

# Package ‘seewave’

April 11, 2012

**Type** Package

**Title** Time wave analysis and graphical representation

**Version** 1.6.3

**Date** 2012-04-10

**Author** Jerome Sueur <sueur@mnhn.fr>, Thierry Aubin, Caroline Simonis

**Maintainer** Jerome Sueur <sueur@mnhn.fr>

**Webpage** <http://rug.mnhn.fr/seewave>

**Depends** R(>= 2.15.0)

**Encoding** UTF-8

**SystemRequirements** FLAC, FFTW

**Imports** fftw, rgl, rpanel, tcltk, tuneR,

**Suggests** audio, sound

**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://rug.mnhn.fr/seewave>

**Repository** CRAN

**Date/Publication** 2012-04-11 13:38:05

**R topics documented:**

addsilw	4
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
dynoscillo	42
dynspec	43
echo	46
env	47
export	49
fadew	50
fbands	51
fdoppler	53
ffilter	54
field	56
fir	57
fma	59
fpeaks	60
ftwindow	63
fund	64
H	65
hilbert	66
ifreq	67
itakura.dist	69
kl.dist	70

ks.dist . . . . .	72
lfs . . . . .	73
listen . . . . .	75
localpeaks . . . . .	76
logspec.dist . . . . .	77
meandB . . . . .	78
meanspec . . . . .	79
mel . . . . .	82
micsens . . . . .	83
moredB . . . . .	84
mutew . . . . .	85
noisew . . . . .	86
notefreq . . . . .	87
octaves . . . . .	88
orni . . . . .	89
oscillo . . . . .	90
oscilloST . . . . .	92
pastew . . . . .	94
peewit . . . . .	95
pellucens . . . . .	96
phaseplot . . . . .	97
pulse . . . . .	98
Q . . . . .	99
repw . . . . .	100
resamp . . . . .	102
revw . . . . .	103
rmam . . . . .	104
rmnoise . . . . .	105
rmoffset . . . . .	106
rms . . . . .	107
roughness . . . . .	108
ruogo . . . . .	109
savewav . . . . .	110
seedata . . . . .	111
seewave . . . . .	112
setenv . . . . .	112
sfm . . . . .	114
sh . . . . .	115
sheep . . . . .	117
simspec . . . . .	118
smoothw . . . . .	120
spec . . . . .	121
specprop . . . . .	123
spectro . . . . .	126
spectro3D . . . . .	129
symba . . . . .	131
synth . . . . .	133
th . . . . .	135

tico . . . . .	137
timer . . . . .	138
wasp . . . . .	139
wav2flac . . . . .	141
wf . . . . .	143
zapsilw . . . . .	145
zc . . . . .	146

<b>Index</b>	<b>148</b>
--------------	------------

---

addsilw	<i>Add or insert a silence section</i>
---------	--

---

### Description

Add or insert a silence section to a time wave.

### Usage

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

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> graphical parameters.

### Value

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

### Author(s)

Jerome Sueur <sueur@mnhn.fr>

**See Also**

[oscillo](#), [cutw](#), [deletew](#), [fadew](#), [pastew](#), [mutew](#), [revw](#), [zapsilw](#)

---

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, output = "matrix", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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. The class of the returned object is set with the argument output.

**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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">meanspec</a> 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:

<code>spec</code>	the spectrum computed
<code>peaks</code>	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**

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

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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> graphical parameters.

**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, fmin, fmax, threshold = NULL, plot = TRUE,
xlab = "Time (s)", ylab = "Frequency (kHz)", ylim = c(0, f/2000), pb =
FALSE, ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	length of the window for the analysis (even number of points, by default = 512).
fmin	the minimum frequency to detect (in Hz). See details.
fmax	the maximum frequency to detect (in Hz). See details
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.
pb	if TRUE returns a text progress bar in the console.
...	other <a href="#">plot</a> graphical parameters.

**Details**

'fmin' and 'fmax' can help by reducing computing time but can also produce less accurate results.

**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 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](#), [acf](#)

**Examples**

```
data(sheep)
# fundamental frequency of a sheep
res<-autoc(sheep, f=8000, threshold=5, fmin=100, fmax=700)
spectro(sheep, f=8000, ovlp=75, scale=FALSE)
points(res, pch=20)
legend(0.5, 3.6, "Fundamental frequency", pch=20, bty=0, cex=0.7)
```

**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",
     xlab = "Time (s)", ylab = "Frequency (kHz)",
     scalelab = "Coherence",
     main = NULL,
     scalefontlab = 1, scalecexlab = 0.75, axisX = TRUE, axisY = TRUE,
     flim = NULL, flimd = NULL,
     ...)
```

**Arguments**

wave1	a first R object
wave2	a second R object
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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.
xlab	label of the time axis.
ylab	label of the frequency axis.
scalelab	label fo the amplitude scale.
main	label of the main title.
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 <a href="#">contour</a> and <a href="#">oscillo</a> 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

This function returns a list of three items:

time	a numeric vector corresponding to the time axis.
freq	a numeric vector corresponding to the frequency axis.
amp	a numeric matrix corresponding to the coherence. 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,
      tidentify = FALSE, fidentify = FALSE, col = "black", cex = 1, plot = TRUE,
      qlab = "Quefreny (bottom: s, up: Hz)", alab = "Amplitude",
      qlim = NULL, alim = NULL, type = "l", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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).
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.
plot	logical, if TRUE plots the cepstrum.
qlab	title of the quefreny axis.

alab	title of the amplitude axis.
qlim	range of quefrequency axis.
alim	range of amplitude axis.
type	if plot is TRUE, type of plot that should be drawn. See <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> graphical parameters.

### Details

The cepstrum of a time wave is the inverse Fourier transform of the logarithm of the Fourier transform. The cepstrum of a wave  $s$  is then calculated as follows:

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

The independent variable of a cepstral graph is called the quefrequency. The quefrequency 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 coordinate. 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.

When using `tidentify` or `tidentify`, press 'stop' tools bar button to return values in the console.

### Value

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

### Warning

The argument peaks is no more available (version > 1.5.6). See the function [fpeaks](#) for peak(s) detection.

### 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>

### References

Oppenheim, A.V. and Schafer, R.W. 2004. From frequency to quefrequency: 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)
```

---

cepstro	<i>2D-cepstrogram of a time wave</i>
---------	--------------------------------------

---

**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",
xlab = "Time (s)", ylab = "Quefreny (ms)",
scalelab = "Amplitude", main = NULL, scalefontlab = 1, scalecexlab = 0.75,
axisX = TRUE, axisY = TRUE, ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 over-plot (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.
xlab	label of the time axis.
ylab	label of the quefrequency axis.
main	label of the main title.
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 <a href="#">contour</a> 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

This function returns a list of three items:

time	a numeric vector corresponding to the time axis.
freq	a numeric vector corresponding to the quefrequency axis.
amp	a numeric matrix corresponding to the 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 quefrequency: a history of the cepstrum. *Signal Processing Magazine IEEE*, 21: 95-106.

**See Also**

[ceps](#), [fund](#), [autoc](#)

**Examples**

```
data(sheep)
sheepc <- cutw(sheep, f=8000, from = 0.19, to = 2.3)
cepstro(sheepc, 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 first R object.
wave2	a second R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> 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/m<sup>2</sup>) 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**

<code>x</code>	a numeric vector or a matrix describing SPL values (in dB).
<code>d</code>	the distance from the sound source where SPL values have been measured (in meter) (by default = 1m)
<code>Iref</code>	reference intensity (in Watt/m <sup>2</sup> ) (by default = 10e-12)
<code>pref</code>	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", pb = FALSE, ...)
```

**Arguments**

wave1	a first R object.
wave2	a second R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
envt	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See <a href="#">env</a> .
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 <a href="#">env</a> .
ksmooth	kernel smooth via <a href="#">kernel</a> . See <a href="#">env</a> .
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 <a href="#">cor</a> ).
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 <a href="#">plot</a> for details (by default "l" for lines).
pb	if TRUE returns a text progress bar in the console.
...	other <a href="#">plot</a> 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 column 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 <a href="#">cor</a> ).
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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> 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 column 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("spearman 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", pb = FALSE, ...)

```

## Arguments

wave1            a first R object.  
wave2            a second R object.

f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	length of the window for the analysis (even number of points, by default = 512).
wn	window name, see <a href="#">ftwindow</a> (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 <a href="#">cor</a> ).
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 <a href="#">plot</a> for details (by default "l" for lines).
pb	if TRUE returns a text progress bar in the console.
...	other <a href="#">plot</a> 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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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:

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

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, fftw = FALSE, threshold = NULL,
plot = TRUE, xlab = "Times (s)", ylab = "Spectral Entropy",
ylim = c(0, 1.1), type = "l", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	if <code>at</code> is not null, length of the window for the analysis (even number of points, by default = 512).
wn	window name, see <a href="#">ftwindow</a> (by default "hanning").
ovlp	overlap between two successive windows (in %).
fftw	if TRUE calls the function FFT of the library <code>fftw</code> . See Notes of the <code>spectro</code> .
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 <code>plot</code> is TRUE, type of plot that should be drawn. See <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> 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](mailto: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 effects
# 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 <a href="#">spec</a> or <a href="#">meanspec</a> .
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, output="matrix", ...)

```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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. The class of the returned object is set with the argument `output`.

