for class 'wpmf'
set_values(obj, newval)

## S3 method for class 'wpmf'
set_dat(obj, newval)

## S3 method for class 'wpmf'
set_wtopt(obj, newval)

## S3 method for class 'wpmf'
set_signif(obj, newval)

## S3 method for class 'wpmf'
get_times(obj)

## S3 method for class 'wpmf'
get_timescales(obj)

## S3 method for class 'wpmf'
get_values(obj)

## S3 method for class 'wpmf'
get_dat(obj)

## S3 method for class 'wpmf'
get_wtopt(obj)
```
# S3 method for class 'wpmf'
get_signif(obj)

Arguments

- object, x, obj: An object of class wpmf
- ...: Not currently used. Included for argument consistency with existing generics.
- newval: A new value, for the set_* methods

Value

summary.wpmf produces a summary of a wpmf object. A print.wpmf method is also available. For wpmf objects, set_* and get_* methods are available for all slots, i.e., * equal to times, timescales, wtopt, values, dat, and signif. The set_* methods just throw an error, to prevent breaking the consistency between the slots of a wpmf object.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also

wpmf

Examples

times<-1:30  #generate time steps
#generate fake count data for 20 locations
dat<-matrix(rpois(20*length(times),20),nrow=20,ncol=length(times))
dat<-cleandat(dat=dat,times=times,clev=2)$cdat  #detrend and demean
h<-wpmf(dat,times)
get_times(h)
summary(h)
print(h)

wt

Computes the wavelet transform of a timeseries. Also the creator function for the wt class.

Description

Computes the wavelet transform of a timeseries. Also the creator function for the wt class. The wt class inherits from the tts class, which inherits from the list class.
wt

Usage

wt(t.series, times, scale.min = 2, scale.max.input = NULL, sigma = 1.05, f0 = 1)

Arguments

t.series A timeseries of real values
times A vector of time step values (e.g., years), spacing 1
scale.min The smallest scale of fluctuation that will be examined. At least 2.
scale.max.input The largest scale of fluctuation that is guaranteed to be examined
sigma The ratio of each time scale examined relative to the next timescale. Should be greater than 1.
f0 The ratio of the period of fluctuation to the width of the envelope. Defaults to 1.

Value

wt returns an object of class wt. Slots are:

values A matrix of complex numbers, of dimensions length(t.series) by the number of timescales. Entries not considered reliable (longer timescales, near the edges of the time span) are set to NA.
times The time steps specified (e.g. years)
wtopt The inputted wavelet transform options scale.min, scale.max.input, sigma, f0 in a list
timescales The timescales (1/frequency) computed for the wavelet transform
dat The data vector from which the transform was computed

Note

Important for interpreting the phase: the phases grow through time, i.e., they turn anti-clockwise.

Author(s)

Lawrence Sheppard <lwsheppard@ku.edu>, Jonathan Walter <jaw3es@virginia.edu>, Daniel Reuman <reuman@ku.edu>

See Also

wt_methods, tts, plotmag, plotphase, browseVignettes("wsyn")
Examples

time1<-1:100
time2<-101:200
ts1p1<-sin(2*pi*time1/15)
ts1p2<-0*time1
ts2p1<-0*time2
ts2p2<-sin(2*pi*time2/8)
ts1<-ts1p1+ts1p2
ts2<-ts2p1+ts2p2
ts<-c(ts1,ts2)
ra<-rnorm(200,mean=0,sd=0.5)
t.series<-ts+ra
t.series<-t.series-mean(t.series)
times<-c(time1,time2)
res<-wt(t.series, times)

---

wt_methods

Basic methods for the wt class

Description

Set, get, summary, and print methods for the wt class.

Usage

```r
## S3 method for class 'wt'
summary(object, ...)

