Package: hht (via r-universe)

August 22, 2024

Type Package

Title The Hilbert-Huang Transform: Tools and Methods

Version 2.1.6

Date 2023-03-12

Depends R (>= 3.1.1)

Imports EMD (>= 1.5.5), fields (>= 6.7.6)

Description 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) and the complete ensemble empirical mode decomposition (CEEMD) methods 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)

Maintainer Daniel C. Bowman <danny.c.bowman@gmail.com>

Author Daniel C. Bowman [aut, cre], Jonathan M. Lees [ctb]

NeedsCompilation no

Date/Publication 2023-03-13 14:30:06 UTC

Repository https://dannycbowman.r-universe.dev

RemoteUrl https://github.com/cran/hht

RemoteRef HEAD

RemoteSha 493c5d0fbeaa2f530eb9e86010f485e6f7a2e176

Contents

`CEEMD` *Complete Ensemble Empirical Mode Decomposition*

Description

This function implements the complete ensemble empirical mode decomposition (CEEMD) algorithm.

Usage

```
CEEMD(sig, tt, noise.amp, trials, verbose = TRUE,
    spectral.method = "arctan", diff.lag = 1, tol = 5, max.sift = 200,
    stop.rule = "type5", boundary = "wave", sm = "none",
    smlevels = c(1), spar = NULL, max.imf = 100, interm = NULL,
    noise.type = "gaussian", noise.array = NULL)
```

Arguments

`sig`	a time series to be decomposed (vector)
`tt`	The sample times of `sig`
`noise.amp`	Amplitude of white noise to use in denoising algorithm
`trials`	Number of times to run EMD
`verbose`	If TRUE, notify when each trial is complete
`spectral.method`	See `Sig2IMF`.
`diff.lag`	See `Sig2IMF`.
`tol`	See `Sig2IMF`.

max.sift	See Sig2IMF.
stop.rule	See Sig2IMF.
boundary	See Sig2IMF.
sm	See Sig2IMF.
smlevels	See Sig2IMF.
spar	See Sig2IMF.
max.imf	See Sig2IMF.
interm	See Sig2IMF.
noise.type	If unspecified or gaussian, produce a Gaussian noise series with length length(sig) and standard deviation noise.amp. If uniform, produce a uniform random distribution with length length(sig) and maximum absolute value of noise.amp. If custom, then use a custom noise array as defined in input parameter noise.array (see below).
noise.array	If noise.type = "custom", this array must be a TRIALS x LENGTH(TT) collection of time series to be used in the place of uniform or gaussian noise. Each row in the array corresponds to the noise series added for that particular trial during the CEEMD run. By default, noise.array = NULL.

Details

This function performs the complete ensemble empirical mode decomposition, a noise assisted empirical mode decomposition algorithm. The CEEMD works by adding a certain amplitude of white noise to a time series, decomposing it via EMD, and saving the result. In contrast to the Ensemble Empirical Mode Decomposition (EEMD) method, the CEEMD also ensures that the IMF set is quasi-complete and orthogonal. The CEEMD can ameliorate mode mixing and intermittency problems. Keep in mind that the CEEMD is a computationally expensive algorithm and may take significant time to run.

Value

ceemd.result	The final result of the CEEMD algorithm

.

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

References

Torres, M. E., Colominas, M. A., Schlotthauer, G., Flandrin, P. (2011). A complete ensemble empirical mode decomposition with adaptive noise. *2011 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp.4144-4147, doi: 10.1109/ICASSP.2011.5947265.

See Also

EEMD, Sig2IMF, PlotIMFs.

Examples

```
## Not run:

data(PortFosterEvent)
noise.amp <- 6.4e-07
trials <- 100

ceemd.result <- CEEMD(sig, tt, noise.amp, trials)
PlotIMFs(ceemd.result, imf.list = 1:6, time.span = c(5, 10))

## End(Not run)
```

`CombineTrials` *Gather EEMD trial files*

Description

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

Usage

```
CombineTrials(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 (`TRUE`) or move them (`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 <danny.c.bowman@gmail.com>

See Also

EEMD, EEMDCompile

Examples

```
#Suppose you have run 3 different EEMD sets of 100 trials each
#and saved the results in EEMD1, EEMD2, EEMD3, respectively:
in.dirs <- c("/home/user/EEMD1", "/home/user/EEMD2/", "/home/user/EEMD3")
out.dir <- "/home/user/all.trials"
## Not run: CombineTrials(in.dirs, out.dir)
#Now all your trials should be located in /home/user/all.trials,
#numbered 1 through 300
```

EEMD *Ensemble Empirical Mode Decomposition*

Description

This function performs ensemble empirical mode decomposition (EEMD).

Usage

