Package 'hht'

March 6, 2013

Type Package

Title The Hilbert-Huang Transform: Tools and Methods

Version 1.0.0

Date 2013-01-09

Depends R (>= 2.15.1), EMD (>= 1.5.2), RSEIS

Description

This package builds on the EMD package to provide additional tools for empirical mode decomposition (EMD) and Hilbert spectral analysis. It also implements the ensemble empirical decomposition (EEMD) method to avoid mode mixing and intermittency problems found in EMD analysis. The package comes with several plotting methods that can be used to view intrinsic mode functions, the HHT spectrum, and the Fourier spectrum.

License GPL (>= 3)

URL http://www.r-project.org

Author Daniel Bowman [aut, cre]

Maintainer Daniel Bowman <daniel.bowman@unc.edu>

R topics documented:

combine_trials *Gather trial files*

Description

This function gathers trial files from multiple directories, renumbers them, and saves them to a single directory for processing using eemd_compile.

Usage

```
combine_trials(in_dirs, out_dir, copy=TRUE)
```

Arguments

in_dirs	Directories containing trial file sets from one EEMD run.
out_dir	Directory in which to save all trial files.
copy	Copy files (copy=TRUE) or move them (copy=FALSE).

Details

Parallel processing is an efficient method for running EEMD. However, this will result in several directories, each with trial files numbered from 1 to N. These files cannot simply be copied together into the same directory, because then they would overwrite each other. This function gathers all trial files in multiple directories, renumbers them, and saves them in a different directory.

Value

The trial files are saved in the directory specified by out_dir.

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>

See Also

eemd, eemd_compile

Examples

```
#Suppose you have run 3 different EEMD sets of 100 trials each and saved the results in e
in_dirs=c("/home/user/eemd1", "/home/user/eemd2/", "/home/user/eemd3")
out_dir="/home/user/all_trials"
## Not run: combine_trials(in_dirs, out_dir)
#Now all your trials should be located in /home/user/all_trials, numbered 1 through 300
```

dcb_emd *Empirical Mode Decomposition*

Description

This function performs empirical mode decomposition. It is the `emd` function in the `EMD` package with the S stoppage rule added.

Usage

```
dcb_emd(xt, tt=NULL, tol=sd(xt)*0.1^2, max.sift=20, stoprule="type1",
    boundary="periodic", smlevels=c(1), sm="none", spar=NA, weight=20,
    check=FALSE, max.imf=10, plot.imf=TRUE, interm=NULL)
```

Arguments

`xt`	observation or signal observed at time `tt`
`tt`	observation index or time index
`tol`	tolerance for stopping rule of sifting
`max.sift`	the maximum number of sifting
`stoprule`	stopping rule of sifting: "type1" (envelope mean <=tol), "type2" (standard deviation between two siftings <= tol), "type3" (S stoppage criterion)
`boundary`	specifies boundary condition from "none" "wave", "symmetric", "periodic" or "evenodd".
`smlevels`	specifies which level of the IMF is obtained by smoothing other than interpolation.
`sm`	specifies whether envelope is constructed by smoothing spline.
`spar`	specifies user-supplied smoothing parameter of spline.
`weight`	the smoothness of spline is determined by `weight` times smoothing parameter of GCV.
`check`	specifies whether the sifting process is displayed. If `check=TRUE`, click the plotting area to start the next step.
`max.imf`	the maximum number of IMF's
`plot.imf`	specifies whether each IMF is displayed. If `plot.imf=TRUE`, click the plotting area to start the next step.
`interm`	specifies vector of periods to be excluded from the IMF's.

Details

This function performs empirical mode decomposition.

Value

`imf`	IMFs
`residue`	residue signal after extracting IMFs from observations `xt`
`nimf`	the number of IMFs

References

Huang, N. E., Shen, Z., Long, S. R., Wu, M. L. Shih, H. H., Zheng, Q., Yen, N. C., Tung, C. C. and Liu, H. H. (1998) The empirical mode decomposition and Hilbert spectrum for nonlinear and nonstationary time series analysis. *Proceedings of the Royal Society London A*, **454**, 903–995.

