Package 'DecomposeR'

June 1, 2020

Type Package

Title Empirical Mode Decomposition for Cyclostratigraphy

Version 1.0.2

Author Sebastien Wouters [aut, cre]

Maintainer Sebastien Wouters <sebastien.wouters@doct.uliege.be>

Description Tools to apply Ensemble Empirical Mode
Decomposition (EEMD) for cyclostratigraphy purposes. Mainly: a new
algorithm, extricate, that performs EEMD in seconds, a linear interpolation
algorithm using the greatest rational common divisor of depth or time,
different algorithms to compute instantaneous amplitude, frequency and
ratios of frequencies, and functions to verify and visualise the outputs.
The functions were developed during the CRASH project (Checking the
Reproducibility of Astrochronology in the Hauterivian). When using for
publication please cite Wouters, S., Da Silva, A.C. Crucifix, M., Sinnesael,
M., Zivanovic, M., Boulvain, F., Devleeschouwer, X., 2019, ``Litholog
generation with the 'StratigrapheR' package and signal decomposition for
cyclostratigraphic purposes''. Geophysical Research Abstracts Vol. 21,
EGU2019-5520, 2019, EGU General Assembly 2019.
<http://hdl.handle.net/2268/234402>.

License GPL-3

Depends R (>= 3.6.0)

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.1.0

Imports graphics, stats, utils, StratigrapheR (>= 1.0.1), hht, grid,
hexbin, colorRamps, dplyr (>= 1.0.0)

Suggests EMD, astrochron, tidyverse

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-01 17:10:02 UTC

R **topics documented:**

---

as.emd *Create / Check emd objects*

Description

Allows to convert the result of a decomposition into a standard list. The warnings of the is.emd checking function allow to identify the problems.

Usage

```
as.emd(
  xy,
  dt,
  imf,
  residue = NULL,
  ini = NULL,
  mode = NULL,
  repl = 1,
  order = NA
)

is.emd(emd)
```

Arguments

xy	a vector of length n for the original signal at each dt
dt	a vector of length n for the depth or time reference
imf	a data.frame or matrix of n rows of the IMFs
residue	a vector of length n for the residue of the decomposition
ini	an optional vector of length n of the eventual initial Intrinsic Mode Function xy would be a demodulation of, if it is a demodulation.
mode	the mode sequence index to give to each replicated IMFs
repl	the id of each replicates. The length of unique(repl) defines the amount of replicates.
order	the order of the imf, typically from higher frequency to lower frequency
emd	an emd object to check

Value

a list made of $xy (original signal), $dt (depth/time), $m (a matrix of the decomposition), $repl (the replicate id of each point) and $mode (the mode id of each point).

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

s30  <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)
s240 <- 2 * sin(t*2*pi/p2)
sn   <- rnorm(n, sd = 0.5)
```

```
xy <- s30 + s240 + sn

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt  <- cumsum(inter_dt)

dec <- as.emd(xy = xy, dt = dt, imf = matrix(c(sn, s30, s240), ncol = 3))

plot_emd(dec, pdf = FALSE)

is.emd(dec)


dec$xy <- 1
is.emd(dec)
```

---

as.pulse   *Create / Check pulse objects*

Description

Allows to convert instantaneous frequency determination results into a single 'pulse' object. This is the format generated by inst.pulse (and gzc if output = 2)

Usage

```
as.pulse(
  dt,
  f,
  a = NULL,
  m = NULL,
  idt = NULL,
  mode = NULL,
  repl = 1,
  order = NA
)

is.pulse(pulse)
```

Arguments

dt	a vector of length n for the depth or time reference
f	a data.frame or matrix of n rows of the instantaneous frequencies
a	a data.frame or matrix of n rows of the instantaneous amplitudes
m	a data.frame or matrix of n rows of the components from which the frequencies and amplitudes were computed from

idt	data.frame or matrix of n rows of identity tuning: new dt coordinates to remove the frequency modulation
mode	the mode sequence index to give to each replicated IMFs
repl	a vector for the number of replicates or a matrix, indicating in which replicate set each point is
order	the order of the imf, typically from higher frequency to lower frequency
pulse	a pulse object to check

Value

a list made of $dt (depth/time), $f (instantaneous frequency), $a (instantaneous amplitude) if a is provided, $repl (the replicate id of each point) and $mode (the mode id of each point).

Examples

```
set.seed(42)

n <- 600
dt <- seq_len(n)

p1 <- 30
p2 <- 240

s30  <- (1 + 0.6 * sin(dt*2*pi/p2)) * sin(dt*2*pi/p1)
s240 <- 2 * sin(dt*2*pi/p2)

xy <- s30 + s240

dec <- as.emd(xy = xy, dt = dt, imf = matrix(c(s30, s240), ncol = 2))

plot_emd(dec, pdf = FALSE, style = 1)

pulse <- inst.pulse(dec, last = TRUE, breaks = 200, bins = 40, cut = 10)

is.pulse(pulse)

simp.pulse <- as.pulse(pulse$dt, pulse$f)

str(simp.pulse)
```

---

condense	*Condenses columns of matrix*

Description

Condenses columns of a matrix by averaging or summing them. The condensing can be done partially: a multiple of the repetitions can be averaged or summed to keep some repetitions.

Usage

```
condense(m, n, fun = "mean")
```

Arguments

m	matrix of repeated signal, each column being a repetition
n	the number of repetitions that will be averaged/summed
fun	the function to apply to each repetition: "mean" or "sum".

Value

a matrix with n times less columns

Examples

```
m <- matrix(rep(seq(100, 800, 100), each = 10) + rep(1:10, 8), ncol = 8)

m

condense(m, 4)
```

DecomposeR   *DecomposeR: Empirical Mode Decomposition for Cyclostratigraphy*

Description

This package provides tools to apply Ensemble Empirical Mode Decomposition (EEMD) for cyclostratigraphy purposes. It proposes a new algorithm, that performs EEMD in seconds, a linear interpolation algorithm using the greatest rational common divisor of depth or time, different algorithms to compute instantaneous amplitude, frequency and ratios of frequencies, and functions to verify and visualise the outputs.

Details

Package: DecomposeR

Type: R package

Version: 1.0.2 (update to dplyr 1.0.0)

License: GPL-3

Note

If you want to use this package for publication or research purposes, please cite Wouters, S., Da Silva, A.C. Crucifix, M., Sinnesael, M., Zivanovic, M., Boulvain, F., Devleeschouwer, X., 2019, Litholog generation with the StratigrapheR package and signal decomposition for cyclostratigraphic purposes. Geophysical Research Abstracts Vol. 21, EGU2019-5520, 2019, EGU General Assembly 2019. <http://hdl.handle.net/2268/234402>

Author(s)

Sebastien Wouters

Maintainer: Sebastien Wouters <sebastien.wouters@doct.uliege.be>

DecomposeR.Datasets	*Datasets for Testing DecomposeR*

Description

Datasets for testing DecomposeR: the ace dataset is from from Sinnesael et al. (2016), the cip2 and cip3 data sets are from the signals 2 and 3 of the CIP project, respectively, and cip1 was derived from cip1_raw which is a rasterisation of the .tif image provided as signal 1 of the CIP project.

Details

dt Values of the signal

xy Depth or time of the signal

References

Sinnesael, M., Zivanovic, M., De Vleeschouwer, D., Claeys, P. & Schoukens, J. (2016). Astronomical component estimation (ACE v.1) by time-variant sinusoidal modeling. Geoscientific Model Development. 9. 3517-3531. https://doi.org/10.5194/gmd-9-3517-2016.

---

dq.algorithm	*Calculates instantaneous frequency of freqeuncy carriers using the DQ method*

Description

Calculates instantaneous frequency of frequency carriers using the direct quadrature method from Huang et al., 2009.

Usage

```
dq.algorithm(fc, dt)
```

Arguments

fc	a matrix of amplitude between -1 and 1, making up the frequency carrier
dt	a vector of depth or time values

Value

a list of the depth/time (dt), frequency (f), and identity tuning (idt), i.e. depths adapted to transform the frequency carrier into a cosine of period 1.

References

Huang, Norden E., Zhaohua Wu, Steven R. Long, Kenneth C. Arnold, Xianyao Chen, and Karin Blank. 2009. "On Instantaneous Frequency". Advances in Adaptive Data Analysis 01 (02): 177–229. https://doi.org/10.1142/S1793536909000096.

Examples