```
EEMD(sig, tt, noise.amp, trials, nimf, trials.dir = NULL, verbose = TRUE,
    spectral.method = "arctan", diff.lag = 1, tol = 5, max.sift = 200,
    stop.rule = "type5", boundary = "wave", sm = "none",
    smlevels = c(1), spar = NULL, max.imf = 100, interm = NULL,
    noise.type = "gaussian", noise.array = NULL)
```

Arguments

sig	a time series to be decomposed (vector)
tt	The sample times of sig
noise.amp	Amplitude of white noise to use in denoising algorithm
trials	Number of times to run EMD
nimf	Number of IMFs to record, IMFs past this number will not be saved
trials.dir	Directory where EEMD trial files will be stored, defaults to "trials." This will create a directory if none exists.
verbose	If TRUE, notify when each trial is complete
spectral.method	See Sig2IMF.
diff.lag	See Sig2IMF.
tol	See Sig2IMF.
max.sift	See Sig2IMF.
stop.rule	See Sig2IMF.

boundary	See Sig2IMF.
sm	See Sig2IMF.
smlevels	See Sig2IMF.
spar	See Sig2IMF.
max.imf	See Sig2IMF.
interm	See Sig2IMF.
noise.type	If unspecified or gaussian, produce a Gaussian noise series with length length(sig) and standard deviation noise.amp. If uniform, produce a uniform random distribution with length length(sig) and maximum absolute value of noise.amp. If custom, then use a custom noise array as defined in input parameter noise.array (see below).
noise.array	If noise.type = "custom", this array must be a TRIALS x LENGTH(TT) collection of time series to be used in the place of uniform or gaussian noise. Each row in the array corresponds to the noise series added for that particular trial during the EEMD run. By default, noise.array = NULL.

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 EEMDCompile 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 files in directory trials.dir (i.e. it is not returned by EEMD.)

Note

Previous versions of this function used a uniform random noise distribution (i.e. runif) to generate the noise time series. The default noise time series is now Gaussian in accordance with existing EEMD literature.

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

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, CombineTrials, EEMDCompile, PlotIMFs.

Examples

```
data(PortFosterEvent)
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, tt, noise.amp, trials, nimf, trials.dir = trials.dir)

#Compile the results
## Not run: EEMD.result <- EEMDCompile(trials.dir, trials, nimf)

#Plot the IMFs
time.span <- c(5, 10)
imf.list <- 1:3
os <- TRUE
res <- TRUE
## Not run: PlotIMFs(EEMD.result, time.span, imf.list, os, res)
```

EEMDCompile *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

```
EEMDCompile(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 per EMD run. IMFs past this number will not be saved.

Details

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

Value

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

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

See Also

EEMD, CombineTrials

Examples

```
data(PortFosterEvent)
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, tt, noise.amp, trials, nimf, trials.dir = trials.dir)

#Compile the results
## Not run: EEMD.result <- EEMDCompile(trials.dir, trials, nimf)

#Plot the IMFs
time.span <- c(5, 10)
imf.list <- 1:3
os <- TRUE
res <- TRUE
## Not run: PlotIMFs(EEMD.result, time.span, imf.list, os, res)
```

EEMDResift *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

```
EEMDResift(EEMD.result, resift.rule, spectral.method = "arctan",
    diff.lag = 1, tol = 5, max.sift = 200, stop.rule = "type5",
    boundary = "wave", sm = "none", smlevels = c(1),
    spar = NULL, max.imf = 100, interm = NULL)
```

Arguments

`EEMD.result`	The averaged IMF set and individual Hilbert spectra of EMD trials generated through EEMD.
`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!
`spectral.method`	See `Sig2IMF`.
`diff.lag`	See `Sig2IMF`.
`tol`	See `Sig2IMF`.
`max.sift`	See `Sig2IMF`.
`stop.rule`	See `Sig2IMF`.
`boundary`	See `Sig2IMF`.
`sm`	See `Sig2IMF`.
`smlevels`	See `Sig2IMF`.
`spar`	See `Sig2IMF`.
`max.imf`	See `Sig2IMF`.
`interm`	See `Sig2IMF`.