Huang, N. E. and Wu Z. A. (2008) A review on Hilbert-Huang Transform: Method and its applications to geophysical studies. *Reviews of Geophysics*, **46**, RG2006.

See Also

dcb_extractimf, eemd, plot_imfs

Examples

```
### Empirical Mode Decomposition
ndata <- 3000
tt2 <- seq(0, 9, length=ndata)
xt2 <- sin(pi * tt2) + sin(2* pi * tt2) + sin(6 * pi * tt2)  + 0.5 * tt2

par(mfrow=c(3,1), mar=c(2,1,2,1))
try <- dcb_emd(xt2, tt2, boundary="wave")

### Plotting the IMFs
par(mfrow=c(3,1), mar=c(2,1,2,1))
X11(); par(mfrow=c(try$nimf+1, 1), mar=c(2,1,2,1))
rangeimf <- range(try$imf)
for(i in 1:try$nimf) {
    plot(tt2, try$imf[,i], type="l", xlab="", ylab="", ylim=rangeimf,
    main=paste(i, "-th IMF", sep="")); abline(h=0)
}
plot(tt2, try$residue, xlab="", ylab="", main="residue", type="l",
axes=FALSE); box()
```

---

dcb_extractimf *Intrinsic Mode Function*

Description

This function extracts intrinsic mode function from a signal. Most of this code is from the "extractimf" function in the EMD package.

Usage

```
dcb_extractimf(residue, tt=NULL, tol=sd(residue)*0.1^2, max.sift=20,
    stoprule="type1", boundary="periodic", sm="none", spar=NA,
    weight=20, check=FALSE)
```

Arguments

residue	observation or signal observed at time tt
tt	observation index or time index
tol	tolerance for stopping rule of sifting

max.sift	the maximum number of sifting
stoprule	stopping rule of sifting: "type1" (envelope mean <=tol), "type2" (standard deviation between two siftings <= tol), "type3" (S stoppage criterion)
boundary	specifies boundary condition from "none", "wave", "symmetric", "periodic" or "evenodd".
sm	specifies whether envelope is constructed by smoothing spline.
spar	specifies user-supplied smoothing parameter of spline.
weight	the smoothness of spline is determined by weight times smoothing parameter of GCV.
check	specifies whether the sifting process is displayed. If check=TRUE, click the plotting area to start the next step.

Details

This function extracts intrinsic mode functions from a signal.

Value

imf	imf
residue	residue signal after extracting the finest imf from residue
niter	the number of iteration to obtain the imf

Author(s)

Donghoh Kim <donghoh.kim@gmail.com>, Hee-Seok Oh, Daniel Bowman <daniel.bowman@unc.edu>

References

Huang, N. E., Shen, Z., Long, S. R., Wu, M. L. Shih, H. H., Zheng, Q., Yen, N. C., Tung, C. C. and Liu, H. H. (1998) The empirical mode decomposition and Hilbert spectrum for nonlinear and nonstationary time series analysis. *Proceedings of the Royal Society London A*, **454**, 903–995.

Huang, N. E. and Wu Z. A. (2008) A review on Hilbert-Huang Transform: Method and its applications to geophysical studies. *Reviews of Geophysics*, **46**, RG2006.

See Also

dcb_emd

Examples

```
### Generating a signal
ndata <- 3000
X11(); par(mfrow=c(1,1), mar=c(1,1,1,1))
tt2 <- seq(0, 9, length=ndata)
xt2 <- sin(pi * tt2) + sin(2* pi * tt2) + sin(6 * pi * tt2)  + 0.5 * tt2
plot(tt2, xt2, xlab="", ylab="", type="l", axes=FALSE); box()

### Extracting the first IMF by sifting process
tryimf <- dcb_extractimf(xt2, tt2, check=FALSE)
```

dt — *Ocean Bottom Seismometer Sample Rate*

Description

This is the sample rate for the instrument that recorded `sig`.

Usage

```
data(port_foster_event)
```

Format

A float describing the sample rate (1/second).