## S3 method for class 'wt'
print(x, ...)

## S3 method for class 'wt'
set_times(obj, newval)

## S3 method for class 'wt'
set_timescales(obj, newval)

## S3 method for class 'wt'
set_values(obj, newval)

## S3 method for class 'wt'
set_dat(obj, newval)

## S3 method for class 'wt'
set_wtopt(obj, newval)
```

---

wt methods
get_times(obj)
## S3 method for class 'wt'
get_timescales(obj)
## S3 method for class 'wt'
get_values(obj)
## S3 method for class 'wt'
get_dat(obj)
## S3 method for class 'wt'
get_wtopt(obj)

Arguments

object, x, obj  An object of class wt
...  Not currently used. Included for argument consistency with existing generics.
newval  A new value, for the set_* methods

Value

summary.wt produces a summary of a wt object. A print.wt method is also available. For wt objects, set_* and get_* methods are available for all slots, i.e., * equal to times, timescales, wtopt, values, and dat. The set_* methods just throw an error, to prevent breaking the consistency between the slots of a wt object.

Author(s)

Daniel Reuman, <reuman@ku.edu>

See Also

wt

Examples

time1<-1:100
time2<-101:200
tsl1p<-sin(2*pi*time1/15)
tsl2p<--0*time1
ts2p1<--0*time2
tsl<-ts1p+ts1p2
tsl2<-ts2p1+ts2p2
tsl<-c(tsl,ts2)
ra<-rnorm(200,mean=0, sd=0.5)
t.series<-ts+ra
t.series<-t.series-mean(t.series)
times<-c(time1,time2)
h<-wt(t.series, times)
get_times(h)
summary(h)
print(h)
Index

addranks, 3
addwmfs, 3, 5, 13
addwpmfs, 4, 4, 13
bandtest, 3, 6, 18, 32, 37, 62
bctrans, 7
cleandat, 8, 8, 18
cludeigen, 10, 13, 27, 29
clust, 4, 5, 9, 10, 11, 15, 27, 29, 34, 53
clust_methods, 13, 14, 41
coh, 3, 7, 9, 16, 21, 32, 35, 37, 49, 53, 62
coh_methods, 18, 19, 41
ercheck_stdat, 21
ercheck_times, 22
ercheck_tsdat, 22
ercheck_wavparam, 23
fastcohtest, 24
fftsurrog, 25
get_adj (set_adj), 42
get_adj.clust (clust_methods), 14
get_bandp (set_adj), 42
get_bandp.coh (coh_methods), 19
get_bandp.wlmtest (wlmtest_methods), 62
get_clusters (set_adj), 42
get_clusters.clust (clust_methods), 14
get_coefs (set_adj), 42
get_coefs.wlm (wlmt_methods), 64
get_coher (set_adj), 42
get_coher.coh (coh_methods), 19
get_coher.wlm (wlmt_methods), 64
get_coords (set_adj), 42
get_coords.clust (clust_methods), 14
get_dat (set_adj), 42
get_dat.clust (clust_methods), 14
get_dat.wlm (wlmt_methods), 64
get_dat.wmf (wmf_methods), 67
get_dat.wpmf (wpmf_methods), 71
get_dat.wt (wt_methods), 74
get_dat1 (set_adj), 42
get_dat1.coh (coh_methods), 19
get_dat2 (set_adj), 42
get_dat2.coh (coh_methods), 19
get_drop (set_adj), 42
get_drop.wlmtest (wlmtest_methods), 62
get_methods spec (set_adj), 42
get_methods spec.clust (clust_methods), 14
get_mns (set_adj), 42
get_mns.clust (clust_methods), 14
get_modres (set_adj), 42
get_modres.clust (clust_methods), 14
get_modval (set_adj), 42
get_modres.wlmtest (wlmtest_methods), 62
get_norm (set_adj), 42
get_norm.coh (coh_methods), 19
get_norm.wlm (wlmt_methods), 64
get_ranks (set_adj), 42
get_ranks.coh (coh_methods), 19
get_ranks.wlmtest (wlmtest_methods), 62
get_sigmeth (set_adj), 42
get_sigmeth.coh (coh_methods), 19
get_signif (set_adj), 42
get_signif.coh (coh_methods), 19
get_signif.wlmtest (wlmtest_methods), 62
get_signif.wpmf (wpmf_methods), 71
get_times (set_adj), 42
get_times.clust (clust_methods), 14
get_times.coh (coh_methods), 19
get_times.tts (tts_methods), 54
get_times.wlm (wlmt_methods), 64
get_times.wmf (wmf_methods), 67
get_times.wpmf (wpmf_methods), 71
get_times.wt (wt_methods), 74
get_timescales (set_adj), 42
get_timescales.coh (coh_methods), 19
get_timescales.tts (tts_methods), 54
get_timescales.wlm (wlm_methods), 64
get_timescales.wmf (wmf_methods), 67
get_timescales.wpmf (wpmf_methods), 71
get_timescales.wt (wt_methods), 74
get_values (set_adj), 42
get_values.tts (tts_methods), 54
get_values.wmpf (wmpf_methods), 67
get_values.wpmf (wpmf_methods), 71
get_values.wt (wt_methods), 74
get_wlmobj (set_adj), 42
get_wlmobj.wlmtest (wlmtest_methods), 62
get_wmfs (set_adj), 42