Details

The function `EEMDCompile` 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 <danny.c.bowman@gmail.com>

See Also

EEMD, EEMDCompile

Examples

```
data(PortFosterEvent)

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, tt, noise.amp, trials, nimf, noise.amp, trials.dir = trials.dir)

#Compile the results
## Not run: EEMD.result <- EEMDCompile(trials.dir, trials, nimf)


resift.rule="max.var"
## Not run: resift.result <- EEMDResift(EEMD.result, resift.rule)

#Plot the IMFs
time.span=c(5, 10)
imf.list=1:3
os=TRUE
res=TRUE
## Not run: PlotIMFs(resift.result, time.span, imf.list, os, res)
```

EvolutiveFFT *Calculate the evolutive Fourier spectrogram.*

Description

Generates the evolutive Fourier spectrogram of a signal, and returns it for use in FTGramImage.

Usage

```
EvolutiveFFT(sig, dt, ft, freq.span, taper = 0.05)
```

Arguments

sig	Signal to analyze.
dt	Sample rate (must be constant).
ft	Fourier transform input parameters • ft$nfft The frequency resolution, should be in powers of 2

- ft$ns Number of samples in a window
- ft$nov Number of samples to overlap, must be less than `ft$ns`

`freq.span`	Frequency range to return.
`taper`	Amount of cosine taper to apply.

Details

This is an internal function and users will likely not call it directly.

Value

`z`	Power spectrum
`y`	Frequency
`x`	Time
`original.signal`	The input signal
`tt`	Sample times based on input sample rate `dt`

Note

This is a modification of the `evolfft` function in the `RSEIS` package.

Author(s)

Daniel C. Bowman <danny.c.bowman@gmail.com>, Jonathan M. Lees

References

Jonathan M. Lees (2012). RSEIS: Seismic Time Series Analysis Tools. R package version 3.1-3.

`FTGramImage` *Display Fourier spectrogram*

Description

This function displays a Fourier spectrogram using the same plot structure and options as `HHGramImage`. It uses the function `EvolutiveFFT` to generate a spectrogram, then wraps it in the same plotting format as `HHGramImage`.

Usage

```
FTGramImage(sig, dt, ft, time.span = NULL, freq.span = NULL,
     amp.span = NULL, taper=0.05, scaling = "none", grid=TRUE,
    colorbar=TRUE, backcol=c(0, 0, 0), colormap=NULL, pretty=FALSE, ...)
```

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. NULL will draw the spectrogram over the entire signal.
freq.span	Frequency span to render spectrogram over. NULL plots everything up to the Nyquist frequency.
amp.span	Amplitude range to plot. NULL plots everything.
taper	Taper value to use for spectrogram (default is 0.05), see spec.taper in the base package.
scaling	determines whether to apply logarithmic (log), or square root (sqrt) scaling to the amplitude data
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
pretty	Boolean - choose nice axes values, some adjustment may result
...	This function supports some optional parameters as well:

- trace.format - the format of the trace minima and maxima in sprintf format
- img.x.format - the format of the X axis labels of the image in sprintf format
- img.y.format - the format of the Y axis labels of the image in sprintf format
- colorbar.format - the format of the colorbar labels in sprintf format
- cex.lab - the font size of the image axis labels
- cex.colorbar - the font size of the colorbar
- cex.trace - the font size of the trace axis labels
- img.x.lab - the X - axis label of the image, it defaults to "time"
- img.y.lab- the Y - axis label of the image, it defaults to "frequency"
- main - figure title

Details

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

Value

img	The spectrogram image, suitable for plotting with the generic image function

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

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

HHGramImage, EvolutiveFFT

Examples

```
data(PortFosterEvent)

dt <- mean(diff(tt))

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)
FTGramImage(sig, dt, ft, time.span = time.span, freq.span = freq.span,
    amp.span = amp.span, pretty = TRUE, img.x.format = "%.1f",
    img.y.format = "%.0f",
    main = "Port Foster Event - Fourier Spectrogram")
```

HHGramImage *Display Hilbert spectrogram*

Description

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

Usage

```
HHGramImage(hgram, time.span = NULL, freq.span = NULL, amp.span = NULL,
    clustergram = FALSE, cluster.span = NULL, imf.list = NULL,
    fit.line = FALSE, scaling = "none", grid = TRUE, colorbar = TRUE,
    backcol = c(0, 0, 0), colormap = NULL, pretty = FALSE, ...)
```