Source

Ocean bottom seismometer records from the 2005 TOMODEC active source tomography experiment, Deception Island, Antarctica.

eemd — *Ensemble Empirical Mode Decomposition*

Description

This function performs ensemble empirical mode decomposition (EEMD).

Usage

```
eemd(sig, dt, trials, nimf, noise_amp, emd_config, trials_dir=NULL)
```

Arguments

`sig`	a time series to be decomposed (vector)
`dt`	The sample rate of `sig`
`trials`	Number of times to run EMD
`nimf`	Number of IMFs to record, IMFs past this number will not be saved
`noise_amp`	Amplitude of white noise to use in denoising algorithm
`emd_config`	Configuration parameters for EMD algorithm, see `emd` function in the `EMD` package for a detailed description of what each option does. • `emd_config$tol` sifting stop criterion • `emd_config$stop_rule` EMD stop rules • `emd_config$boundary` how the start and stop of the time series are handled in the splining process • `emd_config$sm` spline smoothing • `emd_config$spar` smoothing parameter • `emd_config$weight` spline weight
`trials_dir`	Directory where EEMD trial files will be stored, defaults to "trials." This will create a directory if none exists.

Details

This function performs ensemble empirical mode decomposition, a noise assisted version of the EMD algorithm. The EEMD works by adding a certain amplitude of white noise to a time series, decomposing it via EMD, and saving the result. If this is done enough times, the averages of the noise perturbed IMFs will approach the "true" IMF set. The EEMD can ameliorate mode mixing and intermittency problems (see references section).

This EEMD algorithm creates a directory `trials_dir` and saves each EMD trial into this directory. The number of trials is defined using `trials`. The trial files in this directory can then be processed using `eemd_compile` to produce the averaged IMF set, or to plot the Hilbert spectrogram of the data. Keep in mind that the EEMD is an expensive algorithm and may take significant time to run.

Value

`emd_result` The result of each individual EMD trial. This is saved directly to file in directory `trials_dir` (i.e. it is not returned by `eemd`.)

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>

References

Wu, Z. A. and Huang, N. E. (2009) Ensemble empirical mode decomposition: A noise assisted data analysis method. *Advances in Adaptive Data Analysis*, **1**, 1-41.

See Also

`sig2imf`, `eemd_compile`, `plot_imfs`.

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
emd_config$tol=0.2
emd_config$stop_rule="type5"
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20
emd_config$S=5

trials=10
nimf=10
noise_amp=6.4e-07
trials_dir="test"

set.seed(628)
#Run EEMD (this may take some time)
## Not run: eemd(sig, dt, trials, nimf, noise_amp, emd_config, trials_dir)

#Compile the results
```

```
## Not run: eemd_result=eemd_compile(trials_dir, trials, nimf)

#Plot the IMFs
time_span=c(5, 10)
imf_list=1:3
os=TRUE
res=TRUE
## Not run: plot_imfs(eemd_result, time_span, imf_list, os, res)
```

`eemd_compile` *Process EEMD results*

Description

This function compiles individual trial files from an EEMD run, allowing other functions to plot IMFs and Hilbert spectrograms of the data.

Usage

```
eemd_compile(trials_dir, trials, nimf)
```

Arguments

`trials_dir`	Directory where previously generated EEMD trial files are stored.
`trials`	Number of trial files to read. This will warn users if the number of requested trials is greater than the number of files in the directory.
`nimf`	Number of IMFs to record, IMFs past this number will not be saved.

Details

The EEMD algorithm can generate hundreds of files, resulting in massive amounts of data. The `eemd_compile` function processes these files, generating an averaged IMF set and compiling the Hilbert spectrogram of each EMD run. The output of `eemd_compile` can be used in `plot_imfs` and `hhspec_image`. The averaged IMF set from `eemd_compile` can be resifted using `eemd_resift`.

Value

`eemd_result`	The averaged IMF set and individual Hilbert spectra of EMD trials generated through EEMD.

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>

See Also

`eemd`

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
emd_config$tol=0.2
emd_config$stop_rule="type5"
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20
emd_config$S=5

trials=10
nimf=10
noise_amp=6.4e-07
trials_dir="test"
set.seed(628)
#Run EEMD (this may take some time)
## Not run: eemd(sig, dt, trials, nimf, noise_amp, emd_config, trials_dir)

#Compile the results
## Not run: eemd_result=eemd_compile(trials_dir, trials, nimf)

#Plot the IMFs
time_span=c(5, 10)
imf_list=1:3
os=TRUE
res=TRUE
## Not run: plot_imfs(eemd_result, time_span, imf_list, os, res)
```