**Author(s)**

Jerome Sueur

**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**

<code>collevels</code>	a set of levels which are used to partition the amplitude range of the spectrogram (in dB).
<code>palette</code>	a color palette function to be used to assign colors in the plot, see note.
<code>side</code>	side of the axis.
<code>textlab</code>	text of the label.
<code>cexlab</code>	character size of the label.
<code>fontlab</code>	font of the label.
<code>collab</code>	colour of the label.
<code>colaxis</code>	colour of the axis.
<code>...</code>	other <a href="#">axis</a> 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 Wikipedia 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, output = "matrix", ...)
```

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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. The class of the returned object is set with the argument output.

**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, fftw= FALSE, at = NULL, threshold
= NULL, bandpass = NULL, clip = NULL,
plot = TRUE, xlab = "Times (s)", ylab = "Frequency (kHz)",
ylim = c(0, f/2000), ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	length of the window for the analysis (even number of points, by default = 512).
wn	window name, see <a href="#">ftwindow</a> (by default "hanning").

ovlp	overlap between two successive analysis windows (in %).
fftw	if TRUE calls the function FFT of the library <code>fftw</code> . See Notes of the <code>spectro</code> .
at	time position where the dominant frequency has to be computed (in s.).
threshold	amplitude threshold for signal detection (in %).
bandpass	a numeric vector of length two, giving the lower and upper limits of a frequency bandpass filter (in Hz).
clip	a numeric value to select dominant frequency values according to their amplitude in reference to a maximal value of 1 for the whole signal (has to be $>0$ & $< 1$ ).
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.
...	other <a href="#">plot</a> 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 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](mailto:sueur@mnhn.fr)>

### See Also

[spec](#), [meanspec](#), [spectro](#).

### Examples

```
data(tico)
f <- 22050
# default
dfreq(tico,f)
# using the amplitude threshold and changing the graphical output
dfreq(tico, f, ovlp=50,threshold=5, type="l", col=2)
# using 'at' argument for specific positions along the time axis
dfreq(tico, f, at=c(0.25, 0.75, 1.2, 1.6))
dfreq(tico, f, at=seq(0.5, 1.4, by=0.005), threshold=5)
# a specific number of measures on a single note
dfreq(tico, f, at=seq(0.5, 0.9, len=100), threshold=5, xlim=c(0.5,0.9))
# overlap on spectrogram
# and use of 'clip' argument to better track the dominant frequency
```

```
# in noisy conditions
op <- par()
tico <- tico@left/max(tico@left) + noisew(d=length(tico@left)/f, f)
spectro(tico, f)
res <- dfreq(tico, f, clip=0.2, plot=FALSE)
points(res, col=2, pch =13)
par(op)
```

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 first R object.
wave2	a second R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
envt	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See <a href="#">env</a> .
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 <a href="#">env</a> .
ksmooth	kernel smooth via <a href="#">kernel</a> . See <a href="#">env</a> .
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 <a href="#">plot</a> graphical parameters.

**Details**

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

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

**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) ; tico <- tico@left
data(orni) ; orni <- orni@left
# selection in tico of two waves with similar duration
tico2<-tico[1:length(orni)]
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 <a href="#">spec</a> or <a href="#">meanspec</a> (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 <a href="#">spec</a> or <a href="#">meanspec</a> (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 <a href="#">spec</a> or <a href="#">meanspec</a> .
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 <a href="#">plot</a> 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 <a href="#">plot</a> graphical parameters.

**Details**

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

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

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](#), [kl.dist](#), [ks.dist](#), [logspec.dist](#), [itakura.dist](#)

**Examples**

```
a<-noisew(f=8000,d=1)
b<-synth(f=8000,d=1,cf=2000)
c<-synth(f=8000,d=1,cf=1000)
d<-noisew(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, w1 = 512, envt = "hil",  
msmooth = NULL, ksmooth = NULL)
```

**Arguments**

wave1	a first R object.
wave2	a second R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
w1	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 <a href="#">env</a> .
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 <a href="#">env</a> .
ksmooth	kernel smooth via <a href="#">kernel</a> . See <a href="#">env</a> .

**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 `w1`, `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) ; tico <- tico@left
data(orni) ; orni <- orni@left
# selection in tico to have two waves of similar duration (length)
tico <- tico[1:length(orni)]
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 <code>ts</code> , <a href="#">Sample</a> (left channel), or <a href="#">Wave</a> (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 comparison 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 *Increasing*,
- *D* if the series is *Decreasing*,
- *F* if the series remains *Flat*,

when `symb` is set to 5, each value will be replaced by either:

- *I* if the series is *Increasing*,
- *D* if the series is *Decreasing*,
- *F* if the series remains *Flat*,
- *P* if the series shows a *Peak*,
- *T* if the series shows a *Trough*.

**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, output = "matrix")
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".

### 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. The class of the returned object is set with the argument output.

### 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 envelope
# stop (ESC on Windows; right mouse button on Linux)
# check the result on the second graphics device opened thanks to plot=TRUE
```

---

dynoscillo

*Dynamic oscillogram*

---

### Description

This graphical function displays a time wave as an windowed oscillogram.

### Usage

```
dynoscillo(wave, f, wd = NULL, wl = NULL, wnb = NULL, title = TRUE, ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wd	a numerical vector, duration of the window (in seconds)
wl	a numerical vector, length of the window (in number of points).
wnb	a numerical vector, number of windows (no unit).
title	a logical, if TRUE displays a title with information regarding window size and number.
...	other <a href="#">plot</a> graphical parameters.

**Details**

The arguments wd, wl and wn have to be used isolated, not in conjunction. They basically do the same, ie they set the duration of the zooming window that is slided along the signal. For instance, for a 5 seconds sound with a sampling rate (f) at 44.1 kHz,  $wl = 4096$  is equivalent to  $wd = 4096 / 44100 = 0.093$  s and equivalent to  $wnb = 5 * 4096 / 44100 = 53$ .

**Note**

This function requires the package `rpanel`.

**Author(s)**

Jerome Sueur

**See Also**

[oscillo](#), [oscilloST](#), [dyspec](#).

**Examples**

```
require(rpanel)
data(tico)
dynoscillo(tico, wn=4)
```

---

dyspec

*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**

```
dyspec(wave, f, wl = 512, wn = "hanning", zp = 0,
ovlp = 0, fftw = FALSE, norm = FALSE, dB = NULL, dBref = NULL, 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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">ftwindow</a> (by default "hanning").
zp	zero-padding (even number of points), see Details.
ovlp	overlap between two successive windows (in %).
fftw	if TRUE calls the function FFT of the library fftw. See Notes of the spectro.
norm	logical, if TRUE compute a normalised sliding spectrum.
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).
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 <a href="#">plot</a> 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 plotted: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See <a href="#">env</a> .
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 <a href="#">env</a> .
ksmooth	when env is not NULL, kernel smooth via <a href="#">kernel</a> . See <a href="#">env</a> .
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)
require(rpanel)
dyspec(sheep, f=8000, w1=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, output = "matrix", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
amp	a vector describing the relative amplitude of the successive echoes. Each value of the vector should be 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.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> graphical parameters.

**Details**

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

**Value**

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

**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, ssmooth = NULL, norm = FALSE, plot = TRUE, k = 1, j = 1, ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">kernel</a> . See examples.
ssmooth	sum smooth.
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 <a href="#">oscillo</a> 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% overlapping mean sliding window
env(tico,f=22050,msmooth=c(10,50))
# smoothing kernel
env(tico,f=22050,ksmooth=kernel("daniell",10))
# sum smooth
env(tico,f=22050,ssmooth=50)
# 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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">write.table</a> 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, output = "matrix", ...)
```

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> graphical parameters.

### Value

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

### Author(s)

Jerome Sueur <sueur@mnhn.fr>

### See Also

[oscillo](#), [addsilw](#), [cutw](#), [deletew](#), [mutew](#), [pastew](#), [revw](#), [zapsilw](#)

## Examples

```
a<-noisew(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)
```

---

fbands	<i>Frequency bands plot</i>
--------	-----------------------------

---

## Description

This graphical function returns a frequency spectrum as a bar plot.

## Usage

```
fbands(spec, f = NULL, bands = 10, width = FALSE, plot = TRUE,
       xlab = "Frequency (kHz)", ylab = "Relative amplitude", ...)
```

## Arguments

spec	a data set resulting of a spectral analysis obtained with <a href="#">spec</a> or <a href="#">meanspec</a> . Can be in dB.
f	sampling frequency of spec (in Hz). Not requested if the first column of spec contains the frequency axis.
bands	a numeric vector. If vector of length 1, then sets the number of bands dividing in equal parts the spectrum. If of length > 1, then takes the values as kHz limits of the bands dividing the spectrum. These bands can be of different size. See details and examples.
width	logical, if TRUE and that bands is an irregular series of values, then the width of the bands will be proportional to the frequency limits defined in bands.
plot	logical, if TRUE, a plot showing the peaks is returned.
xlab	label of the x-axis.
ylab	label of the y-axis.
...	other <a href="#">plot</a> graphical parameters.

## Details

The function proceeds as follows

- divides the spectrum in bands. The limits of the bands are set with the argument bands. There are two options:

- you set a number of bands with equal size by giving a single value to bands. For instance, setting bands to a value of 10 will slice the spectrum in 10 equal parts and return 10 local peaks.
  - you set the limits of the bands. This is achieved by giving a numeric vector to bands. The limits can follow a regular or irregular series. For instance attributing the vector `c(0,2,4,8)` will generate the following bands [0,2[, [2,4[, [4,8] kHz. Be aware that the last value should not exceed half the sampling frequency used to obtain the spectrum spec.
- uses the function `barplot`.

### Value

A two-column matrix, the first column corresponding to the frequency values (x-axis, mean of the bars limits) and the second column corresponding to height values (y-axis) of the bars.

### Note

The value below bars is the mean between the corresponding frequency limits.

### Author(s)

Jerome Sueur

### See Also

`meanspec`, `spec`, `barplot`.

### Examples