```
n <- 600

t <- seq_len(n)

p1 <- 30

xy <- sin(t*2*pi/p1 + 50)

int <- c(rep(1, 99 + 100), seq(1,3,2/100), seq(3,1,-2/100), rep(1,100 + 99))

dt <- cumsum(int)

cond <- dt < 75

xy <- xy[!cond]
dt <- dt[!cond]/1.2 - 62.5

res <- dq.algorithm(xy, dt)

opar <- par("mfrow")

par(mfrow = c(3,1))

plot(dt, xy, type = "o", pch = 19, main = "Frequency carrier")

plot(dt, 1/res$f, pch = 19, type = "l", log = "y", lwd = 2, ylim = c(25,80),
     main = "Period (Direct Quadrature method)", ylab = "Period")

plot(res$idt[,1], xy, type = "o", pch = 19,
     main = "Identity tuning", axes = FALSE, ylab = "xy", xlab = "dt")

ap <- approx(x = dt, y = res$idt[,1], xout = seq(0,600, by = 20))

axis(1, at = ap$y, labels = ap$x)
axis(2)
box()

par(mfrow = opar)
```

extremist	*Gives local extrema and zero crossings intervals*

Description

Gives local minimas, maximas and zero crossings. Optimised for large data sets; the sky is the limit (and by the sky I mean the ability of R and your computer to memorise large data sets; but within this limit the algorithm can handle millions of points quickly).

Usage

```
extremist(xy, bound = FALSE, local = TRUE, zc = TRUE)
```

Arguments

xy	the values where to find the local extremas
bound	whether to consider the first and last points as both minima and maxima, for special purposes. Default is F, has it should be.
local	whether to consider the first and last points as local minima and maxima, if TRUE by default, otherwise these first and last points will be ignored
zc	whether to return the zero crossings

Value

a list of the indexes of the left (l) and right (r) boundaries for the minima (minindex), maxima (maxindex) and zero crossing (cross), along with the number of extrema and zero crossings

Examples

```
# Function script ----

xy <- c(1,0,0,0,4,5,5,0.5,-0.5,0.5,0,2,2,1,-1,-1,1,1,0,0,-4,-2,2,1,0,0.5,0,
        NA, 0.5,0,-0.5,3,2,3,0,0.5,4,4,0)

impressme <- 0 # Increase up to 5 or 6 to be impressed (bugs if your system
               # can't handle the size of the data).
               # If you increase it, do not run the plot script.

xy <- rep(xy, round(10^impressme))

print(paste("You are running ", length(xy), " points", sep = ""))

res <- extremist(xy)



# Plot script: do not run if you increase the impressme parameter ----

mini <- unique(c(res$minindex[[1]], res$minindex[[2]]))
maxi <- unique(c(res$maxindex[[1]], res$maxindex[[2]]))
zeri <- unique(c(res$cross[[1]], res$cross[[2]]))

l <- length(xy)
```

```
opar <- par("mfrow")

par(mfrow = c(3,1))

plot(1:l, xy, type = "o",pch = 19)
points(mini, xy[mini], pch = 19, col = "blue")

plot(1:l, xy, type = "o",pch = 19)
points(maxi, xy[maxi], pch = 19, col = "red")

plot(1:l, xy, type = "o",pch = 19)
points(zeri, xy[zeri], pch = 19, col = "green")
abline(h = 0, col = "grey")

par(mfrow = opar)
```

---

extricate   *Extricate a signal: an EEMD algorithm*

Description

Performes EEMD

Usage

```
extricate(
  xy,
  dt,
  nimf,
  ini = NULL,
  repl = 1,
  comb = 100,
  mirror_noise = TRUE,
  factor_noise = 3,
  unit_noise = "1stdiff",
  sifting = 10,
  output_sifting = FALSE,
  remove = "trend",
  bind = FALSE,
  speak = FALSE,
  plot_process = FALSE,
  pdf = TRUE,
  name = "extricate",
  ext = ".pdf",
  dir = tempdir(),
  width = 10,
  height = 20,
```

```
    track = TRUE,
    openfile = TRUE
)
```

Arguments

xy	signal, detrended and demeaned, maybe linearly interpolated to have regular sampling interval
dt	depth/time
nimf	number of modes/components/intrinsic mode functions to decompose the signal into
ini	an optional vector of length n of the eventual initial Intrinsic Mode Function xy would be a demodulation of, if it is a demodulation. In that case the mode indexes will start at 2.
repl	the amount of decompositions to output
comb	the amount of decompositions each output decomposition will be a combination of. Has to be a multiple of 2 (even and odd extension stacks have to be combined in any case)
mirror_noise	whether to generate a mirrored noise signal (for even and odd extension) that will cancel perfectly when combining the decompositions
factor_noise	a factor for the amplitude of white noise (finite amplitude obtained via [runif](#)). By default it will be multiplied with the mean of the lagged-one difference to define the noise amplitude
unit_noise	whether to multiply factor_noise by the mean of the lagged-one difference (unit_noise = "1stdiff") or not (unit_noise = "native")
sifting	amount of iterations of the sifting process
output_sifting	whether to output each sifting
remove	whether to remove the linear trend (remove = "trend") or the mean (remove = "mean") prior to decomposition. The removed part will be added back after the decomposition. If remove is anything else, nothing will be removed, which can be problematic for the even and odd extension scheme used.
bind	whether to bind the removed trend or mean to the last component (T), or to add it as another component (F)
speak	whether to print a sentence at each sifting: it gives the stack (even or odd), the mode number and sifting number
plot_process	whether to have a plot of the entire sifting process. This slows down the algorithm, use with low 'repl' and 'comb' values for visualisation purposes
pdf	whether the plot be directly set as a pdf file
name, ext, dir, width, height, track, openfile	
	arguments to provide to pdfDisplay if plot_process and pdf are TRUE

Value

a list made of $xy (original signal), $dt (depth/time), $m (a matrix of the decomposition), $repl (the replicate id of each point) and $mode (the mode id of each point). If output_sifting is TRUE, additional $even_sifting and $odd_sifting data.tables are provided, giving the condensed siftings for the even and odd extensions.

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7, repl = 1, comb = 40, factor_noise = 10,
                 speak = TRUE, output_sifting = TRUE)

integrity(xy, dec)

parsimony(dec)

plot_emd(dec, select = c(4, 6), pdf = FALSE)


plot_emd(dec, li = list(v = 0), dir = tempdir())
```

---

gzc   *Calculates instantaneous frequency using the GZC method*

Description

Calculates instantaneous frequency using the Generalised Zero-Crossing method from Huang et al., 2009. General wrapper for the `gzc.algorithm` function that does all the actual work.

Usage

```
gzc(
  emd = NULL,
  ini = NULL,
```

```
    m = NULL,
    dt = NULL,
    repl = 1,
    mode = NULL,
    dtout = NULL,
    output = 1,
    warn = TRUE
)
```

Arguments

emd	emd-type object
ini	an optional vector of length n of the eventual initial Intrinsic Mode Function xy would be a demodulation of, if it is a demodulation. It will be integrated to the results as mode 1.
m	a matrix of the amplitude values (xy) of the components, each column being a component. Each column should have the same number of non NA values. Vectors, for 1 component, are accepted. Is overridden by emd.
dt	the depth or time value. Is overridden by emd.
repl	the amount of replicates in m. Is overridden by emd.
mode	the mode sequence index to give to each replicated IMFs
dtout	the dt values to sample the frequency and amplitude from if output = 2.
output	the style of the output, whether 0, 1 or 2. 0 provides the raw output of gzc.algorithm, 1 and 2 provides a matrix with $dt (depth/time), $f (frequency) and $a ()amplitude, but with output = 1 the matrix provides the dt only at the extremas and zero-crossings, whereas with output = 2 the dt values are the ones provided with the dtout parameter. 1 is better for plots, 2 allows easier calculations to be performed downstream.
warn	whether to warn if the sampling interval defined by the dtout parameter is to small (redirected from StratigrapheR::tie.lim)

Value

depending on the output parameter:

output = 0 provides the raw output of gzc.algorithm, with $ldt and $rdt (the left and right boundaries of the depth/time intervals), $f (frequency) and $a (amplitude). To that are added $repl (the replicate id) and $mode (the mode id)

output = 1 or 2 provides a matrix with $dt, $f and $a, but with output = 1 the matrix provides the dt only at the extremas and zero-crossings, whereas with output = 2 the dt values are the ones provided with the out parameter. 1 is better for plots, 2 allows easier calculations to be performed downstream.