`eemd_resift` *Resift averaged IMFs from EEMD*

Description

Averaged IMFs produced by EEMD may not satisfy the strict definition of an IMF, and therefore they may not have meaningful Hilbert spectrograms. Huang and Wu (2008) suggest another round of sifting to ensure that the averaged IMFs are made to satisfy the IMF definition. This function resifts the averaged IMF set and saves the results based on rules described in the input `resift_rule`.

Usage

```
eemd_resift(eemd_result, emd_config, resift_rule)
```

Arguments

`eemd_result`	The averaged IMF set and individual Hilbert spectra of EMD trials generated through EEMD.
`emd_config`	Configuration parameters for the resifting emd algorithm, see the `emd` function in the `EEMD` package for a detailed description of each option:

- `emd_config$max_sift` maximum number of IMF sifts
- `emd_config$max_imf` maximum number of IMFs that can be returned
- `emd_config$tol` sifting stop criterion
- `emd_config$stop_rule` EMD stop rules
- `emd_config$boundary` how the start and stop of the time series are handled in the splining process
- `emd_config$sm` spline smoothing
- `emd_config$spar` smoothing parameter
- `emd_config$weight` spline weight

`resift_rule` How the resifting algorithm chooses which IMF to save

- Integer - Which IMF in the resifted set will be saved (so if resift_rule=1, the first IMF will be saved, the rest will be discarded)
- "last" - The last IMF will be saved (not terribly useful)
- "max_var" - The IMF with the most variance will be saved. This will get the most "significant" IMF out of each resifted set.
- "all" - Every single new IMF generated from resifting the averaged IMFs will be saved. There may be a lot of them!

Details

The function `eemd_compile` generates a list of averaged IMFs from EEMD trials. These averaged IMFs often do not satisfy the definition of an IMF, usually because some of them are mixtures of different time scales. This is a consequence of the noise perturbation method of EEMD, but it complicates the attempt to create a meaningful Hilbert spectrogram from the averaged IMF set. The resifting algorithm takes each averaged IMF and performs EMD, thereby splitting each one into multiple "sub-IMFs", each of which satisfy the strict definition of an IMF. The question then is: which of these sub-IMFs best represent the averaged IMF The most rigorous solution is to set `resift_rule` to `"all"`, but that tends to make a large number of sub-IMFs, many with very low amplitude. Another solution is to accept the sub-IMF with the most variance, as that probably represents the fundamental information content of the original averaged IMF.

Value

`resift_result`

The resifted results of the averaged IMF set and the individual Hilbert spectra of each resifted IMF.

Author(s)

Daniel Bowman `<daniel.bowman@unc.edu>`

See Also

`eemd`, `eemd_compile`

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
```

```
emd_config$tol=0.2
emd_config$stop_rule="type5"
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20

trials=10
nimf=10
noise_amp=6.4e-07
trials_dir="test"

set.seed(628)

#Run EEMD (this may take some time)
## Not run: eemd(sig, dt, trials, nimf, noise_amp, emd_config, trials_dir)

#Compile the results
## Not run: eemd_result=eemd_compile(trials_dir, trials, nimf)


resift_rule="max_var"
## Not run: resift_result=eemd_resift(eemd_result, emd_config, resift_rule)

#Plot the IMFs
time_span=c(5, 10)
imf_list=1:3
os=TRUE
res=TRUE
## Not run: plot_imfs(resift_result, time_span, imf_list, os, res)
```

`evolutive_fft` *Generate Fourier spectrogram*

Description

This function generates a spectrogram using the evolutive FFT method. It is a modified copy of the `evolfft` function in the RSEIS package, and still depends on this package to work.

Usage