get_wmfs.clust (clust_methods), 14
get_wpmfs (set_adj), 42
get_wpmfs.clust (clust_methods), 14
get_wtopt (set_adj), 42
get_wtopt.coh (coh_methods), 19
get_wtopt.wlm (wlm_methods), 64
get_wtopt.wmf (wmf_methods), 67
get_wtopt.wpmf (wpmf_methods), 71
get_wtopt.wt (wt_methods), 74
get_wts (set_adj), 42
get_wts.wlm (wlm_methods), 64

is.connected, 26
makeunweighted, 27
mnphase, 28
modularity, 10, 13, 28
normforcoh, 30
plotmag, 18, 30, 35, 37, 40, 67, 70, 73
plotmap, 13, 33
plotphase, 18, 32, 34, 37, 73
plotrank, 18, 42, 35, 36, 62
power, 38, 40
predsync, 39, 50, 59
print.clust (clust_methods), 14
print.coh (coh_methods), 19
print.summary_wsyn, 40
print.tts (tts_methods), 54
print.wlm (wlm_methods), 64
print.wlmtest (wlmtest_methods), 62
print.wmf (wmf_methods), 67
print.wpmf (wpmf_methods), 71
print.wt (wt_methods), 74
set_adj, 42
set_adj.clust (clust_methods), 14
set_bandp (set_adj), 42
set_bandp.coh (coh_methods), 19
set_bandp.wlmtest (wlmtest_methods), 62
set_clusters (set_adj), 42
set_clusters.clust (clust_methods), 14
set_coefs (set_adj), 42
set_coefs.wlm (wlm_methods), 64
set_coher (set_adj), 42
set_coher.coh (coh_methods), 19
set_coher.wlm (wlm_methods), 64
set_coords (set_adj), 42
set_coords.clust (clust_methods), 14
set_dat (set_adj), 42
set_dat.clust (clust_methods), 14
set_dat.wlm (wlm_methods), 64
set_dat.wmf (wmf_methods), 67
set_dat.wpmf (wpmf_methods), 71
set_dat.wt (wt_methods), 74
set_dat1 (set_adj), 42
set_dat1.coh (coh_methods), 19
set_dat2 (set_adj), 42
set_dat2.coh (coh_methods), 19
set_drop (set_adj), 42
set_drop.wlmtest (wlmtest_methods), 62
set_methodspecs (set_adj), 42
set_methodspecs.clust (clust_methods), 14
set_mns (set_adj), 42
set_mns.clust (clust_methods), 14
set_modres (set_adj), 42
set_modres.clust (clust_methods), 14
set_modval (set_adj), 42
set_modval.wlm (wlm_methods), 64
set_norm (set_adj), 42
set_norm.coh (coh_methods), 19
set_norm.wlm (wlm_methods), 64
set_ranks (set_adj), 42
set_ranks.coh (coh_methods), 19
set_ranks.wlmtest (wlmtest_methods), 62
set_signif (set_adj), 42
set_signif.coh (coh_methods), 19
set_signif.wlmtest (wlmtest_methods), 62
set_signif.wpmf (wpmf_methods), 71
set_times (set_adj), 42
set_times.clust (clust_methods), 14
INDEX

set_times.coh (coh_methods), 19
set_times.tts (tts_methods), 54
set_times.wlm (wlm_methods), 64
set_times.wmf (wmf_methods), 67
set_times.wpmf (wpmf_methods), 71
set_times.wt (wt_methods), 74
set_timescales (set_adj), 42
set_timescales.coh (coh_methods), 19
set_timescales.tts (tts_methods), 54
set_timescales.wlm (wlm_methods), 64
set_timescales.wmf (wmf_methods), 67
set_timescales.wpmf (wpmf_methods), 71
set_timescales.wt (wt_methods), 74
set_values (set_adj), 42
set_values.tts (tts_methods), 54
set_values.wmf (wmf_methods), 67
set_values.wpmf (wpmf_methods), 71
set_wlmobj (set_adj), 42
set_wlmobj.wlmtest (wlmtest_methods), 62
set_wmfs (set_adj), 42
set_wmfs.clust (clust_methods), 14
set_wpms (set_adj), 42
set_wpms.clust (clust_methods), 14
set_wtopt (set_adj), 42
set_wtopt.coh (coh_methods), 19
set_wtopt.wlm (wlm_methods), 64
set_wtopt.wmf (wmf_methods), 67
set_wtopt.wpmf (wpmf_methods), 71
set_wtopt.wt (wt_methods), 74
set_wts (set_adj), 42
set_wts.wlm (wlm_methods), 64
setget_methods (set_adj), 42
setmints, 41
summary.clust (clust_methods), 14
summary.coh (coh_methods), 19
summary.tts (tts_methods), 54
summary.wlm (wlm_methods), 64
summary.wlmtest (wlmtest_methods), 62
summary.wmf (wmf_methods), 67
summary.wpmf (wpmf_methods), 71
summary.wt (wt_methods), 74
surrog, 48, 53
synexpl, 40, 49, 59
synmat, 13, 49, 51

tts, 32, 35, 39, 40, 53, 55, 67, 70, 73
tts_methods, 41, 54, 54

warray, 55
wavmatwork, 56
wlm, 7, 9, 40, 50, 57, 62, 66
wlm_methods, 41, 59, 64
wlmfit, 59
wlmtest, 3, 7, 9, 32, 37, 49, 50, 59, 60, 63
wlmtest_methods, 41, 62, 62
wmf, 9, 32, 35, 39, 40, 54, 66, 69, 70
wmf_methods, 41, 67, 67
wpmf, 9, 32, 35, 39, 49, 54, 67, 69, 72
wpmf_methods, 41, 70, 71
wt, 9, 32, 35, 39, 54, 72, 75
wt_methods, 41, 73, 74