References

Huang, Norden E., Zhaohua Wu, Steven R. Long, Kenneth C. Arnold, Xianyao Chen, and Karin Blank. 2009. "On Instantaneous Frequency". Advances in Adaptive Data Analysis 01 (02): 177–229. https://doi.org/10.1142/S1793536909000096.

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)
dec <- extricate(xy, dt, nimf = 7, repl = 1, comb = 50,
                 factor_noise = 10, speak = TRUE)


plot_emd(dec, dir = tempdir())

integrity(xy, dec)
parsimony(dec)

res <- gzc(dec)

numb <- 4

opar <- par('mfrow')

par(mfrow = c(1,2))

plot(dec$m[,numb], dec$dt, type = "l",
     main = paste("Mode", numb, " + Amplitude"),
     xlab = "xy", ylab = "dt", ylim = c(0, 600))
lines(res$a[,numb], res$dt[,numb], col = "red", lwd = 2)

plot(1/res$f[,numb], res$dt[,numb], ylim = c(0,600),
     xlab = "Period", ylab = "dt", log = "x",
     type = "l", col = "red", lwd = 2, main = "Period")

par(mfrow = opar)
```

---

gzc.algorithm   *Calculates instantaneous frequency of simplified IMF using the GZC method*

Description

Calculates instantaneous frequency of simplified IMF using the Generalised Zero-Crossing method from Huang et al., 2009.

Usage

```
gzc.algorithm(xy, dt)
```

Arguments

xy	a matrix of amplitude
dt	a vector of depth or time values

Details

the GZC method is precise to 1/4th of a period, so the results are provided between left and right points, i.e. either an extrema or a zero-crossing.

Value

a list of $ldt (left position), $rdt (right position), $f (frequency) and $a (amplitude)

References

Huang, Norden E., Zhaohua Wu, Steven R. Long, Kenneth C. Arnold, Xianyao Chen, and Karin Blank. 2009. 'On Instantaneous Frequency'. Advances in Adaptive Data Analysis 01 (02): 177–229. https://doi.org/10.1142/S1793536909000096.

Examples

```
xyi  <- c(0.5,0,-0.5,0,0.5,0,-0.5,0,0.5,0,-0.5,0,0.5,0,-0.5,0,0.5,0,-0.5,0,
          1,1,0,-1,-1,0,1,1,0,-1,-1,0,1,1,0,-1,-1)

dti <- 1:length(xyi)

d <- simp.emd(m = xyi, dt = dti)

xy <- d$xy
dt <- d$dt

res <- gzc.algorithm(xy, dt)

opar <- par('mfrow')

par(mfrow = c(2,1))

plot(dti, xyi, pch = 19, type = "o", ylab = "xy", xlab = "dt")
points(dt, xy, pch = 19, col = "green")
points(res$ldt, res$a, pch = 19, col = "red")
points(res$rdt, res$a, pch = 19, col = "red")
```

```
plot(dt, rep(max(res$f, na.rm = TRUE), length(dt)), type = "n",
     ylab = "Frequency", xlab = "dt",
     ylim = c(0, 2 * max(res$f, na.rm = TRUE)))
points(res$ldt, res$f, pch = 19)
points(res$rdt, res$f, pch = 19)

par(mfrow = opar)
```

---

gzc.departure   *departure of instantaneous frequency to generalized zero-crossing*

Description

departure of instantaneous frequency to generalized zero-crossing of instantaneous freqeuncy. The departure is calculated as the exponential of the absolute difference of logarithms of frequencies obtained using a robust generalized zero-crossing method through the gzc function (where the components are simplified into extrema separated by zero-crossings) and instantaneous frequency computed from another method

Usage

```
gzc.departure(
  pulse = NULL,
  dt = NULL,
  m = NULL,
  f = NULL,
  repl = 1,
  mode = NULL,
  simplify = TRUE
)
```

Arguments

pulse	a pulse object object
dt	the depth or time. Is overridden by pulse.
m	a matrix of the modes to calculate the gzc frequency from. Is overridden by pulse.
f	a matrix of the frequencies to compare to gzc.
repl	the amount of replicates in m. Is overridden by emd.
mode	the mode sequence index to give to each replicated IMFs. Is overridden by emd.
simplify	whether to average the value for each component of each replicate

Value

If simplify is TRUE, the function returns the average gzc departure as a data frame where the columns stand for the modes and the rows for the replicates. If simplify if FALSE, the function returns the functions returns local gzc departure.

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec1 <- extricate(xy, dt, nimf = 5, repl = 1, comb = 10, sifting = 1,
                  factor_noise = 10, bind = TRUE, speak = TRUE)

dec2 <- extricate(xy, dt, nimf = 6, repl = 1, comb = 100, sifting = 5,
                  factor_noise = 50, bind = TRUE, speak = TRUE)


plot_emd(dec1, name = "EMD 1", dir = tempdir())
plot_emd(dec2, name = "EMD 2", dir = tempdir())

parsimony(dec1)
parsimony(dec2)

f1 <- inst.pulse(dec1, plot = FALSE)
f2 <- inst.pulse(dec2, plot = FALSE)

gzc.departure(f1)
gzc.departure(f2)
```

---

inst.pulse   *Computes instantaneous frequency using the Hilbert transform*

Description

Calculates instantaneous frequency using the Hilbert transform (HT), normalised Hilbert transform (NHT) or the direct quadrature (DQ) methods. Normalisation is done for NHT and DQ using Huang et al., 2009 algorithm, but the empirical normalisation scheme can fail due to overshoot or undershoot of the spline. Additional research is necessary for that last feature.

Usage

```
inst.pulse(
```

```
  emd = NULL,
  imf = NULL,
  m = NULL,
  dt = NULL,
  ini = NULL,
  repl = 1,
  mode = NULL,
  last = FALSE,
  plot = TRUE,
  method = "HT",
  delta = NULL,
  tolerance = 8,
  relative = TRUE,
  breaks = 500,
  bins = 100,
  cut = 18,
  lines = NULL
)
```

Arguments

emd	an emd object
imf	a matrix of same frequency modes to calculate the frequency from. Is overridden by emd. This allows to calculate and visualise the results for single IMFs more clearly than in a population plot.
m	a matrix of the modes to calculate the frequency from. Is overridden by emd and imf.
dt	the depth or time. Is overridden by emd.
ini	an optional vector of length n of the eventual initial Intrinsic Mode Function xy would be a demodulation of, if it is a demodulation. It will be integrated to the results as mode 1.
repl	the amount of replicates in m. Is overridden by emd.
mode	the mode sequence index to give to each replicated IMFs. Is overridden by emd.
last	whether to use the last mode (trend/residue).
plot	whether to have a plot summary of the output.
method	the IF calculation method: "HT" for Hilbert transform (default), "NHT" for normalised Hilbert transform, and "DQ" for direct quadrature. The two last require normalisation, which can sometimes fail.
delta, tolerance, relative	
	parameters to feed to [respace](#) for interpolation
breaks, bins, cut	
	parameter for the plots: breaks is fed to [plot_hist](#), bins is fed to [plot_hex](#), and cut defines the number of color cuts for [plot_hex](#). For better control use [plot_hist](#) and [plot_hex](#) directly.
lines	the period of lines to be added to the plots for better visualisation

Value

a list made of $dt (depth/time), $f (instantaneous frequency), $a (instantaneous amplitude),$repl (the replicate id of each point) and $mode (the mode id of each point)

References

Huang, Norden E., Zhaohua Wu, Steven R. Long, Kenneth C. Arnold, Xianyao Chen, and Karin Blank. 2009. "On Instantaneous Frequency". Advances in Adaptive Data Analysis 01 (02): 177–229. https://doi.org/10.1142/S1793536909000096.

