Package ‘seewave’

June 26, 2009

Type Package

Title Time wave analysis and graphical representation

Version 1.5.5

Date 2009-6-26

Author Jerome Sueur <sueur@mnhn.fr>, Thierry Aubin <thierry.aubin@u-psud.fr>, Caroline Simonis <csimonis@mnhn.fr>

Maintainer Jerome Sueur <sueur@mnhn.fr>

Webpage http://sueur.jerome.perso.neuf.fr/seewave.html

Depends R(>= 2.2.0), sound

Encoding latin1

SystemRequirements FLAC

Suggests rgl, rpanel, tcltk, signal, tuneR, audio

ZipData no

Description seewave provides functions for analysing, manipulating, displaying, editing and synthesizing time waves (particularly sound). This package processes time analysis (oscillograms and envelopes), spectral content, resonance quality factor, entropy, cross correlation and autocorrelation, zero-crossing, dominant frequency, analytic signal, frequency coherence, 2D and 3D spectrograms and many other analyses.

License GPL (>= 2)

URL http://sueur.jerome.perso.neuf.fr/seewave.html

Repository CRAN

Date/Publication 2009-06-26 17:53:24

R topics documented:

addsilw . . . 3
afilter . . . 5
ama . . . 6
attenuation . . . 7
autoc . . . 8
ccoh . . . 10
ceps . . . 12
cepstro . . . 14
coh . . . 16
convSPL . . . 17
corenv . . . 18
corspec . . . 20
covspectro . . . 22
crest . . . 24
csh . . . 25
cutspec . . . 27
cutw . . . 28
dBscale . . . 29
dBweight . . . 30
deletew . . . 32
dfreq . . . 33
diffenv . . . 35
diffspec . . . 36
diffwave . . . 38
discrets . . . 40
drawenv . . . 41
dynspec . . . 42
echo . . . 45
env . . . 46
export . . . 48
fadew . . . 49
fdoppler . . . 50
ffilter . . . 51
field . . . 53
fir . . . 54
fma . . . 56
ftwindow . . . 57
fund . . . 58
H . . . 60
hilbert . . . 61
ifreq . . . 62
lfs . . . 64
listen . . . 65
meanspec . . . 66
mel . . . 69
micsens . . . 70

moredB . . . 71
mutew . . . 72
noise . . . 73
orni . . . 74
oscillo . . . 75
oscilloST . . . 77
pastew . . . 79
peewit . . . 80
pellucens . . . 81
pulse . . . 81
Q . . . 82
repw . . . 84
resamp . . . 85
revw . . . 86
rmam . . . 87
rmnoise . . . 88
rmoffset . . . 89
rms . . . 90
savewav . . . 91
seewave . . . 92
setenv . . . 93
sfm . . . 94
sh . . . 95
sheep . . . 97
simspec . . . 97
spec . . . 99
specprop . . . 102
spectro . . . 104
spectro3D . . . 107
symba . . . 109
synth . . . 111
th . . . 113
tico . . . 114
timer . . . 115
wasp . . . 116
wav2flac . . . 118
wf . . . 120
zapsilw . . . 122
zc . . . 123

Index **125**

addsilw *Add or insert a silence section*

Description

Add or insert a silence section to a time wave.

Usage

```
addsilw(wave, f, at = "end", choose = FALSE, d = NULL,
plot = FALSE, Sample = FALSE,...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
at	position where to add the silence section (in s). Can be also specified as "start", "middle" or "end".
choose	logical, if TRUE the point where silence will be added into wave2 (=at) can be graphically chosen with a cursor.
d	duration of the silence section to add (in s).
plot	logical, if TRUE returns an oscillographic plot of wave with the new silence section (by default TRUE).
Sample	if TRUE and plot is FALSE returns an object of class Sample
...	other oscillo graphical parameters.

Value

If plot is FALSE, a new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

oscillo, cutw,deletew, fadew,pastew, mutew,revw, zapsilw

Examples

```
data(tico)
addsilw(tico,f=22050,d=0.2)
addsilw(tico,f=22050,at="end",d=0.5)
addsilw(tico,f=22050,at=0.33,d=0.46)
```

afilter — *Amplitude filter*

Description

This function deletes all signal which amplitude is below a selected threshold.

Usage

```
afilter(wave, f, threshold = 5, plot = TRUE,
listen = FALSE, Sample = FALSE,...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`threshold`	amplitude threshold (in %).
`plot`	logical, if `TRUE` plots the new oscillogram (by default `TRUE`).
`listen`	if `TRUE` the new sound is played back.
`Sample`	a logical, if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `oscillo` graphical parameters.

Details

The `threshold` value is in % relative to the maximal value of `wave`. Signal inferior to this value is clipped.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Note

This function is used as an argument (`threshold`) in the following functions: `autoc`, `csh`, `dfreq`, `timer` and `zc`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`ffilter`, `oscillo`

Examples

```
data(orni)
op<-par(mfrow=c(2,1))
afilter(orni,f=22050)
title(main = "threshold level = 5")
afilter(orni,f=22050,threshold=0.5,colwave="blue")
title(main = "threshold level = 0.5")
par(op)
```

---

ama *Amplitude modulation analysis of a time wave*

Description

This function computes the Fourier analysis of a time wave envelope. This allows to detect periodicity, in particular those generated by amplitude modulations.

Usage

```
ama(wave, f, envt = "hil", wl = 512, plot = TRUE, type = "l", ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`envt`	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope.
`wl`	length of the window for the analysis (even number of points, by default = 512).
`plot`	logical, if `TRUE` the spectrum of the envelope (by default `TRUE`).
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `meanspec` parameters.

Details

This function is based on `env` and `meanspec`.
The envelope of `wave` is first computed and the spectrum of this envelope is then processed. All `env` and `meanspec` arguments can be set up. Be sure to set up `wl` large enough if you want to detect low amplitude modulation periodicity.

Value

If `plot` is `FALSE`, `ama` returns a numeric vector corresponding to the computed spectrum. If `peaks` is not `NULL`, `ama` returns a list with two elements:

`spec`	the spectrum computed
`peaks`	the peaks values (in kHz).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`env`, `fma`, `meanspec`

Examples

```
data(orni)
# detection of 2 main amplitude modulations in a cicada song:
# one with a 0.020 kHz frequency (due to signal/silence periodicity)
# one with a 0.258 kHz frequency (due to pulses in the echemes)
# one with a 2.369 kHz frequency (fundamental frequency)
ama(orni,f=22050,wl=1024)
# these amplitude modulations can be identify with a cursor:
ama(orni,f=22050,wl=1024,identify=TRUE)
```

`attenuation` *Generate sound intensity attenuation data*

Description

This function generates dB data following theoretical spherical attenuation of sound.

Usage

```
attenuation(lref, dref = 1, dstop, n, plot = TRUE,
xlab = "Distance (m)", ylab = "dB", type = "l", ...)
```

Arguments

`lref`	reference intensity or pressure level (in dB).
`dref`	reference distance corresponding to `lref` (in m.) (by default = 1).
`dstop`	maximal distance of propagation (in m.).
`n`	number of points generated between `dref` and `dstop`.
`plot`	logical, if `TRUE` plots attenuation against distance of propagation (by default `TRUE`).

`xlab`	title of the x axis.
`ylab`	title of the y axis.
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `plot` graphical parameters.
ss	

Value

If `plot` is `FALSE` return a numeric vector with the data generated.

Note

Sound attenuation in a free, unbounded medium behaves in accordance with the inverse square law. `attenuation` generates data following this rule from a reference point where sound intensity level (SIL) or sound pressure level (SPL) is known. Such theoretical data can be compared with experimental data collected in a real environment.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Hartmann, W. M. 1998 *Signals, sound and sensation*. New York: Springer.

See Also

`convSPL`, `moredB`

Examples

```
# theoretical attenuation up to 150 m of a 100 dB/1m sound source
attenuation(lref=100,dstop=150,n=200)
```

---

autoc *Short-term autocorrelation of a time wave*

Description

This function returns the fundamental frequency of a harmonic time wave. This is achieved by computing a correlation of the signal with itself after a time delay.

Usage

```
autoc(wave, f, wl = 512, fmax, threshold = NULL, plot = TRUE,
xlab = "Time (s)", ylab = "Frequency (kHz)", ylim = c(0, f/2000),...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
wl	length of the window for the analysis (even number of points, by default = 512).
fmax	the maximum frequency to detect (in Hz).
threshold	amplitude threshold for signal detection (in %).
plot	logical, if TRUE plots the fundamental frequency against time (by default TRUE).
xlab	title of the x-axis.
ylab	title of the y-axis.
ylim	the range of y values.
...	other plot graphical parameters.

Details

Autocorrelation process can be time consuming.

Value

When plot is FALSE, autoc returns a two-column matrix, the first column corresponding to time in seconds (*x*-axis) and the second column corresponding to to fundamental frequency in kHz (*y*-axis).
NA corresponds to pause sections in wave (see threshold).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Thierry Aubin ⟨thierry.aubin@u-psud.fr⟩

References

Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds) 1998. *Animal acoustic communication*. Springer, Berlin, Heidelberg.

See Also

ceps

Examples

```
data(sheep)
# fundamental frequency of a sheep
autoc(sheep,f=8000,threshold=5,fmax=700)
# overlay on spectrogram
spectro(sheep,f=8000,ovlp=75,zp=16,scale=FALSE)
par(new=TRUE)
autoc(sheep,f=8000,wl=512,threshold=5,fmax=700,col="black",pch=20,
```

```
    xaxs="i",yaxs="i",ann=FALSE,yaxt="n")
legend(0.5,3.6,"Fundamental frequency",pch=20,col="black",bty=0,cex=0.7)
```

---

ccoh *Continuous coherence function between two time waves*

Description

This function returns a two-dimension coherence representation between two time waves. The function corresponds to a sliding coherence function along the two signals. This produces a 2-D density plot. An amplitude contour plot can be overlaid.

Usage

```
ccoh(wave1, wave2, f, wl = 512, ovlp = 0, plot = TRUE,
grid = TRUE, scale = TRUE, cont = FALSE,
collevels = seq(0, 1, 0.01), palette = rev.heat.colors,
contlevels = seq(0, 1, 0.01), colcont = "black",
colbg="white", colgrid = "black",
colaxis = "black", collab="black",
plot.title = title(main = "", xlab = "Time (s)",
ylab = "Frequency (kHz)"), scalelab = "Coherence",
scalefontlab = 1, scalecexlab =0.75, axisX = TRUE, axisY = TRUE,
flim = NULL, flimd = NULL,
...)
```

Arguments

`wave1`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`wave2`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave1` and `wave2` (in Hz). Does not need to be specified if `wave1` and/or `wave2` are/is of class `ts`, `Sample`, or `Wave`.
`wl`	window length for the analysis (even number of points, by default = 512).
`ovlp`	overlap between two successive windows (in %).
`plot`	logical, if `TRUE` plots the continuous coherence function (by default `TRUE`).
`grid`	logical, if `TRUE` plots a y-axis grid (by default `TRUE`).
`scale`	logical, if `TRUE` plots a dB colour scale on the right side of the plot (by default `TRUE`).
`cont`	logical, if `TRUE` overplots contour lines on the plot (by default `FALSE`).
`collevels`	a set of levels which are used to partition the amplitude range of the coherence (should be between 0 and 1.
`palette`	a color palette function to be used to assign colors in the plot, see `Details`.

`contlevels`	a set of levels which are used to partition the amplitude range for contour overplot (in dB).
`colcont`	colour for `cont` plotting.
`colbg`	background colour.
`colgrid`	colour for `grid` plotting.
`colaxis`	color of the axes.
`collab`	color of the labels.
`plot.title`	statements which add titles to the plot.
`scalelab`	amplitude scale label.
`scalefontlab`	font of the amplitude scale label.
`scalecexlab`	cex of the amplitude scale label.
`axisX`	logical, if `TRUE` plots time X-axis (by default `TRUE`).
`axisY`	logical, if `TRUE` plots frequency Y-axis (by default `TRUE`).
`flim`	modifications of the frequency Y-axis limits.
`flimd`	dynamic modifications of the frequency Y-axis limits. New `wl` and `ovlp` arguments are applied to increase time/frequency resolution.
`...`	other `contour` and `oscillo` graphical parameters.

Details

Coherence is a frequency domain function computed to show the degree of a relationship between two signals. The value of the coherence function ranges between zero and one, where a value of zero indicates there is no causal relationship between the signals. A value of one indicates the existence of linear frequency response between the two signals. This can be used, for instance, to compare the input and output signals of a system.
Any colour palette can be used. In particular, it is possible to use other palettes coming with **seewave**: `temp.colors`, `rev.gray.colors.1`, `rev.gray.colors.2`, `spectro.colors`, `rev.terrain.colors`, `rev.topo.colors`, `rev.cm.colors` corresponding to the reverse of `terrain.colors`, `topo.colors`, `cm.colors`.
Use `locator` to identify points.

Value

If `plot` is `FALSE`, this function returns a matrix. Each column corresponds to a coherence function of length `wl`.

Note

This function is based on `spec.pgram`, `contour` and `filled.contour`. See `spectro` for graphical changes.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ but this function is mainly based on `spec.pgram` by Martyn Plummer, Adrian Trapletti and B.D. Ripley

See Also

coh, spectro, spec.pgram.

Examples

```
wave1<-synth(d=1,f=4000,cf=500)
wave2<-synth(d=1,f=4000,cf=800)
ccoh(wave1,wave2,f=4000)
```

---

ceps *Cepstrum or real cepstrum*

Description

This function returns the cepstrum of a time wave allowing fundamental frequency detection.

Usage

```
ceps(wave, f, wl = 512, at = NULL, from = NULL, to = NULL,
tpeaks = NULL, fpeaks = NULL, tidentify = FALSE,
fidentify = FALSE, col = "black", cex = 1,
colpeaks = "red", cexpeaks = 0.75, fontpeaks = 1, plot = TRUE,
qlab = "Quefrency (bottom: s, up: Hz)", alab = "Amplitude",
qlim = NULL, alim = NULL, type = "l", ...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
wl	if at is not null, length of the window for the analysis (even number of points, by defaults = 512).
at	position where to compute the cepstrum (in s).
from	start position where to compute the cepstrum (in s).
to	end position to compute the cepstrum (in s).
tpeaks	returns peaks value for a given span according to time scale (s)(see details).
fpeaks	returns peaks value for a given span according to frequency scale (Hz)(see details).
tidentify	to identify time values on the plot with the help of a cursor.
fidentify	to identify frequency values on the plot with the help of a cursor.
col	colour of the cepstrum.
cex	pitch size of the cepstrum.

colpeaks	colour of peaks value plotted on the cepstrum.
cexpeaks	character size of peaks value plotted on the cepstrum.
fontpeaks	font of peaks value plotted on the cepstrum.
plot	logical, if TRUE plots the cepstrum.
qlab	title of the quefrency axis.
alab	title of the amplitude axis.
qlim	range of quefrency axis.
alim	range of amplitude axis.
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
...	other plot graphical parameters.

Details

The cepstrum of a time wave is the Fourier transform of the logarithm of the Fourier transform. The cepstrum of a wave *s* is then calcuated as follows:

$$C(s) = Re[FFT^{-1}(\log(|FFT(s)|)]$$

The independent variable of a cepstral graph is called the quefrency. The quefrency is a measure of time, though not in the sense of a signal in the time domain. A correspondence with the frequency domain is obtained by simply computing the reverse of the temporal x co0rdinate. For instance if a peak appears at 0.005 s, this reveals a frequency peak at 200 Hz (=1/0.005). This explain the two scales plotted when plot is TRUE.
If at, from or to are FALSE then ceps computes the cepstrum of the whole signal.
tpeaks and fpeaks setting corresponds to dimension of embed.
When using tidentify or tidentify, press ‘stop’ tools bar button to return values in the console. tpeaks and fpeaks just differ in the unit of the results.

Value

When plot is FALSE, ceps returns the cesptral profile as a two-column matrix, the first column corresponding to quefrency (*x*-axis) and the second corresponding to amplitude (*y*-axis).

Note

Cepstral analysis is mainly used in speech processing. This analysis allows to extract the fundamental frequency, see the examples.
This function is based on fft.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Martin Maechler ⟨maechler@stat.math.ethz.ch⟩ for peaks.

References

Oppenheim, A.V. and Schafer, R.W. 2004. From frequency to quefrency: a history of the cepstrum. *Signal Processing Magazine IEEE*, 21: 95-106.

See Also

`cepstro`, `fund`, `autoc`

Examples

```
data(sheep)
ceps(sheep,f=8000,at=0.4,wl=1024)
# peaks detection in Hertz, the fundamental is at 160 Hz.
ceps(sheep,f=8000,at=0.4,wl=1024,fpeaks=63)
```

---

cepstro *2D-cepstrogram of a time wave*

Description

This function returns a two-dimension cepstrographic representation of a time wave. The function corresponds to a short-term cepstral transform. An amplitude contour plot can be overlaid.

Usage

```
cepstro(wave, f, wl = 512, ovlp = 0, plot = TRUE, grid = TRUE,
scale = TRUE, cont = FALSE, collevels = seq(0, 1, 0.01),
palette = rev.heat.colors, contlevels = seq(0, 1, 0.01),
colcont = "black", colbg="white", colgrid = "black",
colaxis = "black", collab = "black",
plot.title = title(main = "", xlab = "Time (s)", ylab = "Quefrency (kHz)"),
scalelab = "Amplitude", scalefontlab = 1, scalecexlab = 0.75,
axisX = TRUE, axisY = TRUE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`wl`	if `at` is not null, length of the window for the analysis (even number of points, by defaults = 512).
`ovlp`	overlap between two successive windows (in %).
`plot`	logical, if `TRUE` plots the cepstrogram (by default `TRUE`).
`grid`	logical, if `TRUE` plots a y-axis grid (by default `TRUE`).

scale	logical, if TRUE plots a dB colour scale on the right side of the cesptrogram (by default TRUE).
cont	logical, if TRUE overplots contour lines on the cepstrogram (by default FALSE).
collevels	a set of levels which are used to partition the amplitude range of the cepstrogram (in dB).
palette	a color palette function to be used to assign colors in the plot, see Details.
contlevels	a set of levels which are used to partition the amplitude range for contour overplot (in dB).
colcont	colour for cont plotting.
colbg	background colour.
colgrid	colour for grid plotting.
colaxis	color of the axes.
collab	color of the labels.
plot.title	statements which add titles to the plot.
scalelab	amplitude scale label.
scalefontlab	font of the amplitude scale label.
scalecexlab	cex of the amplitude scale label.
axisX	if TRUE plots time X-axis (by default TRUE).
axisY	if TRUE plots frequency Y-axis (by default TRUE).
...	other contour graphical parameters.

Details

It is unfortunately not possible to turn the *y*-axis to a frequency scale.
See spectro for the use of the graphical arguments.

Value

When plot is FALSE, a matrix is returned with the successive cepstral profiles computed along time.

Note

This function is based on ceps.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩.

References

Oppenheim, A.V. and Schafer, R.W. 2004. From frequency to quefrency: a history of the cepstrum. *Signal Processing Magazine IEEE*, 21: 95-106.

See Also

ceps, fund, autoc

Examples

```
data(sheep)
cepstro(sheep,f=8000)
```

---

coh *Coherence between two time waves*

Description

This function returns the frequency coherence between two time waves.

Usage

```
coh(wave1, wave2, f, plot =TRUE, xlab = "Frequency (kHz)",
ylab = "Coherence", xlim = c(0,f/2000), type = "l",...)
```

Arguments

wave1	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
wave2	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave1 and wave2 (in Hz). Does not need to be specified if wave1 and/or wave2 are/is of class ts, Sample, or Wave.
plot	logical, if TRUE plots the continuous coherence function (by default TRUE).
xlab	title of the frequency X-axis.
ylab	title of the coherence Y-axis.
xlim	range of frequency X-axis.
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
...	other plot graphical parameters.

Details

Coherence is a frequency domain function computed to show the degree of a relationship between two signals. The value of the coherence function ranges between zero and one, where a value of zero indicates there is no causal relationship between the signals. A value of one indicates the existence of linear frequency response between the two signals. This can be used, for instance, to compare the input and output signals of a system.

Value

When `plot` is `FALSE`, this `coh` returns a two-column matrix, the first column being the frequency axis in kHz (*x*-axis) and the second column being the coherence (*y*-axis).

Note

This function is based on `spec.pgram`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ but this function is based on `spec.pgram` by Martyn Plummer, Adrian Trapletti and B.D. Ripley.

See Also

`ccoh`, `spectro`, `spec.pgram`.