Arguments

hgram	Data structure generated by HHRender.
time.span	Time span to render spectrogram over. NULL will draw the spectrogram over the entire signal.
freq.span	Frequency span to render spectrogram over. NULL plots everything up to the max frequency set when HHRender was run.
amp.span	This is the amplitude span to plot, everything below is set to backcol, everything above is set to max color, NULL scales to the range in the signal.
clustergram	If TRUE, plot the number of times data occupies a given pixel instead of plotting the signal amplitude. This is akin to the weight component returned by the as.image function in the fields package. Only applies when using EEMD. Default is FALSE.
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 using EEMD. The pixel range is defined as c(AT LEAST, AT MOST).
imf.list	A vector of IMFs to plot on the spectrogram, the others will not be shown. You must set combine.imfs = FALSE in HHRender for this to work correctly.
fit.line	If TRUE, plot a red line on the trace that shows the part of the signal represented by the spectrogram
.	
scaling	determines whether to apply a logarithmic ("log"), or square root ("sqrt") scaling to the amplitude data, default is "none"
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
pretty	Boolean - to choose nice axes values or to use exactly the ranges given
...	This function supports some optional parameters as well: • trace.format - the format of the trace minima and maxima in sprintf format • img.x.format - the format of the X axis labels of the image in sprintf format • img.y.format - the format of the Y axis labels of the image in sprintf format • colorbar.format - the format of the colorbar labels in sprintf format • cex.lab - the font size of the image axis labels • cex.colorbar - the font size of the colorbar • cex.trace - the font size of the trace axis labels • img.x.lab - the X - axis label of the image, it defaults to "time" • img.y.lab - the Y - axis label of the image, it defaults to "frequency" • main - adds a title to the figure

Details

This function plots the image generated by HHRender along with the original signal trace. 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.

Value

img	The spectrogram image, suitable for plotting with the generic image function

Note

Using the option combine.imfs = FALSE in HHRender will cause HHGramImage to run much, much slower. Unless you have a compelling reason to plot certain IMFs (as opposed to all of them combined), I recommend against using this.

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 HHRender, 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 <danny.c.bowman@gmail.com>

See Also

FTGramImage, HHRender, HHSpecPlot

Examples

```
data(PortFosterEvent)

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, tt, noise.amp, trials, nimf, trials.dir = trials.dir)

#Compile the results
## Not run: EEMD.result <- EEMDCompile(trials.dir, trials, nimf)

#Calculate spectrogram
```

```
dt  <-  0.1
dfreq  <-  0.1
## Not run: hgram <- HHRender(EEMD.result, dt, dfreq)


#Plot spectrogram
time.span <- c(4, 10)
freq.span <- c(0, 25)
## Not run: HHGramImage(hgram, time.span, freq.span,
pretty = TRUE, img.x.format = "%.1f", img.y.format = "%.0f",
main = "Port Foster event - ensemble Hilbert spectrogram")
## End(Not run)

#Plot intersections

## Not run: HHGramImage(hgram, time.span, freq.span, amp.span = c(1, 5),
clustergram = TRUE, pretty = TRUE, img.x.format = "%.1f", colorbar.format = "%.0f",
img.y.format = "%.0f", main = "Port Foster event - signal stability")
## End(Not run)

#Decluster
#show only areas with stable signal
#i.e. each pixel has data from at least 3 EEMD trials
## Not run: HHGramImage(hgram, time.span = time.span, freq.span = freq.span,
cluster.span = c(, 10), pretty = TRUE, img.x.format = "%.1f",
img.y.format = "%.0f",
main = "Port Foster event - ensemble Hilbert spectrogram")
## End(Not run)

#Log amplitude plot

## Not run: HHGramImage(hgram, time.span = time.span, freq.span = freq.span,
scaling = "log", pretty = TRUE, img.x.format = "%.1f", img.y.format = "%.0f",
main = "Port Foster event - ensemble Hilbert spectrogram with log amplitude")
## End(Not run)

#Log frequency plot
dfreq <- 0.001
## Not run: hgram=HHRender(EEMD.result, dt, dfreq, scaling = "log")
## Not run: HHGramImage(hgram, time.span, freq.span = c(0, 2),
pretty = TRUE, img.x.format = "%.1f", img.y.format = "%.2f",
img.y.lab = "log frequency",
main = "Port Foster event - ensemble Hilbert spectrogram with log frequency")
## End(Not run)

#Only show IMF 1
dfreq <- 0.1
## Not run: hgram=HHRender(EEMD.result, dt, dfreq, combine.imfs = FALSE)
## Not run: HHGramImage(hgram, time.span, freq.span, imf.list = 1,
pretty = TRUE, img.x.format = "%.1f", img.y.format = "%.0f",
main = "Port Foster event - ensemble Hilbert spectrogram of IMF 1")
## End(Not run)
```

HHRender *Render Hilbert spectrogram*

Description

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

Usage

```
HHRender(hres, dt, dfreq, time.span = NULL, freq.span = NULL, scaling = "none",
    combine.imfs = TRUE, verbose = TRUE)
```

Arguments

hres	This is the output generated by Sig2IMF, EEMDCompile, EEMDResift , or CEEMD.
dt	Time resolution of spectrogram. Must be greater than the sample rate of the time series to avoid gaps.
dfreq	Frequency resolution of spectrogram.
time.span	Time span to render spectrogram over; NULL means over the whole time series.
freq.span	Frequency span to include in spectrogram; NULL means render all the frequencies in the time series
scaling	If "log", render a log frequency spectrogram. Defaults to "none" (linear).
combine.imfs	If TRUE, add the spectra for all IMFS together in one ensemble Hilbert spectrogram, if FALSE, keep separate so you can investigate individual IMFs in HHGramImage.
verbose	If TRUE, print progress messages.