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)
dec <- extricate(xy, dt, nimf = 7, repl = 10, comb = 10,
                 factor_noise = 10, speak = TRUE)


plot_emd(dec, dir = tempdir())

integrity(xy, dec)
parsimony(dec)

ht   <- inst.pulse(dec, lines = c(30, 240))
gzcr <- gzc(dec)

imf <- dec$m[,4]

inst.pulse(imf = imf, dt = dt, method = "DQ")
```

---

inst.ratio   *Computes instantaneous ratio of frequency*

Description

Computes instantaneous ratio of frequency

Usage

```
inst.ratio(
  pulse = NULL,
  dt = NULL,
  f = NULL,
  a = NULL,
  repl = 1,
  plot = TRUE,
  sqrt.rpwr = TRUE,
  style = "b",
  select = NA,
  bins = 100,
  cut = 18,
  lines = NULL,
  width = 10,
  height = 10,
  name = "Ratio",
  ext = ".pdf",
  dir = tempdir(),
  track = TRUE,
  openfile = TRUE
)
```

Arguments

pulse	a pulse object (created by inst.pulse for instance)
dt	depth/time. Is overridden by pulse.
f	instantaneous frequency. Is overridden by pulse.
a	instantaneous amplitude. Is overridden by pulse.
repl	number of replicates in f
plot	whether to plot an output
sqrt.rpwr, style, select, bins, cut, lines, width, height
	parameters to feed to plot_ratio for the plots
name, ext, dir, track, openfile
	parameters to feed to pdfDisplay in plot_ratio for pdf plot.

Value

a list of depth/time ($dt), frequency ($f), ratio of frequency ($ratio), if a is provided; the ratio power ($rpwr) i.e. the multiplication of the instantaneous amplitudes of the modes two by two, the replicates id ($repl)and id for the first and second frequency modes used for the ratio ($l for the first, $r for the second, or $lr for the two combined)

Examples

```
set.seed(42)
```

```
n    <- 600
time <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(time *2*pi/p2)) * sin(time *2*pi/p1)  +
  2 * sin(time *2*pi/p2) + rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7,
                 repl = 10, comb = 10, factor_noise = 10,
                 speak = TRUE)


plot_emd(dec, dir = tempdir())

integrity(xy, dec)
parsimony(dec)

ht    <- inst.pulse(dec, lines = c(30, 240))
ratio <- inst.ratio(ht, style = "s", lines = 8)
```

---

integrity   *Integrity of a decomposition*

Description

The function additions each component of a decomposition by depth/time, subtract it with the original signal, and provides the absolute of this subtraction. This is allows to verify if the decomposition is computed correctly.

The bulk value is the cumulated value of this proxy. If the decomposition is done right the value should be very small, but non-zero due to the floating-point arithmetics used by computers that generate tiny errors. Its actually interesting: the first computations of the orbital solutions were strongly affected by this error, as the chaotic behaviour of the equations enhanced the effect of these tiny tiny errors.

Usage

```
integrity(xy, emd = NULL, m = NULL, repl = 1, bulk = TRUE)
```

Arguments

xy   the signal

emd	an emd object to test. The emd$xy original signal is not used, to avoid confusion: you always have to provide the xy signal yourself.
m	a matrix with columns of same length that xy, made of the decomposition of the signal. Is overridden by emd.
repl	the replication of decompositions in m. Is overridden by emd.
bulk	whether to have a bulk value each decomposition replication, or for each dt of each replication

Value

a matrix with each column being a replication, or a list of bulk values for each replication

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7, repl = 10, comb = 10, factor_noise = 10,
                 speak = TRUE, output_sifting = TRUE)

integrity(xy, dec)
```

---

is.ratio	*Check ratio objects*

Description

Check ratio objects

Usage

```
is.ratio(ratio)
```

Arguments

ratio	a ratio object to check

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7,
                 repl = 10, comb = 10, factor_noise = 10,
                 speak = TRUE)

ht    <- inst.pulse(dec)
ratio <- inst.ratio(ht, plot = FALSE)

is.ratio(ratio)
```

---

is.simp.emd	*Tests for simplified EMD*

Description

Tests whether each column of a matrix is an alternation of -minima zero-crossing maxima zero-crossing-

Usage

```
is.simp.emd(xy)
```

Arguments

xy   a vector or matrix of values to test

Examples

```
xytest1 <- c(0.5, 1,-1,-0.85,-0.5,-1,-0.5,-1,1,0.5,0,-1,0,
             1,-1,0,1,2,-2,1,2,1,3,0,-1,-1,3,0)

xytest2 <- c(0, 1,-1,-0.85,-0.5,-1,-0.5,-1,1,0.5,0,0,
             1,1,1,1,2,-2,1,2,1,3,0,-1,-1,3,0)
```

```
dat1 <- simp.emd(m = xytest1, dt = 1:length(xytest1))

dat2 <- simp.emd(m = xytest2, dt = 1:length(xytest2))

is.simp.emd(dat1$xy)

is.simp.emd(dat2$xy)

# There is a problem when two maxima or minima are separeted by a point at 0
# that does not cross any further, creating a false simplified IMF. THis is
# not considered as a simplified IMF by this function. However this scenario
# should be very rare in EMDs, but you never really know.
```

---

mode.in   *Add / Remove / Bind modes in emd objects*

Description

Add / Remove / Bind modes in emd objects

Usage

```
mode.in(emd, xy, mode = NA, adjust = TRUE, name = "Added")

mode.out(obj, keep = NULL, lose = NULL, adjust = F, reorder = F)

mode.bind(emd, mode = NA, xy = NULL, adjust = T, name = "bound")
```

Arguments

emd	emd-type object
xy	an Instrinsic Mode Function to add
mode, keep, lose	
	[mode.in] the position where to add the mode / [mode.out] the modes to keep or lose / [mode.bind] the modes to merge
adjust	whether to adapt the initial signal of an emd object ($xy in the emd object) when adding or removing a mode
name	the name of the new mode
obj	emd or pulse type object
reorder	whether to reinitialise the index of modes when suppressing one

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7,
                 repl = 10, comb = 10, factor_noise = 10,
                 speak = TRUE)

opar <- par('mfrow')

par(mfrow = c(2,1))

integrity(xy, dec)

ht  <- inst.pulse(dec, plot = FALSE)

plot_hist(x = 1/ht$f, breaks = 500, id = ht$mode,
          xlog = TRUE, text = TRUE, xlab = "Period",
          main = "Initial Decomposition")

bound <- mode.bind(dec, mode = c(6,7))

ht2  <- inst.pulse(bound, plot = FALSE)

plot_hist(x = 1/ht2$f, breaks = 500, id = ht2$mode,
          xlog = TRUE, text = TRUE, xlab = "Period",
          main = "Binding of modes 6 and 7")

par(mfrow = opar)


plot_emd(bound, dir = tempdir(), adapt.axis = TRUE)
```

---

n.extrema   *Number of extrema/zero-crossings*

Description

Computes the number of extrema and zero-crossings for different groups of data, by their id or separated by NA values

Usage

```
n.extrema(
  xy,
  id = NULL,
  use.names = TRUE,
  bound = FALSE,
  local = FALSE,
  zc = TRUE
)
```

Arguments

xy	signal or decomposed signal
id	the id for different groups. If any NA value is in xy, it will also separate two groups of data
use.names	whether to use the names in id
bound, local, zc	
	parameters to feed to extremist

Value

a list of the number of minima ($n.min), maxima ($n.max), and, if zc = TRUE, zero-crossings ($n.cross)

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5)

xy <- xy - mean(xy)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7, repl = 1, comb = 40, factor_noise = 10,
                 speak = TRUE)
```

```
integrity(xy, dec)
parsimony(dec)

n.extrema(dec$m, dec$mode)

plot_emd(dec, select = c(6,8,9), pdf = FALSE, adapt.axis = TRUE)

plot_emd(dec, li = list(v = 0), adapt.axis = TRUE, dir = tempdir())
```

---

normalise *Empirical AM and FM decomposition*

Description

Applies the normalisation scheme of Huang et al., 2009 to decompose any Intrinsic Mode Functions obtained (usually via Empirical Mode Decomposition) into an Frequency Modulated component of amplitude 1, also called carrier, and its Amplitude Modulated enveloppe. The carrier can then be used to compute the instantaneous frequency via the Normalised Hilbert Transform (NHT) or by calculating its Direct Quadrature (DQ) (Huang et al., 2009). HOWEVER THIS FUNCTION CAN FAIL due to overshoot or undershoot of the spline fitting. Additional research is necessary.

Usage