```
data(sheep)
spec <- meanspec(sheep, f=8000, plot=FALSE)
# default plot
fbands(spec)
# setting a specific number of bands
fbands(spec, bands=6)
#setting specific regular bands limits
fbands(spec, bands=seq(0,4,by=0.25))
# some plot tuning
op <- par(las=1)
fbands(spec, bands=seq(0,4,by=0.1),
       horiz=TRUE, col=heat.colors(41),
       xlab="", ylab="",
       cex.axis=0.75, cex.names = 0.75,
       axes=FALSE)
par(op)
# showing or not the width of the bands
oct <- octaves(440,3)/1000
op <- par(mfrow=c(2,1))
fbands(spec, bands=oct, col="blue")
fbands(spec, bands=oct, width = TRUE, col="red")
par(op)
# kind of horizontal zoom
```

```

op <- par(mfrow=c(2,1))
fbands(spec, bands=seq(0,4,by=0.2), col=c(rep(1,10),
      rep("orange",5),rep(1,5)), main="all frequency range")
fbands(spec, bands=seq(2,3,by=0.2),
      col="orange", main="a subset or zoom in")
par(op)
# kind of dynamic frequency bands
specs <- dynspec(sheep, f=8000, plot= FALSE)$amp
out <- apply(specs, f=8000, MARGIN=2, FUN = fbands, bands = seq(0,4,by=0.2), col = 1, ylim=c(0,max(specs)))

```

---

 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

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

$v_s$  = speed of the source,

$v_o$  = 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", fftw = FALSE, output="matrix")
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">spec</a> and <a href="#">meanspec</a> . w1 is no more required. See examples.
w1	window length for the analysis (even number of points).
wn	window name, see <a href="#">ftwindow</a> (by default "hanning").
fftw	if TRUE calls the function FFT of the library fftw. See Notes of the <a href="#">spectro</a> .
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".

**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**

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**See Also**

[afilter](#), [lfs](#), [fir](#)

**Examples**

```
a<-noisew(f=8000,d=1)
# low-pass
b<-ffilter(a, f=8000, to=1500)
spectro(b, f=8000, w1=512)
# high-pass
c<-ffilter(a, f=8000, from=2500)
spectro(c, f=8000, w1=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 than 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  $k \times d$  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  $k \times d$  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,
    w1 = 512, wn = "hanning", listen = FALSE, output = "matrix")
```

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">spec</a> and <a href="#">meanspec</a> . w1 is no more required. See examples.
w1	window length of the impulse filter (even number of points).
wn	window name, see <a href="#">ftwindow</a> (by default "hanning").
listen	if TRUE the new sound is played back.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".

### 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

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

### 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](#)

### Examples

```
a<-noisew(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<-noisew(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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
threshold	amplitude threshold for signal detection (in %).
plot	logical, if TRUE the spectrum of the instantaneous frequency (by default TRUE).
...	other <a href="#">spec</a> 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:

<code>spec</code>	the spectrum computed
<code>peaks</code>	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)
```

---

fpeaks

*Frequency peak detection*

---

**Description**

This function searches for peaks of a frequency spectrum.

**Usage**

```
fpeaks(spec, f = NULL, nmax = NULL, amp = NULL, freq = NULL,
        threshold = NULL, plot = TRUE, title = TRUE,
        xlab = "Frequency (kHz)", ylab = "Amplitude",
        labels = TRUE, legend = TRUE, collab = "red", ...)
```

**Arguments**

<code>spec</code>	a data set resulting of a spectral analysis obtained with <a href="#">spec</a> or <a href="#">meanspec</a> . Can be in dB.
<code>f</code>	sampling frequency of <code>spec</code> (in Hz). Not requested if the first column of <code>spec</code> contains the frequency axis.
<code>nmax</code>	maximal number of peaks detected. Overrides <code>amp</code> , <code>freq</code> and <code>threshold</code> . See details.
<code>amp</code>	amplitude slope parameter, a numeric vector of length 2. Refers to the amplitude slope of the peak. The first value is the left slope and the second value is the right slope. Only peaks with higher slopes than threshold values will be kept. See details.

freq	frequency threshold parameter. If the frequency difference of two successive peaks is less than this threshold, then the peak of highest amplitude will be kept only. See details.
threshold	amplitude threshold parameter. Only peaks above this threshold will be considered. See details.
plot	logical, if TRUE, a plot showing the peaks is returned.
title	logical, if TRUE add the number of peaks detected as a plot title.
xlab	label of the x-axis.
ylab	label of the y-axis.
labels	logical, if TRUE peak labels are plotted.
legend	logical, if TRUE a legend returning the different selection parameters (nmax, amp, freq, threshold, threshold) is added to the plot.
collab	labels color.
...	other <a href="#">plot</a> graphical parameters.

### Details

Here are some details regarding the different selection parameters:

- **nmax**: this parameter is to be used if you wish to get a specific number of peaks. The peaks selected are those with the highest slopes. It then does not work in conjunction with the other parameters.
- **freq**: this parameter allows to remove from the selection successive peaks with a small frequency difference. Imagine you have two successive peaks at 1200 Hz and 1210 Hz and at 0.5 and 0.25 in amplitude. If you set `freq` to 50 Hz, then only the first peak will be kept.
- **amp**: this parameter allows to remove from the selection peaks with low slopes. You can make the selection on both slopes or on a single one. Imagine you have an asymmetric peak with a 0.01 left slope and a 0.02 right slope. The peak will be discarded for the following settings: both values higher than 0.02 (e.g. `amp = c(0.03, 0.04)`), the first value higher than 0.01 (e.g. `amp = c(0.02, 0.001)`), the second value higher than 0.02 (e.g. `amp = c(0.001, 0.03)`). If you do not want apply the selection on one of the slope use 0. For instance, a selection on the left slope only will be achieved with: `amp = c(0.02, 0)`.
- **threshold**: this parameter can be used to do a rough selection on the spectrum. Peaks with an amplitude value (not a slope) lower than this threshold will be automatically discarded. This can be useful when you want to remove peaks of a low-amplitude background noise.

### Value

A two-column matrix, the first column corresponding to the frequency values (x-axis) and the second column corresponding to the amplitude values (y-axis) of the peaks.

### Note

You can also use `fpeaks` with other kind of spectrum, for instance a cepstral spectrum. See examples.

**Author(s)**

Jerome Sueur and Amandine Gasc

**See Also**

[localpeaks](#), [meanspec](#), [spec](#)

**Examples**

```

data(tico)
spec <- meanspec(tico, f=22050, plot=FALSE)
specdB <- meanspec(tico, f=22050, dB="max0", plot=FALSE)
# all peaks
fpeaks(spec)
# 10 highest peaks
fpeaks(spec, nmax=10)
# highest peak (ie dominant frequency)
fpeaks(spec, nmax=1)
# peaks that are separated by more than 500 Hz
fpeaks(spec, freq=500)
# peaks with a left slope higher than 0.1
fpeaks(spec, amp=c(0.1,0))
# peaks with a right slope higher than 0.1
fpeaks(spec, amp=c(0,0.1))
# peaks with left and right slopes higher than 0.1
fpeaks(spec, amp=c(0.1,0.1))
# peaks above a 0.5 threshold
fpeaks(spec, threshold=0.5)
# peaks of a dB spectrum with peaks showing slopes higher than 3 dB
fpeaks(specdB, amp=c(3,3))
# comparing different parameter settings
meanspec(tico, f=22050)
col <- c("#ff000090", "#0000ff75", "#00ff00")
cex <- c(2,1.25,1.5)
pch <- c(19,17,4)
title(main="Peak detection \n (spectrum with values between 0 and 1)")
res1 <- fpeaks(spec, plot = FALSE)
res2 <- fpeaks(spec, amp=c(0.02,0.02), plot =FALSE)
res3 <- fpeaks(spec, amp=c(0.02,0.02), freq=200, plot = FALSE)
points(res1, pch=pch[1], col=col[1], cex=cex[1])
points(res2, pch=pch[2], col=col[2], cex=cex[2])
points(res3, pch=pch[3], col=col[3], cex=cex[3])
legend("topright", legend=c("all peaks", "amp", "amp & freq"), pch=pch,
pt.cex=cex, col=col, bty="n")
# example with a cepstral spectrum
data(sheep)
res <- ceps(sheep, f=8000, at=0.4, w1=1024, plot=FALSE)
fpeaks(res, nmax=4, xlab="Quefreny (s)")

```

---

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,
     from = NULL, to = NULL,
     plot = TRUE, xlab = "Time (s)", ylab = "Frequency (kHz)",
     ylim = c(0, f/2000), pb = FALSE, ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	if <code>at</code> 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 %).
from	start position where to compute the fundamental frequency (in s).
to	end position to compute the fundamental frequency (in s).
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.
pb	if TRUE returns a text progress bar in the console.
...	other <a href="#">plot</a> 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 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 quefrequency: 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% overlap between successive sliding windows, time zoom and
# 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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	window length for spectral entropy analysis (even number of points). See <a href="#">sh</a> .
envt	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See <a href="#">env</a> .
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 <a href="#">env</a> .
ksmooth	kernel smooth via <a href="#">kernel</a> . See <a href="#">env</a> .

**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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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](mailto: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](#)

**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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> graphical parameters.

**Details**

The instantaneous phase is the argument of the analytic signal obtained through 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 ( <i>x</i> -axis) and the second column corresponding to wrapped instantaneous phase in radians ( <i>y</i> -axis).
f	a two-column matrix, the first column corresponding to time in seconds ( <i>x</i> -axis) and the second column corresponding to instantaneous frequency in kHz ( <i>y</i> -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)
```

---

itakura.dist

*Itakuro-Saito distance*

---

**Description**

Compare two distributions (e.g. two frequency spectra) by computing the Itakuro-Saito distance

**Usage**

```
itakura.dist(spec1, spec2)
```

**Arguments**

spec1 any distribution, especially a spectrum 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 any distribution, especially a spectrum obtained with [spec](#) or [meanspec](#) (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).