Details

HHRender processes Hilbert spectral data prior to displaying with HHGramImage. HHRender returns the ensemble Hilbert spectrogram if combine.imfs = TRUE, otherwise it returns an IMF-by-IMF Hilbert spectrogram of dimensions [time, freq, imf]. The user can then choose which IMFs to plot when he or she calls HHGramImage, but this makes HHGramImage run about 40x slower on my machine. The trade off is in speed versus flexibility for HHGramImage; the combine.imfs option does not affect HHRender very much.

Value

hgram	A data structure containing the spectrogram image and other information required by HHGramImage.

Note

The HHRender 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 HHRender 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. See options for HHGramImage for ways to plot data based on signal stability.

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

See Also

EEMDCompile, HHGramImage

Examples

```
data(PortFosterEvent)

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, tt, noise.amp, trials, nimf, noise.amp, trials.dir <- trials.dir)

#Compile the results
## Not run: EEMD.result <- EEMDCompile(trials.dir, trials, nimf)

#Calculate spectrogram
dt <- 0.1
dfreq <- 0.1

## Not run: hgram <- HHRender(EEMD.result, dt, dfreq)

#Plot spectrogram
time.span <- c(5, 10)
freq.span <- c(0, 25)
amp.span <- c(1e-6, 2.5e-5)
## Not run: HHGramImage(hgram, time.span = time.span,
    freq.span = freq.span, amp.span = amp.span)
## End(Not run)
```

HHSpecPlot *Display Hilbert periodogram*

Description

This function displays the Hilbert periodogram, with options to plot individual IMFs and also the Fourier periodogram for comparison.

Usage

```
HHSpecPlot(hspec, freq.span = NULL, scaling = "none", imf.list = NULL,
    show.total = TRUE, show.fourier = FALSE, scale.fourier = FALSE,
    show.imfs = FALSE, legend = TRUE, ...)
```

Arguments

hspec	Data structure returned by HHSpectrum
freq.span	Frequency range to plot, NULL plots all of them
scaling	Amplitude scaling, can be "log" (log 10), "sqrt" (square root), defaults to "none".
imf.list	Which IMFs to plot, requires show.imfs = TRUE.
show.total	Show the ensemble Hilbert spectrogram
show.fourier	Show the Fourier periodogram
scale.fourier	Scale Fourier and Hilbert spectra to each other for easier comparison
show.imfs	Plot individual IMF spectra
legend	Determines whether or not a legend is shown
...	This function supports some optional parameters as well: • xlab - X axis label • ylab - Y axis label • legend.location - where to put the legend • total.col - color of ensemble Hilbert periodogram • total.lwd - lwd of ensemble Hilbert periodogram • total.lty - lty of ensemble Hilbert periodogram • imf.cols - colors of IMF periodogram • imf.lwd - lwds of IMF periodogram • imf.lty - ltys of IMF periodogram • fourier.col - color of Fourier periodogram • fourier.lwd - lwd of Fourier periodogram • fourier.lty - lty of Fourier periodogram • main - figure title

Details

This function plots the Hilbert periodogram of a signal, with options to show periodograms of individual IMFs. You can also plot a simple Fourier periodogram for comparison.

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

See Also

HHSpectrum, HHGramImage

Examples

```
#Here we see how the EMD produces a dyadic filter bank for uniform random noise
#The frequency distributions of all but the first IMF display a Chi-Square distribution
#See Huang, N. E. & Wu, Z.
#A review on Hilbert-Huang Transform: Method and its applications to geophysical studies.
#Reviews of Geophysics, 2008, 46, RG2006

#The EMD of this signal may take a couple of minutes to run

set.seed(628)
sig  <-  runif(10000)
tt  <-  seq_len(length(sig)) * 0.01

## Not run: emd.result  <-  Sig2IMF(sig, tt)

dfreq  <-  0.1
## Not run: hspec  <-  HHSpectrum(emd.result, dfreq)

## Not run: HHSpecPlot(hspec, show.imfs = TRUE,
imf.list = 1:10, show.total = TRUE, scaling = "sqrt",
imf.lwd = rep(2, 10), total.lty = 3)
## End(Not run)
```

HHSpectrum *Generate Hilbert spectrum*

Description

Generates a Hilbert periodogram from the results of Sig2IMF and EEMD.