```
normalise(emd = NULL, m = NULL, dt = NULL, repl = 1, last = TRUE, speak = TRUE)
```

Arguments

emd	an emd object
m	a matrix of the modes to calculate the amplitude and the frequency carrier from. Is overridden by emd.
dt	the depth or time. Is overridden by emd.
repl	the amount of replicates in m. Is overridden by emd.
last	whether to use the last mode (trend/residue).
speak	whether to print a sentence at each iteration

Value

a list of two matrices: $fc (frequency carrier) and $a (instantaneous amplitude)

References

Huang, Norden E., Zhaohua Wu, Steven R. Long, Kenneth C. Arnold, Xianyao Chen, and Karin Blank. 2009. 'On Instantaneous Frequency'. Advances in Adaptive Data Analysis 01 (02): 177–229. https://doi.org/10.1142/S1793536909000096.

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
        rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7,
               repl = 1, comb = 100, factor_noise = 10,
               speak = TRUE)

plot_emd(dec, pdf = FALSE, select = 4)

integrity(xy, dec)
parsimony(dec)

m  <- dec$m

res <- normalise(dt = dt, m = m, last = FALSE)

numb <- 4

opar <- par('mfrow')

par(mfrow = c(1,2))

plot(m[,numb], dt, type = "l", xlab = "xy",
     main = paste("Mode", numb, "and AM enveloppe"))
lines(res$a[,numb], dt, col = "red", lty = 5, lwd = 2)

plot(res$fc[,numb], dt, type = "l", xlab = "xy",
     main = "FM carrier")

par(mfrow = opar)
```

---

oscillate   *Modify a signal using a Van der Pol oscillator*

Description

Modify a signal using a Van der Pol oscillator

Usage

```
oscillate(
  xy,
  dt,
  period,
  delta = 0.05,
  damp = 5e-05,
  f.noise = 5,
  f.signal = 0.95,
  dx = function(x, y, beta, damp) beta * y - x * (x^2 + y^2 - 1) * damp,
  dy = function(x, y, beta, damp) -beta * x - y * (x^2 + y^2 - 1) * damp,
  xi = if (length(xy) != 0) xy[1] else 0.5,
  yi = if (length(xy) != 0) xy[1] else 0.5,
  normalise = TRUE,
  limit = TRUE
)
```

Arguments

xy	initial signal (vector or matrix)
dt	depth/time (same length than length/rows of xy)
period	the period of the oscillator (length 1 or n)
delta	the sampling interval for iteration (length 1 or n)
damp	damping parameter
f.noise	a factor of the amount of noise (length 1 or n)
f.signal	a factor of the amount of signal (length 1 or n)
dx, dy	the differentials used in the oscillator. They should be provided as functions needing x, y, beta (2*pi/period) and damp (damping) parameters
xi	the initial x value
yi	the initial y value
normalise	whether to recenter the output signal on the initial signal
limit	whether to warn when parameters are irrealistic (subjective)

Examples

```
set.seed(42)

n <- 800

dt <- seq(0,n, 1)

p1 <- 100
p2 <- 40

xy <- (1 + 0.6 * sin(dt*2*pi/p1)) * sin(dt*2*pi/p2)  + 2 * sin(dt*2*pi/p1) + 1
```

```
xyout <- oscillate(xy, dt, period = 30)

opar <- par("mfrow")

par(mfrow = c(1,1))

plot(xy, dt, type = "l",
     main = "Initial signal (bold) & oscillated signal (dashed)",
     lwd = 2, xlim = c(-4, 6))
lines(xyout, dt, type = "l", col = "grey50", lwd = 2, lty = 5)

par(mfrow = opar)
```

---

parsimony   *Parsimony of a decomposition*

Description

The function additions the absolute values of each component of a decomposition by depth/time, and computes the ratio of that with the absolute values of the signal. This is done either by depth/time or on the time/depth-cumulated signal (i.e. the bulk signal).

This is a proxy for parsimony: it is the factor of amplitude added by the decomposition. A perfect decomposition, that does not 'invent' wiggles, should approach 1, but will logically always be higher. However it is influenced by the absolute value of the initial signal: if the original signal is not centered around 0, the parsimony is not significative (it will artificially be closer to 1). To correct for that, the residue (part of the decomposition that is not centered around zero) has to be removed from the original signal.

Usage

```
parsimony(
  emd = NULL,
  xy = NULL,
  m = NULL,
  mode = NULL,
  repl = 1,
  bulk = TRUE,
  correct = NA
)
```

Arguments

emd	an emd object
xy	the signal
m	a matrix with columns of same length that xy, made of the decomposition of the signal

mode	the mode sequence index to give to each replicated IMFs
repl	the replication of decompositions in m
bulk	whether to have a bulk value each decomposition replication, or for each dt of each replication
correct	the modes to remove from the original signal and decomposition for a significative parsimony calculation. If NA, it removes the last mode, considered as the residue. Can be a vector of several integers, standing for the columns of m. If NULL, no mode is removed

Value

a matrix with each column being a replication, or a list of bulk values for each replication

Examples

```
set.seed(42)

n <- 500

dt <- seq_len(n)
xy <- rnorm(n, mean = 0, sd = 1) + 10

dec <- extricate(xy, dt, nimf = 7, comb = 10,
                 factor_noise = 1, speak = TRUE)

plot_emd(dec, dir = tempdir())

parsimony(dec, correct = NULL)

parsimony(dec)
```

---

pile.down   *Destacks a pile.up() signal*

Description

Destacks a signal stacked by [pile.up](#) by averaging each repetition back to n multiples.

Usage

```
pile.down(x, stack, even, n = length(unique(stack$id)) - 2)
```

Arguments

x	Treated signal
stack	Initial stack from which the x signal is from
even	Whether the x signal comes from even extension part of the initial stack (if FALSE, it would come from the odd extension part)
n	The multiple of destacking (has to be a multiple of n/2 (n being the parameter used in `pile.up`), in other words a multiple of length(unique(stack$id)) - 2 (minus 2 as the upper an lower extension are to be removed)

Value

a matrix or a vector of the destacked signal

Examples

```
set.seed(42)

n <- 200
t <- seq_len(n)

p1 <- 25
p2 <- 75

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)
inter_dt[20] <- 20

dt <- cumsum(inter_dt)

opar <- par()$mfrow
par(mfrow = c(1,1))

res <- pile.up(xy, dt, 4)

par(mfrow = c(2,1))
plot(res$ndt, res$even, type = "l", col = "blue")
plot(res$ndt, res$odd,  type = "l", col = "red")

par(mfrow = c(opar))

# Small number of repetitions ----

opar <- par("mfrow")
par(mfrow = c(1,2))

stack <- pile.up(xy, dt, 10)

signal <- stack$even + runif(length(stack$even), -3, 3)
```

```
res <- pile.down(signal, stack, even = TRUE, n = 5)

plot(xy, dt, type = "l", lwd = 2, main = "Low number of repetitions")
lines(res, dt,  type = "l", lty = 5, col = "red")

# High number of repetitions ----

stack <- pile.up(xy, dt, 1000)

signal <- stack$even + runif(length(stack$even), -3, 3)

res <- pile.down(signal, stack, even = TRUE, n = 500)

plot(xy, dt, type = "l", lwd = 2, main = "High number of repetitions")
lines(res, dt,  type = "l", lty = 5, col = "red")

par(mfrow = c(opar))
```

---

pile.up   *Repeat and stack a signal in central and line symmetry*

Description

Repeats and stacks a signal duplicated in central (even) and line (odd) symmetry to apply Ensemble Empirical Mode Decomposition (EEMD) on one single vector following the simple boundary rule of Zeng and He (2004). This allows to avoid the iterations that are typical of EEMD. A complete set of signal is added by default at the upper and lower part of the stack, to be removed in the end process.