**Details**

The Itakura-Saito (I-S) distance is a non-symmetric measure of the difference between two probability distributions. It is here adapted for frequency spectra. The distance is asymmetric, ie computing the I-S distance between spec1 and spec2 is not the same as computing it between spec2 and spec1. A symmetry can be obtained by calculating the mean between the two directions. The distance is obtained following:

$$D_{I-S}(spec1||spec2) = \sum \frac{spec1}{spec2} - \log\left(\frac{spec1}{spec2}\right) - 1$$

**Value**

The function returns a list of three items:

D1	The I-S distance of 'spec2' with respect to 'spec1' ( <i>i.e.</i> D(spec1    spec2))
D2	The I-S distance of 'spec1' with respect to 'spec2' ( <i>i.e.</i> D(spec2    spec1))
D	The symmetric distance ( <i>i.e.</i> D = 0.5*(D1+D2))

**Author(s)**

Jerome Sueur

**See Also**

[kl.dist](#), [ks.dist](#), [logspec.dist](#), [simspec](#), [diffspec](#)

**Examples**

```
# Comparison of two spectra
data(tico)
tico1 <- spec(tico, at=0.65, plot=FALSE)
tico2 <- spec(tico, at=1.1, plot=FALSE)
itakura.dist(tico1, tico2)
```

---

kl.dist

*Kullback-Leibler distance*

---

**Description**

Compare two distributions (e.g. two frequency spectra) by computing the Kullback-Leibler distance

**Usage**

```
kl.dist(spec1, spec2, base = 2)
```

**Arguments**

spec1	any distribution, especially a spectrum obtained with <a href="#">spec</a> or <a href="#">meanspec</a> (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
spec2	any distribution, especially a spectrum obtained with <a href="#">spec</a> or <a href="#">meanspec</a> (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
base	the logarithm base used to compute the distance. See <a href="#">log</a> .

**Details**

The Kullback-Leibler distance or relative entropy is a non-symmetric measure of the difference between two probability distributions. It is here adapted for frequency spectra. The distance is asymmetric, ie computing the K-L distance between spec1 and spec2 is not the same as computing it between spec2 and spec1. A symmetry can be obtained by calculating the mean between the two directions.

The distance is obtained following:

$$D_{K-L}(spec1||spec2) = \sum spec1 \times \log\left(\frac{spec1}{spec2}\right)$$

**Value**

The function returns a list of three items:

D1	The K-L distance of 'spec2' with respect to 'spec1' ( <i>i.e.</i> D(spec1    spec2))
D2	The K-L distance of 'spec1' with respect to 'spec2' ( <i>i.e.</i> D(spec2    spec1))
D	The symmetric K-L distance ( <i>i.e.</i> D = 0.5*(D1+D2))

**Note**

The base of the logarithm can be changed using the argument base. When sets to base 2, the information is measured in units of bits. When sets to base  $e$ , the information is measured in nats.

**Author(s)**

Jerome Sueur

**References**

Kullback, S., Leibler, R.A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22: 79-86

**See Also**

[ks.dist](#), [logspec.dist](#), [simspec](#), [diffspec](#)

**Examples**

```
# Comparison of two spectra
data(tico)
tico1 <- spec(tico, at=0.65, plot=FALSE)
tico2 <- spec(tico, at=1.1, plot=FALSE)
kl.dist(tico1, tico2) # log2 (binary logarithm)
kl.dist(tico1, tico2, base=exp(1)) # ln (natural logarithm)
```

ks.dist

*Kolmogorov-Smirnov distance***Description**

Compare two distributions (e.g. two frequency spectra) by computing the Kolmogorov-Smirnov distance

**Usage**

```
ks.dist(spec1, spec2, f = NULL, plot = FALSE, type = "l",
        col = c("blue", "red"),
        lty = c(2, 4), flab = "Frequency (kHz)",
        alab = "Cumulated amplitude", flim = c(0, f/2000),
        alim = c(0, 1), title = TRUE, legend = TRUE, ...)
```

**Arguments**

spec1	any distribution, especially a spectrum obtained with <a href="#">spec</a> or <a href="#">meanspec</a> (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
spec2	any distribution, especially a spectrum obtained with <a href="#">spec</a> or <a href="#">meanspec</a> (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 <a href="#">spec</a> or <a href="#">meanspec</a> .
plot	logical, if TRUE plots both cumulated spectra and their maximal distance ( <i>i.e.</i> the K-S distance).
type	if plot is TRUE, type of plot that should be drawn. See <a href="#">plot</a> for details (by default "l" for lines).
col	a vector of length 2 for the colour of spec1 and spec2.
lty	a vector of length 2 for the line type of spec1 and spec2 if type="l".
flab	title of the frequency axis.
alab	title of the amplitude axis.
flim	the range of frequency values.

alim	range of amplitude axis.
title	logical, if TRUE, adds a title with D and F values.
legend	logical, if TRUE adds a legend to the plot.
...	other <a href="#">plot</a> graphical parameters.

### Details

The Kolmogorov distance is the maximal distance between the cumulated spectra. The function returns this distance and the corresponding frequency. This is an adaptation of the statistic computed by the non-parametric Kolmogorov-Smirnov test (see [ks.test](#)).

### Value

The function returns a list of two items

D	the Kolmogorov-Smirnov distance
F	the frequency (in KHz) where the Kolmogorov-Smirnov distance was found

### Note

There is no p-value associated to the K-S distance.

### Author(s)

Jerome Sueur

### See Also

[kl.dist](#), [simspec](#), [diffspec](#)

### Examples

```
# Comparison of two spectra and plot of the cumulated spectra with the K-S distance
data(tico)
tico1 <- spec(tico, at=0.65, plot=FALSE)
tico2 <- spec(tico, at=1.1, plot=FALSE)
ks.dist(tico1, tico2, plot=TRUE)
```

---

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", fftw = FALSE, output = "matrix")
```

## Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">ftwindow</a> (by default "hanning").
fftw	if TRUE calls the function FFT of the library fftw. See Notes of the <a href="#">spectro</a> .
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".

## 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. The class of the returned object is set with the argument `output`.

## Author(s)

Jerome Sueur <[sueur@mnhn.fr](mailto:sueur@mnhn.fr)> and Thierry Aubin <[thierry.aubin@u-psud.fr](mailto: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	<i>Play a sound wave</i>
--------	--------------------------

---

**Description**

Play a sound wave

**Usage**

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

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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-column matrix, data.frame, time-series and Sample objects.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr> but the original [play](#) function is by Uwe Ligges (package **tuneR**).

**See Also**

[play](#)

**Examples**

```
## NOT RUN
# data(tico)
# listen(tico, f=22050)
# listen(tico, f=22050, from=0.5, to=1.5)
# listen(noise(d=1, f=8000, Wave=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)
```

---

 localpeaks

*Local maximum frequency peak detection*


---

### Description

This functions searches for local peaks of a frequency spectrum

### Usage

```
localpeaks(spec, f = NULL, bands = 10, plot = TRUE,
  xlab = "Frequency (kHz)", ylab = "Amplitude", labels = TRUE, ...)
```

### Arguments

spec	a data set resulting of a spectral analysis obtained with <a href="#">spec</a> or <a href="#">meanspec</a> . Can be in dB.
f	sampling frequency of spec (in Hz). Not requested if the first column of spec contains the frequency axis.
bands	a numeric vector. If vector of length 1, then sets the number of bands dividing in equal parts the spectrum. If of length > 1, then takes the values as kHz limits of the bands dividing the spectrum. These bands can be of different size. See details and examples.
plot	logical, if TRUE, a plot showing the peaks is returned.
xlab	label of the x-axis.
ylab	label of the y-axis.
labels	logical, if TRUE peak labels are plotted.
...	other <a href="#">plot</a> graphical parameters.

### Details

The function proceed as follows

- divides the spectrum in bands. The limits of the bands are set with the argument bands. There are two options:
  - you set a number of bands with equal size by giving a single value to bands. For instance, setting bands to a value of 10 will slice the spectrum in 10 equal parts and return 10 local peaks.
  - you set the limits of the bands. This is achieve by giving a numeric vector to bands. The limits can follow a regular or irregular series. For instance attributing the vector c(0,2,4,8) will generate the following bands [0,2[, [2,4[, [4,8] kHz. Be aware that the last value should not exceed half the sampling frequency used to obtain the spectrum spec.
- uses the function [fpeaks](#) with the argument nmax set to 1.

**Value**

A two-column matrix, the first column corresponding to the frequency values (x-axis) and the second column corresponding to the amplitude values (y-axis) of the peaks.

**Author(s)**

Jerome Sueur

**See Also**

[fpeaks](#), [meanspec](#), [spec](#)

**Examples**

```
data(sheep)
spec <- meanspec(sheep, f=8000)
# a specific number of bands with all the same size
localpeaks(spec, bands=5)
# bands directly specified with a regular sequence
localpeaks(spec, bands=seq(0,8/2,by=0.5))
# bands directly specified with an irregular sequence
localpeaks(spec, bands=c(0,0.5,1,1.5,3,4))
# Amaj octave bands, note that there is no peak detection
# in the higher part of the spectrum as sequence stops at 3520 Hz
localpeaks(spec, bands=octaves(440, below=3, above=3)/1000)
```

---

logspec.dist

*Log-spectral distance*

---

**Description**

Compare two distributions (e.g. two frequency spectra) by computing the log-spectral distance

**Usage**