Examples

```
wave1<-synth(d=1,f=4000,cf=500)
wave2<-synth(d=1,f=4000,cf=800)
coh(wave1,wave2,f=4000)
```

`convSPL` *Convert sound pressure level in other units*

Description

This function converts sound pressure level (in dB) in sound power (Watt), intensity (Watt/m2) and pressure (Pa). By default, these conversions are applied to air-borne sound.

Usage

```
convSPL(x, d = 1, Iref = 10^-12, pref = 2*10^-5)
```

Arguments

`x`	a numeric vector or a matrix describind SPL values (in dB).
`d`	the distance from the sound source where SPL values have been measured (in meter) (by default = 1m)
`Iref`	reference intensity (in Watt/m2) (by default = 10e-12)
`pref`	reference pressure (in Pa) (by default = 2.10e-5)

Value

`convSPL` returns a list containing three components:

`P`	data converted in sound power (in Watt).
`I`	data converted in sound intensity (in Watt/m2).
`p`	data converted in sound pressure (in Pa).

Note

`Iref` and `pref` correspond to a 1 kHz sound in air.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Hartmann, W. M. 1998 *Signals, sound and sensation*. New York: Springer.

See Also

`moredB`, `dBweight`, `attenuation`

Examples

```
# conversion of two SPL measurements taken at 0.5 m from the source
convSPL(c(80,85),d=0.5)
```

`corenv` *Cross-correlation between two time wave envelopes*

Description

This function tests the similarity between two time wave envelopes by returning their maximal correlation and the time shift related to it.

Usage

```
corenv(wave1, wave2, f, envt="hil", msmooth = NULL, ksmooth = NULL,
plot = TRUE, plotval = TRUE,
method = "spearman", col = "black", colval = "red",
cexval = 1, fontval = 1, xlab = "Time (s)",
ylab = "Coefficient of correlation (r)", type = "l", ...)
```

Arguments

wave1	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
wave2	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave1 and wave2 (in Hz). Does not need to be specified if wave1 and/or wave2 are/is of class ts, Sample, or Wave.
envt	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See env.
msmooth	a vector of length 2 to smooth the amplitude envelope with a mean sliding window. The first component is the window length (in number of points). The second component is the overlap between successive windows (in %). See env.
ksmooth	kernel smooth via kernel. See env.
plot	logical, if TRUE plots r values against frequency shift (by default TRUE).
plotval	logical, if TRUE adds to the plot maximum r value and frequency offset (by default TRUE).
method	a character string indicating which correlation coefficient is to be computed ("pearson", "spearman", or "kendall") (see cor).
col	colour of r values.
colval	colour of r max and frequency offset values.
cexval	character size of r max and frequency offset values.
fontval	font of r max and frequency offset values.
xlab	title of the frequency axis.
ylab	title of the r axis.
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
...	other plot graphical parameters.

Details

Successive correlations between the envelopes of wave1 and wave2 are computed when regularly sliding forward and backward wave2 along wave1.
The maximal correlation is obtained at a particular shift (time offset). This shift may be positive or negative.
The higher smooth is set up, the faster will be the computation but less precise the results will be.
The corresponding p value, obtained with cor.test, is plotted. Inverting wave1 and wave2 may give slight different results.

Value

If plot is FALSE, corenv returns a list containing four components:

r	a two-column matrix, the first colum corresponding to the time shift (frequency x-axis) and the second column corresponding to the successive r correlation values between env1 and env2 (correlation y-axis).

rmax	the maximum correlation value between x and y.
p	the p value corresponding to rmax.
t	the time offset corresponding to rmax.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

env,corspec,covspectro, cor,cor.test.

Examples

```
data(orni)
# cross-correlation between two echemes of a cicada song
wave1<-cutw(orni,f=22050,from=0.3,to=0.4,plot=FALSE)
wave2<-cutw(orni,f=22050,from=0.58,to=0.68,plot=FALSE)
corenv(wave1,wave2,f=22050)
```

---

corspec *Cross-correlation between two frequency spectra*

Description

This function tests the similarity between two frequency spectra by returning their maximal correlation and the frequency shift related to it.

Usage

```
corspec(spec1, spec2, f = NULL, plot = TRUE, plotval = TRUE,
method = "spearman", col = "black", colval = "red",
cexval = 1, fontval = 1, xlab = "Frequency (kHz)",
ylab = "Coefficient of correlation (r)", type="l",...)
```

Arguments

spec1	a first data set resulting of a spectral analysis obtained with spec or meanspec (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
spec2	a first data set resulting of a spectral analysis obtained with spec or meanspec (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
f	sampling frequency of waves used to obtain spec1 and spec2 (in Hz). Not necessary if spec1 and/or spec2 is a two columns matrix obtained with spec or meanspec.
plot	logical, if TRUE plots r values against frequency shift (by default TRUE).

`plotval`	logical, if `TRUE` adds to the plot maximum r value and frequency offset (by default `TRUE`).
`method`	a character string indicating which correlation coefficient is to be computed ("pearson", "spearman", or "kendall") (see `cor`).
`col`	colour of r values.
`colval`	colour of r max and frequency offset values.
`cexval`	character size of r max and frequency offset values.
`fontval`	font of r max and frequency offset values.
`xlab`	title of the frequency axis.
`ylab`	title of the r axis.
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `plot` graphical parameters.

Details

It is important not to have data in dB.
Successive correlations between `spec1` and `spec2` are computed when regularly shifting `spec2` towards lower or higher frequencies.
The maximal correlation is obtained at a particular shift (frequency offset). This shift may be positive or negative.
The corresponding p value, obtained with `cor.test`, is plotted.
Inverting `spec1` and `spec2` may give slight different results, see examples.

Value

If `plot` is `FALSE`, `corspec` returns a list containing four components:

`r`	a two-column matrix, the first colum corresponding to the frequency shift (frequency x-axis) and the second column corresponding to the successive r correlation values between `spec1` and `spec2` (correlation y-axis).
`rmax`	the maximum correlation value between `spec1` and `spec2`.
`p`	the p value corresponding to `rmax`.
`f`	the frequency offset corresponding to `rmax`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds) 1998. *Animal acoustic communication*. Springer, Berlin, Heidelberg.

See Also

`spec`, `meanspec`, `corspec`, `covspectro`, `cor`, `cor.test`.

Examples

```
data(tico)
# compare the two first notes spectra
a<-spec(tico,f=22050,wl=512,at=0.2,plot=FALSE)
c<-spec(tico,f=22050,wl=512,at=1.1,plot=FALSE)
op<-par(mfrow=c(2,1), mar=c(4.5,4,3,1))
spec(tico,f=22050,at=0.2,col="blue")
par(new=TRUE)
spec(tico,f=22050,at=1.1,col="green")
legend(x=8,y=0.5,c("Note A", "Note C"),lty=1,col=c("blue","green"),bty="o")
par(mar=c(5,4,2,1))
corspec(a,c, ylim=c(-0.25,0.8),xaxs="i",yaxs="i",las=1)
par(op)
# different correlation methods give different results...
op<-par(mfrow=c(3,1))
corspec(a,c,xaxs="i",las=1, ylim=c(-0.25,0.8))
title("spearmann correlation (by default)")
corspec(a,c,xaxs="i",las=1,ylim=c(0,1),method="pearson")
title("pearson correlation")
corspec(a,c,xaxs="i",las=1,ylim=c(-0.23,0.5),method="kendall")
title("kendall correlation")
par(op)
# inverting x and y does not give exactly similar results
op<-par(mfrow=c(2,1),mar=c(2,4,3,1))
corspec(a,c)
corspec(c,a)
par(op)
```

---

covspectro *Covariance between two spectrograms*

Description

This function tests the similarity between two spectrograms by returning their maximal covariance and the time shift related to it.

Usage

```
covspectro(wave1, wave2, f, wl = 512, wn = "hanning", n,
plot = TRUE, plotval = TRUE,
method = "spearman", col = "black", colval = "red", cexval = 1,
fontval = 1, xlab = "Time (s)",
ylab = "Normalised covariance (cov)", type = "l", ...)
```

Arguments

`wave1`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).

`wave2`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave1` and `wave2` (in Hz). Does not need to be specified if `wave1` and/or `wave2` are/is of class `ts`, `Sample`, or `Wave`.
`wl`	length of the window for the analysis (even number of points, by default = 512).
`wn`	window name, see `ftwindow` (by default `"hanning"`).
`n`	number of covariances computed between `wave1` and `wave2` when sliding `wave2` along `wave1`.
`plot`	logical, if `TRUE` plots r values against frequency shift (by default `TRUE`).
`plotval`	logical, if `TRUE` adds to the plot maximum R value and frequency offset (by default `TRUE`).
`method`	a character string indicating which correlation coefficient is to be computed ("pearson", "spearman", or "kendall") (see `cor`).
`col`	colour of r values.
`colval`	colour of r max and frequency offset values.
`cexval`	character size of r max and frequency offset values.
`fontval`	font of r max and frequency offset values.
`xlab`	title of the frequency axis.
`ylab`	title of the r axis.
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `plot` graphical parameters.

Details

Successive covariances between the spectrogram of `wave1` and the spectrogram of `wave2` are computed when regularly sliding forward and backward `wave2` along `wave1`.
The maximal covariance is obtained at a particular shift (time offset). This shift may be positive or negative.
`n` sets in how many steps `wave2` will be slided along `wave1`. Time process can be then decreased by setting low `n` value.
Inverting `wave1` and `wave2` may give slight different results.

Value

If `plot` is `FALSE`, `covspectro` returns a list containing three components:

`cov`	the successive covariance values between `wave1` and `wave2`.
`covmax`	the maximum covariance between `wave1` and `wave2`.
`t`	the time offset corresponding to `cov`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds) 1998. *Animal acoustic communication*. Springer, Berlin, Heidelberg.

See Also

`corspec`, `corenv`, `spectro`, `cor`,

Examples

```
# covariance between two notes of a birdsong
data(tico)
note1<-cutw(tico, f=22050, from=0.5, to=0.9)
note2<-cutw(tico, f=22050, from=0.9, to=1.3)
covspectro(note1,note2,f=22050,n=37)
```

`crest` *Crest factor and visualization*

Description

This function returns the crest factor and localizes the different crest(s).

Usage

```
crest(wave, f, plot = FALSE, col = 2, cex = 3, symbol = "*", ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`plot`	if `TRUE` plots the oscillograme of `wave` and indicates the location of the crest(s)
`col`	color of the symbol indicating the localisation of the crest(s)
`cex`	symbol magnification
`symbol`	symbol indicating the localisation of the crest(s)
`...`	other

Details

The crest factor of a time series s is calculated according to:

$$C = \frac{max(s)}{rms(s)}$$

with rms the root-mean-square (see `rms`).

Value

The function returns a list of three items

`C`	crest factor
`val`	value of the crest(s)
`loc`	location of the crest(s)

Note

There might be several crests (maxima) along the time wave but there is a single crest factor.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Hartmann, W. M. 1998 *Signals, sound and sensation*. New York: Springer.

See Also

`oscillo`, `rms`

Examples

```
data(tico)
crest(tico, f=22050)
# see the crest location and change the default graphical parameters
crest(tico, f=22050, plot=TRUE, sym="-")
```

`csh` *Continuous spectral entropy*

Description

This function computes the continuous spectral entropy (H) of a time wave.

Usage

```
csh(wave, f, wl = 512, wn = "hanning", ovlp = 0, threshold = NULL,
plot = TRUE, xlab = "Times (s)", ylab = "Spectral Entropy",
ylim = c(0, 1.1), type = "l", ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`wl`	if `at` is not null, length of the window for the analysis (even number of points, by default = 512).
`wn`	window name, see `ftwindow` (by default `"hanning"`).
`ovlp`	overlap between two successive windows (in %).
`threshold`	amplitude threshold for signal detection (in %).
`plot`	logical, if `TRUE` plots the spectral entropy against time (by default `TRUE`).
`xlab`	title of the x axis.
`ylab`	title of the y axis.
`ylim`	the range of y values.
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `plot` graphical parameters.

Details

See `sh` for computing method.

Value

When `plot` is `FALSE`, `csh` returns a two-column matrix, the first column being time in seconds (*x*-axis) and the second column being the spectral entropy (*y*-axis) computed along time.
NA corresponds to pause sections in `wave` (see `threshold`).

Note

The spectral entropy of a noisy signal will tend towards 1 whereas the spectral entropy of a pure tone signal will tend towards 0.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Toh, A. M., Togneri, R. & Nordholm, S. 2005 Spectral entropy as speech features for speech recognition. *Proceedings of PEECS*, pp. 60-65.

See Also

`sh`, `th`

Examples

```
data(orni)
csh(orni,f=22050,wl=512,ovlp=50)
# using the threshold argument can lead to some edge effets
# here sh=1 at the end of echemes
csh(orni,f=22050,wl=512,ovlp=50,threshold=5)
```

`cutspec` *Cut a frequency spectrum*

Description

This function can be used to select (cut) a specific part of a frequency spectrum.

Usage

```
cutspec(spec, f = NULL, flim, norm = FALSE, PMF = FALSE)
```

Arguments

`spec`	a vector or a two-column matrix set resulting of a spectral analysis (not in dB) This can be the value obtained with `spec` or `meanspec`.
`f`	sampling frequency of `spec` (in Hz).
`flim`	a vector of length 2 to specify the new frequency range (in kHz).
`norm`	a logical, if `TRUE` the spectrum returned is normalised between 0 and 1.
`PMF`	a logical, if `TRUE` the spectrum returned is a probability mass function.

Value

A new spectrum is returned. The class of the returned object is the one of the input object (`spec`)

Note

The sampling frequency `f` is not necessary if `spec` has been obtained with either `spec` or `meanspec`. This function can be used before calling analysis function like `sh` or `sfm`. See examples.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`spec`, `meanspec`

Examples

```
data(orni)
a<-meanspec(orni,f=22050,plot=FALSE)
b<-cutspec(a,flim=c(4,8))
# quick check with a plot
plot(b,type="l")
# effects on spectral entropy
sfm(a)
sfm(b)
```

---

cutw *Cut a section of a time wave*

Description

This function selects and cuts a section of data describing a time wave. Original and cut sections can be plotted as oscillograms for comparison.

Usage

```
cutw(wave, f, from = NULL, to = NULL, choose = FALSE,
plot = FALSE, marks = TRUE, Sample = FALSE,...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`from`	start mark (in s).
`to`	end mark (in s).
`choose`	logical, if `TRUE` start (=`from`) and end (=`to`) points can be graphically chosen with a cursor on the oscillogram.
`plot`	logical, if `TRUE` returns an oscillographic plot of original and cut sections (by default `FALSE`).
`marks`	logical, if `TRUE` shows the start and end mark on the plot (by default `TRUE`).
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `oscillo` graphical parameters.

Details

If `plot` is `TRUE` returns a two-frame plot with both original and cut sections.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`oscillo`, `addsilw`,`deletew`, `fadew`,`mutew`,`pastew`,`revw`, `zapsilw`

Examples

```
# a 0.4 s section in a bird song
data(tico)
a<-cutw(tico,f=22050,from=0.5,to=0.9)
oscillo(a,22050)
# a direct way to see what has been cut
cutw(tico,f=22050,from=0.5,to=0.9,plot=TRUE)
```

`dBscale` *dB colour scale for a spectrogram display*

Description

This function displays a vertical or horizontal dB colour scale to be used with `spectro` plots.

Usage

```
dBscale(collevels, palette = spectro.colors, side = 4,
textlab = "Amplitude\n(dB)", cexlab = 0.75,
fontlab = 1, collab = "black", colaxis = "black",...)
```

Arguments

`collevels`	a set of levels which are used to partition the amplitude range of the spectrogram (in dB).
`palette`	a color palette function to be used to assign colors in the plot, see `note`.
`side`	side of the axis.
`textlab`	text of the label.
`cexlab`	character size of the label.
`fontlab`	font of the label.
`collab`	colour of the label.
`colaxis`	colour of the axis.
`...`	other `axis` arguments.

Note

This function, based on `filled.contour` by Ross Ihaka, is not supposed to be used by itself but as a legend of `spectro`.
Any colour palette can be used. In particular, it is possible to use other palettes coming with **seewave**: `rev.gray.colors.1`, `rev.gray.colors.2`, `rev.heat.colors`, `rev.terrain.colors`, `rev.topo.colors`, `rev.cm.colors` corresponding to the reverse of `heat.colors`, `terrain.colors`, `topo.colors`, `cm.colors`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

See Also

`spectro`.

Examples

```
data(pellucens)
# place the scale on the left and not on the right as spectro() does
def.par <- par(no.readonly = TRUE)
layout(matrix(c(1, 2), nc = 2), widths = c(1, 5))
par(mar=c(5,3,4,2))
dBscale(collevels=seq(-30,0,1),side=2)
par(mar=c(5,4,4,2))
spectro(pellucens, f=22050,wl=512,scale=FALSE)
par(def.par)
# place the scale on the top and not on the right as spectro() does
def.par <- par(no.readonly = TRUE)
layout(matrix(c(0,1,2,2), nc = 2, byrow=TRUE),widths=c(1,2),heights=(c(1,5.5)))
par(mar=c(0.5,3,4,2))
dBscale(collevels=seq(-30,0,1), textlab = "",side=3)
mtext("Amplitude (dB)",side=2,line = 1,at=0.6,cex=0.75)
par(mar=c(5,4,0.5,2))
spectro(pellucens, f=22050,wl=512,scale=FALSE)
par(def.par)
```

---

dBweight *dB weightings*

Description

This function returns the four most common dB weightings.

Usage

```
dBweight(f, dBref = NULL)
```

Arguments

`f`	frequency (in Hz).
`dBref`	dB reference level (by default `NULL`).

Details

By default, the function returns four weightings. When `dBref` is not `NULL` then the function returns the conversion from a dB reference level to four dB weighting levels.

Value

`dBweight` returns a list of four items corresponding to four dB weightings.

`A`	dB (A)
`B`	dB (B)
`C`	dB (C)
`D`	dB (D)

Note

The transfer equations used here come from Wipipedia but they were originally coming from the appendix of an international standard on the design performance of sound level meters IEC 651:1979 (Neil Glenister, pers. com.).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Zev Ross

References

http://en.wikipedia.org/wiki/A-weighting.

See Also

`convSPL`, `moredB`

Examples

```
# weight for a 50 Hz frequency
dBweight(f=50)
# A weight for the 1/3 Octave centre frequencies.
dBweight(f=c(20,25,31.5,40,50,63,80,100,125,160,200,250,
315,400,500,630,800,1000,1500,
1600,2000,2500,3150,4000,5000,
6300,8000,10000,12500,16000,20000))$A
# correction for a 50 Hz sound emitted at 100 dB
dBweight(f=50, dB=100)
# weighting curves plot
f <- seq(10,20000,by=10)
par(las=1)
```

```
plot(f, dBweight(f)$A, type="n", log="x",
xlim=c(10,10^5),ylim=c(-80,20),xlab="",ylab="",xaxt="n",yaxt="n")
abline(v=c(seq(10,100,by=10),seq(100,1000,by=100),
seq(1000,10000,by=1000),seq(10000,100000,by=10000),
c(100,1000,10000,100000)),col="lightgrey",lty=2)
abline(v=c(100,1000,10000,100000),col="grey")
abline(h=seq(-80, 20, 20),col="grey")
par(new=TRUE)
plot(f, dBweight(f)$A, type="l", log="x",
xlab="Frequency (Hz)", ylab="dB",lwd=2, col="blue", xlim=c(10,10^5),ylim=c(-80,20))
title(main="Acoustic weighting curves (10 Hz -20 kHz)")
lines(x=f, y=dBweight(f)$B, col="green",lwd=2)
lines(x=f, y=dBweight(f)$C, col="red",lwd=2)
lines(x=f, y=dBweight(f)$D, col="black",lwd=2)
legend("bottomright",legend=c("dB(A)","dB(B)","dB(C)","dB(D)"),
lwd=2,col=c("blue","green","red","black"),bty="o",bg="white")
```

`deletew` *Delete a section of a time wave*

Description

This function selects and delete a section of data describing a time wave. Original section and section after deletion can be plotted as oscillograms for comparison.

Usage

```
deletew(wave, f, from = NULL, to = NULL, choose = FALSE, plot = FALSE,
marks = TRUE, Sample = FALSE,...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`from`	start position (in s).
`to`	end position (in s).
`choose`	logical, if `TRUE` start (=`from`) and end (=`to`) points can be graphically chosen with a cursor on the oscillogram.
`plot`	logical, if `TRUE` returns an oscillographic plot of original and cut sections (by default `FALSE`).
`marks`	logical, if `TRUE` shows the start and end mark on the plot (by default `TRUE`).
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `oscillo` graphical parameters.