Usage

```
HHSpectrum(hres, dfreq, freq.span = NULL, time.span = NULL,
    scaling = "none", verbose = TRUE)
```

Arguments

hres	This is the output generated by EEMDCompile or EEMDResift
dfreq	Frequency resolution of spectrum
time.span	Time span to render spectrum over; NULL means over the whole time series
freq.span	Frequency span to include in spectrum; NULL means render all the frequencies in the time series
scaling	If "log", render a log10 frequency spectrum. Defaults to "none" (linear).
verbose	If TRUE, print progress messages

Details

HHSpectrum sums Hilbert spectral data over the time domain to produce the equivalent of a periodogram. The result can be plotted using HHSpecPlot.

Value

hspec	A data structure containing the spectrum of each IMF.

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

See Also

HHRender, HHSpecPlot

Examples

```
## Not run:
data(PortFosterEvent)

emd.result <- Sig2IMF(sig, tt)

dfreq <- 0.1
hspec <- HHSpectrum(emd.result, dfreq)
HHSpecPlot(hspec, show.fourier = TRUE, scale.fourier = TRUE)


## End(Not run)
```

HHTPackagePlotter *Set up spectrogram figure*

Description

Sets up the figure window for HHGramImage and FTGramImage. This is an internal function and will likely never be called by a user

Usage

```
HHTPackagePlotter(img, trace, amp.span, img.x.lab, img.y.lab,
fit.line = NULL, window = NULL, colormap = NULL, backcol = c(0, 0, 0),
pretty = FALSE, grid = TRUE, colorbar = TRUE, opts = list())
```

Arguments

img	Fourier or Hilbert spectrogram image.
trace	Time series corresponding to the spectrogram.
amp.span	Amplitudes over which to plot.
img.x.lab	Specifies the X axis label on the image part of the figure, defaults to "time"
img.y.lab	Specifies the Y axis label on the image part of the figure, defaults to "frequency"
fit.line	Plots a line corresponding to the IMF sum on the trace, if requested
window	The Fourier window length, if applicable
colormap	The image color map
backcol	The background color of the image (what shows up for pixels with value NA)
pretty	Adjusts image axes to have nice values, see the pretty function in the base package included in R
grid	Determines whether to plot grid lines on the spectrogram
colorbar	Whether to plot a color bar for amplitude values
opts	Other possible options passed from HHGramImage and FTGramImage

Value

INTERNAL

Author(s)

Daniel Bowman <danny.c.bowman@gmail.com>

HilbertEnvelope *Instantaneous amplitude*

Description

Generates the instantaneous amplitude of an analytic signal given by HilbertTransform

Usage

```
HilbertEnvelope(asig)
```

Arguments

asig	The analytic signal returned by HilbertTransform

Value

envelope	Instantaneous amplitude

Author(s)

Daniel C. Bowman <danny.c.bowman@gmail.com>

See Also

HilbertTransform, InstantaneousFrequency

Examples

```
tt <- seq(1000) * 0.01
sig <- sin(4 * pi * tt) + sin(3.4 * pi * tt)
asig <- HilbertTransform(sig)
env <- HilbertEnvelope(asig)
plot(tt, sig, type = "l")
lines(tt, env, col = "red")
lines(tt, -env, col = "red")
```

HilbertTransform *The Hilbert transform*

Description

Creates the analytic signal using the Hilbert transform.

Usage

```
HilbertTransform(sig)
```

Arguments

sig	Signal to transform.

Details

Creates the real and imaginary parts of a signal.

Value

asig	Analytic signal

Author(s)

Daniel C. Bowman <danny.c.bowman@gmail.com>

See Also

HilbertEnvelope, InstantaneousFrequency

Examples

```
tt <- seq(1000) * 0.01
sig <- sin(pi * tt)
asig <- HilbertTransform(sig)
plot(tt, sig, xlim = c(0, 12))
lines(tt, Re(asig), col = "green")
lines(tt, Im(asig), col = "red")
legend("topright", col = c("black", "green", "red"),
lty = c(NA, 1, 1), pch = c(1, NA, NA),
legend = c("Signal", "Real", "Imaginary"))
```

`InstantaneousFrequency` *Derive instantaneous frequency*

Description

Calculates instantaneous frequency from an analytic signal.