```
logspec.dist(spec1, spec2)
```

**Arguments**

spec1	any distribution, especially a spectrum obtained with <a href="#">spec</a> or <a href="#">meanspec</a> (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).
spec2	any distribution, especially a spectrum obtained with <a href="#">spec</a> or <a href="#">meanspec</a> (not in dB). This can be either a two-column matrix (col1 = frequency, col2 = amplitude) or a vector (amplitude).

**Details**

The distance is computed according to:

$$D_{LS}(spec1||spec2) = D_{LS}(spec2||spec1) = \sqrt{\sum 10 \times \log_{10}\left(\frac{spec1}{spec2}\right)^2}$$

**Value**

A numeric vector of length 1 returning the D distance.

**Note**

The distance is symmetric.

**Author(s)**

Jerome Sueur

**See Also**

[ks.dist](#), [kl.dist](#), [itakura.dist](#), [simspec](#), [diffspec](#)

**Examples**

```
# Comparison of two spectra
data(tico)
tico1 <- spec(tico, at=0.65, plot=FALSE)
tico2 <- spec(tico, at=1.1, plot=FALSE)
logspec.dist(tico1, tico2)
```

---

meandB

*mean of dB values*

---

**Description**

This functions calculates the mean of dB values

**Usage**

```
meandB(x)
```

**Arguments**

x                    a numeric vector or a matrix.

**Details**

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

**Value**

A numeric vector of length 1 is returned.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr> and Zev Ross

**References**

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

**See Also**

moredB, sddB, [convSPL](#), [dBweight](#)

**Examples**

```
meandB(c(89, 90, 95))
```

---

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, w1 = 512, wn = "hanning", ovlp = 0, fftw = FALSE, PSD = FALSE,
PMF = FALSE, dB = NULL, dBref = NULL, from = NULL, to = NULL,
identify = FALSE, col = "black", cex = 1, plot = 1, flab = "Frequency (kHz)",
alab = "Amplitude", flim = NULL, alim = NULL, type = "l", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
w1	length of the window for the analysis (even number of points, by default = 512).
wn	window name, see <a href="#">ftwindow</a> (by default "hanning").
ovlp	overlap between two successive analysis windows (in %).
fftw	if TRUE calls the function FFT of the library fftw. See Notes of spectro.
PSD	if TRUE return Power Spectra Density, <i>i. e.</i> the square of the spectra.
PMF	if TRUE return Probability Mass Function, <i>i. e.</i> the probability distribution of frequency.

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 $2 \times 10^{-5}$ 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).
identify	to identify frequency and amplitude values on the plot with the help of a cursor.
col	colour of the spectrum.
cex	pitch size.
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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> graphical parameters.

### Details

See examples of [spec](#). This function is based on [fft](#).

### 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

### Warning

The argument peaks is no more available (version > 1.5.6). See the function [fpeaks](#) for peak(s) detection.

### Note

The argument fftw can be used to try to speed up process time. When set to TRUE, the Fourier transform is computed through the function FFT of the package fftw. This package is a wrapper around the fastest Fourier transform of the free C subroutine library FFTW (<http://www.fftw.org/>). FFT should be then installed on your OS.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**See Also**

[spec](#), [fpeaks](#), [localpeaks](#), [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)
# log scale on frequency axis
meanspec(orni, f=22050, log="x")
```

---

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:

$$m = 1127.01048 \times \log\left(1 + \left(\frac{f}{700}\right)\right)$$

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

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

$$f = 700 \times \left(e^{\frac{m}{1127.01048}} - 1\right)$$

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:

$$S_{dB} = 20 \times \log_{10}\left(\frac{s}{s_{ref}}\right)$$

with  $s$  the measured sensitivity in mv/Pa and  $s_{ref}$  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 matrix.

**Details**

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

**Value**

A numeric vector of length 1 is returned.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**References**

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

**See Also**

`meandB`, `sddb`, [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	<i>Replace time wave data by 0 values</i>
-------	---

---

### 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,  
output = "matrix", ...)
```

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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. The class of the returned object is set with the argument output.

### 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)
```

---

noisew	<i>Generate noise</i>
--------	-----------------------

---

## Description

This function generates noise.

## Usage

```
noisew(f, d, type="unif", listen = FALSE, output = "matrix")
```

## 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.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".

## Details

Uniform noise is generated using [runif](#) and gaussian noise is based on [rnorm](#)

## Value

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

## Author(s)

Jerome Sueur <sueur@mnhn.fr>

## See Also

[synth](#), [pulse](#)

## Examples

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

---

notefreq	<i>Frequency of a musical note</i>
----------	------------------------------------

---

**Description**

This function computes the frequency of a musical note (Equal temperament)

**Usage**

```
notefreq(note, ref = 440, octave = 3)
```

**Arguments**

note	a numerical or a character vector. See Note.
ref	a numerical vector of length 1 for the reference frequency.
octave	a numerical vector of length for the octave number.

**Details**

The frequency is computed according to:

$$f = ref \times 2^{octave-3 + \frac{note-10}{12}}$$

with:

*ref* = reference frequency,

*octave* = octave number, and

*note* = rank of the note along the scale.

**Value**

The frequency in Hz is returned.

**Note**

The note can be given in two ways. The first solution is to give the rank of the note along the scale (e.g. rank 10 for A) or to give its names in characters with the following notation: C, D, E, F, G, A, B.

**Author(s)**

Jerome Sueur

**See Also**

[octaves](#)

**Examples**

```
# Some notes frequency (use apply-like functions when dealing with character strings)
sapply(c("C", "A", "Gb"), notefreq)

# C major scale plot
n <- 1:12
freq <- notefreq(n)
names <- c("C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B")
plot(n, freq, pch=19, cex=1.5,
      xlab = "Note name",
      ylab = "Frequency (Hz)",
      xaxt="n", las=1, main="Third octave")
axis(side=1, at=n, labels=names)
abline(h=freq, col="lightgrey")

# C major scale sound
f <- 2000 # sampling rate
s <- NULL
for (i in 1:length(freq))
{
  tmp <- synth(d=0.5, f=f, cf=freq[i])
  s <- pastew(s, tmp, at="start", f)
}
spectro(s, f, ovlp=75)
```

---

octaves

*Octave values*


---

**Description**

This functions returns the frequency values of the octaves below and above a specific frequency

**Usage**

```
octaves(x, below = 3, above = 3)
```

**Arguments**

x	a numeric vector, frequency of the note in Hz or Khz.
below	the number of octaves below x.
above	the number of octaves above x.

**Value**

A numeric vector with the octave series in frequency (Hz or kHz depending on x unit).

**Author(s)**

Jerome Sueur

**See Also**[notefreq](#)**Examples**

```

names <- c("C","D","E","F","G","A","B")
values <- c(261.63, 293.66, 329.64, 349.23, 392, 440, 493.88)
res <- sapply(values, FUN=octaves)/1000
op <- par(las=1,mfrow=c(2,1))
par(mar=c(0,4,1,1))
matplot(x=1:7, y=res, t="o", pch=names, xlab="",
        ylab="Frequency (kHz) [linear scale]", col=rainbow(7), xaxt="n")
par(mar=c(4.5,4,0,1))
matplot(x=1:7, y=res, t="o", pch=names, xlab="Octave",
        ylab="Frequency (kHz) [log scale]", col=rainbow(7), ylog=TRUE, log="y")
par(op)

```

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 Wave object.

**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", cexaxis = 1, coly0 = "lightgrey",
tcl = 0.5, title = FALSE, xaxt="s", yaxt="n", type="l", bty = "l")
```

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 columns (by default TRUE).
identify	returns the time and amplitude coordinates 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.
cexaxis	magnification for axes annotation.
coly0	colour of the y=0 line.
tcl	length of tick marks.
title	TRUE to add a title with information on wave duration and f, FALSE to live it blank, or a character string to add any desired title.
xaxt	equivalent to xaxt of <code>par</code> (by default = "s").
yaxt	equivalent to yaxt of <code>par</code> (by default = "n").
type	type of plot, by default "l". Use "n" for no plot.
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 two-column matrix with the time and amplitude coordinates 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**

[dynoscillo](#), [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
require(rpanel)
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",
cexaxis = 1, coly01 = "grey47", coly02 = "black", title = FALSE,
bty = "l")

```

## Arguments

wave1	a first R object.
wave2	a second R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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
cexaxis	mangification for axes annotation.
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](#), [dynoscillo](#)

### 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", join = FALSE, choose = FALSE, plot = FALSE,
marks = TRUE, output = "matrix", ...)
```

### Arguments

wave1	a first R object.
wave2	a second R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
at	wave2 position in seconds where wave1 will be pasted into. Can be also specified as "start", "middle" or "end".
join	if TRUE the two waves will be pasted and jointed by removing the last point of wave2. See examples.
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).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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. The class of the returned object is set with the argument output.

**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)
# setting the argument 'join' to TRUE might be useful
# to smooth pasting when some phase problem occur
# generate two sine waves
a <- synth(cf=50, f=400, d=0.1)
b <- synth(cf=100, f=400, d=0.1)
# paste it with 'join' turned to FALSE
# there is a click at the junction between the two waves
pastew(a, b, f=400, plot=TRUE)
# that can be removed by setting 'join' to TRUE
pastew(a, b, f=400, join=TRUE, 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 Wave object.

**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 Wave object.

**Details**

Duration = 3.309 s. Sampling frequency = 11025 hz.

**Source**

Recording by Jerome Sueur.

**Examples**

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

---

`phaseplot`*Phase-phase 2D or 3D plot of a time wave*

---

**Description**

This function returns a 2D or 3D representation of a time wave according to its first, second and possibly third derivatives.

**Usage**