Details

If `plot` is `TRUE` returns a two-frame plot with both original and resulting sections.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`oscillo`, `addsilw`,`cutw`, `fadew`, `mutew`, `pastew`, `revw`, `zapsilw`

Examples

```
# deletion a 0.4 s section in a bird song
data(tico)
a<-deletew(tico,f=22050,from=0.5,to=0.9)
oscillo(a,22050)
# a direct way to see what has been cut
deletew(tico,f=22050,from=0.5,to=0.9,plot=TRUE)
```

`dfreq` *Dominant frequency of a time wave*

Description

This function gives the dominant frequency (i. e. the frequency of highest amplitude) of a time wave.

Usage

```
dfreq(wave, f, wl = 512, wn = "hanning", ovlp = 0, threshold = NULL,
plot = TRUE, xlab = "Times (s)", ylab = "Frequency (kHz)",
ylim = c(0, f/2000), type ="l",...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`wl`	length of the window for the analysis (even number of points, by default = 512).
`wn`	window name, see `ftwindow` (by default `"hanning"`).

`ovlp`	overlap between two successive analysis windows (in %).
`threshold`	amplitude threshold for signal detection (in %).
`plot`	logical, if `TRUE` plots the dominant frequency against time (by default `TRUE`).
`xlab`	title of the x axis.
`ylab`	title of the y axis.
`ylim`	the range of y values.
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `plot` graphical parameters.

Value

When `plot` is `FALSE`, `dfreq` returns a two-column matrix, the first column corresponding to time in seconds (*x*-axis) and the second column corresponding to to dominant frequency in kHz (*y*-axis).
NA corresponds to pause sections in `wave` (see `threshold`).

Note

This function is based on `fft`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`spec`, `meanspec`,`spectro`.

Examples

```
data(tico)
dfreq(tico,f=22050,ovlp=50,threshold=5)
# overlay on spectrogram
spectro(tico,f=22050,ovlp=50,zp=16,scale=FALSE,
    collevels=seq(-40,0,1),palette=rev.terrain.colors)
par(new=TRUE,las=1)
dfreq(tico,f=22050,ovlp=50,threshold=6,col="red",lwd=2,
    ann=FALSE)
```

diffenv *Difference between two amplitude envelopes*

Description

This function estimates the surface difference between two amplitude envelopes.

Usage

```
diffenv(wave1, wave2, f, envt = "hil", msmooth = NULL, ksmooth = NULL,
plot = FALSE, lty1 = 1, lty2 = 2, col1 = 2, col2 = 4, cold = 8,
xlab = "Time (s)", ylab = "Amplitude", ylim = NULL, legend = TRUE, ...)
```

Arguments

wave1	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
wave2	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave1 and wave2 (in Hz). Does not need to be specified if wave1 and/or wave2 are/is of class ts, Sample, or Wave.
envt	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See env.
msmooth	a vector of length 2 to smooth the amplitude envelope with a mean sliding window. The first component is the window length (in number of points). The second component is the overlap between successive windows (in %). See env.
ksmooth	kernel smooth via kernel. See env.
plot	logical, if TRUE plots both envelopes and their surface difference (by default FALSE).
lty1	line type of the first envelope (envelope of wave1).
lty2	line type of the second envelope (envelope of wave2).
col1	colour of the first envelope (envelope of wave1).
col2	colour of the second envelope (envelope of wave2).
cold	colour of the surface difference.
xlab	title of the time axis.
ylab	title of the amplitude axis.
ylim	range of amplitude axis.
legend	logical, if TRUE adds a legend to the plot.
...	other plot graphical parameters.

Details

Envelopes of both waves are first transformed as probability mass functions (PMF).
Envelope difference is then computed according to:

$$D = \frac{\sum |env1 - env2|}{2}, withD \in [0,1].$$

Value

The difference is returned. This value is without unit. When `plot` is `TRUE`, both envelopes and their difference surface are plotted on the same graph.

Note

This method can be used as a relative distance estimation between different envelopes.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩.

References

Sueur, J., Pavoine, S., Hamerlynck, O. & Duvail, S. (2008) - Rapid acoustic survey for biodiversity appraisal. *PLoS ONE*, 3(12): e4065.

See Also

`env`, `corenv`, `diffspec`, `diffwave`

Examples

```
data(tico)
data(orni)
# selection in tico of two waves with similar duration (dim)
tico2<-as.matrix(tico[1:nrow(orni),1])
diffenv(tico2,orni,f=22050,plot=TRUE)
# smoothing the envelope gives a better graph but slightly changes the result
diffenv(tico2,orni,f=22050,msmooth=c(20,0),plot=TRUE)
```

`diffspec` *Difference between two frequency spectra*

Description

This function estimates the surface difference between two frequency spectra.

Usage

```
diffspec(spec1, spec2, f = NULL, dB = FALSE, plot = FALSE, type="l",
lty1 = 1, lty2 = 2, col1 = 2, col2 = 4, cold = 8,
flab = "Frequency (kHz)", alab = "Amplitude",
flim = NULL, alim = NULL, legend = TRUE, ...)
```

Arguments

`spec1`	a first data set resulting of a spectral analysis obtained with `spec` or `meanspec` (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
`spec2`	a first data set resulting of a spectral analysis obtained with `spec` or `meanspec` (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
`f`	sampling frequency of waves used to obtain `spec1` and `spec2` (in Hz). Not necessary if `spec1` and/or `spec2` is a two-column matrix obtained with `spec` or `meanspec`.
`dB`	logical, if `TRUE` return the spectra and their surface difference in dB (by default `FALSE`).
`plot`	logical, if `TRUE` plots both spectra and their surface difference (by default `FALSE`).
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`lty1`	line type of `spec1` if `type`="l".
`lty2`	line type of `spec2` if `type`="l".
`col1`	colour of `spec1`.
`col2`	colour of `spec2`.
`cold`	colour of the surface difference.
`flab`	title of the frequency axis.
`alab`	title of the amplitude axis.
`flim`	the range of frequency values.
`alim`	range of amplitude axis.
`legend`	logical, if `TRUE` adds a legend to the plot.
`...`	other `plot` graphical parameters.

Details

Both spectra are first transformed as probability mass functions (PMF).
Spectral difference is then computed according to:

$$D = \frac{\sum |spec1 - spec2|}{2}, withD \in [0, 1].$$

with 0 < D <1.

Value

The difference is returned. This value is without unit. If `dB` is `TRUE`, the same value is returned in dB.
When `plot` is `TRUE`, both spectra and their difference surface are plotted on the same graph.

Note

This method can be used as a relative distance estimation between different spectra.
The dB value obtained can be very different from the one visually estimated when looking at the graph (`plot=TRUE`).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Sandrine Pavoine ⟨pavoine@mnhn.fr⟩.

References

Sueur, J., Pavoine, S., Hamerlynck, O. and Duvail, S. (2008). Rapid acoustic survey for biodiversity appraisal. *PLoS One*, 3(12): e4065.

See Also

`spec`, `meanspec`, `corspec`, `simspec`, `diffenv`

Examples

```
a<-noise(f=8000,d=1)
b<-synth(f=8000,d=1,cf=2000)
c<-synth(f=8000,d=1,cf=1000)
d<-noise(f=8000,d=1)
speca<-spec(a,f=8000,wl=512,at=0.5,plot=FALSE)
specb<-spec(b,f=8000,wl=512,at=0.5,plot=FALSE)
specc<-spec(c,f=8000,wl=512,at=0.5,plot=FALSE)
specd<-spec(d,f=8000,wl=512,at=0.5,plot=FALSE)
diffspec(speca,speca,f=8000)
#[1] 0 => similar spectra of course !
diffspec(speca,specb)
diffspec(speca,specc,plot=TRUE)
diffspec(specb,specc,plot=TRUE)
diffspec(speca,specd,plot=TRUE)
```

`diffwave` *Difference between two time waves*

Description

This function estimates the difference between two waves by computing the product between envelope surface difference and frequency surface difference.

Usage

```
diffwave(wave1, wave2, f, wl = 512, envt = "hil", msmooth = NULL, ksmooth = NULL)
```

Arguments

`wave1`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`wave2`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave1` and `wave2` (in Hz). Does not need to be specified if `wave1` and/or `wave2` are/is of class `ts`, `Sample`, or `Wave`.
`wl`	window length for spectral analysis (even number of points).
`envt`	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See `env`.
`msmooth`	a vector of length 2 to smooth the amplitude envelope with a mean sliding window. The first component is the window length (in number of points). The second component is the overlap between successive windows (in %). See `env`.
`ksmooth`	kernel smooth via `kernel`. See `env`.

Details

This function computes the product between the values obtained with `diffspec` and `diffenv` functions.
This then gives a global (time and frequency) estimation of dissimilarity.
The frequency mean spectrum and the amplitude envelope needed for computing respectively `diffspec` and `diffenv` are automatically generated. They can be controlled through `wl`, `msmooth` and `ksmooth` arguments respectively.
See examples below and examples in `diffspec` and `diffenv` for implications on the results.

Value

A single value varying between 0 and 1 is returned. The value has no unit.

Note

This method can be used as a relative distance estimation between different waves.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Sueur, J., Pavoine, S., Hamerlynck, O. & Duvail, S. (2008) - Rapid acoustic survey for biodiversity appraisal. *PLoS ONE*, 3(12): e4065.

See Also

diffspec, diffenv

Examples

```
data(tico)
data(orni)
# selection in tico to have two waves of similar duration (length)
tico<-as.matrix(tico[1:nrow(orni),1])
diffwave(tico,orni,f=22050)
# changing the frequency parameter (wl)
diffwave(tico,orni,f=22050,wl=1024)
# changing the temporal parameter (msmooth)
diffwave(tico,orni,f=22050,msmooth=c(20,0))
```

---

discrets *Time series discretisation*

Description

This function transforms a numeric (time) series into a sequence of symbols

Usage

```
discrets(x, symb = 5, collapse = TRUE)
```

Arguments

x	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
symb	the number of symbols used for the discretisation, can be set to 3 or 5 only.
collapse	logical, if TRUE, the symbols are pasted in a character string of length 1.

Details

The function partitions the numeric (time) series into a sequence of finite number of symbols. This symbols result of the comparaison of each series value with its temporal neighbours.
They are two discretisations available:
when symb is set to 3, each value will be replaced by either:
- *I* if the series is *I*ncreasing,
- *D* if the series is *D*ecreasing,
- *F* if the series remains *F*lat,
when symb is set to 5, each value will be replaced by either:
- *I* if the series is *I*ncreasing,
- *D* if the series is *D*ecreasing,
- *F* if the series remains *F*lat,
- *P* if the series shows a *P*eak,
- *T* if the series shows a *T*rough.

Value

A character string of length 1 if `collapse` is `TRUE`. Otherwise, a character string of length *n-2* if `symbol=3` (the first and last values cannot be replaced with a symbol) or *n-1* if `symbol=3` (the first value cannot be replaced with a symbol.)

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Cazelles, B. 2004 Symbolic dynamics for identifying similarity between rhythms of ecological time series. *Ecology Letters*, 7: 755-763.

See Also

`symba`

Examples

```
# a random variable
discrets(rnorm(30))
discrets(rnorm(30),symb=3)
# a frequency spectrum
data(tico)
spec1<-spec(tico,f=22050,at=0.2,plot=FALSE)
discrets(spec1[,2])
```

`drawenv` *Draw the amplitude envelope of a time wave*

Description

This function lets the user modifying the amplitude envelope of a time wave by drawing it with the graphics device

Usage

```
drawenv(wave, f, n = 20, plot = FALSE, listen = FALSE, Sample = FALSE)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`n`	the maximum number of points to draw the new envelope. Valid values start at 1.

plot	if TRUE returns the oscillogram of the new time wave (by default FALSE).
listen	if TRUE the new sound is played back.
Sample	if TRUE and plot is FALSE returns an object of class Sample.

Details

The function first plots an oscillogram view of wave.
The user has then to choose points on the positive side of the y-axis (amplitude). The junction of these points will draw a new amplitude envelope.
The order of points along the x-axis (time) is not important but points cannot be cancelled. When this process is finished the new time wave is returned in the console or as an oscillogram in a second graphics device if plot is TRUE.
The function uses locator.

Value

If plot is FALSE, a new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

setenv, env, synth

Examples

```
a<-synth(d=1,f=22050,cf=1000)
# drawenv(a,f=22050,plot=TRUE)
# choose points on the oscillogram view to draw a new enveloppe
# stop (ESC on Windows; right mouse button on Linux)
# check the result on the second graphics device opened thanks to plot=TRUE
```

---

dynspec *Dynamic sliding spectrum*

Description

This function plots dynamically a sliding spectrum along a time wave. This basically corresponds to a short-term Fourier transform.

Usage

```
dynspec(wave, f, wl = 512, wn = "hanning", zp = 0,
ovlp = 0, norm = FALSE, dB = FALSE, plot = TRUE,
title = TRUE, osc = FALSE, flab = "Frequency (kHz)",
alab = "Amplitude", alim = NULL, flim = c(0, f/2000),
type = "l", from = NULL, to = NULL, envt = NULL,
msmooth = NULL, ksmooth = NULL, colspec = "black",
coltitle = "black", colbg = "white", colline = "black",
colaxis = "black", collab = "black", cexlab = 1,
fontlab = 1, colwave = "black",
coly0 = "lightgrey", colcursor = "red", bty = "l")
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`wl`	if `at` is not null, length of the window for the analysis (even number of points, by defaults = 512).
`wn`	window name, see `ftwindow` (by default `"hanning"`).
`zp`	zero-padding (even number of points), see `Details`.
`ovlp`	overlap between two successive windows (in %).
`norm`	logical, if `TRUE` compute a normalised sliding spectrum.
`dB`	logical, if `TRUE` returns the sliding spectrum in dB (by default `FALSE`).
`plot`	logical, if `TRUE` plots in an ew graphics device the successive spectra sliding along the time wave (by default `TRUE`).
`title`	logical, if `TRUE` adds a title with the time position of the current spectrum along the time wave.
`osc`	logical, if `TRUE` plots an oscillogram beneath the sliding spectrum with a cursor showing the position of the current spectrum (by default `FALSE`).
`flab`	title of the frequency axis.
`alab`	title of the amplitude axis.
`flim`	range of frequency axis.
`alim`	range of amplitude axis.
`type`	type of plot that should be drawn for the sliding spectrum. See `plot` for details (by default "l" for lines).
`from`	start mark where to compute the sliding spectrum (in s).
`to`	end mark where to compute the sliding spectrum (in s).
`envt`	the type of envelope to be plooted: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See `env`.

msmooth	when env is not NULL, a vector of length 2 to smooth the amplitude envelope with a mean sliding window. The first component is the window length (in number of points). The second component is the overlap between successive windows (in %). See env.
ksmooth	when env is not NULL, kernel smooth via kernel. See env.
colspec	colour of the sliding spectrum.
coltitle	if title is TRUE, colour of the title.
colbg	background colour.
colline	colour of axes line.
colaxis	colour of the axes.
collab	colour of axes title.
cexlab	character size for axes title.
fontlab	font for axes title.
colwave	colour of the oscillogram or of the envelope (only when osc is TRUE).
coly0	colour of the y=0 line (only when osc is TRUE).
colcursor	colour of oscillogram cursor (only when osc is TRUE).
bty	the type of box to be drawn around the oscillogram (only when osc is TRUE).

Details

Use the slider panel to move along the time wave.
Use the argument norm if you wish to have each spectrum normalised, *i.e.* with values between 0 and 1 or maximised to 0 dB when dB is TRUE.
The function requires the package **rpanel** that is based on the package **tcltk**.

Value

If plot is FALSE, this function returns a matrix which columns correspond to the spectra computed along the time wave.

Note

This function is very similar to a spectrogram. See the Details of spectro for some information regarding the short term Fourier transform.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

See Also

spectro, spectro3D, wf, spec, fft, oscillo.

Examples

```
data(sheep)
dynspec(sheep,f=8000,wl=1024,ovlp=50,osc=TRUE)
dev.off()
```

echo *Echo generator*

Description

This function generate echoes of a time wave.

Usage

```
echo(wave, f, amp, delay, plot = FALSE,
listen = FALSE, Sample = FALSE, ...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
amp	a vector describing the relative amplitude of the successive echoes. Each value of the vector should in [0,1]
delay	a vector describing the time delays of the successive echoes from the beginning of wave (in s.)
plot	logical, if TRUE returns an oscillographic plot of the wave modified (by default FALSE).
listen	if TRUE the new sound is played back.
Sample	if TRUE and plot is FALSE returns an object of class Sample
...	other oscillo graphical parameters.

Details

amp and delay should strictly have the same length corresponding to the number of desired echoes.

Value

When plot is FALSE, a new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Note

This function is based on a convolution (convolve) between the input wave and a pulse echo filter.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Stoddard, P. K. (1998). Application of filters in bioacoustics. *In*: Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds), *Animal acoustic communication*. Springer, Berlin, Heidelberg,pp. 105-127.

See Also

synth

Examples

```
# generation of the input wave
a<-synth(f=11025,d=1,cf=2000,shape="tria",am=c(50,10),fm=c(1000,10,1000))
# generation of three echoes
# with respectively a relative amplitude of 0.8, 0.4, and 0.2
# and with a delay of 1s, 2s, and 3s  from the beginning of the input wave
aecho<-echo(a,f=11025,amp=c(0.8,0.4,0.2),delay=c(1,2,3))
# oscillographic output to see what we have generated
op<-par(mfrow=c(2,1))
oscillo(a,f=11025,title="Input signal")
oscillo(aecho,f=11025,colwave="blue",title="Signal with echoes",coltitle="blue")
par(op)
# another echo with time delays overlapping with the input wave
echo(a,f=11025,amp=c(0.4,0.2,0.4),delay=c(0.6,0.8,1.5),plot=TRUE,listen=TRUE)
```

---

env *Amplitude envelope of a time wave*

Description

This function returns the absolute or Hilbert amplitude envelope of a time wave.

Usage

```
env(wave, f, envt = "hil", msmooth = NULL,
ksmooth = NULL, norm = FALSE, plot = TRUE, k = 1, j = 1, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`envt`	the type of envelope to be returned: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See Details section.
`msmooth`	a vector of length 2 to smooth the amplitude envelope with a mean sliding window. The first component is the window length (in number of points). The second component is the overlap between successive windows (in %). See examples.

ksmooth	kernel smooth via kernel. See examples.
norm	a logical, if TRUE the amplitude of the envelope is normalised between 0 and 1.
plot	logical, if TRUE returns a plot of wave envelope (by default TRUE).
k	number of horizontal sections when plot is TRUE (by default =1).
j	number of vertical sections when plot is TRUE (by default =1).
...	other oscillo graphical parameters.

Details

When envt is set as "abs", the amplitude envelope returned is the absolute value of wave.
When envt is set as "hil", the amplitude envelope returned is the modulus (Mod) of the analytical signal of wave obtained through the Hilbert transform (hilbert).

Value

Data are returned as one-column matrix when plot is FALSE.

Note

Be aware that smoothing with either msmooth or ksmooth changes the original number of points describing wave.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

oscillo,hilbert

Examples

```
data(tico)
# Hilbert amplitude envelope
env(tico,f=22050)
# absolute amplitude envelope
env(tico,f=22050,envt="abs")
# smoothing with a 10 points and 50
env(tico,f=22050,msmooth=c(10,50))
# smoothing kernel
env(tico,f=22050,ksmooth=kernel("daniell",10))
# overplot of oscillographic and envelope representations
oscillo(tico,f=22050)
par(new=TRUE)
env(tico,f=22050,colwave=2)
```

`export` *Export sound data*

Description

Export sound data as a text file that can be read by a sound player like 'Goldwave'

Usage

```
export(wave, f, filename = NULL, header=TRUE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`filename`	name of the new file. (by default the name of `wave`).
`header`	either a logical or a character vector, if `TRUE` add a header to be read by Goldwave, if `FALSE` does not add any header, if a character vector add the character vector as a header.
`...`	other `write.table` parameters.

Details

Creates a new text file with a header describing the main features of the sound (wave). For instance, for a 2 s sound with a sampling frequency of 8000 Hz, the header will be: [ASCII 8000Hz, Channels: 1, Samples: 160000, Flags: 0]. This type of file can be read by sound players like Goldwave (`http://www.goldwave.com/`).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

Examples

```
a<-synth(f=8000,d=2,cf=2000,plot=FALSE)
export(a,f=8000)
unlink("a.txt")
```

fadew *Fade in and fade out of a time wave*

Description

This function applies a “fade in” and/or a “fade out” to a time wave following a linear, exponential or cosinus-like shape.