Usage

```
InstantaneousFrequency(asig, tt, method = "arctan", lag = 1)
```

Arguments

`asig`	Analytic signal produced by `HilbertTransform`
`tt`	Sample times
`method`	How the instantaneous frequency is calculated. `"arctan"` uses the arctangent of the real and imaginary parts of the Hilbert transform, taking the numerical derivative of phase for frequency. `"chain"` uses the analytical derivative of the arctangent function prior to performing the numerical calculation.
`lag`	Differentiation lag, see the `diff` function in the `base` package.

Value

`instfreq`	Instataneous frequency in 1/time

Note

The `"arctan"` method was adapted from the `hilbertspec` function in the `EMD` package.

!!IMPORTANT!! The numeric differentiation may be unstable for certain signals. For example, high frequency sinusoids near the Nyquist frequency can give inaccurate results when using the `"chain"` method. When in doubt, use the `PrecisionTester` function to check your results!

Author(s)

Daniel C. Bowman <danny.c.bowman@gmail.com>

See Also

`PrecisionTester`

PlotIMFs *Display IMFs*

Description

This function displays IMFs generated using Sig2IMF, EEMDCompile. or EEMDResift

Usage

```
PlotIMFs(sig, time.span = NULL, imf.list = NULL, original.signal = TRUE,
    residue = TRUE, fit.line = FALSE, lwd = 1, cex = 1, ...)
```

Arguments

sig	Data structure returned by Sig2IMF, EEMDCompile, or EEMDResift.
time.span	Time span over which to plot IMFs. NULL will draw the entire signal.
imf.list	Which IMFs to plot, NULL plots all of them.
original.signal	whether or not to plot the original signal.
residue	whether to plot the residue of the EMD method.
fit.line	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 EEMDCompile.

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 <danny.c.bowman@gmail.com>

See Also

HHGramImage

Examples

```
data(PortFosterEvent)

#Run EMD
emd.result <- Sig2IMF(sig, tt, sm = "polynomial")

#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

PlotIMFs(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
PlotIMFs(emd.result, time.span, imf.list, original.signal, residue, fit.line)
```

PrecisionTester *Test numerically determined instantaneous frequency against exact instantaneous frequency*

Description

This function compares the performance of InstantaneousFrequency against signals of known instantaneous frequency. The known signal is of the form

$$x(t) = a\sin(\omega_1 + \varphi_1) + b\sin(\omega_2 + \varphi_2) + c$$

One can create quite complicated signals by choosing the various amplitude, frequency, and phase constants.

Usage

```
PrecisionTester(tt = seq(0, 10, by = 0.01), method = "arctan", lag = 1,
    a = 1, b = 1, c = 1, omega.1 = 2 * pi, omega.2 = 4 * pi,
    phi.1 = 0, phi.2 = pi/6, plot.signal = TRUE,
    plot.instfreq = TRUE, plot.error = TRUE, new.device = TRUE, ...)
```

Arguments

tt	Sample times.
method	How the numeric instantaneous frequency is calculated, see InstantaneousFrequency
lag	Differentiation lag, see the diff function in the base package.
a	Amplitude coefficient for the first sinusoid.
b	Amplitude coefficient for the second sinusoid.

c	DC shift
omega.1	Frequency of the first sinusoid.
omega.2	Frequency of the second sinusoid.
phi.1	Phase shift of the first sinusoid.
phi.2	Phase shift of the second sinusoid.
plot.signal	Whether to show the time series.
plot.instfreq	Whether to show the instantaneous frequencies, comparing the numerical and analytical result.
plot.error	Whether to show the difference between the numerical and analytical result.
new.device	Whether to open each plot as a new plot window (defaults to TRUE). However, Sweave doesn't like dev.new(). If you want to use PrecisionTester in Sweave, be sure that new.device = FALSE
...	Plotting parameters.

Value

instfreq$sig	The time series
instfreq$analytic	The exact instantaneous frequency
instfreq$numeric	The numerically-derived instantaneous frequency from InstantaneousFrequency

Author(s)