Usage

```
pile.up(xy, dt, n, warn = TRUE)
```

Arguments

xy	the signal
dt	the depth/time positions of each xy
n	the number of replicates you want. It has to be a multiple of two, as you will generate two stacks: the even and the odd one.
warn	whether you want to be annoyed

Value

a dataframe of the original dt (odt), the stack-modified dt (ndt), the inversion factor to change the even stack into the odd one and vice-versa (invert), the even xy stack (even) and the odd one (odd)

Examples

```
set.seed(42)

n <- 200
t <- seq_len(n)

p1 <- 25
p2 <- 75

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)
inter_dt[20] <- 20

dt <- cumsum(inter_dt)

opar <- par()$mfrow
par(mfrow = c(1,1))

res <- pile.up(xy, dt, 4)

par(mfrow = c(2,1))
plot(res$ndt, res$even, type = "l", col = "blue")
plot(res$ndt, res$odd,  type = "l", col = "red")

par(mfrow = c(opar))
```

---

plot_emd	*Plot a decomposition*

Description

General plot for a complete decomposition (that can be summed back to the original signal)

Usage

```
plot_emd(
  emd = NULL,
  xy = NULL,
  ini = NULL,
  dt = NULL,
  m = NULL,
  mode = NULL,
  repl = 1,
  size.xy = 5,
  size.dt = 25,
  style = 2,
```

```
        xylim = NULL,
        dtlim = NULL,
        inilim = NULL,
        vertical = TRUE,
        adapt.axis = FALSE,
        adapt.last = TRUE,
        select = NULL,
        over = NULL,
        s = list(type = "o", pch = 19, cex = 0.5),
        o = list(type = "l", col = "blue", lwd = 2),
        i = list(type = "o", pch = 19, cex = 0.5),
        e = list(type = "l", col = "red", lwd = 2),
        la = list(h = c(), v = c(), col = "red", xpd = FALSE),
        ls = list(),
        li = list(col = "grey", lty = 5),
        box = TRUE,
        ax = list(),
        ay = list(),
        parg = list(),
        title = TRUE,
        t1 = "Signal",
        t2 = "Mode",
        pdf = TRUE,
        name = "EMD",
        ext = ".pdf",
        dir = tempdir(),
        track = TRUE,
        openfile = TRUE
)
```

Arguments

emd	an emd object
xy	the original signal. Is overridden by emd.
ini	an optional vector of length n of the eventual initial Intrinsic Mode Function xy would be a demodulation of, if it is a demodulation.
dt	the depth/time. Is overridden by emd.
m	a matrix with columns of same length that xy, made of the decomposition of the signal. Is overridden by emd.
mode	which modes/decompositions to plot
repl	the replication of decompositions in m. Is overridden by emd.
size.xy, size.dt	
	the size i inches of each individual plot in pdf
style	whether to not plot the original signal (style = 0), to plot it as the first signal (style = 1), or to plot it before each individual mode (style = 2, is the default)

xylim, dtlim, inilim	
	the boundaries for the plots (inilim stands for the xy boundaries of the plot of the initial IMF xy is a demodulation of, if applicable)
vertical	whether to have the depth/time [dt] axis vertically (geologist convention) or horizontaly (climatologist convention)
adapt.axis	whether to let the plot adapt the axis to see the variability of the decompositions. The default os to have a comparable x axis for each plots
adapt.last	whether to adapt the last plot as a residue (if TRUE the x axis will be identical to the one of the signal, not centered on 0)
select	the components to plot
over	which modes/decompositions will be cumulated and added to the signal plotted at their left or above them (if style = 2)
s, o, i, e	lists of parameters to feed lines, for the original signal, the cumulated modes/decompositions overlapping it, the modes/decompositions themselves, and the enveloppe of the initial signal used for demodulation if it applies, respectively.
la, ls, li	lists of parameters to provide the abline function (makes personalised lines for you to have a better grasp of the data). la will plot on all panels, ls on the signal ones, and li on the modes ones.
box	whether to draw boxes around the plots
ax, ay	lists of parameters to feed minorAxis, the function making the axes, for the x and y axes
parg	list of parameters to feed par
title	whether to write titles
t1	the title for the signal
t2	the title for the modes
pdf	whether to plot as a pdf
name, ext, dir, track, openfile	
	parameters for the pdfDisplay function, namely the name of the pdf file, its extension (if you want to make a .svg file you can), the directory of the file, whether to track the changes (if you use sumatrapdf as a default pdf reader you can set it to F and it will avoid creating too many pdf files), and whether to directly open the file

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + 0.01 * t
```

```
inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7,
                 repl = 10, comb = 10, factor_noise = 10,
                 speak = TRUE)

plot_emd(dec, select = c(4,6), pdf = FALSE)

plot_emd(dec, dir = tempdir())
```

---

plot_hex   *Group and/or log-scale hexagonal binning*

Description

Group and/or log-scale hexagonal binning. Provides a legend indicating the count representations.
USES THE GRID GRAPHICAL SYSTEM, BASE GRAPHICS NOT SUPPORTED. To add lines,
polygons or text, use the l, g and t arguments.

Usage

```
plot_hex(
  x,
  y,
  id = NA,
  select = NA,
  uniform = TRUE,
  bins = 60,
  xbnds = range(x, na.rm = TRUE),
  ybnds = range(y, na.rm = TRUE),
  xlim = xbnds,
  ylim = ybnds,
  log = "",
  shape = 1,
  mincnt = 1,
  maxcnt = NA,
  colorcut = seq(0, 1, length = 17),
  colramp = function(n) matlab.like(length(colorcut) - 1),
  trans = NULL,
  inv = NULL,
  border = NULL,
  lwd = 0.1,
  cex = 1,
```

```
  main = "",
  xlab = "x",
  ylab = "y",
  xaxis = TRUE,
  yaxis = TRUE,
  xaxs = "r",
  yaxs = "r",
  box = TRUE,
  mar = c(0.15, 0.125, 0.15, 0.2),
  legend = TRUE,
  leg_sep = 0.1,
  xpd_hex = 0.75,
  xpd_leg = 1.5,
  l = list(x = NULL, y = NULL, default.units = "native"),
  g = list(x = NULL, y = NULL, default.units = "native"),
  t = list(label = NULL, default.units = "native"),
  plot = TRUE
)
```

Arguments

x, y	vectors giving the coordinates of the bivariate data points to be binned.
id	a vector of ids for each x value, to separate different groups of data
select	the groups of ids to plot
uniform	whether to keep the creaks defined by the entire matrixes when selecting only a part of it
bins	the number of bins partitioning the range of xbnds.
xbnds, ybnds	horizontal and vertical limits of the binning region in x or y units respectively; must be numeric vector of length 2.
xlim, ylim	the limits of the plot
log	a character string which contains "x" if the x axis is to be logarithmic, "y" if the y axis is to be logarithmic and "xy" or "yx" if both axes are to be logarithmic.
shape	the theoretical shape = yheight/xwidth of the plotting. This adapts the form of the hexagons accordingly.
mincnt, maxcnt	fraction of cell area for the lowest and largest count, respectively
colorcut	vector of values covering [0, 1] that determine hexagon color class boundaries and hexagon legend size boundaries. Alternatively, an integer (<= maxcnt) specifying the number of equispaced colorcut values in [0,1].
colramp	function accepting an integer n as an argument and returning n colors.
trans	a transformation function for the counts such as [log10](log10)
inv	the inverse transformation function (if trans = [log10](log10), inv should for instance be function(x) 10^x.
border	the color of the border of the hexagons. By default it will be the color of the filling

`lwd`	the width of the border of the hexagons.
`cex`	the magnification of text.
`main`	main title.
`xlab, ylab`	x and y axis labels respectively.
`xaxis, yaxis`	whether to plot the x and y axes respectively.
`xaxs, yaxs`	The style of axis interval calculation to be used for the axes. By default the style "r" (regular) first extends the data range by 4 percent at each end and then finds an axis with pretty labels that fits within the extended range. Style "i" (internal) just finds an axis with pretty labels that fits within the original data range.
`box`	whether to plot a box.
`mar`	a numerical vector of the form c(bottom, left, top, right) which gives the room the give to the margins in Normalised Parent Coordinates (see `grid` package for more information)
`legend`	whether to plot the legend.
`leg_sep`	the distance between hexagons and text f the legend in Normalised Parent Coordinates left on the right margin
`xpd_hex`	factor to expand the legend hexagons
`xpd_leg`	factor to expand the height of the legend
`l`	a list of arguments to feed to `grid::grid.polyline` ATTENTION the grid package has to be loaded
`g`	a list of arguments to feed to `grid::grid.polygon` ATTENTION the grid package has to be loaded
`t`	a list of arguments to feed to `grid::grid.text` ATTENTION the grid package has to be loaded
`plot`	whether to plot. If FALSE, returns a grob.

Examples