```
evolutive_fft(sig, dt, ft, freq_span, taper = 0.05)
```

Arguments

`sig`	The signal to process
`dt`	sample rate
`ft`	Fourier spectrogram options • `ft$nfft` is the fft length • `ft$ns` is the number of samples in a window • `ft$nov` is the number of samples to overlap
`freq_span`	Frequency span to render spectrogram over. `c(0, -1)` plots everything up to the Nyquist frequency.
`taper`	Taper value to use for spectrogram (default is 0.05)

Details

This function is a simple Fourier spectrogram plotter. It's useful to compare this image with images generated by hhspec_image to see how the Fourier and Hilbert spectrograms differ.

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>, Jonathan M. Lees

References

Jonathan M. Lees (2012). RSEIS: Seismic Time Series Analysis Tools. R package version 3.0-6. http://CRAN.R-project.org/package=RSEIS

See Also

ftspec_image

Examples

```
data(port_foster_event)

ft=list()
ft$nfft=4096
ft$ns=30
ft$nov=29

freq_span=c(0, 25)
ev = evolutive_fft(sig, dt, ft, freq_span)

#Plot raw spectrogram
image_z=t(ev$DSPEC[1:(ev$numfreqs/2),])
f_ind=(ev$freqs>=freq_span[1] & ev$freqs <= freq_span[2])
image_z=t(ev$DSPEC[1:(ev$numfreqs/2),])
image_z=image_z[,f_ind]
image(image_z)
```

---

ftspec_image *Display Fourier spectrogram*

Description

This function displays a Fourier spectrogram using the same plot structure and options as hhspec_image. It uses the function evolutive_fft to generate a spectrogram, then wraps it in the same plotting format as hhspec_image. ftspec_image depends on functions contained in the RSEIS package to work.

Usage

```
ftspec_image(sig, dt, ft, time_span, freq_span, amp_span, amp_units=NULL,
     amp_unit_conversion=NULL, taper=0.05, grid=TRUE, colorbar=TRUE,
     backcol=c(0, 0, 0), colormap=NULL, pretty=TRUE, cex=1, main="")
```

Arguments

sig	The signal to process
dt	sample rate
ft	Fourier spectrogram options • ft$nfft is the fft length • ft$ns is the number of samples in a window • ft$nov is the number of samples to overlap
time_span	Time span to render spectrogram over. c(0, -1) will draw the spectrogram over the entire signal.
freq_span	Frequency span to render spectrogram over. c(0, -1) plots everything up to the Nyquist frequency.
amp_span	Amplitude range to plot. c(0, -1) plots everything.
amp_units	What to call the amplitude units.
amp_unit_conversion	How to convert amplitude units of the input signal to amplitude units on the image
taper	Taper value to use for spectrogram (default is 0.05)
grid	Boolean - whether to display grid lines or not
colorbar	Boolean - whether to display amplitude colorbar or not
backcol	What background color to use behind the spectrogram, in a 3 element vector: c(red, green, blue)
colormap	What palette object to use for the spectrogram, defaults to rainbow(colorbins,start=0,end
pretty	Boolean - to choose nice axes values or to use exactly the ranges given
cex	Font scaling.
main	Title of main plot

Details

This function is a simple Fourier spectrogram plotter. It's useful to compare this image with images generated by hhspec_image to see how the Fourier and Hilbert spectrograms differ.

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>

References

Jonathan M. Lees (2012). RSEIS: Seismic Time Series Analysis Tools. R package version 3.0-6. http://CRAN.R-project.org/package=RSEIS

See Also

hhspec_image, evolutive_fft

Examples

```
data(port_foster_event)

ft=list()
ft$nfft=4096
ft$ns=30
ft$nov=29

time_span=c(5, 10)
freq_span=c(0, 25)
amp_span=c(1e-5, 0.0003)
ftspec_image(sig, dt, ft, time_span, freq_span, amp_span)
```

`hhspec_image` *Display Hilbert Huang spectrogram*

Description

This function displays the Hilbert spectrogram of EMD and EEMD results.

Usage