```
phaseplot(wave, f, dim = 3, plot = TRUE, type = "l", xlab = "1st derivative", ylab = "2nd derivative", zlab = "3rd derivative")
```

**Arguments**

<code>wave</code>	an R object.
<code>f</code>	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
<code>dim</code>	a vector of length 1, the number of dimensions of the plot. Can be either 2 or 3.
<code>plot</code>	logical, if TRUE plots phase-phase plot (by default TRUE).
<code>type</code>	type of plot that should be drawn. See <a href="#">plot</a> for details (by default "l" for lines).
<code>xlab</code>	title of the x axis.
<code>ylab</code>	title of the y axis.
<code>zlab</code>	title of the z axis.
<code>...</code>	other <a href="#">plot</a> or <a href="#">plot3d</a> graphical parameters

**Value**

If `plot` is FALSE then a 2 or 3 column matrix is returned. The position of the column is related to the order of the derivative (*i. e.* first column = first derivative).

**Note**

Phase-phase plot can be used to test non-linearity.

**Author(s)**

Jerome Sueur

**References**

For use of such plots see: Rice AN, Land BR, Bass AH (2011) - Nonlinear acoustic complexity in a fish 'two-voice' system. *Proceedings of the Royal Society B*, in press.

**Examples**

```
require(rgl)
data(tico)
phaseplot(tico)
```

---

pulse	<i>Generate rectangle pulse</i>
-------	---------------------------------

---

**Description**

This function generates a rectangle pulse.

**Usage**

```
pulse(dbefore, dpulse, dafter, f, plot = FALSE, output = "matrix", ...)
```

**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).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">plot</a> parameters.

**Value**

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**See Also**

[synth](#), [noisew](#)

**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 <a href="#">spec</a> , or <a href="#">meanspec</a> (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 <a href="#">spec</a> or <a href="#">meanspec</a> .
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 amplitude axis.
type	if plot is TRUE, type of plot that should be drawn. See <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> graphical parameters.

### Details

A high Q value indicates a highly resonant system.

### Value

A list is returned with the following four items:

Q	a numeric vector of length 1 returning the Q factor (no units)
dfreq	a numeric vector of length 1 the dominant frequency (kHz)
fmin	a numeric vector of length 1 returning the minimum frequency of the -dB level bandwidth (kHz)

`fmax` a numeric vector of length 1 returning the minimum frequency of the -dB level bandwidth (kHz)

`bwd` a numeric vector of length 1 returning the bandwidth, i. e. `fmax-fmin` (kHz)

**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, join = FALSE, plot = FALSE, output= "matrix", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
times	a numeric of length 1 describing the number of times the wave has to be repeated.
join	if TRUE the last point of wave will be removed for smoothing junction between repetitions. See examples.
plot	logical, if TRUE plots the repeated time wave.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> graphical parameters.

**Value**

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

**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)
# use 'join' for smooth pasting
par(mfrow=c(2,1))
a <- synth(cf=50, f=400, d=0.1)
repw(a, f=400, plot=TRUE)
title(main="join is FALSE")
points(x=0.1, y=0, cex=2, col=2)
repw(a, f=400, join=TRUE, plot=TRUE)
title(main="join is TRUE")
points(x=0.1, y=0, cex=2, col=2)
```

---

`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, output="matrix")
```

**Arguments**

<code>wave</code>	an R object.
<code>f</code>	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
<code>g</code>	new sampling frequency of wave (in Hz).
<code>output</code>	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".

**Value**

If `plot` is FALSE, a new wave is returned. The class of the returned object is set with the argument `output`.

**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, output = "matrix", ...)
```

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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. The class of the returned object is set with the argument output.

### 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, output = "matrix", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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 wave is returned. The class of the returned object is set with the argument output.

**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](#).

## 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, output = "matrix", ...)
```

## Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">smooth.spline</a> 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. The class of the returned object is set with the argument output.

## Note

Low frequency noise might not be removed out properly.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**See Also**

[afilter](#), [noisew](#)

**Examples**

```
# synthesis of a 440 Hz sound with background noise
n <- noisew(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, output = "matrix", ...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
plot	logical, if TRUE returns an oscillographic plot of the wave after removing the offset (by default FALSE).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> graphical parameters.

**Value**

If plot is FALSE, a new wave is returned. The class of the returned object is set with the argument output.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**See Also**[oscillo](#)**Examples**

```
data(tico)
# artificially 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 computed 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)
```

---

roughness	<i>Roughness or total curvature</i>
-----------	-------------------------------------

---

**Description**

This function computes the roughness or total curvature of a curve, i.e. of a time wave or of a spectrum

**Usage**

```
roughness(x, std = FALSE)
```

**Arguments**

`x` a vector  
`std` a logical, if set to TRUE then `x` is standardized by its maximum.

**Details**

Roughness or total curvature is the integrated squared second derivative :

$$roughness = \int [D^2x(t)]^2 dt$$

.

**Value**

A vector of length 1.

**Note**

The value has not unit.

**Author(s)**

Jerome Sueur

**References**

Ramsay JO, Silverman BW (2005) *Functional data analysis*. Springer, Berlin.

**See Also**

[rugo](#), [rms](#), [sh](#), [th](#), [H](#).

**Examples**

```
data(tico)
spec <- meanspec(tico, plot=FALSE)[,2]
roughness(spec)
```

---

rugo

*Rugosity of a time wave*

---

**Description**

This function computes the rugosity of a time wave or time series

**Usage**

```
rugo(x, ...)
```

**Arguments**

`x` a vector  
`...` other [mean](#) parameters.

**Details**

The formula has been slightly modified from Mezquida & Martinez (2009: 826) to fit with the classical definition of the root-mean-square (see [rms](#)).

The rugosity is then computed as following:

$$rugo = \sqrt{\frac{\sum_{i=1}^{n-1} (x_{i+1} - x_i)^2}{n}}$$

for a vector `x` of length `n`.

**Value**

A vector of length 1.

**Note**

The rugosity 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

## References

Mezquida DA, Martinez JL (2009) - Platform for beehives monitoring based on sound Analysis. A perpetual warehouse for swarm's daily acitivity. *Spanish Journal of Agricultural Research* **7**, 824-828.

## See Also

[roughness](#), [rms](#), [sh](#), [th](#), [H](#).

## Examples

```
data(tico) ; tico <-tico@left
# rugosity of the original recording
rugo(tico/max(tico))
# add artificially some background noise
noise <- noisew(d=length(tico)/22050, f=22050)
ticon1 <- tico/max(tico) + 0.5*noise
# new rugosity (higher)
rugo(ticon1/max(ticon1))
```

---

savewav

*Save a .wav file*

---

## Description

Save sound data as .wav file

## Usage

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

## Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
filename	name of the new file. (by default the name of wave).

## Details

This function uses three functions from the package **tuneR**: [Wave](#), [normalize](#) and [writeWave](#).

## Note

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

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**See Also**

[export](#).

**Examples**

```
require(tuneR)
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")
```

---

seedata

*A quick look at quantitative data*

---

**Description**

See quantitative data at a glance

**Usage**

```
seedata(data, na.rm = FALSE, col = "grey")
```

**Arguments**

data	a numeric vector describing quantitative data.
na.rm	logical, if TRUE removes NA.
col	main color.

**Details**

The red curves depict the corresponding Normal law (same mean and sd as data).

**Value**

A multi-plot graphic is returned.

**Author(s)**

Caroline Simonis <csimonis@mnhn.fr> and Jerome Sueur <sueur@mnhn.fr>.

**Examples**

```
seedata(rnorm(1000))
```

---

 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: 1.6.3  
 Date: 2012-04-10  
 License: GPL version 2 or newer  
 Contributors : Amandine Gasc, Jonathan Lees, Sandrine Pavoine, Luis J. Villanueva-Rivera, Zev Ross, Carl G. Witthorn  
 Acknowledgments: Michel Baylac, Charlotte Cure, Arnold Fertin, Kurt Hornik, Emiliano A. Laca, Uwe Ligges, Emmanuel  
 Webpage: <http://rug.mnhn.fr/seewave>  
 Discussion group : <http://groups.google.com/group/seewave>  
 Reference: Sueur J, Aubin T, Simonis C - Seewave: a free modular tool for sound analysis and synthesis. Bioacoust

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>  
 Thierry Aubin  
 Caroline Simonis

Maintainer: Jerome Sueur <sueur@mnhn.fr>

---

 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, output = "matrix", ...)
```

**Arguments**

wave1	a first R object.
wave2	a second R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
envt	the type of envelope to be used for wave2: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See <a href="#">env</a> .
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 <a href="#">env</a> .
ksmooth	kernel smooth via <a href="#">kernel</a> to apply to the amplitude envelope of wave2. See <a href="#">env</a> .
plot	if TRUE returns the oscillogram of the new time wave (by default FALSE).
listen	if TRUE the new sound is played back.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">oscillo</a> 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. The class of the returned object is set with the argument output.

**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' amplitude 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)
spec<-spec(a,f=8000,at=0.5,plot=FALSE)
sfm(spec)
# [1] 0
b<-noisew(d=1,f=8000)
specb<-spec(b,f=8000,at=0.5,plot=FALSE)
sfm(specb)
# [1] 0.8233202

```

sh

*Shannon and Renyi spectral entropy***Description**

This function computes the Shannon or Renyi entropy of a frequency spectrum

**Usage**

```
sh(spec, alpha = NULL)
```

**Arguments**

spec	a data set resulting of a spectral analysis obtained with <a href="#">spec</a> or <a href="#">meanspec</a> (not in dB).
alpha	by default NULL to compute Shannon entropy, otherwise superior to 0 but different to 1 to compute Renyi entropy.

**Details**

Shannon 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.

Renyi spectral entropy of order alpha is calculated according to:

$$R = \frac{1}{1-\alpha} \times \log_2\left(\sum_{i=1}^N y_i^\alpha\right)$$

with

$$\alpha \geq 0$$

and

$$\alpha \neq 1$$

### Value

A numeric vector of length 1 is returned. The value has no unit.