```
library(grid) # To use the gpar function

set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
        rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7, repl = 10, comb = 10, factor_noise = 10,
```

```
                        speak = TRUE)


  plot_emd(dec, dir = tempdir())

  integrity(xy, dec)
  parsimony(dec)

  ht  <- inst.pulse(dec, plot = FALSE)

  plot_hex(x = 1/ht$f, y = ht$a, bins = 100, ybnds = c(0,2),
           log = "x", trans = log10, inv = function(x) 10^x,
           main = "Spectral Population", xlab = "Period", ylab = "Amplitude")

  plot_hex(x = 1/ht$f, y = ht$a, bins = 100, ybnds = c(0,2),
           log = "x", trans = log10, inv = function(x) 10^x,
           main = "Spectral Population", xlab = "Period", ylab = "Amplitude",
           id = ht$mode, select = c(4,6,7),
           l = list(x = c(30, 30, 240, 240), y = unit(c(0,1,0,1), "npc"),
                    id = c(1,1,2,2), gp = gpar(col = c("red", "blue"), lwd = 2)),
         g = list(x = c(18, 50, 50, 18, 18, 50, 50, 18),
                    y = c(0, 0, 1.9, 1.9, 2.05, 2.05, 1.95, 1.95),
                    id = c(1,1,1,1,2,2,2,2),
                    gp = gpar(col = c("red", NA), fill = c(NA, "white"), lwd = 2)),
         t = list(label = "Mode 4", x = 30, y = 2, gp = gpar(col = "red")))
```

---

plot_hist   *Group and/or log-scale histogram*

Description

Specialised histogram: allows to work in log-scale (for x) and to distinguish different groups of data

Usage

```
plot_hist(
  x,
  breaks = 100,
  id = NA,
  select = NA,
  pile = TRUE,
  line = FALSE,
  mids = FALSE,
  xlim = NA,
  ylim = NA,
  xlog = FALSE,
  axes = TRUE,
  xa = list(),
```

```
    ya = list(),
    main = "",
    xlab = "X",
    ylab = "Counts",
    col = NA,
    border = NA,
    text = FALSE,
    labels = NA,
    t = list(adj = c(0.5, -2), font = 2),
    add = FALSE
)
```

Arguments

x	vector or matrix
breaks	one of:

- a vector giving the breakpoints between histogram cells,
- a function to compute the vector of breakpoints,
- a single number giving the number of cells for the histogram,
- a character string naming an algorithm to compute the number of cells (see 'Details' in [hist](#)),
- a function to compute the number of cells.

In the last three cases the number is a suggestion only; as the breakpoints will be set to pretty values, the number is limited to 1e6 (with a warning if it was larger). If breaks is a function, the x vector is supplied to it as the only argument (and the number of breaks is only limited by the amount of available memory).

id	a vector of ids for each x value, to separate different groups of data
select	a vector of id values idenifying the groups of data to plot and their order
pile	whether to cumulate the different one on the other
line	whether to plot as lines or rectangles
mids	if lines is TRUE, whether the nodes of the lines are the middle positions or the upper corner of the rectangles.
xlim, ylim	the boundaries for the plots. If ylim = NA the upper ylim will be increased by 10% to allow for text (see 'text' parameter)
xlog	whether to set the x axis in log scale
axes	whether to plot the axes
xa, ya	list of arguments to feed minorAxis for the x and y axes respectively
main, xlab, ylab	
	the main title and the labels of the x and y axes
col	a function or a character vector defining the colors of the different modes
border	the colour of the borders, by default identical to col
text	if there are different groups, whether to add a number above each of them to distinguish them

labels   the labels to put on top of each group

t   a list of parameters to feed text()

add   whether to add the plot to a preexisting plot

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7, repl = 10, comb = 10, factor_noise = 10,
                 speak = TRUE)


plot_emd(dec, dir = tempdir())

integrity(xy, dec)
parsimony(dec)

ht  <- inst.pulse(dec, plot = FALSE)

opar <- par('mfrow')

par(mfrow = c(2,1))

plot_hist(x = 1/ht$f, breaks = 500,
          xlog = TRUE, xlab = "Period")

plot_hist(x = 1/ht$f, breaks = 500, id = ht$mode,
          xlog = TRUE, text = TRUE, add = TRUE, line = TRUE, pile = FALSE)

abline(v = c(p1, p2), col = "red", lwd = 2, lty = 5)

plot_hist(x = 1/ht$f, breaks = 500, id = ht$mode,
          xlog = TRUE, text = TRUE, xlab = "Period")

abline(v = c(p1, p2), col = "red", lwd = 2, lty = 5)

par(mfrow = opar)
```

---

plot_imf *Plot IMFs characteristics*

Description

General plot for the envelope, instantaneous frequency (period) and identity tuning of an intrinsic mode function (IMF)

Usage

```
plot_imf(
  pulse,
  dtlim = NULL,
  xylim = NULL,
  flim = NULL,
  fclim = NULL,
  dtline = NULL,
  fline = NULL,
  fcline = NULL,
  vertical = FALSE,
  n = 10,
  at.maj = NULL,
  ls = list(type = "o", pch = 19),
  le1 = list(lwd = 2),
  le2 = list(lty = 2),
  lid = list(type = "p", pch = 19),
  lcos = list(),
  ldt = list(lty = 5, lwd = 2),
  lf = list(lty = 5),
  lfc = list(lty = 5),
  box = TRUE
)
```

Arguments

pulse	a pulse object
dtlim, xylim, flim, fclim	
	the boundaries for the plots, respectively for the depth/time, amplitude, frequency and freqeuncy carrier
dtline, fline, fcline	
	coordinates to add vertical/horizontal lines
vertical	whether to have the depth/time [dt] axis vertically
n	the the number of intervals defined by minor ticks (geologist convention) or horizontaly (climatologist convention)
at.maj	the positions at which major tick-marks are to be drawn.

ls, le1, le2, lid, lcos

                  lists of parameters to feed lines, for the original signal, the upper and lower
                  envelope, the identity tuning, and the cosine line in the identity tuning

ldt, lf, lfc   lists of parameters to provide the abline function (makes personalised lines for
                  you to have a better grasp of the data).

box   whether to draw boxes around the plots

Details

the line in the identity tuning plot is a genuine cosine, independent from the signal. This is evident
when riding waves generate dephasing.

Examples

```
n <- 600

t <- seq_len(n)

p1 <- 30
p2 <- 40 * 21

am <- sin(t*2*pi/p2 + 50) + 0.03

xy <- sin(t*2*pi/p1 + 50) * 3 * am

int <- c(rep(1, 99 + 100), seq(1,3,2/100), seq(3,1,-2/100), rep(1,100 + 99))

dt <- cumsum(int)

samp <- approx(dt, xy, xout = seq(1,802, by = 2))

xy <- samp$y
dt <- samp$x

e <- normalise(m = xy, dt = dt)$a

cond <- dt < 75

xy <- xy[!cond]
dt <- (dt[!cond] - 75) / 1.2
e  <- e[!cond]

dq   <- dq.algorithm(xy/e, dt)

pulse <- as.pulse(dt = dt, m = xy, f = dq$f, a = e, idt = dq$idt,
                  repl = 1)

plot_imf(pulse, fline = 25, dtline = c(222, 489))
```

| plot_pulse | *Visualise the instantaneous frequencies and amplitudes of a decomposition* |

Description

Visualise the instantaneous frequencies and amplitudes of a decomposition

Usage

```
plot_pulse(
  pulse,
  style = "b",
  breaks = 500,
  bins = 100,
  cut = 18,
  lines = NULL,
  keep = NULL,
  lose = NULL
)
```

Arguments

pulse	a pulse object (created by [inst.pulse](#) or [as.pulse](#))
style	whether to plot the distribution of frequency ('d'), the spectral population ('p') or both ('b', is the default)
breaks, bins, cut	
	parameter for the plots: breaks is fed to [plot_hist](#), bins is fed to [plot_hex](#), and cut defines the number of color cuts for [plot_hex](#). For better control use [plot_hist](#) and [plot_hex](#) directly.
lines	the period of lines to be added to the plots for better visualisation
keep, lose	which modes to plot or to not (keep overrides lose)