```
hhspec_image(hspec, time_span, freq_span, amp_span, cluster_span=NULL,
amp_units=NULL, amp_unit_conversion=NULL, grid=TRUE, colorbar=TRUE,
backcol=c(0, 0, 0), colormap=NULL, pretty=TRUE, cex=1, main="")
```

Arguments

`hspec`	Data structure generated by `hh_render`.
`time_span`	Time span to render spectrogram over. `c(0, -1)` will draw the spectrogram over the entire signal.
`freq_span`	Frequency span to render spectrogram over. `c(0, -1)` plots everything up to the max frequency set when `hh_render` was run.
`amp_span`	This is the amplitude span to plot, everything below is set to `backcol`, everything above is set to max color, `c(0, -1)` scales to the range in the signal.
`cluster_span`	Plots only parts of the signal that have a certain number of data points per pixel (see notes below). This only applies when you're plotting EEMD data. The pixel range is defined as `c(AT LEAST, AT MOST)`.
`amp_units`	What to call the amplitude units.
`amp_unit_conversion`	How to convert amplitude units of the input signal to amplitude units on the image
`grid`	Boolean - whether to display grid lines or not
`colorbar`	Boolean - whether to display amplitude colorbar or not
`backcol`	What background color to use behind the spectrogram, in a 3 element vector: `c(red, green, blue)`
`colormap`	What palette object to use for the spectrogram, defaults to `rainbow(colorbins,start=0,end`
`pretty`	Boolean - to choose nice axes values or to use exactly the ranges given
`cex`	Font scaling.
`main`	Title of main plot

Details

This function plots the image generated by `hh_render` along with the original signal. The plotter can use data from both EMD and EEMD runs. When it plots EEMD data, it shows the time/frequency plot of every single trial at once. The `cluster_span` option is useful in this case because it can distinguish "signal" (pixels where multiple trials intersect) from "noise" (whether from EEMD or from nature) where only one trial contributes data.

Note

It may take some trial and error to get a nice image. For example, if the data points are too small (and thus the spectrogram looks like a mist of fine points rather than continuous frequency bands), try rerunning `hh_render`, but with lower frequency resolution. If the spectrogram is extremely noisy, try defining `cluster_span` - this usually gets rid of most of the random noise. For example, a `cluster_span` of `c(3, 10)` only keeps pixels that have data from at least 3 trials, up to 10. Most noise pixels will only have one trial contributing data. The upper limit (10) is a formality - it does not make much sense at this point to put an upper limit on trial intersections unless you are interested in the **noise** component isolated from the signal.

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>

See Also

`ftspec_image`, `hh_render`

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
emd_config$tol=0.2
emd_config$stop_rule="type5"
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20
emd_config$S=5

trials=10
nimf=10
noise_amp=6.4e-07
trials_dir="test"

set.seed(628)
#Run EEMD (this may take some time)
## Not run: eemd(sig, dt, trials, nimf, noise_amp, emd_config, trials_dir)

#Compile the results
## Not run: eemd_result=eemd_compile(trials_dir, trials, nimf)

#Calculate spectrogram
max_freq=25
```

```
freq_step=0.01
## Not run: hspec=hh_render(eemd_result, max_freq, freq_step)

#Plot spectrogram
time_span=c(5, 10)
freq_span=c(0, 25)
amp_span=c(1e-6, 2.5e-5)
## Not run: hhspec_image(hspec, time_span, freq_span, amp_span)
```

`hhtransform` *Hilbert transform wrapper*

Description

This function is a wrapper for the Hilbert transform functions included in the EMD package.

Usage

```
hhtransform(imf_set)
```

Arguments

`imf_set`	A data structure returned by `sig2imf`, `eemd`, or `eemd_resift`.

Details

This function determines instantaneous amplitude and frequency from IMFs generated by the EMD method.

Value

`hht_result`	The input data structure `imf_set`, but with fields `imf_set$hinstfreq` (frequency) and `imf_set$hamp` (amplitude).

Author(s)

Daniel Bowman `<daniel.bowman@unc.edu>`

References

Huang, N. E., Shen, Z., Long, S. R., Wu, M. L. Shih, H. H., Zheng, Q., Yen, N. C., Tung, C. C. and Liu, H. H. (1998) The empirical mode decomposition and Hilbert spectrum for nonlinear and nonstationary time series analysis. *Proceedings of the Royal Society London A*, **454**, 903–995.