Usage

```
fadew(wave, f, din = 0, dout = 0, shape = "linear", plot = FALSE,
listen = FALSE, Sample = FALSE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`din`	fade in duration
`dout`	fade out duration
`shape`	fade shape, `"linear"`, `"exp"` for exponential, `"cos"` for cosinus-like, (by default `"linear"`)
`plot`	logical, if `TRUE` returns an oscillographic plot of the wave modified (by default `FALSE`).
`listen`	if `TRUE` the new sound is played back.
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `oscillo` graphical parameters.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`oscillo`, `addsilw`, `cutw`, `deletew`,`mutew`, `pastew`, `revw`, `zapsilw`

Examples

```
a<-noise(d=5,f=4000)
op<-par(mfrow=c(3,1))
fadew(a,f=4000,din=1,dout=2,plot=TRUE,title="Linear",cexlab=0.8)
fadew(a,f=4000,din=1,dout=2,shape="exp",plot=TRUE,title="Exponential shape",
    colwave="blue",coltitle="blue",cexlab=0.8)
fadew(a,f=4000,din=1,dout=2,shape="cos",plot=TRUE,title="Cosinus-like shape",
    colwave="red",coltitle="red",cexlab=0.8)
par(op)
```

---

fdoppler *Doppler effect*

Description

This function computes the altered frequency of a moving source due to the Doppler effect.

Usage

```
fdoppler(f, c = 340, vs, vo = 0, movs = "toward", movo = "toward")
```

Arguments

f	original frequency produced by the source (in Hz or kHz)
c	speed of sound in meters/second.
vs	speed of the source in meters/second.
vo	speed of the observer in meters/second. The observer is static by default *i.e.* vo = 0
movs	movement direction of the source in relation with observer position, either "toward" (by default) or "away".
movo	movement direction of the observer in relation with the source position, either "toward" (by default, but be aware that the observer is static by default) or "away".

Details

The altered frequency *f'* is computed according to:

$$f' = f \times \frac{c \pm v_o}{c \pm v_s}$$

with *f* = original frequency produced by the source (in Hz or kHz),
vs = speed of the source,
vo = speed of the observer.

Value

The altered frequency is returned in a vector.

Note

You can use `wasp` to have exact values of `c`. See examples.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

http://www.kettering.edu/~drussell/Demos/doppler/doppler.html.

See Also

`wasp`

Examples

```
# a 400 Hz source moving toward or away from the observer at 85 m/s
fdoppler(f=400,vs=85)
# [1] 533.3333
fdoppler(f=400,vs=85,movs="away")
# [1] 320
# use wasp() if you wish to have exact sound speed at a specific temperature
fdoppler(f=wasp(f=400,t=25)$c, vs=85)
# [1] 461.8667
# Doppler effect at different source speeds
f<-seq(1,10,by=1); lf<-length(f)
v<-seq(10,300,by=20); lv<-length(v)
res<-matrix(numeric(lf*lv),ncol=lv)
for(i in 1:lv) res[,i]<-fdoppler(f=f,vs=v[i])
op<-par(bg="lightgrey")
matplot(x=f,y=res,type="l",lty=1,las=1,col= spectro.colors(lv),
xlab="Source frequency (kHz)", ylab="Altered frequency (kHz)")
legend("topleft",legend=paste(as.character(v),"m/s"),
lty=1,col= spectro.colors(lv))
title(main="Doppler effect at different source speeds")
par(op)
```

---

ffilter *Frequency filter*

Description

This function filters out a selected frequency section of a time wave (low-pass, high-pass, low-stop, high-stop, bandpass or bandstop frequency filter).

Usage

```
ffilter(wave, f, from = FALSE, to = FALSE, bandpass = TRUE,
custom = NULL, wl = 512, wn = "hanning", Sample = FALSE)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
from	start frequency (in Hz) where to apply the filter.
to	end frequency (in Hz) where to apply the filter.
bandpass	if TRUE a band-pass filter is applied between from and to, if FALSE a band-stop filter is applied between from and to (by default TRUE).
custom	a vector describing the frequency response of a custom filter. This can be manually generated or obtained with spec and meanspec. wl is no more required. See examples.
wl	window length for the analysis (even number of points).
wn	window name, see ftwindow (by default "hanning").
Sample	if TRUE and plot is FALSE returns an object of class Sample.

Details

A short-term Fourier transform is first applied to the signal (see spectro), then the frequency filter is applied and the new signal is eventually generated using the reverse of the Fourier Transform (fft).
There is therefore neither temporal modifications nor amplitude modifications.

Value

A new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

afilter,lfs,fir

Examples

```
a<-noise(f=8000,d=1)
# low-pass
b<-ffilter(a,f=8000,to=1500)
spectro(b,f=8000,wl=512)
# high-pass
c<-ffilter(a,f=8000,from=2500)
spectro(c,f=8000,wl=512)
# band-pass
d<-ffilter(a,f=8000,from=1000,to=2000)
spectro(d,f=8000,wl=512)
```

```
# band-stop
e<-ffilter(a,f=8000,from=1500,to=2500,bandpass=FALSE)
spectro(e,f=8000,wl=512)
# custom
myfilter1<-rep(c(rep(0,32),rep(1,32)),4)
g<-fir(a,f=8000,custom=myfilter1)
spectro(g,f=8000)
```

`field` *Near field and far field limits*

Description

This function helps in knowing whether you are working in the near or far field.

Usage

```
field(f, d)
```

Arguments

`f`	frequency (Hz)
`d`	distance from the sound source (m)

Details

Areas very close to the sound source are in the near-field where the contribution of particle velocity to sound energy is greater thant that of sound pressure and where these components are not in phase. Sound propagation properties are also different near or far from the source. It is therefore important to know where the microphone was from the source.
To know this, the product k*d is computed according to:

$$k \times d = \frac{f}{c} \times d$$

with d = distance from the source (m), f = frequency (Hz) and c = sound celerity (m/s).
If k*d is greatly inferior 1 then the microphone is in the near field.
The decision help returned by the function follows the rule:
far field:

$$k \times d > 1$$

between near and far field limits:

$$0.1 \leq k \times d \leq 1$$

near field:

$$k \times d < 0.1$$

.

Value

A list of two values is returned:

`kd`	the numeric value k*d used to take a decision
`d`	a character string giving the help decision.

Note

This function works for air-borne sound only.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

Examples

```
# 1 kHz near field at 1 cm from the source
field(f=1000,d=0.01)
# playing with distance from source and sound frequency
op<-par(bg="lightgrey")
D<-seq(0.01,0.5,by=0.01); nD<-length(D)
F<-seq(100,1000,by=25); nF<-length(F)
a<-matrix(numeric(nD*nF),nrow=nD)
for(i in 1:nF) a[,i]<-field(f=F[i],d=D)$kd
matplot(x=D,y=a,type="l",lty=1,col= spectro.colors(nF),
  xlab="Distance from the source (m)", ylab="k*d")
title("Variation of the product k*d with distance and frequency")
text(x=c(0.4,0.15),y=c(0.02,1), c("Near Field","Far Field"),font=2)
legend(x=0.05,y=1.4,c("100 Hz","1000 Hz"),lty=1,
  col=c(spectro.colors(nF)[1],spectro.colors(nF)[nF]),bg="grey")
abline(h=0.1)
par(op)
```

`fir` *Finite Impulse Response filter*

Description

This function is a FIR filter that filters out a selected frequency section of a time wave (low-pass, high-pass, low-stop, high-stop, bandpass or bandstop frequency filter).

Usage

```
fir(wave, f, from = FALSE, to = FALSE, bandpass = TRUE, custom = NULL,
wl = 512, wn = "hanning", listen = FALSE, Sample= FALSE)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
from	start frequency (in Hz) where to apply the filter.
to	end frequency (in Hz) where to apply the filter.
bandpass	if TRUE a band-pass filter is applied between from and to, if FALSE a band-stop filter is applied between from and to (by default TRUE).
custom	a vector describing the frequency response of a custom filter. This can be manually generated or obtained with spec and meanspec. wl is no more required. See examples.
wl	window length of the impulse filter (even number of points).
wn	window name, see ftwindow (by default "hanning").
listen	if TRUE the new sound is played back.
Sample	if TRUE and plot is FALSE returns an object of class Sample

.

Details

This function is based on the reverse of the Fourier Transform (fft) and on a convolution (convolve) between the wave to be filtered and the impulse filter.

Value

A new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Stoddard, P. K. (1998). Application of filters in bioacoustics. *In*: Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds), *Animal acoustic communication*. Springer, Berlin, Heidelberg,pp. 105-127.

See Also

ffilter,lfs, afilter, fir1, fir2

Examples

```
a<-noise(f=8000,d=1)
# low-pass
b<-fir(a,f=8000,to=1500)
spectro(b,f=8000)
# high-pass
c<-fir(a,f=8000,from=2500)
spectro(c,f=8000)
# band-pass
d<-fir(a,f=8000,from=1000,to=2000)
spectro(d,f=8000)
# band-stop
e<-fir(a,f=8000,from=1500,to=2500,bandpass=FALSE)
spectro(e,f=8000)
# custom filter manually generated
myfilter1<-rep(c(rep(0,32),rep(1,32)),4)
g<-fir(a,f=8000,custom=myfilter1)
spectro(g,f=8000)
# custom filter generated using spec()
data(tico)
myfilter2<-spec(tico,f=22050,at=0.7,wl=512,plot=FALSE)
b<-noise(d=1,f=22050)
h<-fir(b,f=22050,custom=myfilter2)
spectro(h,f=22050)
```

`fma` *Frequency modulation analysis*

Description

This function computes the Fourier analysis of the instantaneous frequency of a time wave. This allows to detect periodicity in frequency modulation.

Usage

```
fma(wave, f, threshold = NULL, plot = TRUE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`threshold`	amplitude threshold for signal detection (in %).
`plot`	logical, if `TRUE` the spectrum of the instantaneous frequency (by default `TRUE`).
`...`	other `spec` parameters.

Details

This function is based on `ifreq` and `spec`.
The instantaneous frequency of `wave` is first computed and the spectrum of this frequency modulation is then processed. All `env` and `spec` arguments can be set up.

Value

If `plot` is `FALSE`, `fma` returns a numeric vector corresponding to the computed spectrum. If `peaks` is not `NULL`, `fma` returns a list with two elements:

`spec`	the spectrum computed
`peaks`	the peaks values (in kHz).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`ifreq`, `hilbert`, `spec`, `ama`

Examples

```
# a sound with a 1 Khz sinusoid FM
a<-synth(d=1, f=8000, cf=1500, fm=c(1000,1000,0))
fma(a,f=8000)
```

`ftwindow` *Fourier transform windows*

Description

Generates different Fourier Transform windows.

Usage

```
ftwindow(wl, wn = "hamming")
```

Arguments

`wl`	window length
`wn`	window name: `bartlett`, `blackman`, `flattop`, `hamming`, `hanning`, or `rectangle` (by default `hamming`).

Value

A vector of length `wl`.

Note

Try the example to see windows shape.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Harris, F.J., 1978. On the use of windows for harmonic analysis with the discrete Fourier Transform. *Proceedings of the IEEE*, 66(1): 51-83.

See Also

covspectro, dfreq, meanspec, spec, spectro, spectro3D

Examples

```
a<-ftwindow(512)
b<-ftwindow(512,wn="bartlett")
c<-ftwindow(512,wn="blackman")
d<-ftwindow(512,wn="flattop")
e<-ftwindow(512,wn="hanning")
f<-ftwindow(512,wn="rectangle")
all<-cbind(a,b,c,d,e,f)
matplot(all,type="l",col=1:6,lty=1:6)
legend(legend=c("hamming","bartlett","blackman","flattop","hanning","rectangle"),
x=380,y=0.95,col=1:6,lty=1:6,cex=0.75)
```

---

fund *Fundamental frequency track*

Description

This function tracks the fundamental frequency through a short-term cepstral transform.

Usage

```
fund(wave, f, wl = 512, ovlp = 0, fmax, threshold = NULL,
plot = TRUE, xlab = "Time (s)", ylab = "Frequency (kHz)",
ylim = c(0, f/2000), ...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
wl	if at is not null, length of the window for the analysis (even number of points, by defaults = 512).
ovlp	overlap between two successive windows (in %).
fmax	the maximum frequency to detect (in Hz).
threshold	amplitude threshold for signal detection (in %).
plot	logical, if TRUE plots the fundamental frequency modulations against time (by default TRUE).
xlab	title of the time axis (s).
ylab	title of the frequency axis (Hz).
ylim	the range of frequency values.
...	other plot graphical parameters.

Value

When plot is FALSE, fund returns a two-column matrix, the first column corresponding to time in seconds (*x*-axis) and the second column corresponding to to fundamental frequency in kHz (*y*-axis).
NA corresponds to pause sections in wave (see threshold).

Note

This function is based on ceps.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩.

References

Oppenheim, A.V. and Schafer, R.W. 2004. From frequency to quefrency: a history of the cepstrum. *Signal Processing Magazine IEEE*, 21: 95-106.

See Also

cepstro, ceps, autoc

Examples

```
data(sheep)
fund(sheep,f=8000,fmax=300,type="l")
# with 50
# amplitude filter (threshold)
fund(sheep,f=8000,fmax=300,type="b",ovlp=50,threshold=5,ylim=c(0,1),cex=0.5)
# overlaid on a spectrogram
spectro(sheep,f=8000,ovlp=75,zp=16,scale=FALSE,palette=rev.gray.colors.2)
par(new=TRUE)
fund(sheep,f=8000,fmax=300,type="p",pch=24,ann=FALSE,
  xaxs="i",yaxs="i",col="black",bg="red",threshold=6)
```

H *Total entropy*

Description

This function estimates the total entropy of a time wave.

Usage

```
H(wave, f, wl = 512, envt="hil", msmooth = NULL, ksmooth = NULL)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`wl`	window length for spectral entropy analysis (even number of points). See `sh`.
`envt`	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See `env`.
`msmooth`	a vector of length 2 to smooth the amplitude envelope with a mean sliding window. The first component is the window length (in number of points). The second component is the overlap between successive windows (in %). See `env`.
`ksmooth`	kernel smooth via `kernel`. See `env`.

Details

This function computes the product between the values obtained with `sh` and `th` functions.
This then gives a global (time and frequency) estimation of signal entropy.
The frequency mean spectrum and the amplitude envelope needed for computing respectively `sh` and `th` are automatically generated. They can be controlled through `wl` and `smooth` arguments respectively. See examples below and examples in `sh` and `th` for implications on the results.

Value

A single value varying between 0 and 1 is returned. The value has no unit.

Note

The entropy of a noisy signal will tend towards 1 whereas the entropy of a pure tone signal will tend towards 0.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Sueur, J., Pavoine, S., Hamerlynck, O. & Duvail, S. (2008) - Rapid acoustic survey for biodiversity appraisal. *PLoS ONE*, 3(12): e4065.

See Also

sh, th, csh

Examples

```
data(orni)
H(orni,f=22050)
# changing the spectral parameter (wl)
H(orni,f=22050,wl=1024)
# changing the temporal parameter (msmooth)
H(orni,f=22050,msmooth=c(20,0))
```

---

hilbert *Hilbert transform and analytic signal*

Description

This function returns the analytic signal of a time wave through Hilbert transform.

Usage

```
hilbert(wave, f)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.

Details

The analytic signal is useful to get the amplitude envelope (see argument `henv` of `oscillo` and the instantaneous phase or frequency (see `ifreq`) of a time wave.

Value

`hilbert` returns the analytic signal as a complex matrix. The imaginary part of this matrix is the Hilbert transform.

Note

To get the Hilbert component only, use `Im(Hilbert(wave))`.

Author(s)

Jonathan Lees ⟨jonathan.lees@unc.edu⟩

References

Mbu Nyamsi, R. G., Aubin, T. & Bremond, J. C. 1994 On the extraction of some time dependent parameters of an acoustic signal by means of the analytic signal concept. Its application to animal sound study. *Bioacoustics*, 5: 187-203.

See Also

`ifreq`, argument `henv` of `oscillo`

Examples

```
a<-synth(f=8000, d=1, cf=1000)
aa<-hilbert(a, f=8000)
```

`ifreq` *Instantaneous frequency*

Description

This function returns the instantaneous frequency (and/or phase) of a time wave through the computation of the analytic signal (Hilbert transform).

Usage

```
ifreq(wave, f, phase = FALSE, threshold = NULL,
plot = TRUE, xlab = "Time (s)", ylab = NULL,
ylim = NULL, type = "l", ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`phase`	if `TRUE` and `plot` is also `TRUE` plots the instantaneous phase instead of the instantaneous frequency.
`threshold`	amplitude threshold for signal detection (in %).
`plot`	logical, if `TRUE` plots the instantaneous frequency or phase against time (by default `TRUE`).
`xlab`	title of the x axis.
`ylab`	title of the y axis.
`ylim`	the range of y values.
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `plot` graphical parameters.

Details

The instantaneous phase is the argument of the analytic signal obtained throught the Hilbert transform.
The instantaneous phase is then unwrapped and derived against time to get the instantaneous frequency.
There may be some edge effects at both start and end of the time wave.

Value

If `plot` is `FALSE`, `ifreq` returns a list of two components:

`p`	a two-column matrix, the first column corresponding to time in seconds (*x*-axis) and the second column corresponding to wrapped instantaneous phase in radians (*y*-axis).
`f`	a two-column matrix, the first column corresponding to time in seconds (*x*-axis) and the second column corresponding to instantaneous frequency in kHz (*y*-axis).

Note

This function is based on the analytic signal obtained with the Hilbert transform (see `hilbert`).
The function requires the package **signal**.
The matrix describing the instantaneous phase has one more row than the one describing the instantaneous frequency.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Mbu Nyamsi, R. G., Aubin, T. & Bremond, J. C. 1994 On the extraction of some time dependent parameters of an acoustic signal by means of the analytic signal concept. Its application to animal sound study. *Bioacoustics*, 5: 187-203.

See Also

`hilbert`, `zc`

Examples

```
# generate a sound with sine and linear frequency modulations
a<-synth(d=1, f=8000, cf=1500, fm=c(200,10,1000))
# plot on a single graphical device the instantaneous frequency and phase
op<-par(mfrow=c(2,1))
ifreq(a,f=8000,main="Instantaneous frequency")
ifreq(a,f=8000,phase=TRUE,main="Instantaneous phase")
par(op)
```

---

lfs *Linear Frequency Shift*

Description

This function linearly shifts all the frequency content of a time wave.

Usage

```
lfs(wave, f, shift, wl = 128, wn = "hanning", Sample = FALSE)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`shift`	positive or negative frequency shift to apply (in Hz).)
`wl`	window length for the analysis (even number of points, by default = 512).
`wn`	window name, see `ftwindow` (by default `"hanning"`).
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`

.

Details

A short-term Fourier transform is first applied to the signal (see `spectro`), then the frequency shift is applied and the new signal is eventually generated using the reverse of the Fourier Transform (`fft`).
There is therefore neither temporal modifications nor amplitude modifications.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Thierry Aubin ⟨thierry.aubin@u-psud.fr⟩

References

Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds) 1998. *Animal acoustic communication*. Springer, Berlin, Heidelberg.

See Also

`ffilter`, `spectro`

Examples

```
data(orni)
a<-lfs(orni,f=22050,shift=1000)
spectro(a,f=22050)
# to be compared with the original signal
spectro(orni,f=22050)
```

`listen` *Play a sound wave*

Description

Play a sound wave

Usage

```
listen(wave, f, from = NULL, to = NULL, choose = FALSE)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`from`	start of play (in s).
`to`	end of play (in s).
`choose`	logical, if `TRUE` start (=`from`) and end (=`to`) points can be graphically chosen with a cursor on the oscillogram.

Note

This function is based on `play` but allows to read one-colum matrix, data.frame and Sample objects.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ but the original `play` function is by Matthias Heymann (package **sound**).

See Also

`play`

Examples

```
data(tico)
listen(tico,f=22050)
listen(tico,f=22050,from=0.5,to=1.5)
listen(noise(d=1,f=8000,Sample=TRUE))
# change f to play the sound a different speed
data(sheep)
# normal
listen(sheep,f=8000)
# two times faster
listen(sheep,f=8000*2)
# two times slower
listen(sheep,f=8000/2)
```

`meanspec` *Mean frequency spectrum of a time wave*

Description

This function returns the mean frequency spectrum (i.e. the mean relative amplitude of the frequency distribution) of a time wave. Results can be expressed either in absolute or dB data.