### Note

The Shannon spectral entropy of a noisy signal will tend towards 1 whereas the Shannon spectral entropy of a pure tone signal will tend towards 0. See Han *et al.* for details regarding the Renyi entropy.

### Author(s)

Jerome Sueur

### References

Han, NC, Muniandy SV, Dayou J (2011) Acoustic classification of Australian anurans based on hybrid spectral-entropy approach. *Applied Acoustics*.

Nunes RR, Almeida de MP, Sleigh JW (2004) - Spectral entropy: a new method for anesthetic adequacy. *Revista Brasileira de Anestesiologia*, **54**, 413-422.

Renyi A (1961) - On measures of information and entropy. Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability 1960. pp. 547-561.

### See Also

[csh](#), [th](#), [H](#), [sfm](#)

### Examples

```
a<-synth(f=8000,d=1,cf=2000,plot=FALSE)
spec<-spec(a,f=8000,at=0.5,plot=FALSE)
#####
# Shannon spectral entropy
#####
sh(spec)
# [1] 0.2336412
b<-noisew(d=1,f=8000)
specb<-spec(b,f=8000,at=0.5,plot=FALSE)
```

```
sh(specb)
# close to 1
#####
# Renyi spectral entropy
#####
sh(speca, alpha=2)
sh(speca, alpha=3)
```

---

sheep

*Sheep bleat*

---

### Description

Recording of a sheep bleat.

### Usage

```
data(sheep)
```

### Format

A Wave object.

### 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 <a href="#">spec</a> or <a href="#">meanspec</a> (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 <a href="#">spec</a> or <a href="#">meanspec</a> (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 <a href="#">spec</a> or <a href="#">meanspec</a> .
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 <a href="#">plot</a> 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 <a href="#">plot</a> graphical parameters.

**Details**

Spectra similarity is assessed according to:

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

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](#), [kl.dist](#), [ks.dist](#), [logspec.dist](#), [itakura.dist](#)

**Examples**

```
a<-noisew(f=8000,d=1)
b<-synth(f=8000,d=1,cf=2000)
c<-synth(f=8000,d=1,cf=1000)
d<-noisew(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)
```

smoothw

*A function to tentavily smooth a time wave*

---

**Description**

This function tries to smooth with a sum sliding window a time wave, and then to remove residual noise.

**Usage**

```
smoothw(wave, f, wl, padding=TRUE, output = "matrix")
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	window length in number of points (samples).
padding	a logical, if TRUE add 0 values at the start and end of the file to match wave length (duration).
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".

**Details**

A window slides along the signal and sums up the sample amplitude values. Zero values are added at the end of the wave to keep wave length (duration).

**Value**

A new wave is returned. The class of the returned object is set with the argument output. If padding is TRUE, the new wave starts and ends up with 0 values to match the size of wave.

**Warning**

This function should be used with care as this kind of filter may change the frequency content of the sound. See the examples section for an illustration.

**Author(s)**

Jerome Sueur

**See Also**

[fir](#), [filter](#)

**Examples**

```
# An example to show that smoothw() may change
# the frequency content of your sound
data(orni)
orni2 <- smoothw(orni, wl=2, out="Wave")
orni10 <- smoothw(orni, wl=10, out="Wave")
orni50 <- smoothw(orni, wl=50, out="Wave")
orni100 <- smoothw(orni, wl=100, out="Wave")
meanspec(orni)
lines(meanspec(orni2, plot=FALSE), col=2)
lines(meanspec(orni10, plot=FALSE), col=3)
lines(meanspec(orni50, plot=FALSE), col=4)
lines(meanspec(orni100, plot=FALSE), col=5)
legend("topright", col=1:5, lty=1, legend=c("original", "wl=2", "wl=10", "wl=50", "wl=100"))
```

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", fftw = FALSE, PSD = FALSE,
     PMF = FALSE, dB = NULL, dBref = NULL,
     at = NULL, from = NULL, to = NULL,
     identify = FALSE, col = "black", cex = 1,
     plot = 1, flab = "Frequency (kHz)",
     alab = "Amplitude", flim = NULL,
     alim = NULL, type="l",...)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">ftwindow</a> (by default "hanning").
fftw	if TRUE calls the function FFT of the library fftw. See Notes of the <code>spectro.x</code>
PSD	if TRUE return Power Spectra Density, <i>i. e.</i> the square of the spectra.
PMF	if TRUE return Probability Mass Function, <i>i. e.</i> the probability distribution of frequency.

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).
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.
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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> graphical parameters.

### Details

If at, from or to are FALSE then spec computes the spectrum of the whole signal.

### 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

### Warning

The argument peaks is no more available (version > 1.5.6). See the function [fpeaks](#) for peak(s) detection.

### Note

This function is based on [fft](#).

**Author(s)**

Jerome Sueur

**See Also**[meanspec](#), [fpeaks](#), [localpeaks](#), [dynspec](#), [corspec](#), [fft](#).**Examples**

```

data(tico)
# spectrum of the whole signal, in absolute or dB amplitude,
# horizontally 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)
# 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)
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 <a href="#">spec</a> or <a href="#">meanspec</a> (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 specify 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 <a href="#">plot</a> for details (by default "l" for lines).
...	other <a href="#">plot</a> 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 <a href="#">mean</a> )
sd	standard deviation of the mean (see <a href="#">sd</a> )
sem	standard error of the mean
median	median frequency (see <a href="#">median</a> )
mode	mode frequency, <i>i.e.</i> the dominant frequency
Q25	first quartile (see <a href="#">quantile</a> )
Q75	third quartile (see <a href="#">quantile</a> )
IQR	interquartile range (see <a href="#">IQR</a> )
cent	centroid, see note
skewness	skewness, a measure of asymmetry, see note
kurtosis	kurtosis, a measure of peakedness, see note
sfm	spectral flatness measure (see <a href="#">sfm</a> )
sh	spectral entropy (see <a href="#">sh</a> )
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 symmetric,  
 $S > 0$  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, fftw = FALSE, 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",
        cexlab = 1, cexaxis = 1,
        tlab = "Time (s)",
        flab = "Frequency (kHz)",
        alab = "Amplitude",
        scalelab = "Amplitude\n(dB)",
        main=NULL,
        scalefontlab = 1, scalecexlab =0.75,
        axisX = TRUE, axisY = TRUE, tlim = NULL, trel = TRUE,
        flim = NULL, flimd = NULL,
        widths = c(6,1), heights = c(3,1),
        oma = rep(0,4),
        listen=FALSE,
        ...)
```

### Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
wl	window length for the analysis (even number of points). (by default = 512)
zp	zero-padding (even number of points), see <i>Details</i> .
ovlp	overlap between two successive windows (in %).
fftw	if TRUE calls the function FFT of the library <code>fftw</code> . See <i>Notes</i> .
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 <code>ftwindow</code> (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 <code>Details</code> .
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.
cexlab	size of the labels.
cexaxis	size of the axes.
tlab	label of the time axis.
flab	label of the frequency axis.
alab	label of the amplitude axis.
scalelab	amplitude scale label.
main	label of the main title.
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.
trrel	time X-axis with a relative scale when <code>tlim</code> is not null, <i>i.e.</i> relative to wave.
flim	modifications of the frequency Y-axis limits.
flimd	dynamic modifications of the frequency Y-axis limits. New <code>wl</code> and <code>ovlp</code> arguments are applied to increase time/frequency resolution.
widths	a vector of length 2 to control the relative widths of columns on the device when <code>scale</code> is TRUE.
heights	a vector of length 2 to control the relative heights of rows on the device when <code>osc</code> is TRUE.
oma	a vector of length 4 to control the size of outer margins when either <code>scale</code> or <code>osc</code> is TRUE.
listen	if TRUE the sound is played back (by default FALSE).
...	other <code>contour</code> and <code>oscillo</code> 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 **see-wave**: `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

This function returns a list of three items:

<code>time</code>	a numeric vector corresponding to the time axis.
<code>freq</code>	a numeric vector corresponding to the frequency axis.
<code>amp</code>	a numeric matrix corresponding to the amplitude values. Each column is a Fourier transform of length <code>wl</code> .

## Note

The argument `fftw` can be used to try to speed up process time. When set to `TRUE`, the Fourier transform is computed through the function `FFT` of the package `fftw`. This package is a wrapper around the fastest Fourier transform of the free C subroutine library `FFTW` (<http://www.fftw.org/>). `FFT` should be then installed on your OS.

## Note

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

## Author(s)

Jerome Sueur and Caroline Simonis.

## 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=22050, dBref=2*10e-5)
# 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, w1 = 512, wn = "hanning", zp = 0,
  ovlp = 0, fftw = FALSE, dB = "max0", dBref = NULL, plot = TRUE,
  magt = 10, magf = 10, maga = 2,
  palette = rev.terrain.colors)
```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
w1	length of the window for the analysis (even number of points).
wn	window name, see <code>ftwindow</code> (by default "hanning").
zp	zero-padding (even number of points), see Details.
ovlp	overlap between two successive windows (in %).
fftw	if TRUE calls the function FFT of the library <code>fftw</code> . See Notes of the spectro.
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 $2 \cdot 10^{-5}$ 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 `w1` value. Choosing a high `w1` value will increase the frequency resolution but reduce the temporal one, and *vice versa*. The frequency precision is obtained by calculating the ratio  $f/w1$ , and the temporal precision is obtained by calculating the reverse ratio  $w1/f$ . 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 **see-wave**: `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**

This function returns a list of three items:

time	a numeric vector corresponding to the time axis.
freq	a numeric vector corresponding to the frequency axis.
amp	a numeric matrix corresponding to the amplitude values. Each column is 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](#), [dysnpec](#), [wf](#), [fft](#).