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)
```

```
dt <- cumsum(inter_dt)
dec <- extricate(xy, dt, nimf = 7, repl = 10, comb = 10,
                 factor_noise = 10, speak = TRUE)


plot_emd(dec, dir = tempdir())


integrity(xy, dec)
parsimony(dec)

ht   <- inst.pulse(dec, plot = FALSE)

plot_pulse(ht, lines = c(30, 240))
```

---

plot_ratio   *Visualise the instantaneous frequencies ratios of a decomposition*

Description

Visualise the instantaneous frequencies ratios of a decomposition

Usage

```
plot_ratio(
  ratio,
  sqrt.rpwr = TRUE,
  style = "b",
  select = NA,
  bins = 100,
  cut = 18,
  lines = NULL,
  plot = TRUE,
  width = 10,
  height = 10,
  name = "Ratio",
  ext = ".pdf",
  dir = tempdir(),
  track = TRUE,
  openfile = TRUE
)
```

Arguments

ratio   a ratio object (created by inst.ratio

sqrt.rpwr	whether to use the squared ratio power (i.e. the squared multiplication of the instantaneous amplitudes of the modes two by two) rather than the ratio power itself.
style	whether to plot a single plot in the graphics device ('s'), the to plot an ensemble of all the ratios combinations in a pdf ('e'), or both ('b', is the default)
select	the groups of ratios combinations to plot in the single plot (in the "1/2" form)
bins, cut	parameter for the plots: bins is fed to [plot_hex](), and cut defines the number of color cuts for [plot_hex](). For better control use [plot_hex]() directly.
lines	the ratio of lines to be added to the plots for better visualisation
plot	whether to plot. Otherwise output a grob of the single plot.
width, height	the width and height in inches of each separate plot in the ensemble of all the ratios combinations
name, ext, dir, track, openfile	parameters for the pdfDisplay function, namely the name of the pdf file, its extension (if you want to make a .svg file you can), the directory of the file, whether to track the changes (if you use sumatrapdf as a default pdf reader you can set it to F and it will avoid creating too many pdf files), and whether to directly open the file

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)
dec <- extricate(xy, dt, nimf = 7, repl = 10, comb = 10,
                 factor_noise = 10, speak = TRUE)


plot_emd(dec, dir = tempdir())

integrity(xy, dec)
parsimony(dec)

ht    <- inst.pulse(dec, plot = FALSE)
ratio <- inst.ratio(ht, plot = FALSE)

plot_ratio(ratio, lines = c(8), style = "s")
plot_ratio(ratio, lines = c(8), style = "s", select = c("4/6"))
```

```
plot_ratio(ratio, lines = c(8), style = "e", dir = tempdir())
```

---

ratios   *Computes ratios of numerical values*

Description

Computes ratios of numerical values

Usage

```
ratios(x)
```

Arguments

x   values to compute the ratio from

Value

a dataframe of $ratio, $x1 and $x2

Examples

```
ratios(c(20,40,100,400))
```

---

repl.out   *Remove / Bind replicates in emd objects*

Description

Remove / Bind replicates in emd objects

Usage

```
repl.out(emd, keep = NULL, lose = NULL, reorder = FALSE)

repl.bind(emd, comb)
```

Arguments

emd	emd-type object
keep, lose	the modes to keep or lose
reorder	whether to reinitialise the index of replicates when suppressing one
comb	the number of replicates that have to be bound together

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7, repl = 20, comb = 2, factor_noise = 10,
                 speak = TRUE, output_sifting = TRUE)

reduced  <- repl.out(dec, keep = c(3,4))

parsimony(reduced)

plot_emd(reduced, pdf = FALSE, select = c(4,6))

combined <- repl.bind(dec, 10)

parsimony(combined)

plot_emd(combined, pdf = FALSE, select = c(4,6))
```

---

respace	*Interpolate with even spacing*

Description

Interpolate with even spacing. Can determine on its own the most conservative sampling interval (using the Greatest Common Rational Divisor)

Usage

```
respace(dt, xy = NULL, delta = NULL, tolerance = 8, relative = TRUE)
```

Arguments

dt	depth/time (same length than length/rows of xy)
xy	signal (vector or matrix)
delta	the new sampling interval. If NA, uses the Greatest Common Rational Divisor

tolerance, relative

                parameters for the `divisor` function (`StratigrapheR` package), to compute the
                Greatest Common Rational Divisor

Value

a list of interpolated xy and dt values ($xy and $dt), plus a vector of logicals indicating whether
each point was part of the initial input or was added by interpolation

Examples

```
set.seed(42)

n <- 50
t <- seq_len(n)

xy <- (1 + 0.6 * sin(t*0.025)) * sin(t*0.2)  + 2 * sin(t*0.025) +
        rnorm(n, sd = 0.5)

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5), 1)

dt <- cumsum(inter_dt)

res <- respace(xy = xy, dt = dt)

opar <- par("mfrow")

par(mfrow = c(1,1))

plot(res$xy, res$dt, type = "l")
points(res$xy[res$initial], res$dt[res$initial], pch = 19, col = "green")
points(res$xy[!res$initial], res$dt[!res$initial],
        pch = 19, col = "red", cex = 0.5)

par(mfrow = opar)
```

---

simp.emd   *Simplifies the components of an EMD*

Description

Simplifies the component of an EMD to only extremas and zero-crossings, and outputs problematic
extrema: multiple extrema (extrema not separated by zero-crossings) and crossing extrema (extrema
at zero).

Usage

```
simp.emd(emd = NULL, m = NULL, dt = NULL, repl = 1, use.names = FALSE)
```

Arguments

emd	emd-type object
m	a matrix of the amplitude values (xy) of the components, each column being a component. Each column should have the same number of non NA values. Vectors, for 1 component, are accepted. Is overridden by emd.
dt	the depth or time value. Is overridden by emd.
repl	the amount of replicates in m. Is overridden by emd.
use.names	whether to use the column names to identify problematic extrema

Value

a list of the depth or time values ($dt) of the simplified IMF (Intrinsic Mode Function), of their amplitude ($xy), and of the position and component of problematic multiple extrema ($multiple_extrema) and crossing extrema ($crossing_extrema)

Examples

```
xytest <- c(0.5, 1,-1,-0.85,-0.5,-1,-0.5,-1,1,0.5,0,0,
            1,-1,0,1,2,-2,1,2,1,3,0,-1,-1,3,0)

repeatafterme <- 2

m  <- matrix(rep(xytest,repeatafterme), ncol = repeatafterme)
dt <- 1:length(xytest)

res <- simp.emd(m = m, dt = dt, repl = repeatafterme)

opar <- par("mfrow")

par(mfrow = c(1,1))

plot(dt, xytest, type = "o", pch = 19)
abline(h = 0, col = "grey")

me <- res$multiple_extrema$dt[res$multiple_extrema$repl == 1]
ce <- res$crossing_extrema$dt[res$multiple_extrema$repl == 1]

abline(v = me, col = "orange")
abline(v = ce, col = "darkred")

points(res$dt[,1], res$xy[,1], col = "red", pch = 19)

par(mfrow = opar)
```

---

symmetry   *Symmetry of components*

Description

The function returns the highest factor of amplitude either in negative or positive values. This quantifies the symmetry of components.

Usage

```
symmetry(xy, names = "num")
```

Arguments

xy   signal (vector or matrix)

names   the names to use for the resulting vector. If NULL no names are provided, if NA
                its the names of the columns of the xy matrix, if "num" it the column index of
                the matrix xy

Examples

```
set.seed(42)

n <- 600
t <- seq_len(n)

p1 <- 30
p2 <- 240

xy <- (1 + 0.6 * sin(t*2*pi/p2)) * sin(t*2*pi/p1)  + 2 * sin(t*2*pi/p2) +
  rnorm(n, sd = 0.5) + t * 0.01

inter_dt <- round(runif(length(xy), min = 0.5, max = 1.5),1)

dt <- cumsum(inter_dt)

dec <- extricate(xy, dt, nimf = 7, repl = 1, comb = 40, factor_noise = 10,
                 speak = TRUE, output_sifting = TRUE)

symmetry(dec$m)

plot_emd(dec, select = c(6,8,9), pdf = FALSE, adapt.axis = TRUE)
```

Index