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
emd_config$tol=0.2
emd_config$stop_rule="type5"
```

```
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20
emd_config$S=5

#Run EMD (this may take some time)
emd_result=sig2imf(sig, dt, emd_config)

#Get instantaneous amplitude and frequency
emd_result=hhtransform(emd_result)

#Render spectrogram
max_freq=25
freq_step=0.05
hspec=hh_render(emd_result, max_freq, freq_step)

#Show result
time_span=c(5, 10)
freq_span=c(0, 25)
amp_span=c(0.000001, 0.00001)
hhspec_image(hspec, time_span, freq_span, amp_span)
```

`hh_render` *Render Hilbert spectrogram*

Description

This function prepares results from the Hilbert transform of EMD or EEMD results for display using `hhspec_image`.

Usage

```
hh_render(hres, max_freq, freq_step, imf_list=NULL)
```

Arguments

`hres`	This is the output generated by `eemd_compile`, `eemd_resift`, or `hhtransform`.
`max_freq`	Maximum frequency to plot on spectrogram.
`freq_step`	Frequency resolution.
`imf_list`	IMFs to include in spectrogram

Details

`hh_render` processes Hilbert spectral data prior to display. This function is separate from the plotting function `hhspec_image` because it is computer intensive to generate the spectral data. It is best to generate the spectrogram first, then change the options in `hhspec_image` to display the image you want, rather than recalculating the spectrogram every time.

Value

`hspec`	A data structure containing the spectrogram image and other information required by `hhspec_image`.

Note

The hh_render function also keeps track of which trial contributes what data to the spectrogram. For the EMD, this does not make much sense, because there is only one trial. However, when hh_render is run on EEMD results, it remembers which time/frequency bin gets data from each trial. This is a way to distinguish between noise and signal in data: signal is where multiple trials contribute data to the same time/frequency bin, noise is where only one (or a couple) of trials contribute data. The cluster_span option in hhspec_image function allows users to restrict the spectrogram display to only those pixels that have multiple trial hits in them.

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>

See Also

eemd_compile, hhspec_image

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
emd_config$tol=0.2
emd_config$stop_rule="type5"
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20
emd_config$S=5

trials=10
nimf=10
noise_amp=6.4e-07
trials_dir="test"

set.seed(628)
#Run EEMD (this may take some time)
## Not run: eemd(sig, dt, trials, nimf, noise_amp, emd_config, trials_dir)

#Compile the results
## Not run: eemd_result=eemd_compile(trials_dir, trials, nimf)

#Calculate spectrogram
max_freq=25
freq_step=0.01
## Not run: hspec=hh_render(eemd_result, max_freq, freq_step)

#Plot spectrogram
time_span=c(5, 10)
freq_span=c(0, 25)
amp_span=c(1e-6, 2.5e-5)
## Not run: hhspec_image(hspec, time_span, freq_span, amp_span)
```

plot_imfs *Display IMFs*

Description

This function displays IMFs generated using sig2imf, eemd_compile. or eemd_resift

Usage

```
plot_imfs(sig, time_span, imf_list, original_signal,
    residue, fit_line=FALSE, lwd=1, cex=1, ...)
```

Arguments

sig	Data structure returned by sig2imf, eemd_compile. or eemd_resift.
time_span	Time span over which to plot IMFs. c(0, -1) will draw the entire signal.
imf_list	Which IMFs to plot.
original_signal	(boolean) whether or not to plot the original signal.
residue	(boolean) whether to plot the residue of the EMD method.
fit_line	(boolean) whether to add a red line to the original signal trace showing how much of the original signal is contained in the selected IMFs and/or residual.
lwd	Line weight.
cex	Text size.
...	Pass additional graphics parameters to IMF plotter

Details

This function plots the IMF decomposition of a signal. It can show the original signal and also the residue left over when the IMFs are removed from the signal. The plotter can use data from both EMD and EEMD runs. When it plots EEMD data, it shows the averaged IMFs from the trials processed by eemd_compile.