Usage

```
meanspec(wave, f, wl = 512, wn = "hanning", ovlp = 0, PSD = FALSE,
PMF = FALSE, dB = NULL, dBref = NULL, from = NULL, to = NULL, peaks = NULL,
identify = FALSE, col = "black", cex = 1, colpeaks = "red",
cexpeaks = 1,  fontpeaks = 1, plot = 1, flab = "Frequency (kHz)",
alab = "Amplitude", flim = c(0,f/2000),alim = NULL, type ="l",...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
wl	length of the window for the analysis (even number of points, by default = 512).
wn	window name, see ftwindow (by default "hanning").
ovlp	overlap between two successive analysis windows (in %).
PSD	if TRUE return Power Spectra Density, *i. e.* the square of the spectra.
PMF	if TRUE return Probability Mass Function, *i. e.* the probability distribution of frequencie.
dB	a character string specifying the type dB to return: "max0" for a maximum dB value at 0, "A", "B", "C" and "D" for common dB weights.
dBref	a dB reference value when dB is not NULL. NULL by default but should be set to 2*10e-5 for a 20 microPa reference (SPL).
from	start mark where to compute the spectrum (in s).
to	end mark where to compute the spectrum (in s).
peaks	if not NULL returns peaks value for a given span (see details).
identify	to identify frequency and amplitude values on the plot with the help of a cursor.
col	colour of the spectrum.
cex	pitch size.
colpeaks	colour of peaks value plotted on the spectrum.
cexpeaks	character size of peaks value plotted on the spectrum.
fontpeaks	font of peaks value plotted on the spectrum.
plot	if 1 returns frequency on x-axis, if 2 returns frequency on y-axis, (by default 1).
flab	title of the frequency axis.
alab	title of the amplitude axis.
flim	range of frequency axis (in kHz).
alim	range of amplitude axis.
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
...	other plot graphical parameters.

Details

If from and to are FALSE then spec computes the mean spectrum of the whole signal.
peaks setting corresponds to dimension of embed. See examples of spec.

Value

If `plot` is `FALSE`, `meanspec` returns a two columns matrix, the first column corresponding to the frequency axis, the second column corresponding to the amplitude axis.
If `identify` is `TRUE`, `spec` returns a list with two elements:

`freq`	the frequency of the points chosen on the spectrum
`amp`	the relative amplitude of the points chosen on the spectrum
`spec`	the spectrum computed
`peaks`	the peaks values (in kHz).

Note

See examples of `spec`. This function is based on `fft`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Martin Maechler ⟨maechler@stat.math.ethz.ch⟩ for `peaks`

See Also

`spec`, `dynspec`, `corspec`, `diffspec`, `simspec`, `fft`.

Examples

```
data(orni)
# compute the mean spectrum of the whole time wave
meanspec(orni,f=22050)
# compute the mean spectrum of a time wave section (from 0.32 s to 0.39 s)
meanspec(orni,f=22050,from=0.32,to=0.39)
# different window lengths
op<-par(mfrow=c(3,1))
meanspec(orni,f=22050,wl=256)
title("wl=256")
meanspec(orni,f=22050,wl=1024)
title("wl=1024")
meanspec(orni,f=22050,wl=4096)
title("wl=4096")
par(op)
# different overlap values (almost no effects here...)
op<-par(mfrow=c(3,1))
meanspec(orni,f=22050)
title("ovlp=0")
meanspec(orni,f=22050,ovlp=50)
title("ovlp=50")
meanspec(orni,f=22050,ovlp=95)
title("ovlp=95")
par(op)
# use of flim to zoom in
op<-par(mfrow=c(2,1))
```

```
meanspec(orni,f=22050)
title("zoom in")
meanspec(orni,f=22050,wl=512,flim=c(4,6))
par(op)
# comparaison of spectrum and mean spectrum
op<-par(mfrow=c(2,1))
spec(orni,f=22050)
title("spec()")
meanspec(orni,f=22050)
title("meanspec()")
par(op)
```

---

mel *Hertz / Mel conversion*

Description

This function converts Hertz data in Mel data.

Usage

```
mel(x, inverse = FALSE)
```

Arguments

x	a value in Hertz (or in Mel if `inverse` is `TRUE`)
inverse	logical, if `TRUE` converts the Mel data in Hertz data.

Details

Hertz to mel conversion is computed according to:

$$m = 1127.01048 \times \log{(1 + (\frac{f}{700}))}, with m in Mel and f in Hertz.$$

with *m* in Mel and *f* in Hertz.

Mel to Hertz conversion (when `inverse` is `TRUE`) is therefore computed according to:

$$f = 700 \times (e^{\frac{m}{1127.01048}} - 1), with f in Hertz and m in Mel.$$

with *f* in Hertz and *m* in Mel.

Value

A corresponding **R** object is returned.

Note

The Mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The name Mel comes from the word melody to indicate that the scale is based on pitch comparisons. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 mels.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Stevens, S. S., Volkman, J. and Newman, E. B. 1937. A scale for the measurement of psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8: 185-190.

Examples

```
x<-seq(0,10000,by=50)
y<-mel(x)
plot(x,y,type="l",xlab = "f (hertz)", ylab = "f (mel)",
  main = "Mel scale", col="red")
```

`micsens` *Microphone sensitivity and conversion*

Description

This function converts microphone sensitivity from mV/Pa to dB.

Usage

```
micsens(x, sref = 1, inverse = FALSE)
```

Arguments

`x`	a measured sensitivity in mV/Pa (or in dB if `inverse` is `TRUE`)
`sref`	the sensitivity reference (by default equals to 1 V/Pa)
`inverse`	logical, if `TRUE`, the inverse conversion from dB to mV/Pa is computed.

Details

The sensitivity *S* in dB is calculated according to:

$$S_{dB} = 20 \times log_{10}(\frac{s}{s_{ref}}), withsthemeasuredsensitivityinmv/Paandsreftherefrencesensitivity(bydefault1mV/Pa$$

with *s* the measured sensitivity in mv/Pa and *sref* the reference sensitivity (by default 1 mV/Pa).

Value

A numeric value in dB *re* 1V/Pa with default settings, in mV/Pa if `inverse` is set to `FALSE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`convSPL`

Examples

```
# conversion of a sensitivity of 2 mV/Pa
micsens(2)
# conversion of a sensitivity of -54 dB re 1V/Pa
micsens(-54,inverse=TRUE)
```

`moredB` *Addition of dB values*

Description

This functions calculates the sum of dB values

Usage

```
moredB(x)
```

Arguments

`x`	a numeric vector or a matrix.

Details

The addition of dB values is not linear. See examples.

Value

A numeric vector or a matrix is returned.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Hartmann, W. M. 1998 *Signals, sound and sensation*. New York: Springer.

See Also

convSPL, dBweight

Examples

```
# two sources of 60 dB give an intensity or pressure level of 63 dB
moredB(c(60,60))
# addition of three sources
moredB(c(89,90,95))
```

---

mutew — *Replace time wave data by 0 values*

Description

This functions replaces a time wave or a section of a time wave by 0 values. For a time wave describing a sound, this corresponds in muting the sound or a section of it.

Usage

```
mutew(wave, f, from = NULL, to = NULL, choose = FALSE, plot = TRUE,
Sample = FALSE,...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
from	start of the silence section (in s).
to	end of the silence section (in s).
choose	logical, if TRUE start (=from) and end (=to) points can be graphically chosen with a cursor on the oscillogram.
plot	logical, if TRUE returns an oscillographic plot of wave with the new silence section (by default TRUE).
Sample	if TRUE and plot is FALSE returns an object of class Sample
...	other oscillo graphical parameters.

Details

By default, from and from are NULL, this results in completely muting wave.

Value

If plot is FALSE, a new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

oscillo, addsilw, cutw, deletew,fadew, pastew, revw, zapsilw

Examples

```
data(tico)
mutew(tico,f=22050,from=0.5,to=0.9)
```

---

noise *Generate noise*

Description

This function generates noise.

Usage

```
noise(f, d, type="unif", listen = FALSE, Sample = FALSE)
```

Arguments

f	sampling frequency of the signal to be generated (in Hz)
d	duration of the signal to be generated.
type	a character string to specify the type of noise, either "unif" or "gaussian".
listen	if TRUE the new sound is played back.
Sample	if TRUE and plot is FALSE returns an object of class Sample

.

Details

Uniform noise is generated using runif and gaussian noise is based on rnorm

Value

If plot is FALSE, a new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

synth, pulse

Examples

```
# add noise to a synthetic signal
a<-noise(d=1,f=8000)
b<-synth(f=8000,d=1,cf=2000,plot=FALSE)
c<-a+b
spectro(c,f=8000)
```

---

orni *Song of the cicada Cicada orni*

Description

Recording of a calling song section of the Mediterranean cicada *Cicada orni*.

Usage

```
data(orni)
```

Format

A data frame with 15842 observations on the following variable.

V1 a numeric vector

Details

Duration = 0.719 s. Sampling frequency = 22050 Hz.

Source

Recording by Jerome Sueur.

Examples

```
data(orni)
oscillo(orni,f=22050)
```

oscillo *Show a time wave as an oscillogram*

Description

This graphical function displays a time wave as an oscillogram in a single or multi-frame plot. The envelope of the wave can also be shown.

Usage

```
oscillo(wave, f, from = NULL, to = NULL, scroll = NULL,
zoom = FALSE, k=1, j=1, labels = TRUE, byrow = TRUE,
identify = FALSE, plot = TRUE, colwave = "black",
coltitle = "black", cextitle = 1.2,
fonttitle = 2, collab = "black",
cexlab = 1, fontlab = 1, colline = "black",
colaxis = "black", coly0 = "lightgrey",
title = FALSE, xaxt="s", yaxt="n", bty = "l")
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`from`	start of the oscillogram (in s).
`to`	end of the oscillogram (in s).
`scroll`	a numeric of length 1 allowing to move along the time wave using a slider panel. This numeric corresponds to the number of successive windows dividing the time wave.
`zoom`	time zoom in with start and end points chosen on the oscillogram with a cursor.
`k`	number of horizontal sections (by default =1).
`j`	number of vertical sections (by default =1).
`labels`	if `TRUE` plots time and amplitude labels (by default `TRUE`).
`byrow`	logical, if `TRUE`, the sections are filled by rows, otherwise the sections are filled by colmuns (by default `TRUE`).
`identify`	returns the time coordinate of points chosen with a cursor on the oscillogram.
`plot`	logical, if `TRUE` returns an oscillographic or envelope plot of `wave`(by default `TRUE`).
`colwave`	colour of the oscillogram or of the envelope.
`coltitle`	if `title` is `TRUE`, colour of the title.
`cextitle`	character size for the title.

`fonttitle`	font for the title.
`cexlab`	character size for axes title.
`fontlab`	font for axes title.
`collab`	colour of axes title.
`colline`	colour of axes line.
`colaxis`	colour of the axes.
`coly0`	colour of the y=0 line.
`title`	`TRUE` to add a title with information on `wave` duration and `f`, `FALSE` to live it blanck, or a character string to add any desired title.
`xaxt`	equivalent to `xaxt` of `par` (by default =`"s"`).
`yaxt`	equivalent to `yaxt` of `par` (by default =`"n"`).
`bty`	the type of box to be drawn around the oscillogram.

Value

Data are returned as one-column matrix if `plot` is `FALSE`. `identify` returns a numeric object with the time coordinate of points successively chosen on the oscillogram.

Note

`zoom` is similar to but more visual than `from` and/or `to`. `zoom` and `identify` do work with a single-frame window only (*i. e.* with `k` = 1 and `j` = 1).
Press ‘Stop’ button of the tools bar after choosing the appropriate points on the oscillogram.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

See Also

`oscilloST`,`cutw`, `pastew`, `timer`

Examples

```
data(tico)
# a simple oscillogram of a bird song
oscillo(tico,f=22050)
# zoom in
op<-par(mfrow=c(4,1),mar=c(4.5,4,2,2))
oscillo(tico,22050,cexlab=0.75)
oscillo(tico,22050,from=0.5,to=0.9,cexlab=0.75)
oscillo(tico,22050,from=0.65,to=0.75,cexlab=0.75)
oscillo(tico,22050,from=0.68,to=0.70,cexlab=0.75)
par(op)
# the same divided in four lines
oscillo(tico,f=22050,k=4,j=1)
# the same divided in different numbers of lines and columns
```

```
oscillo(tico,f=22050,k=4,j=4)
oscillo(tico,f=22050,k=2,j=2,byrow=TRUE)
oscillo(tico,f=22050,k=2,j=2,byrow=FALSE)
# overplot of oscillographic and envelope representations
oscillo(tico,f=22050)
par(new=TRUE)
env(tico,f=22050,colwave=2)
# full colour modifications in a two-frame oscillogram
op<-par(bg="grey")
oscillo(tico,f=22050,k=4,j=1,title=TRUE,colwave="black",
    coltitle="yellow",collab="red",colline="white",
    colaxis="blue",coly0="grey50")
par(op)
# change the title
data(orni)
oscillo(orni,f=22050,title="The song of a famous cicada")
# move along the signal using scroll
oscillo(tico,f=22050,scroll=8)
```

---

oscilloST *Show a stereo time wave as oscillograms*

Description

This graphical function displays a stereo (2 channels) time wave as an oscillogram in a two-frame plot. The envelope of the wave can also be shown.

Usage

```
oscilloST(wave1, wave2 = NULL, f, from = NULL, to = NULL,
identify = FALSE, plot = TRUE, colwave1 = "black",
colwave2 = "blue", coltitle = "black",
collab = "black", cexlab = 1, fontlab = 1, colaxis = "black",
coly01 = "grey47", coly02 = "black", title = FALSE,
bty = "l")
```

Arguments

wave1	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
wave2	a vector, a matrix (second column), an object of class ts, Sample (right channel), or Wave (right channel).
f	sampling frequency of wave1 and wave2 (in Hz). Does not need to be specified if wave1 and/or wave2 are/is of class ts, Sample, or Wave.
from	start of the oscillogram (in s).
to	end of the oscillogram (in s).

`identify`	returns the time coordinate of points chosen with a cursor on the bottom oscillogram.
`plot`	logical, if `TRUE` returns an oscillographic or envelope plot of `wave`(by default `TRUE`).
`colwave1`	colour of the oscillogram or of the envelope of `wave1`.
`colwave2`	colour of the oscillogram or of the envelope of `wave2`.
`coltitle`	if `title` is `TRUE`, colour of the title.
`collab`	colour of axes title.
`cexlab`	character size for axes title.
`fontlab`	font for axes title.
`colaxis`	colour of the axes
`coly01`	colour of the y=0 line of `wave1`.
`coly02`	colour of the y=0 line of `wave1`.
`title`	logical, if `TRUE` plots the title with information on time and `f` (by default `FALSE`).
`bty`	the type of box to be drawn around the oscillogram.

Value

Data are returned as two-column matrix if `plot` is `FALSE`. `identify` returns a numeric object with the time coordinate of points successively chosen on the bottom oscillogram.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

See Also

`oscillo`

Examples

```
a<-synth(f=8000,d=1,cf=2000,am=c(50,10),plot=FALSE)
b<-synth(f=8000,d=1,cf=1000,fm=c(0,0,2000),plot=FALSE)
oscilloST(a,b,f=8000)
```

`pastew` *Paste a time wave to another one*

Description

This function pastes a first time wave to a second one. The time wave to be pasted, the time wave to be completed and the resulting time wave can be displayed in a three-frame oscillographic plot.

Usage

```
pastew(wave1, wave2, f, at = "end", choose = FALSE, plot = FALSE,
marks = TRUE, Sample = FALSE,...)
```

Arguments

`wave1`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel) to be pasted to `wave2`.
`wave2`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave1` and `wave2` (in Hz). Does not need to be specified if `wave1` and/or `wave2` are/is of class `ts`, `Sample`, or `Wave`.
`at`	`wave2` position in seconds where `wave1` will be pasted into. Can be also specified as `"start"`, `"middle"` or `"end"`.
`choose`	logical, if `TRUE` the point where `wave1` will be pasted into `wave2` (=`at`) can be graphically chosen with a cursor.
`plot`	logical, if `TRUE` returns an oscillographic plot of `wave1`, `wave2` and `wave1` + `wave2` (by default `FALSE`).
`marks`	logical, if `TRUE` shows where `wave1` has been pasted (by default `TRUE`).
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `oscillo` graphical parameters.

Details

If `plot` is `TRUE` returns a two-frame plot with three waves:
(1) the wave to be pasted (`wave1`),
(2) the wave to be completed (`wave2`),
(3) the resulting wave.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

oscillo, addsilw, cutw, deletew, fadew, mutew , revw, repw, zapsilw

Examples

```
data(tico)
# double a data set describing a bird song
a<-pastew(tico,tico,f=22050)
oscillo(a,f=22050)
# a direct way to see what has been pasted
pastew(tico,tico,f=22050,plot=TRUE)
# cut a section and then paste it at the beginning
a<-cutw(tico, f=22050, from=0.5, to=0.9)
pastew(a,tico,f=22050,at="start",plot=TRUE)
# or paste it at a specific location
pastew(a,tico,f=22050,at=1.4,plot=TRUE)
```

---

peewit *Song of the bird Vanellus vanellus*

Description

Recording of a song emitted by a peewit (lapwing) male *Vanellus vanellus*

Usage

```
data(peewit)
```

Format

A data frame with 15561 observations on the following variable.

V1 a numeric vector

Details

Duration = 0.706 s. Sampling frequency = 22050 hz.

Source

Recording by Thierry Aubin.

Examples

```
data(peewit)
oscillo(peewit,f=22050)
```

pellucens — *Calling song of the tree cricket Oecanthus pellucens*

Description

Recording of a calling song section emitted by the European tree cricket *Oecanthus pellucens*.

Usage

```
data(pellucens)
```

Format

A data frame with 36476 observations on the following variable.

V1 a numeric vector

Details

Duration = 3.309 s. Sampling frequency = 11025 hz.

Source

Recording by Jerome Sueur.

Examples

```
data(pellucens)
oscillo(pellucens,f=11025)
```

pulse — *Generate rectangle pulse*

Description

This function generates a rectangle pulse.

Usage

```
pulse(dbefore, dpulse, dafter, f, plot = FALSE, Sample =FALSE, ...)
```

Arguments

`dbefore`	duration of the silent period before the pulse
`dpulse`	duration of the pulse to generate
`dafter`	duration of silent period after the pulse
`f`	sampling frequency of the signal to be generated (in Hz)
`plot`	logical, if `TRUE` returns an oscillographic plot of the pulse generated (by default `FALSE`).
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `plot` parameters.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`synth`, `noise`

Examples

```
pulse(dbefore=0.5,dpulse=0.1,dafter=0.3,f=8000,plot=TRUE)
```

Q — *Resonance quality factor of a frequency spectrum*

Description

This function estimates the frequency pureness of a time wave by returning the resonant quality factor Q at a specific dB level.

Usage

```
Q(spec, f = NULL, level = -3, plot = TRUE, colval = "red",
cexval = 1, fontval = 1, flab = "Frequency (kHz)",
alab = "Relative amplitude (dB)", type = "l", ...)
```

Arguments

spec	a data set resulting of a spectral analysis obtained with spec, or meanspec (in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
f	sampling frequency of the wave used to obtain spec (in Hz). Not necessary if spec is a two columns matrix obtained with spec or meanspec.
level	frequency bandwidth set by an amplitude value relative to spectrum (in dB).
plot	logical, if TRUE returns the spectrum with Q plotted (by default TRUE).
colval	colour of plotting Q.
cexval	character size of plotting Q.
fontval	font of plotting Q.
flab	title of the frequency axis.
alab	title of the ampltiude axis.
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
...	other plot graphical parameters.

Details

A high Q value indicates a highly resonant system.

Value

Q is returned as a single numeric data.

Note

This function is based on fft.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

spec, meanspec, corspec, fft.

Examples

```
# bird song
data(tico)
t<-spec(tico,f=22050,at=1.1,plot=FALSE,dB="max0")
op<-par(mfrow=c(2,1),las=1)
Q(t,type="l")
Q(t,type="l",xlim=c(3.8,4.2),ylim=c(-60,0))
title("zoom in")
par(op)
```

```
# cricket, changing the dB level
data(pellucens)
p<-spec(pellucens,f=11025,at=0.5,plot=FALSE,dB="max0")
op<-par(mfrow=c(3,1))
Q(p,type="l",xlim=c(1.8,2.6),ylim=c(-70,0))
title("level = - 3 (default value)",col.main="red")
Q(p,type="l",level=-6,
    xlim=c(1.8,2.6),ylim=c(-70,0),colval="blue")
title("level = - 6",col.main="blue")
Q(p,type="l",level=-9,
    xlim=c(1.8,2.6),ylim=c(-70,0),colval="green")
title("level = - 9",col.main="green")
par(op)
```

---

repw *Repeat a time wave*

Description

This function repeats a time wave

Usage

```
repw(wave, f, times = 2, plot = FALSE, Sample = FALSE, ...)
```

Arguments

wave	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
f	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
times	a numeric of length 1 describing the number of times the wave has to be repeated.
plot	logical, if `TRUE` plots the repeated time wave
Sample	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
...	other `oscillo` graphical parameters.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`oscillo`, `addsilw`, `cutw`, `deletew`, `fadew`, `mutew`, `pastew`, `revw`, `zapsilw`

Examples

```
data(tico)
repw(tico,f=22050,plot=TRUE)
```

`resamp` *Resample a time wave*

Description

This function resamples (down- or over-samples) a time wave. This corresponds to a sampling frequency change.

Usage

```
resamp(wave, f, g, Sample = FALSE)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`g`	new sampling frequency of `wave` (in Hz).
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`

.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Note

Resampling might change frequency properties of the time wave.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

Examples

```
data(peewit)
# downsampling
a<-resamp(peewit,f=22050,g=11025)
# oversampling
b<-resamp(peewit,f=22050,g=44100)
```

`revw` *Time reverse of a time wave*

Description

Reverse the wave along the time axis.