Daniel C. Bowman <danny.c.bowman@gmail.com>

See Also

InstantaneousFrequency

Examples

```
#Simple signal
tt <- seq(0, 10, by = 0.01)
a <- 1
b <- 0
c <- 0
omega.1 <- 30 * pi
omega.2 <- 0
phi.1 <- 0
phi.2 <- 0
PrecisionTester(tt, method = "arctan", lag = 1, a, b, c,
    omega.1, omega.2, phi.1, phi.2)

#That was nice - what happens if we use the "chain" method...?

PrecisionTester(tt, method = "chain", lag = 1, a, b, c,
```

```
    omega.1, omega.2, phi.1, phi.2)

#Big problems!  Let's increase the sample rate

tt <- seq(0, 10, by = 0.0005)
PrecisionTester(tt, method = "chain", lag = 1, a, b, c,
omega.1, omega.2, phi.1, phi.2)

#That's better

#Frequency modulations caused by signal that is not symmetric about 0

tt <- seq(0, 10, by = 0.01)
a <- 1
b <- 0
c <- 0.25
omega.1 <- 2 * pi
omega.2 <- 0
phi.1 <- 0
phi.2 <- 0

PrecisionTester(tt, method = "arctan", lag = 1, a, b, c,
omega.1, omega.2, phi.1, phi.2)

#Non-uniform sample rate
set.seed(628)
tt <- sort(runif(500, 0, 10))
a <- 1
b <- 0
c <- 0
omega.1 <- 2 * pi
omega.2 <- 0
phi.1 <- 0
phi.2 <- 0

PrecisionTester(tt, method = "arctan", lag = 1, a, b, c,
omega.1, omega.2, phi.1, phi.2)
```

`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 `tt`. It shows one of several thousand transitory seismic events occurring in Port Foster (the flooded caldera of the volcano).

Usage

```
data(PortFosterEvent)
```

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, tt, spectral.method = "arctan", diff.lag = 1, stop.rule = "type5",
    tol = 5, boundary = "wave", sm = "none", smlevels = c(1), spar = NULL,
    max.sift = 200, max.imf = 100, interm = NULL)
```

Arguments

`sig`	a time series to be decomposed (vector)
`tt`	A vector of sample times for `sig`
`spectral.method`	defines how to calculate instantaneous frequency - whether to use the arctangent of the analytic signal with numeric differentiation ("arctan") or the result of the chain rule applied to the arctangent, then numerically differentiated ("chain"); see `InstantaneousFrequency`.
`diff.lag`	specifies if you want to do naive differentiation (`diff.lag = 1`), central difference method (`diff.lag = 2` or higher difference methods (`diff.lag > 2`) to determine instantaneous frequency; see `InstantaneousFrequency`.
`stop.rule`	As quoted from the EMD package documentation: "The stop rule of sifting. The type1 stop rule indicates that absolute values of envelope mean must be less than the user-specified tolerance level in the sense that the local average of upper and lower envelope is zero. The stopping rules type2, type3, type4 and type5 are the stopping rules given by equation (5.5) of Huang et al. (1998), equation (11a), equation (11b) and S stoppage of Huang and Wu (2008), respectively."
`tol`	Determines what value is used to stop the sifting - this will depend on which stop rule you use.
`boundary`	how the beginning and end of the signal are handled
`sm`	Specifies how the signal envelope is constructed, see Kim et al, 2012.
`smlevels`	Specifies what level of the IMF is obtained by smoothing other than interpolation, see EMD package documentation

spar	User-defined smoothing parameter for spline, kernel, or local polynomial smoothing.
max.sift	How many sifts are allowed - if this value is exceeded the IMF is returned as-is.
max.imf	Maximum number of IMFs allowed.
interm	Specifies vector of periods to be excluded from IMFs to cope with mode mixing.

Details

This function configures and performs empirical mode decomposition using the emd function in the EMD package.

Value

emd.result	The intrinsic mode functions (IMFs), instantaneous frequencies, and instantaneous amplitudes of sig.

References

Kim, D., Kim, K. and Oh, H.-S. (2012) Extending the scope of empirical mode decomposition by smoothing. *EURASIP Journal on Advances in Signal Processing*, **2012**, 168.

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, PlotIMFs

Examples

```
data(PortFosterEvent)

#Run EMD
emd.result=Sig2IMF(sig, tt)

#Display IMFs

time.span <- c(5, 10)
imf.list <- 1:3
original.signal <- TRUE
residue <- TRUE

PlotIMFs(emd.result, time.span, imf.list, original.signal, residue)

#Plot spectrogram
sdt <- tt[2] - tt[1]
dfreq <- 0.25
freq.span <- c(0, 25)
```

```
hgram <- HHRender(emd.result, sdt, dfreq, freq.span = freq.span, verbose = FALSE)

time.span <- c(4, 10)
freq.span <- c(0, 25)
amp.span <- c(0.000001, 0.00001)
HHGramImage(hgram, time.span = time.span,
freq.span = freq.span, amp.span = amp.span,
pretty = TRUE)
```

---

tt *Ocean Bottom Seismometer Sample Rate*

Description

This is the sample times for the instrument that recorded sig.

Usage

```
data(PortFosterEvent)
```

Format

A vector describing the sample times. The sample rate was constant at 125 samples per second.

Source

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

Index