Note

It is very important to inspect the IMF set prior to rendering Hilbert spectrograms. Oftentimes, problems with the EMD are obvious when the IMFs are plotted. The fit_line option can help with this.

Author(s)

Daniel Bowman <daniel.bowman@unc.edu>

See Also

hhspec_image

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
emd_config$tol=5
emd_config$stop_rule="type5"
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20

#Run EMD
emd_result=sig2imf(sig, dt, emd_config)

#Plot the first 4 IMFs of the EEMD of a signal.
time_span=c(5, 10)
imf_list=1:4
original_signal=TRUE
residue=TRUE

plot_imfs(emd_result, time_span, imf_list, original_signal, residue)

#Check how much contribution IMFs 2 and 3 make to the complete signal.
imf_list=c(2, 3)
fit_line=TRUE
plot_imfs(emd_result, time_span, imf_list, original_signal, residue, fit_line)
```

---

sig *Transitory Seismic Event at Deception Island Volcano*

Description

This is 20 seconds of data from the 2005 TOMODEC ocean bottom seismometer network at Deception Island, South Shetland Islands, Antarctica with sample rate `dt`. It shows one of several thousand transitory seismic events occurring in Port Foster (the flooded caldera of the volcano).

Usage

```
data(port_foster_event)
```

Format

A 2500 element vector containing the seismic record. Units are meters per second.

Source

Ocean bottom seismometer records from the 2005 TOMODEC active source tomography experiment, Deception Island, Antarctica.

`sig2imf` *Empirical Mode Decomposition wrapper*

Description

This function wraps the `emd` function in the `EMD` package. `sig2imf` is used in `eemd` and others.

Usage

```
sig2imf(sig, dt, emd_config)
```

Arguments

`sig`	a time series to be decomposed (vector)
`dt`	The sample rate of `sig`
`emd_config`	Configuration information for the EMD algorithm, see the `emd` function in the `EMD` package for a detailed description of what each option does.

- `emd_config$max_sift` maximum number of IMF sifts
- `emd_config$max_imf` maximum number of IMFs that can be returned
- `emd_config$tol` sifting stop criterion
- `emd_config$stop_rule` EMD stop rules
- `emd_config$boundary` how the start and stop of the time series are handled in the splining process
- `emd_config$sm` spline smoothing
- `emd_config$spar` smoothing parameter
- `emd_config$weight` spline weight

Details

This function configures and performs empirical mode decomposition.

Value

`emd_result`	A data structure for input into `eemd_compile`, `plot_imfs`, and `hhtransform`.

References

Huang, N. E., Shen, Z., Long, S. R., Wu, M. L. Shih, H. H., Zheng, Q., Yen, N. C., Tung, C. C. and Liu, H. H. (1998) The empirical mode decomposition and Hilbert spectrum for nonlinear and nonstationary time series analysis. *Proceedings of the Royal Society London A*, **454**, 903–995.

Huang, N. E. and Wu Z. A. (2008) A review on Hilbert-Huang Transform: Method and its applications to geophysical studies. *Reviews of Geophysics*, **46**, RG2006.

See Also

`eemd`, `plot_imfs`

Examples

```
data(port_foster_event)

emd_config=list()
emd_config$max_sift=200
emd_config$max_imf=100
emd_config$tol=5
emd_config$stop_rule="type5"
emd_config$boundary="wave"
emd_config$sm="none"
emd_config$spar=NA
emd_config$weight=20

#Run EMD
emd_result=sig2imf(sig, dt, emd_config)

#Display IMFs

time_span=c(5, 10)
imf_list=1:3
original_signal=TRUE
residue=TRUE

plot_imfs(emd_result, time_span, imf_list, original_signal, residue)

#Get Hilbert transform
emd_result=hhtransform(emd_result)

#Plot spectrogram
max_freq=25
freq_step=0.05
hspec=hh_render(emd_result, max_freq, freq_step)

freq_span=c(0, 25)
amp_span=c(0.000001, 0.00001)
hhspec_image(hspec, time_span, freq_span, amp_span)
```

Index