Usage

```
revw(wave, f, env = TRUE, ifreq = TRUE, plot = FALSE, Sample = FALSE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`env`	logical, if `TRUE` the amplitude envelope is reversed.
`ifreq`	logical, if `TRUE` the instantaneous frequency is reversed.
`plot`	logical, if `TRUE` returns an oscillographic plot of the reversed wave (by default `FALSE`).
`Sample`	logical, if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `oscillo` graphical parameters.

Details

If `plot` is `TRUE` returns an oscillogram of the reversed wave. The amplitude and the instantaneous frequency can be independently reversed thanks to the arguments `env` and `ifreq`. See the examples.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Beeman, K. 1998. Digital signal analysis, editing and synthesis *in* Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds) 1998. *Animal acoustic communication*, pp. 59-103. Springer, Berlin, Heidelberg.

See Also

oscillo, addsilw, deletew, fadew, pastew, mutew

Examples

```
data(tico)
# simple reverse
revw(tico,f=22050,plot=TRUE)
# envelope reverse only
revw(tico,f=22050,ifreq=FALSE, plot=TRUE)
# instantaneous frequency reverse only
revw(tico,f=22050,env=FALSE, plot=TRUE)
```

---

rmam — *Remove the amplitude modulations of a time wave*

Description

This functions removes the amplitude modulation of a time wave through the Hilbert amplitude envelope.

Usage

```
rmam(wave, f, plot = FALSE, listen = FALSE, Sample = FALSE, ...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
plot	logical, if TRUE returns an oscillographic plot of the nwe time wave (by default FALSE).
listen	if TRUE the new sound is played back.
Sample	if TRUE and plot is FALSE returns an object of class Sample
...	other oscillo graphical parameters.

Details

The new time wave is obtained by dividing the original time wave by its Hilbert amplitude envelope.

Value

If `plot` is `FALSE`, a new time wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Mbu Nyamsi, R. G., Aubin, T. & Brï¿½mond, J. C. 1994 On the extraction of some time dependent parameters of an acoustic signal by means of the analytic signal concept. Its application to animal sound study. *Bioacoustics*, 5: 187-203.

See Also

`hilbert`.

Examples

```
# generate a new sound with amplitude modulation
a<-synth(f=8000, d=1, cf=1500, am=c(50,10))
# remove the amplitude modulation and plot the result
rmam(a,f=8000,plot=TRUE)
```

`rmnoise` *Remove noise*

Description

This function removes background noise by smoothing

Usage

```
rmnoise(wave, f, Sample = FALSE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`Sample`	a logical, if `TRUE` returns an object of class `Sample`
`...`	other `smooth.spline` arguments.

Details

This function is based on `smooth.spline`. You can use the arguments of the later to modify the smoothing.

Value

A new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Note

Low frequency noise might not be removed out properly.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`afilter`, `noise`

Examples

```
# synthesis of a 440 Hz sound with background noise
n <- noise(d=1,f=8000)
s <- synth(d=1,f=8000,cf=440)
ns <- n+s
# remove noise (but low frequency content still there)
a <- rmnoise(ns,f=8000)
```

`rmoffset` *Remove the offset of a time wave*

Description

This function removes the offset of a time wave.

Usage

```
rmoffset(wave, f, plot = FALSE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`plot`	logical, if `TRUE` returns an oscillographic plot of the wave after removing the offset (by default `FALSE`).
`...`	other `oscillo` graphical parameters.

Value

If `plot` is `FALSE`, `rmoffset` returns a one-column matrix describing the new wave with the same sampling frequency as the original wave.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`oscillo`

Examples

```
data(tico)
# artifically generates an offset
tico2<-tico+0.1
# see the wave with an offset
oscillo(tico2,f=22050)
# remove the offset
rmoffset(tico2,f=22050,plot=TRUE)
```

`rms` *Root Mean Square*

Description

This function computes the root mean square or quadratic mean.

Usage

```
rms(x, ...)
```

Arguments

`x`	an R object
`...`	further arguments passed to mean

Details

The Root Mean Square or quadratic mean is cimputed according to:

$$RMS = \sqrt{\frac{1}{n} \times \sum_{i=1}^{N} {x_i}^2}$$

Value

A vector

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

mean

Examples

```
# simple rms
rms(1:10)
# rms of a normalized envelope
data(sheep)
env <- env(sheep, f=8000)
rms(env)
```

---

savewav   *Save .wav file*

Description

Save sound data as .wav file

Usage

```
savewav(wave, f,
filename = NULL)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
filename	name of the new file. (by default the name of wave).

Details

This function uses two functions from the package **sound**: Sample and saveSample

Note

The file automatically owerwrites an existing file with the same name.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

as.Sample, saveSample, export.

Examples

```
a<-synth(f=8000,d=2,cf=2000,plot=FALSE)
# the name of the file is automatically the name of the object
# here: "a.wav"
savewav(a,f=22050)
unlink("a.wav")
# if you wish to change the name, use 'file' argument
savewav(a,f=22050,file="b.wav")
unlink("b.wav")
```

---

seewave — *Time wave analysis and graphical representation*

Description

seewave provides functions for analysing, manipulating, displaying, editing and synthesizing time waves (particularly sound). This package processes in particular time analysis (oscillograms and envelopes), spectral content, resonance quality factor, entropy, cross correlation and autocorrelation, zero-crossing, frequency coherence, dominant frequency, analytic signal, 2D and 3D spectrograms.

Details

Package:	seewave
Type:	Package
Version:	1.5.5
Date:	2009-6-26
License:	GPL version 2 or newer
Contributors :	Jonathan Lees, Martin Maechler, Sandrine Pavoine, Luis J. Villanueva-Rivera, Zev Ross, Carl G. Witthof
Acknowledgments:	Michel Baylac, Emmanuel Paradis, Arnold Fertin, Kurt Hornik, Uwe Ligges

Author(s)

Jerome Sueur <sueur@mnhn.fr>
Thierry Aubin <thierry.aubin@u-psud.fr>
Caroline Simonis <csimonis@mnhn.fr>

Maintainer: Jerome Sueur <sueur@mnhn.fr>

http://sueur.jerome.perso.neuf.fr/seewave.html `setenv` *Set the amplitude envelope of a time wave to another one*

Description

This function sets the amplitude envelope of a time wave to another one

Usage

```
setenv(wave1, wave2, f, envt="hil", msmooth = NULL, ksmooth = NULL,
plot = FALSE, listen = FALSE, Sample = FALSE, ...)
```

Arguments

`wave1`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`wave2`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel) describing the time wave which envelope will be used to set `wave1` envelope.
`f`	sampling frequency of `wave1` and `wave2` (in Hz). Does not need to be specified if `wave1` and/or `wave2` are/is of class `ts`, `Sample`, or `Wave`.
`envt`	the type of envelope to be used for `wave2`: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See `env`.
`msmooth`	a vector of length 2 to smooth the amplitude envelope of `wave2` with a mean sliding window. The first component is the window length (in number of points). The second component is the overlap between successive windows (in %). See `env`.
`ksmooth`	kernel smooth via `kernel` to apply to the amplitude envelope of`wave2`. See `env`.
`plot`	if `TRUE` returns the oscillogram of the new time wave (by default `FALSE`).
`listen`	if `TRUE` the new sound is played back.
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`.
`...`	other `oscillo` graphical parameters.

Details

`wave1` and `wave2` can have different duration (length)
Smoothing the envelope with `smooth` or `ksmooth` can significantly change the value returned.

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

drawenv, env, synth

Examples

```
data(tico)
a<-synth(d=1,f=22050,cf=1000)
# apply 'tico' ammplitude envelope to 'a' that has a square amplitude envelope
setenv(a,tico,f=22050,plot=TRUE)
# the same but with smoothing the envelope
setenv(a,tico,f=22050,ksmooth=kernel("daniell",50),plot=TRUE)
```

---

sfm — *Spectral Flatness Measure*

Description

This function estimates the flatness of a frequency spectrum.

Usage

```
sfm(spec)
```

Arguments

spec	a data set resulting of a spectral analysis obtained with spec or meanspec (not in dB).

Details

SFM is calculated as the ratio between the geometric mean and the arithmetic mean :

$$F = N \times \frac{\sqrt[N]{\prod_{i=1}^{N} y_i}}{\sum_{i=1}^{N} y_i}$$

with:
y = relative amplitude of the *i* frequency,
and *N* = number of frequencies.

Value

A single value varying between 0 and 1 is returned. The value has no unit.

Note

The SFM of a noisy signal will tend towards 1 whereas the SFM of a pure tone signal will tend towards 0.
See sh for another measure of signal noisiness/pureness.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

sh, csh

Examples

```
a<-synth(f=8000,d=1,cf=2000,plot=FALSE)
speca<-spec(a,f=8000,at=0.5,plot=FALSE)
sfm(speca)
# [1] 0
b<-noise(d=1,f=8000)
specb<-spec(b,f=8000,at=0.5,plot=FALSE)
sfm(specb)
# [1] 0.8233202
```

---

sh — *Spectral entropy*

Description

This function computes the entropy of a frequency spectrum.

Usage

```
sh(spec)
```

Arguments

spec	a data set resulting of a spectral analysis obtained with spec or meanspec (not in dB).

Details

Spectral entropy is calculated according to:

$$S = -\frac{\sum_{i=1}^{N} y_i log_2(y_i)}{log_2(N)}$$

with:
y = relative amplitude of the *i* frequency,
and

$$\sum_{i=1}^{N} y_i = 1$$

and *N* = number of frequencies.

Value

A single value varying between 0 and 1 is returned. The value has no unit.

Note

The spectral entropy of a noisy signal will tend towards 1 whereas the spectral entropy of a pure tone signal will tend towards 0.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Nunes, R. R., Almeida de, M. P. & Sleigh, J. W. 2004 Spectral entropy: a new method for anesthetic adequacy. *Revista Brasileira de Anestesiologia*, **54**, 413-422.

See Also

csh,th, H, sfm

Examples

```
a<-synth(f=8000,d=1,cf=2000,plot=FALSE)
speca<-spec(a,f=8000,at=0.5,plot=FALSE)
sh(speca)
# [1] 0.2336412
b<-noise(d=1,f=8000)
specb<-spec(b,f=8000,at=0.5,plot=FALSE)
sh(specb)
# close to 1
```

sheep — *Sheep bleat*

Description

Recording of a sheep bleat.

Usage

```
data(sheep)
```

Format

A data frame with 19764 observations on the following variable.

V1 a numeric vector

Details

Duration = 2.47 s. Sampling frequency = 8000 hz.

Source

Recording by Frederic Sebe.

Examples

```
data(sheep)
oscillo(sheep,f=8000)
```

simspec — *Similarity between two frequency spectra*

Description

This function estimates the similarity between two frequency spectra.

Usage

```
simspec(spec1, spec2, f = NULL, plot = FALSE, type = "l",
lty1 = 1, lty2 = 2,
lty3 = 3, col1 = 2, col2 = 4, col3 = 1, flab = "Frequency (kHz)",
alab = "Amplitude (percentage)", flim = c(0, f/2000),
alim = c(0, 100),
legend = TRUE, ...)
```

Arguments

spec1	a first data set resulting of a spectral analysis obtained with spec or meanspec (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
spec2	a first data set resulting of a spectral analysis obtained with spec or meanspec (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
f	sampling frequency of waves used to obtain spec1 and spec2 (in Hz). Not necessary if spec1 and/or spec2 is a two columns matrix obtained with spec or meanspec.
plot	logical, if TRUE plots both spectra and similarity function (by default FALSE).
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
lty1	line type of spec1 if type="l".
lty2	line type of spec2 if type="l".
lty3	line type of the similarity function if type="l".
col1	colour of spec1.
col2	colour of spec2.
col3	colour of the similarity function.
flab	title of the frequency axis.
alab	title of the amplitude axis.
flim	the range of frequency values.
alim	range of amplitude axis.
legend	logical, if TRUE adds a legend to the plot.
...	other plot graphical parameters.

Details

Spectra similarity is assessed according to:

$$S = 100 \times \sum_{i=1}^{N} \frac{\min spec1(i), spec2(i)}{\max spec1(i), spec2(i)}$$

with S in %.

Value

The similarity index is returned. This value is in %.
When plot is TRUE, both spectra and the similarity function are plotted on the same graph. The similarity index is the mean of this function.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Deecke, V. B. and Janik, V. M. 2006. Automated categorization of bioacoustic signals: avoiding perceptual pitfalls. *Journal of the Acoustical Society of America*, 119: 645-653.

See Also

spec, meanspec, corspec, diffspec, diffenv

Examples

```
a<-noise(f=8000,d=1)
b<-synth(f=8000,d=1,cf=2000)
c<-synth(f=8000,d=1,cf=1000)
d<-noise(f=8000,d=1)
speca<-spec(a,f=8000,at=0.5,plot=FALSE)
specb<-spec(b,f=8000,at=0.5,plot=FALSE)
specc<-spec(c,f=8000,at=0.5,plot=FALSE)
specd<-spec(d,f=8000,at=0.5,plot=FALSE)
simspec(speca,speca)
simspec(speca,specb)
simspec(speca,specc,plot=TRUE)
simspec(specb,specc,plot=TRUE)
#[1] 12.05652
simspec(speca,specd,plot=TRUE)
```

---

spec — *Frequency spectrum of a time wave*

Description

This function returns the frequency spectrum (*i.e.* the relative amplitude of the frequency content) of a time wave. Results can be obtained either as absolute or dB data.

Usage

```
spec(wave, f, wl = 512, wn = "hanning", PSD = FALSE,
PMF = FALSE, dB = NULL, dBref = NULL,
at = NULL, from = NULL, to = NULL, peaks = NULL,
identify = FALSE, col = "black", cex = 1, colpeaks = "red",
cexpeaks = 1, fontpeaks = 1, plot = 1, flab = "Frequency (kHz)",
alab = "Amplitude", flim = c(0, f/2000),
alim = NULL, type="l",...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).

f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
wl	if at is not null, length of the window for the analysis (even number of points, by defaults = 512).
wn	window name, see ftwindow (by default "hanning").
PSD	if TRUE return Power Spectra Density, *i. e.* the square of the spectra.
PMF	if TRUE return Probability Mass Function, *i. e.* the probability distribution of frequencie.
dB	a character string specifying the type dB to return: "max0" for a maximum dB value at 0, "A", "B", "C" and "D" for common dB weights.
dBref	a dB reference value when dB is not NULL. NULL by default but should be set to 2*10e-5 for a 20 microPa reference (SPL).
at	position where to compute the spectrum (in s).
from	start mark where to compute the spectrum (in s).
to	end mark where to compute the spectrum (in s).
peaks	if not NULL returns peaks value for a given span (see details).
identify	to identify frequency and amplitude values on the plot with the help of a cursor.
col	colour of the spectrum.
cex	pitch size of the spectrum.
colpeaks	colour of peaks value plotted on the spectrum.
cexpeaks	character size of peaks value plotted on the spectrum.
fontpeaks	font of peaks value plotted on the spectrum.
plot	if 1 returns frequency on x-axis, if 2 returns frequency on y-axis, (by default 1).
flab	title of the frequency axis.
alab	title of the amplitude axis.
flim	range of frequency axis.
alim	range of amplitude axis.
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
...	other plot graphical parameters.

Details

If at, from or to are FALSE then spec computes the spectrum of the whole signal.
peaks setting corresponds to dimension of embed. See examples.

Value

This function returns a two-column matrix, the first column corresponding to the frequency axis, the second column corresponding to the amplitude axis.
If identify is TRUE, spec returns a list with two elements:

`freq`	the frequency of the points chosen on the spectrum
`amp`	the relative amplitude of the points chosen on the spectrum
`spec`	the spectrum computed
`peaks`	the peaks value (in kHz).

Note

This function is based on `fft`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Martin Maechler ⟨maechler@stat.math.ethz.ch⟩ for `peaks`.

See Also

`meanspec`, `dynspec`, `corspec`, `fft`.

Examples

```
data(tico)
# spectrum of the whole signal, in absolute or dB amplitude,
# horizontaly or vertically
op<-par(mfrow=c(2,2))
spec(tico,f=22050)
spec(tico,f=22050,col="red",plot=2)
spec(tico,f=22050,dB="max0",col="blue")
spec(tico,f=22050,dB="max0",col="green",plot=2)
par(op)
# spectra computed at specific locations with peak value
op<-par(mfrow=c(2,2))
spec(tico,f=22050,wl=512,at=0.2,peak=175)
title("Note A")
spec(tico,f=22050,wl=512,at=0.7,peak=175)
title("Note B")
spec(tico,f=22050,wl=512,at=1.1,peak=175)
title("Note C")
spec(tico,f=22050,wl=512,at=1.6,peak=165)
title("Note D")
par(op)
# an indirect way to compare spectra
a<-spec(tico,f=22050,wl=512,at=0.2,plot=FALSE)
b<-spec(tico,f=22050,wl=512,at=0.7,plot=FALSE)
c<-spec(tico,f=22050,wl=512,at=1.1,plot=FALSE)
d<-spec(tico,f=22050,wl=512,at=1.6,plot=FALSE)
all<-cbind(a[,2],b[,2],c[,2],d[,2])
matplot(x=a[,1],y=all,yaxt="n",
    xlab="Frequency (kHz)",ylab="Amplitude",xaxs="i",type="l")
legend(8,0.8,c("Note A","Note B", "Note C", "Note D"),bty="o",
    lty=c(1:4),col=c(1:4))
# spectrum from a particular position to another one
op<-par(mfrow=c(2,1))
```

```
oscillo(tico,f=22050)
abline(v=c(0.5,0.9),col="red",lty=2)
text(c("from 0.5 s","to 0.7 s"),
    x=c(0.5,0.9),y=rep(max(tico/1.1),2),col="red",pos=4)
spec(tico,f=22050,wl=512,from=0.5,to=0.9,col="red")
title("Spectrum of the note B")
par(op)
# spectrum and spectrogram
data(orni)
orni1<-cutw(orni,f=22050,from=0.32,to=0.39)
layout(matrix(c(1,2),nc=2),widths=c(3,1))
par(mar=c(5,4,3,0.5))
spectro(orni1,f=22050,wl=128,zp=8,ovlp=85,scale=FALSE)
par(mar=c(5,1,3,0.5))
spec(orni1,f=22050,col="red",plot=2,flab="",yaxt="n")
```

`specprop` *Spectral properties*

Description

This function returns a list of statistical properties of a frequency spectrum.

Usage

```
specprop(spec, f, str = FALSE, flim=NULL, plot = FALSE, type = "l", ...)
```

Arguments

`spec`	a data set resulting of a spectral analysis obtained with `spec` or `meanspec` (not in dB).
`f`	sampling frequency of `spec` (in Hz).
`str`	logical, if `TRUE` returns the results in a structured table.
`flim`	a vector of length 2 to specifgy the frequency limits of the analysis (in kHz)
`plot`	if `1` returns the spectrum , if `2` returns the cumulative spectrum, both of them with the first quartile, the third quartile, the median and the mode plotted (by default `FALSE`).
`type`	if `plot` is `TRUE`, type of plot that should be drawn. See `plot` for details (by default "l" for lines).
`...`	other `plot` graphical parameters.

Details

The spectrum is converted in a probability mass function (PMF).
If a selected value has to be selected with $, the argument `str` has to be set to `FALSE`.

Value

A list of 15 values is returned

mean	mean frequency (see mean)
sd	standard deviation of the mean (see sd)
sem	standard error of the mean
median	median frequency (see median)
mad	absolute deviation of median (see mad)
mode	mode frequency, *i.e.* the dominant frequency
Q25	first quartile (see quantile)
Q75	third quartile (see quantile)
IQR	interquartile range (see IQR)
cent	centroï¿½d, see note
skewness	skewness, a measure of asymmetry, see note
kurtosis	kurtosis, a measure of peakedness, see note
sfm	spectral flatness measure (see sfm)
sh	spectral entropy (see sh)
prec	frequency precision of the spectrum

Note

Centroid is computed according to:

$$C = \sum_{i=1}^{N} x_i \times y_i$$

with:
x = frequencies, y = relative amplitude of the i frequency,
N = number of frequencies.

Skewness is computed according to:

$$S = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^3}{N-1} \times \frac{1}{\sigma^3}$$

.
S < 0 when the spectrum is skewed to left,
S = 0 when the spectrum is symetric,
S > 0 when the spectrum is skewed to right.
Spectrum asymmetry increases with |S|.

Kurtosis is computed according to:

$$K = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^4}{N-1} \times \frac{1}{\sigma^4}$$

.
K < 3 when the spectrum is platikurtic, *i.e.* it has fewer items at the center and at the tails than the normal curve but has more items in the shoulders,
K = 3 when the spectrum shows a normal shape,
K > 3 when the spectrum is leptokurtic, *i.e.* it has more items near the center and at the tails, with fewer items in the shoulders relative to normal distribution with the same mean and variance.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

Examples

```
data(orni)
a<-meanspec(orni,f=22050,plot=FALSE)
specprop(a,f=22050)
# to get a single measure of the list
specprop(a,f=22050)$mode
# to get the results structured
specprop(a,f=22050,str=TRUE)
# to limit the analysis between 4 and 6 kHz
specprop(a,f=22050,flim=c(4,6),str=TRUE)
# plots
specprop(a,f=22050,plot=1)
specprop(a,f=22050,plot=2)
```

---

spectro *2D-spectrogram of a time wave*

Description

This function returns a two-dimension spectrographic representation of a time wave. The function corresponds to short-term Fourier transform. An amplitude contour plot can be overlaid.

Usage

```
spectro(wave, f, wl = 512, wn = "hanning", zp = 0,
ovlp = 0, dB = "max0", dBref = NULL, plot = TRUE,
grid = TRUE, osc = FALSE, scale = TRUE, cont = FALSE,
collevels = NULL, palette = spectro.colors,
contlevels = NULL, colcont = "black",
colbg = "white", colgrid = "black",
colaxis = "black", collab="black",
plot.title = title(main = "", xlab = "Time (s)",
ylab = "Frequency (kHz)"), scalelab = "Amplitude\n(dB)",
scalefontlab = 1, scalecexlab =0.75,
axisX = TRUE, axisY = TRUE, tlim = NULL, trel = TRUE,
flim = NULL, flimd = NULL, listen=FALSE,
...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
wl	window length for the analysis (even number of points). (by default = 512)
zp	zero-padding (even number of points), see Details.
ovlp	overlap between two successive windows (in %).
dB	a character string specifying the type dB to return: "max0" (default) for a maximum dB value at 0, "A", "B", "C" and "D" for common dB weights.
dBref	a dB reference value when dB is TRUE. NULL by default but should be set to 2*10e-5 for a 20 microPa reference.
wn	window name, see ftwindow (by default "hanning").
plot	logical, if TRUE plots the spectrogram (by default TRUE).
grid	logical, if TRUE plots a y-axis grid (by default TRUE).
osc	logical, if TRUE plots an oscillogram beneath the spectrogram (by default FALSE).
scale	logical, if TRUE plots a dB colour scale on the right side of the spectrogram (by default TRUE).
cont	logical, if TRUE overplots contour lines on the spectrogram (by default FALSE).
collevels	a set of levels which are used to partition the amplitude range of the spectrogram (in dB).
palette	a color palette function to be used to assign colors in the plot, see Details.
contlevels	a set of levels which are used to partition the amplitude range for contour overplot (in dB).
colcont	colour for cont plotting.
colbg	background colour.
colgrid	colour for grid plotting.
colaxis	color of the axes.
collab	color of the labels.
plot.title	statements which add titles to the plot.
scalelab	amplitude scale label.
scalefontlab	font of the amplitude scale label.
scalecexlab	cex of the amplitude scale label.
axisX	logical, if TRUE plots time X-axis (by default TRUE).
axisY	logical, if TRUE plots frequency Y-axis (by default TRUE).
tlim	modifications of the time X-axis limits.
trel	time X-axis with a relative scale when tlim is not null, *i.e.* relative to wave.
flim	modifications of the frequency Y-axis limits.
flimd	dynamic modifications of the frequency Y-axis limits. New wl and ovlp arguments are applied to increase time/frequency resolution.
listen	if TRUE the sound is played back (by default FALSE).
...	other contour and oscillo graphical parameters.

Details

Following Heisenberg uncertainty principle, the short-term Fourier transform cannot be precised in both time and frequency. The temporal and frequency precisions of the function are actually dependent of the `wl` value. Choosing a high `wl` value will increase the frequency resolution but reduce the temporal one, and *vice versa*. The frequency precision is obtained by calculating the ratio `f`/`wl`, and the temporal precision is obtained by calculating the reverse ratio `wl`/`f`. This problem can be reduced in some way with `zp` that adds 0 values on both sides of the analysis window. This increases frequency resolution without altering time resolution.
Any colour palette can be used. In particular, it is possible to use other palettes coming with **seewave**: `temp.colors`, `rev.gray.colors.1`, `rev.gray.colors.2`, `rev.heat.colors`, `rev.terrain.colors`, `rev.topo.colors`, `rev.cm.colors` corresponding to the reverse of `heat.colors`, `terrain.colors`, `topo.colors`, `cm.colors`.
Use `locator` to identify points.

Value

If `plot` is `FALSE`, this function returns a matrix. Each column corresponds to a Fourier transform of length `wl`/2.

Note

This function is based on `fft`, `contour` and `filled.contour`

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

References

Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds) 1998. *Animal acoustic communication*. Springer, Berlin, Heidelberg.

See Also

`spectro3D`, `dynspec`, `wf`, `oscillo`, `dBscale`, `fft`.

Examples

```
data(tico)
data(pellucens)
# simple plots
spectro(tico,f=22050)
spectro(tico,f=22050,osc=TRUE)
spectro(tico,f=22050,scale=FALSE)
spectro(tico,f=22050,osc=TRUE,scale=FALSE)
# change the dB scale by setting a different dB reference value (20 microPa)spectro(tico,f=2
# manipulating wl
op<-par(mfrow=c(2,2))
spectro(tico,f=22050,wl=256,scale=FALSE)
title("wl = 256")
```

```
spectro(tico,f=22050,wl=512,scale=FALSE)
title("wl = 512")
spectro(tico,f=22050,wl=1024,scale=FALSE)
title("wl = 1024")
spectro(tico,f=22050,wl=4096,scale=FALSE)
title("wl = 4096")
par(op)
# vertical zoom using flim
spectro(tico,f=22050, ylim=c(2,6))
spectro(tico,f=22050, ylimd=c(2,6))
# a full plot
pellu2<-cutw(pellucens,f=22050,from=1,plot=FALSE)
spectro(pellu2,f=22050,ovlp=85,zp=16,osc=TRUE,
    cont=TRUE,contlevels=seq(-30,0,20),colcont="red",
    lwd=1.5,lty=2,palette=rev.terrain.colors)
# black and white spectrogram
spectro(pellu2,f=22050,ovlp=85,zp=16,
    palette=rev.gray.colors.1)
# colour modifications
data(sheep)
spectro(sheep,f=8000,palette=temp.colors,collevels=seq(-115,0,1))
spectro(pellu2,f=22050,ovlp=85,zp=16,
palette=rev.cm.colors,osc=TRUE,colwave="orchid1")
spectro(pellu2,f=22050,ovlp=85,zp=16,osc=TRUE,palette=rev.heat.colors,
colbg="black",colgrid="white", colwave="white",colaxis="white",collab="white")
```

---

spectro3D *3D-spectrogram of a time wave*

Description

This function returns a three-dimension spectrographic representation of a time wave. The function corresponds to short-term Fourier transform.

Usage

```
spectro3D(wave, f, wl = 512, wn = "hanning", zp = 0,
ovlp = 0, dB = "max0", dBref = NULL, plot = TRUE,
magt = 10, magf = 10, maga = 2,
palette = rev.terrain.colors)
```

Arguments

wave	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
f	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
wl	length of the window for the analysis (even number of points).

wn	window name, see ftwindow (by default "hanning").
zp	zero-padding (even number of points), see Details.
ovlp	overlap between two successive windows (in %).
dB	a character string specifying the type dB to return: "max0" for a maximum dB value at 0, "A", "B", "C" and "D" for common dB weights.
dBref	a dB reference value when dB is TRUE. NULL by default but should be set to 2*10e-5 for a 20 microPa reference.
plot	logical, if TRUE plots the spectrogram (by default TRUE).
magt	magnification of the time axis.
magf	magnification of the frequency axis.
maga	magnification of the amplitude axis.
palette	a color palette function to be used to assign colors in the plot, see Details.

Details

Following Heisenberg uncertainty principle, the short-term Fourier transform cannot be precised in both time and frequency. The temporal and frequency precisions of the function are actually dependent of the wl value. Choosing a high wl value will increase the frequency resolution but reduce the temporal one, and *vice versa*. The frequency precision is obtained by calculating the ratio f/wl, and the temporal precision is obtained by calculating the reverse ratio wl/f. This problem can be reduced in some way with zp that adds 0 values on both sides of the analysis window. This increases frequency resolution without altering time resolution.
Any colour palette can be used. In particular, it is possible to use other palettes coming with **seewave**: rev.gray.colors.1, rev.gray.colors.2, spectro.colors, temp.colors, rev.heat.colors, rev.cm.colors, rev.topo.colors, corresponding to the reverse of heat.colors,topo.colors, cm.colors.
Use magt, magf and maga to resize the plot.

Value

If plot is FALSE, this function returns a matrix. Each column corresponds to a Fourier transform of length wl/2.

Note

This function requires **rgl** and is based on fft. See examples of spectro for analysis arguments (wl,zp, ovlp).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

See Also

spectro, dynspec, wf, fft.

Examples

```
data(tico)
spectro3D(tico,f=22050,wl=512,ovlp=75,zp=16,maga=4,palette=rev.terrain.colors)
```

---

symba *Symbol analysis of a numeric (time) series*

Description

This function analyses one or two sequences of symbols from numeric (time) series.

Usage

```
symba(x, y = NULL, symb = 5, collapse = TRUE, entropy = "abs",
plot = FALSE, type = "l", lty1 = 1, lty2 = 2, col1 = 2, col2 = 4,
cex1 = 0.75, cex2= 0.75, xlab = "index", ylab = "Amplitude", legend=TRUE, ...)
```

Arguments

x	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
y	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
symb	the number of symbols used for the discretisation, can be set to 3 or 5 only.
collapse	logical, if TRUE, the symbols are pasted in a character string of length 1.
entropy	either "abs" for an absolute value or "rel" for a relative value, i. e. between 0 and 1.
plot	logical, if TRUE plots the series x (and y) and the respective symbols.
type	if plot is TRUE, type of plot that should be drawn. See plot for details (by default "l" for lines).
lty1	line type of the object x if type="l".
lty2	line type of the object y if type="l".
col1	colour of the object x.
col2	colour of the object y.
cex1	character size of x symbols.
cex2	character size of y symbols.
xlab	title of the x axis.
ylab	title of the y axis.
legend	logical, if TRUE and if y is not NULL adds a legend to the plot.
...	other plot graphical parameters.

Details

The analysis consists in transforming the series into a sequence of symbols (see the function discrets) and in computing the absolute frequency of each symbol within the sequence.
The entropy (*H*) is then calculated using the symbol frequencies. Using the argument `entropy`, the entropy can be expressed along an absolute scale or as a relative value varying between 0 and 1.
If two numeric (time) series are provided (`x` and `y`) the absolute symbol frequencies and entropy of each series is returned. Besides the mutual information (*I*) is estimated according to:

$$I = H_x + H_y - Hxy, with H_x the entropy of \mathtt{x} symbol series, H_y the entropy of \mathtt{y} symbol series, and H_{xy} the joint entropy o$$

with with Hx the entropy of `x` symbol sequence, Hy the entropy of `y` symbol sequence, and Hxy the joint entropy of `x` and `y` symbol sequences.

Value

If `y` is `NULL` a list of three items is returned (s1, freq1, h1).
If `y` is not `NULL`, a list of 6 items is returned (s1, freq1, h1, s2, freq2, h2, I):

`s1`	the sequence of symbols of `x`,
`freq1`	the absolute frequency of each `x` symbol,
`h1`	the entropy of `x` symbol sequence,
`s2`	the sequence of symbols of `y`,
`freq2`	the absolute frequency of each `y` symbol,
`h2`	the entropy of `y` symbol sequence,
`I`	the mutual information between `x` and `y`.

Note

It might be useful to round the values of the input series (see examples).
The mutual information (*I*) should increase with the similarity between the series to compare (`x` and `y`).

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

Cazelles, B. 2004 Symbolic dynamics for identifying similarity between rhythms of ecological time series. *Ecology Letters*, 7: 755-763.

See Also

discrets

Examples

```
# analysis of a frequency spectrum
data(tico)
spec1<-spec(tico,f=22050,at=0.2,plot=FALSE)
symba(spec1[,2],plot=TRUE)
# it might be better to round the values
symba(round(spec1[,2],2),plot=TRUE)
# in that case the symbol entropy is almost similar to the spectral entropy
symba(round(spec1[,2],2),entrop="rel")$h1
sh(spec1)
# to compare two frequency spectra
spec2<-spec(tico,f=22050,wl=512,at=1.1,plot=FALSE)
symba(round(spec1[,2],2),round(spec2[,2],2),plot=TRUE)
```

---

synth   *Synthesis of time wave*

Description

This functions synthesize pure tone sound with amplitude modulation (am) and/or frequency modulation (fm).

Usage

```
synth(f, d, cf, a = 1, shape = NULL, p = 0,
am = c(0, 0), fm = c(0, 0, 0),
plot = FALSE, listen = FALSE, Sample = FALSE,...)
```

Arguments

`f`	sampling frequency (in Hz).
`d`	duration (in s).
`cf`	carrier frequency (in Hz).
`a`	amplitude (linear scale, relative when adding different waves).
`shape`	modification of the whole amplitude shape of the wave. See `details`).
`p`	phase (in radians).
`am`	a vector of length 2 describing amplitude modulation parameters, see `details`.
`fm`	a vector of length 3 describing frequency modulation parameters, see `details`.
`plot`	if `TRUE` returns the spectrogram of the synthezised sound (by default `FALSE`).
`listen`	if `TRUE` the new sound is played back.
`Sample`	if `TRUE` and `plot` is `FALSE` returns an object of class `Sample`
`...`	other `spectro` graphical parameters.

Details

`shape` allows to modify the whole amplitude shape of the wave. There are four options to be given as as character string: (i) "incr" : linear increase (ii) "decr" : linear decrease (iii) "sine" : sinusoid-like shape (iv) "tria" : triangular shape

`am` is a vector of length 2 including:
(1) the amplitude modulation depth (in %),
(2) the frequency of the amplitude modulation.

`fm` is a vector of length 3 including:
(1) the maximum excursion of a sinusoidal frequency modulation (in Hz),
(2) the frequency of a sinusoidal frequency modulation,
(3) the maximum excursion of a linear frequency modulation (in Hz).

Value

If `plot` is `FALSE`, a new wave is returned as a one-column matrix or as a `Sample` object if `Sample` is `TRUE`.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩, Thierry Aubin ⟨thierry.aubin@u-psud.fr⟩ and Caroline Simonis ⟨csimonis@mnhn.fr⟩.

References

Hartmann, W. M. 1998 *Signals, sound and sensation*. New York: Springer.

See Also

`noise`, `pulse`, `echo`

Examples

```
# pure tone
synth(f=22050,d=1,cf=4000,plot=TRUE)
# pure tone with sinusoid-like overall shape
synth(f=22050,d=1,cf=4000,shape="sine",plot=TRUE,osc=TRUE)
# pure tones with am
synth(f=22050,d=1,cf=4000,am=c(50,10),plot=TRUE,osc=TRUE)
# pure tone with +2000 Hz linear fm
synth(f=22050,d=1,cf=4000,fm=c(0,0,2000),plot=TRUE)
# pure tone with sinusoidal fm
# (maximum excursion of 1000 Hz, frequency of 10 Hz)
synth(f=22050,d=1,cf=4000,fm=c(1000,10,0),plot=TRUE,wl=256,ovlp=75)
# pure tone with sinusoidal am
# (maximum excursion of 1000 Hz, frequency of 10 Hz)
# and linear fm (maximum excursion of 1000 Hz)
synth(f=22050,d=1,cf=4000,fm=c(1000,10,1000),plot=TRUE,wl=256,ovlp=75)
# the same with am
synth(f=22050,d=1,cf=4000,am=c(50,10),
```

```
    fm=c(1000,10,1000),plot=TRUE,wl=256,ovlp=75,osc=TRUE)
# the same with am and a triangular overall shape
synth(f=22050,d=1,cf=4000,shape="tria",am=c(50,10),
    fm=c(1000,10,1000),plot=TRUE,wl=256,ovlp=75,osc=TRUE)
# more complex sound
F1<-synth(f=22050,cf=2000,d=1,fm=c(500,5,0))
F2<-synth(f=22050,a=0.8,cf=4000,d=1,fm=c(500,5,0))
F3<-synth(f=22050,a=0.6,cf=6000,d=1,fm=c(500,5,0))
F4<-synth(f=22050,a=0.4,cf=8000,d=1,fm=c(500,5,0))
final1<-F1+F2+F3+F4
spectro(final1,f=22050,wl=512,ovlp=75,scale=FALSE)
```

`th` *Temporal entropy*

Description

Compute the entropy of a temporal envelope.

Usage

```
th(env)
```

Arguments

`env` a data set resulting of an envelope obtained using `env`

Details

Temporal entropy is calculated according to:

$$S = -\frac{\sum_{i=1}^{N} y_i log_2(y_i)}{log_2(N)}$$

with:
y = relative amplitude of the *i* envelope point,
and

$$\sum_{i=1}^{N} y_i = 1$$

and *N* = number of envelope points.

Value

A single value varying between 0 and 1 is returned. The value has no unit.

Note

The temporal entropy of a noisy signal with many amplitude modulations will tend towards 1 whereas the temporal entropy of quiet signal will tend towards 0.
Note, however, that a sustained sound with an almost flat envelope will also show a very high temporal entropy. See examples.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

sh, csh, H

Examples

```
# Temporal entropy of a cicada song
data(orni)
envorni<-env(orni,f=22050,plot=FALSE)
th(envorni)
# Smoothing the envelope might slightly change the result.
envorniS<-env(orni,f=22050,smooth=c(50,0),plot=FALSE)
th(envorniS)
# If we mute a part of the cicada song, the temporal entropy decreases
orni2<-mutew(orni,f=22050,from=0.3,to=0.55,plot=FALSE)
envorni2<-env(orni2,f=22050,plot=FALSE)
th(envorni2)
# The temporal entropy of noise tends towards 1
a<-noise(d=1,f=8000)
enva<-env(a,f=8000,plot=FALSE)
th(enva)
# But be aware that the temporal entropy
# of a sustained sound also tends towards 1
b<-synth(f=8000,d=1,cf=2000,plot=FALSE)
envb<-env(b,f=8000,plot=FALSE)
th(envb)
```

---

tico — *Song of the bird Zonotrichia capensis*

Description

Recording of a song emitted by a male of the neotropical sparrow *Zonotrichia capensis*.

Usage

```
data(tico)
```

Format

A data frame with 39578 observations on the following variable.

V1 a numeric vector

Details

Duration = 1.795 s. Sampling frequency = 22050 hz.

Source

Recording by Thierry Aubin.

Examples

```
data(tico)
oscillo(tico,f=22050)
```

---

timer *Time measurements of a time wave*

Description

This function computes and shows the duration of signal periods, pause periods and their ratio.

Usage

```
timer(wave, f, threshold, smooth = NULL, plot = TRUE,
plotthreshold = TRUE, col = "black", colval = "red",
xlab = "Time (s)", ylab = "Amplitude", ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`threshold`	amplitude threshold for signal detection (in %).
`smooth`	smoothes the envelope by averaging the number of points selected
`plot`	logical, if `TRUE` plots the envelope and the measurements (by default `TRUE`).
`plotthreshold`	logical, if `TRUE` plots the threshold as an horizontal line on the graph (by default `TRUE`).
`col`	colour of the envelope.
`colval`	colour of plotted measurements.
`xlab`	title of the x-axis.
`ylab`	title of the y-axis.
`...`	other `plot` graphical parameters.

Value

If `plot` is `FALSE`, `timer` returns a list containing three components:

`s`	duration of signal periods in seconds
`p`	duration of pause periods in seconds
`r`	ratio between the signal periods and silence

.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

`oscillo`, `cutw`, `pastew`.

Examples

```
data(tico)
timer(tico,f=22050,threshold=5,smooth=50)
# to compare with an oscillographic representation
data(orni)
op<-par(mfrow=c(2,1))
timer(orni,f=22050,threshold=5,smooth=40,tck=0.05,
        bty="l",xaxs="i",colval="blue")
title(main="A cicada song made of five echemes",col="blue")
oscillo(orni,f=22050,k=1,j=1)
par(op)
```

`wasp` *WAve length and SPeed of sound*

Description

This function returns the wavelength and the speed of sound of a given frequency in air, fresh-water or sea-water.

Usage

```
wasp(f, t = 20, c = NULL, s = NULL, d = NULL, medium = "air")
```

Arguments

`f`	frequency (Hz).
`t`	temperature (degree Celcius).
`c`	celerity (m/s) if a wavelength is to be found at a particular speed of sound.
`s`	salinity (parts per thousand) when `medium` is `"sea"`.
`d`	depth (m) when `medium` is `"sea"`.
`medium`	medium for sound propagation, either "air", "fresh" for fresh, or pure, water, "sea" for sea water.

Details

Speed of sound in air is computed according to:

$$c = 331.4 + 0.6 \times t$$

Speed of sound in fresh-water is computed according to Marczak equation:

$$\begin{gathered} c = 1.402385.10^3 + 5.038813 \times t - 5.799136.10^{-2} \times t^2 \\ +3.287156.10^{-4} \times t^3 - 1.398845.10^{-6} \times t^4 \\ +2.787860.10^{-9} \times t^5 \end{gathered}$$

with *t* = temperature in degrees Celsius; range of validity: 0-95Â°C at atmospheric pressure.

Speed of sound in sea-water is computed according to Mackenzie equation:

$$\begin{gathered} c = 1448.96 + 4.591 \times t - 5.304.10^{-2} \times t^2 \\ +2.374.10^{-4} \times t^3 + 1.34 \times (s - 35) + 1.63.10^{-2} \times d \\ +1.675.10^{-7} \times d^2 - 1.025.10^{-2} \times t \times (s - 35) \\ -7.139.10^{-13} \times t \times d^3 \end{gathered}$$

with *t* = temperature in degrees Celsius; *s* = salinity in parts per thousand; *d* = depth in meters; range of validity: temperature 2 to 30 Â°C, salinity 25 to 40 parts per thousand, depth 0 to 8000 m.

Wavelength is obtained following:

$$\lambda = \frac{c}{f}$$

with *c* = speed of sound in meters/second; *f* = frequency in Hertz.

Value

A list of two values is returned:

`l`	wavelength in meters
`c`	speed of sound in meters/second.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

References

http://resource.npl.co.uk

Examples

```
# wavelength (m) of a 2000 Hz air-borne sound at 20Â°C
wasp(f=2000)$l
# [1] 0.1717

# sound speed in sea at 0 and -500 m for a respective temperature of 22Â°C and 11Â°C
wasp(f=1000,s=30,d=c(0,500),t=c(22,11),medium="sea")$c
# [1] 1521.246 1495.414

# wavelength (m) of a 1000 Hz sound in a medium unspecified where c = 1497 m/s
wasp(f=1000,c=1497)$l
# [1] 1.497

# variation of wavelength according to frequency and air temperature
op<-par(bg="lightgrey")
a<-seq(1000,20000,by=100) ; na<-length(a)
b<-seq(-20,40,by=10) ; nb<-length(b)
res<-matrix(numeric(na*nb),nrow=na)
for(i in 1:nb) res[,i]<-wasp(a,t=b[i])$l
matplot(x=a,y=res,type="l",lty=1,col= spectro.colors(nb),
  xlab="Frequency (Hz)",ylab="Wavelength (m)")
title("Wavelength of air-borne sound at different temperatures")
legend(x=15000,y=0.3,c("-20Â°C","-10Â°C","0Â°C","10Â°C","20Â°C","30Â°C","40Â°C"),
  lty=1,col= spectro.colors(nb),bg="grey")
par(op)
```

`wav2flac` *wav-flac file conversion*

Description

This function converts .wav files into .flac files and reversely

Usage

```
wav2flac(file, reverse = FALSE, overwrite = FALSE, exename = NULL, path2exe = NULL)
```

Arguments

`file`	the .wav or .flac file to convert.
`reverse`	logical, if `TRUE` converts a .flac file into a .wav file.
`overwrite`	logical, if `TRUE` overwrites the file to convert.
`exename`	a character string specifying the name of the FLAC binary file. If `NULL`, the dedault name "flac" will be used for Linux OS and "flac.exe" for Windows OS.
`path2exe`	a character string giving the path to the FLAC binary file. If `NULL`, the dedault path "c:/Program Files/FLAC/" will be used for Windows OS.

Details

The function runs FLAC. FLAC has then to be installed first: `http://flac.sourceforge.net/`, if not the function will not work.

Value

A new file is created.

Note

Free Lossless Audio Codec (FLAC) is a file format by Josh Coalson for lossless audio data compression. FLAC reduces bandwidth and storage requirements without sacrificing the integrity of the audio source. Audio sources encoded to FLAC are typically reduced in size 40 to 50 percent.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩, Luis J. Villanueva-Rivera ⟨lvillanu@purdue.edu⟩

References

FLAC website: `http://flac.sourceforge.net/`

See Also

`savewav`

Examples

```
if(nzchar(Sys.which("flac"))) # check that FLAC is installed on your system
{
# synthesis of a 1kHz sound
a<-synth(d=10,f=8000,cf=1000)
# save it as a .wav file in the default working directory
savewav(a,f=8000)
# compress it to FLAC format and overwrite on the file a.wav
wav2flac("a.wav", overwrite=TRUE)
# back to .wav format
wav2flac("a.flac", reverse=TRUE)
# remove the files
```

```
unlink(c("a.wav","a.flac"))
}
```

---

wf *Waterfall display*

Description

This function returns a waterfall display of a short-term Fourier transform or of any matrix.

Usage

```
wf(wave, f = NULL, wl = 512, zp = 0, ovlp = 0, dB = "max0", dBref = NULL, wn = "har
x = NULL, hoff = 1, voff = 1, col = heat.colors,
xlab = "Frequency (kHz)", ylab = "Amplitude (dB)", xaxis = TRUE, yaxis =
TRUE, density = NULL, border = NULL, lines = FALSE, ...)
```

Arguments

`wave`	a `vector`, a `matrix` (first column), an object of class `ts`, `Sample` (left channel), or `Wave` (left channel).
`f`	sampling frequency of `wave` (in Hz). Does not need to be specified if `wave` is an object of class `ts`, `Sample`, or `Wave`.
`wl`	window length for the analysis (even number of points). (by default = 512)
`zp`	zero-padding (even number of points), see `Details`.
`ovlp`	overlap between two successive windows (in %).
`dB`	a character string specifying the type dB to return: "max0" for a maximum dB value at 0, "A", "B", "C" and "D" for common dB weights.
`dBref`	a dB reference value when `dB` is `TRUE`. `NULL` by default but should be set to 2*10e-5 for a 20 microPa reference.
`wn`	window name, see `ftwindow` (by default `"hanning"`).
`x`	a matrix if `wave` is not provided.
`hoff`	horizontal 'offset' which shifts actual x-values slightly per row for visibility. Fractional parts will be removed.
`voff`	vertical 'offset' which separates traces.
`col`	a color or a color palette function to be used to assign colors in the plot
`xlab`	title of the frequency x-axis.
`ylab`	title of the amplitude y-axis.
`xaxis`	a logical, if `TRUE` adds the frequency x-axis according to `f`.
`yaxis`	a logical, if `TRUE` adds the amplitude y-axis according.

density	argument of polygon: the density of shading lines, in lines per inch. The default value of 'NULL' means that no shading lines are drawn. A zero value of 'density' means no shading nor filling whereas negative values (and 'NA') suppress shading (and so allow color filling).
border	argument of polygon: the color to draw the border. The default, 'NULL', means to use 'par("fg")'. Use 'border = NA' to omit borders.
lines	a logical, if TRUE plots lines instead of surfaces (polygons).
...	other graphical arguments to passed to plot

Details

Data input can be either a time wave (wave) or a matrix (x). In that case, if xaxis is set to TRUE the x-axis will follow the row index. To change it, turn xaxis to FALSE and use axis afterwards. See examples.

Note

The function is well adapted to display short-term Fourier transform. However, any matrix can be called using the argument x instead of wave.

Author(s)

Carl G. Witthoft and Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

spectro, spectro3D, dynspec

Examples

```
data(tico)
wf(tico,f=22050)
# changing the display parameters
jet.colors <- colorRampPalette(c("blue", "green"))
wf(tico,f=22050, hoff=0, voff=2, col=jet.colors, border = NA)
# matrix input instead of a time wave and transparent lines display
m <- numeric()
for(i in seq(-pi,pi,len=40)) {m <- cbind(m,10*(sin(seq(0,2*pi,len=100)+i)))}
wf(x=m, lines=TRUE, col="#0000FF50",xlab="Time", ylab="Amplitude",
main="waterfall display")
```

zapsilw *Zap silence periods of a time wave*

Description

This function simply delete the silence periods of a time wave.

Usage

```
zapsilw(wave, f, threshold = 5, plot = TRUE, Sample = FALSE, ...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
threshold	amplitude threshold (in %) between silence and signal.
plot	logical, if TRUE plots the orginal and the new oscillograms (by default TRUE).
Sample	if TRUE and plot is FALSE returns an object of class Sample
...	other oscillo graphical parameters.

Value

If plot is FALSE, a new wave is returned as a one-column matrix or as a Sample object if Sample is TRUE.

Note

Use the argument threshold to set the level of silence. See the examples.

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩

See Also

afilter, oscillo

Examples

```
data(orni)
zapsilw(orni,f=22050,colwave="red")
# setting the threshold value
zapsilw(orni,f=22050,threshold=1)
```

zc — *Instantaneous frequency of a time wave by zero-crossing*

Description

This function measures the period of a full oscillating cycle.

Usage

```
zc(wave, f, plot = TRUE, interpol = 1, threshold = NULL,
xlab = "Time (s)", ylab = "Frequency (kHz)", ylim = c(0, f/2000),...)
```

Arguments

wave	a vector, a matrix (first column), an object of class ts, Sample (left channel), or Wave (left channel).
f	sampling frequency of wave (in Hz). Does not need to be specified if wave is an object of class ts, Sample, or Wave.
plot	logical, if TRUE plots the dominant frequency along the time wave(by default TRUE).
interpol	interpolation factor.
threshold	amplitude threshold for signal detection (in %).
xlab	title of the x axis.
ylab	title of the y axis.
ylim	the range of y values.
...	other plot graphical parameters.

Details

If plot is FALSE, zc returns a vector of numeric data with the instantaneous frequency.

Value

If plot is FALSE, zc returns a two-column matrix, the first column corresponding to time in seconds (*x*-axis) and the second column corresponding to the instantaneous frequency of the time wave in kHz (*y*-axis).
'NA's correspond either to pause periods (e. g. detected applying threshold or sections of the time wave not crossing the zero line. To remove 'NA's with na.omit allows to get only instantaneous frequency values but discards information about pause sections.

Note

interpol adds points to the time wave by linear interpolation (through approx). This increases measurement precision but as well time process. Type argument of plot cannot be set to "l".

Author(s)

Jerome Sueur ⟨sueur@mnhn.fr⟩, Caroline Simonis ⟨csimonis@mnhn.fr⟩ and Thierry Aubin ⟨thierry.aubin@u-psud.fr⟩

References

Hopp, S. L., Owren, M. J. and Evans, C. S. (Eds) 1998. *Animal acoustic communication.* Springer, Berlin, Heidelberg.

See Also

ifreq

Examples

```
data(pellucens)
pellu1<-cutw(pellucens,f=22050,from=0,to=1,plot=FALSE)
# without interpolation
zc(pellu1,f=22050,threshold=5,pch=20)
# with interpolation
zc(pellu1,f=22050,threshold=5,interpol=20,pch=20)
# a way to plot with a line and to filter low frequencies
pellu2<-zc(pellu1,f=22050,threshold=5,interpol=20,plot=FALSE)
pellu3<-na.omit(pellu2[,2])
pellu4<-pellu3[pellu3>3]
plot(x=seq(0,nrow(pellu1)/22050,length.out=length(pellu4)),
    y=pellu4,type="l",xlab="Time(s)",ylab="Frequency(kHz)")
```

Index

∗Topic **IO**
export, 46
ftwindow, 55
savewav, 89
wav2flac, 116
∗Topic **datagen**
drawenv, 39
echo, 43
noise, 71
pulse, 79
setenv, 91
synth, 109
∗Topic **datasets**
orni, 72
peewit, 78
pellucens, 79
sheep, 95
tico, 112
∗Topic **data**
attenuation, 5
∗Topic **dplot**
addsilw, 2
ama, 4
autoc, 6
ccoh, 8
ceps, 10
cepstro, 12
coh, 14
corenv, 16
corspec, 18
covspectro, 20
cutw, 26
dBscale, 27
deletew, 30
dfreq, 31
diffenv, 33
diffspec, 34
dynspec, 40
env, 44
fadew, 47
fma, 54
fund, 56
ifreq, 60
meanspec, 64
mutew, 70
oscillo, 73
oscilloST, 75
pastew, 77
Q, 80
repw, 82
resamp, 83
revw, 84
rmoffset, 87
seewave, 90
simspec, 95
spec, 97
specprop, 100
spectro, 102
spectro3D, 105
timer, 113
wf, 118
zc, 121
∗Topic **math**
convSPL, 15
dBweight, 28
fdoppler, 48
mel, 67
micsens, 68
moredB, 69
wasp, 114
∗Topic **ts**
addsilw, 2
afilter, 3
ama, 4
autoc, 6
ccoh, 8
ceps, 10
cepstro, 12

coh, 14
corenv, 16
corspec, 18
covspectro, 20
crest, 22
csh, 23
cutspec, 25
cutw, 26
dBscale, 27
dBweight, 28
deletew, 30
dfreq, 31
diffenv, 33
diffspec, 34
diffwave, 36
discrets, 38
drawenv, 39
dynspec, 40
echo, 43
env, 44
fadew, 47
ffilter, 49
field, 51
fir, 52
fma, 54
ftwindow, 55
fund, 56
H, 58
hilbert, 59
ifreq, 60
lfs, 62
listen, 63
meanspec, 64
mutew, 70
noise, 71
oscillo, 73
oscilloST, 75
pastew, 77
pulse, 79
Q, 80
repw, 82
resamp, 83
revw, 84
rmam, 85
rmnoise, 86
rmoffset, 87
rms, 88
seewave, 90
setenv, 91
sfm, 92
sh, 93
simspec, 95
spec, 97
specprop, 100
spectro, 102
spectro3D, 105
symba, 107
synth, 109
th, 111
timer, 113
wf, 118
zapsilw, 120
zc, 121

addsilw, 2, *27*, *31*, *47*, *71*, *78*, *83*, *85*
afilter, 3, *50*, *53*, *87*, *120*
ama, 4, *55*
approx, *121*
as.Sample, *90*
attenuation, 5, *16*
autoc, *3*, 6, *12*, *14*, *57*
axis, *27*, *119*

ccoh, 8, *15*
ceps, *7*, 10, *13, 14*, *57*
cepstro, 12, *12*, *57*
coh, *10*, 14
contour, *9*, *13*, *103, 104*
convolve, *43*, *53*
convSPL, *6*, 15, *29*, *69, 70*
cor, *17–19*, *21, 22*
cor.test, *17–19*
corenv, 16, *22*, *34*
corspec, 18, *18, 19*, *22*, *36*, *66*, *81*, *97*, *99*
covspectro, *18, 19*, 20, *56*
crest, 22
csh, *3*, 23, *59*, *93, 94*, *112*
cutspec, 25
cutw, *2*, 26, *31*, *47*, *71*, *74*, *78*, *83*, *114*

dBscale, 27, *104*
dBweight, *16*, 28, *70*
deletew, *2*, *27*, 30, *47*, *71*, *78*, *83*, *85*
dfreq, *3*, 31, *56*
diffenv, 33, *36–38*, *97*
diffspec, 34, *34*, *37, 38*, *66*, *97*
diffwave, *34*, 36

discrets, 38, *108*
drawenv, 39, *92*
dynspec, 40, *66*, *99*, *104*, *106*, *119*

echo, 43, *110*
embed, *11*, *65*, *98*
env, *5*, *17, 18, 33, 34*, *37*, *40–42*, 44, *58*, *91, 92*, *111*
export, 46, *90*

fadew, *2*, *27*, *31*, 47, *71*, *78*, *83*, *85*
fdoppler, 48
ffilter, *3*, 49, *53*, *63*
fft, *11*, *32*, *42*, *50*, *53*, *62*, *66*, *81*, *99*, *104*, *106*
field, 51
filled.contour, *9*, *28*, *104*
fir, *50*, 52
fir1, *53*
fir2, *53*
fma, *5*, 54
ftwindow, *21*, *24*, *31*, *41*, *50*, *53*, 55, *62*, *65*, *98*, *103*, *106*, *118*
fund, *12*, *14*, 56

H, 58, *94*, *112*
hilbert, *45*, *55*, 59, *61, 62*, *86*

ifreq, *55*, 60, *60*, *122*
IQR, *101*

kernel, *17*, *33*, *37*, *42*, *45*, *58*, *91*

lfs, *50*, *53*, 62
listen, 63
locator, *9*, *40*, *104*

mad, *101*
mean, *89*, *101*
meanspec, *4, 5*, *18, 19*, *25*, *32*, *35, 36*, *50*, *53*, *56*, 64, *81*, *92, 93*, *96, 97*, *99, 100*
median, *101*
mel, 67
micsens, 68
Mod, *45*
moredB, *6*, *16*, *29*, 69
mutew, *2*, *27*, *31*, *47*, 70, *78*, *83*, *85*

na.omit, *121*
noise, 71, *80*, *87*, *110*

orni, 72
oscillo, *2, 3*, *9*, *23*, *26, 27*, *30, 31*, *42, 43*, *45*, *47*, *60*, *70, 71*, 73, *76–78*, *82–85*, *87, 88*, *91*, *103, 104*, *114*, *120*
oscilloST, *74*, 75

par, *74*
pastew, *2*, *27*, *31*, *47*, *71*, *74*, 77, *83*, *85*, *114*
peewit, 78
pellucens, 79
play, *64*
plot, *4*, *6, 7*, *11*, *14*, *17*, *19*, *21*, *24*, *32, 33*, *35*, *41*, *57*, *61*, *65*, *80, 81*, *96*, *98*, *100*, *107*, *113*, *119*, *121*
polygon, *119*
pulse, *72*, 79, *110*

Q, 80
quantile, *101*

repw, *78*, 82
resamp, 83
revw, *2*, *27*, *31*, *47*, *71*, *78*, *83*, 84
rmam, 85
rmnoise, 86
rmoffset, 87
rms, *22, 23*, 88
rnorm, *71*
runif, *71*

Sample, *2–4*, *7, 8*, *10*, *12*, *14*, *17*, *20–22*, *24*, *26, 27*, *30, 31*, *33*, *37–41*, *43, 44*, *46*, *47*, *50*, *53*, *54*, *57–59*, *61–63*, *65*, *70*, *71*, *73*, *75*, *77*, *80*, *82–87*, *89*, *91*, *97*, *98*, *103*, *105*, *107*, *109, 110*, *113*, *118*, *120, 121*
saveSample, *89, 90*
savewav, 89, *117*
sd, *101*
seewave, 90
seewave-package (*seewave*), 90
setenv, *40*, 91
sfm, *25*, 92, *94*, *101*
sh, *24, 25*, *58, 59*, 93, *93*, *101*, *112*
sheep, 95
simspec, *36*, *66*, 95
smooth.spline, *86, 87*
spec, *18, 19*, *25*, *32*, *35, 36*, *42*, *50*, *53–56*, *65, 66*, *81*, *92, 93*, *96*, 97, *97*, *100*

spec.pgram, *9, 10*, *15*
specprop, 100
spectro, *9, 10*, *13*, *15*, *22*, *27, 28*, *32*, *42*, *50*, *56*, *62, 63*, 102, *106*, *109*, *119*
spectro3D, *42*, *56*, *104*, 105, *119*
symba, *39*, 107
synth, *40*, *44*, *72*, *80*, *92*, 109

th, *24*, *58, 59*, *94*, 111
tico, 112
timer, *3*, *74*, 113

wasp, *49*, 114
wav2flac, 116
Wave, *2–4*, *7, 8*, *10*, *12*, *14*, *17*, *20–22*, *24*, *26*, *30, 31*, *33*, *37–39*, *41*, *43, 44*, *46, 47*, *50*, *53, 54*, *57–59*, *61–63*, *65*, *70*, *73*, *75*, *77*, *82–87*, *89*, *91*, *97, 98*, *103*, *105*, *107*, *113*, *118*, *120, 121*
wf, *42*, *104*, *106*, 118
write.table, *46*

zapsilw, *2*, *27*, *31*, *47*, *71*, *78*, *83*, 120
zc, *3*, *62*, 121