**Examples**

```
require(rgl)
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 first R object.
y	a second R object
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 <a href="#">plot</a> 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 <a href="#">plot</a> 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:

$$I = H_x + H_y - H_{xy}$$

with  $H_x$  the entropy of x symbol series,  $H_y$  the entropy of y symbol series, and  $H_{xy}$  the joint entropy of x and y symbol series.

**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, output = "matrix",...)
```

**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 synthesized sound (by default FALSE).
listen	if TRUE the new sound is played back.
output	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
...	other <a href="#">spectro</a> graphical parameters.

**Details**

shape allows to modify the whole amplitude shape of the wave. There are four options to be given as a 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. The class of the returned object is set with the argument output.

**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

[noisew](#), [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](mailto: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<-noisew(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 Wave object.

### 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 = 5, envt="abs",
      power = 1, msmooth = NULL, ksmooth = NULL,
      ssmooth = NULL, plot = TRUE, plotthreshold = TRUE,
      col = "black", colval = "red",
      xlab = "Time (s)", ylab = "Amplitude", ...)
```

## Arguments

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
threshold	amplitude threshold for signal detection (in %).
envt	the type of envelope to be used: either "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. See <a href="#">env</a> .
power	a power factor applied to the amplitude envelope. Increasing power will reduce low amplitude modulations and increase high amplitude modulations. This can be used to reduce background noise (by default equals to 1, <i>i.e.</i> no change).
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 <a href="#">env</a> .
ksmooth	kernel smooth for the amplitude envelope via <a href="#">kernel</a> . See <a href="#">env</a> .
ssmooth	sum smooth for the amplitude envelope via <a href="#">kernel</a> . See <a href="#">env</a> .
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 <a href="#">plot</a> graphical parameters.

**Value**

A list containing six items:

s	duration of signal period(s) in seconds
p	duration of pause period(s) in seconds
r	ratio between the signal and silence periods(s)
.	
positions	a list containing four elements:
s.start	start position(s) of signal period(s)
s.end	end position(s) of signal period(s)

**Warning**

Setting to high values to msmooth or ssmooth might return unaccurate results. Double check you results if so.

**Author(s)**

Jerome Sueur <sueur@mnhn.fr>

**See Also**

[env](#), [cutw](#), [pastew](#).

**Examples**

```
data(tico)
timer(tico, f=22050, threshold=5, msmooth=c(50, 0))
# to compare with an oscillographic representation
data(orni)
op<-par(mfrow=c(2, 1))
timer(orni, f=22050, threshold=5, msmooth=c(40, 0), tck=0.05,
      bty="l", colval="blue")
title(main="A cicada song made of five echemes", col="blue")
oscillo(orni, f=22050, k=1, j=1)
par(op)
```

**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 Celsius).
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:

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

with  $t$  = temperature in degrees Celsius; range of validity: 0-95 degrees Celcius at atmospheric pressure.

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

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

with  $t$  = temperature in degrees Celsius;  $s$  = salinity in parts per thousand;  $d$  = depth in meters; range of validity: temperature 2 to 30 degrees Celcius, 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 20 degrees Celsius
wasp(f=2000)$l
# [1] 0.1717

# sound speed in sea at 0 and -500 m for a respective temperature of 22 degrees Celcius and 11 degrees Celsius
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 (deg. C)")
legend(x=15000,y=0.3,c("-20","-10","0","10","20","30","40"),
       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 default 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 default 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, w1 = 512, zp = 0, ovlp = 0, fftw= FALSE, dB = "max0",
  dBref = NULL, wn = "hanning", 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, lwd=NULL, ...)

```

**Arguments**

wave	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
w1	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 %).
fftw	if TRUE calls the function FFT of the library fftw. See Notes of the spectro.
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 $2 \times 10^{-5}$ for a 20 microPa reference.
wn	window name, see <a href="#">ftwindow</a> (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 <code>polygon</code> : 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 <code>polygon</code> : 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).
lwd	line width.
...	other graphical arguments to passed to <code>plot</code>

### 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 deletes the silence periods of a time wave.

**Usage**

```
zapsilw(wave, f, threshold = 5, plot = TRUE, output = "matrix", ...)
```

**Arguments**

<code>wave</code>	an R object.
<code>f</code>	sampling frequency of wave (in Hz). Does not need to be specified if embedded in wave.
<code>threshold</code>	amplitude threshold (in %) between silence and signal.
<code>plot</code>	logical, if TRUE plots the original and the new oscillograms (by default TRUE).
<code>output</code>	character string, the class of the object to return, either "matrix", "Wave", "Sample", "audioSample" or "ts".
<code>...</code>	other <a href="#">oscillo</a> graphical parameters.

**Value**

If `plot` is FALSE, a new wave is returned. The class of the returned object is set with the argument `output`.

**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	an R object.
f	sampling frequency of wave (in Hz). Does not need to be specified if embedded in 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 <a href="#">plot</a> 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, 49
- ftwindow, 63
- savewav, 110
- wav2flac, 141

## \*Topic **datagen**

- drawenv, 41
- echo, 46
- noisew, 86
- pulse, 98
- setenv, 112
- synth, 133

## \*Topic **datasets**

- orni, 89
- peewit, 95
- pellucens, 96
- sheep, 117
- tico, 137

## \*Topic **data**

- attenuation, 7

## \*Topic **distribution**

- itakura.dist, 69
- kl.dist, 70
- logspec.dist, 77

## \*Topic **distrubtion**

- ks.dist, 72

## \*Topic **dplot**

- ama, 6
- autoc, 8
- ccoh, 10
- ceps, 12
- cepstro, 14
- coh, 16
- corenv, 18
- corspec, 20
- covspectro, 22
- cutw, 28
- dBscale, 29
- deletew, 32

- dfreq, 33
- diffenv, 35
- diffspec, 36
- dynoscillo, 42
- dynspec, 43
- env, 47
- fadew, 50
- fbands, 51
- fma, 59
- fpeaks, 60
- fund, 64
- ifreq, 67
- localpeaks, 76
- meanspec, 79
- mutew, 85
- oscillo, 90
- oscilloST, 92
- pastew, 94
- phaseplot, 97
- Q, 99
- repw, 100
- resamp, 102
- revw, 103
- rmoffset, 106
- seedata, 111
- seewave, 112
- simspec, 118
- spec, 121
- specprop, 123
- spectro, 126
- spectro3D, 129
- timer, 138
- wf, 143
- zc, 146

## \*Topic **maths**

- notefreq, 87
- octaves, 88

## \*Topic **math**

- convSPL, 17

- dBweight, 30
- fdoppler, 53
- meandB, 78
- mel, 82
- micsens, 83
- moredB, 84
- wasp, 139
- \*Topic **ts**
  - afilter, 5
  - ama, 6
  - autoc, 8
  - ccoh, 10
  - ceps, 12
  - cepstro, 14
  - coh, 16
  - 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
  - dynoscillo, 42
  - dynspec, 43
  - echo, 46
  - env, 47
  - fadew, 50
  - fbands, 51
  - ffilter, 54
  - field, 56
  - fir, 57
  - fma, 59
  - fpeaks, 60
  - ftwindow, 63
  - fund, 64
  - H, 65
  - hilbert, 66
  - ifreq, 67
  - itakura.dist, 69
  - kl.dist, 70
  - ks.dist, 72
  - lfs, 73
  - listen, 75
  - localpeaks, 76
  - logspec.dist, 77
  - meanspec, 79
  - mutew, 85
  - noisew, 86
  - oscillo, 90
  - oscilloST, 92
  - pastew, 94
  - phaseplot, 97
  - pulse, 98
  - Q, 99
  - repw, 100
  - resamp, 102
  - revw, 103
  - rmam, 104
  - rmnoise, 105
  - rmoffset, 106
  - rms, 107
  - roughness, 108
  - rugo, 109
  - seewave, 112
  - setenv, 112
  - sfm, 114
  - sh, 115
  - simspec, 118
  - smoothw, 120
  - spec, 121
  - specprop, 123
  - spectro, 126
  - spectro3D, 129
  - symba, 131
  - synth, 133
  - th, 135
  - timer, 138
  - wf, 143
  - zapsilw, 145
  - zc, 146
- acf, 9
- addsilw, 4, 29, 33, 50, 85, 95, 101, 103
- afilter, 5, 55, 58, 106, 145
- ama, 6, 60
- approx, 146
- attenuation, 7, 18
- autoc, 5, 8, 14, 16, 65

- axis, 29, 144
- barplot, 52
- ccoh, 10, 17
- ceps, 9, 12, 15, 16, 64, 65
- cepstro, 14, 14, 65
- coh, 12, 16
- contour, 11, 15, 127, 128
- convolve, 46, 58
- convSPL, 8, 17, 31, 79, 83, 84
- cor, 19–21, 23, 24
- cor.test, 19–21
- corenv, 18, 24, 36
- corspec, 20, 20, 21, 24, 38, 81, 100, 119, 123
- covspectro, 20, 21, 22, 63
- crest, 24
- csh, 5, 25, 66, 115, 116, 136
- cutspec, 27
- cutw, 5, 28, 33, 50, 85, 91, 95, 101, 139
- dBscale, 29, 128
- dBweight, 18, 30, 79, 84
- deletew, 5, 29, 32, 50, 85, 95, 101, 103
- dfreq, 5, 33, 63
- diffenv, 35, 38, 39, 119
- diffspec, 36, 36, 39, 70, 71, 73, 78, 81, 119
- diffwave, 36, 38
- discrets, 40, 132, 133
- drawenv, 41, 113
- dynoscillo, 42, 91, 93
- dynspec, 43, 43, 81, 123, 128, 131, 144
- echo, 46, 135
- env, 7, 19, 20, 35, 36, 39, 42, 45, 47, 65, 113, 136, 138, 139
- export, 49, 111
- fadew, 5, 29, 33, 50, 85, 95, 101, 103
- fbands, 51
- fdoppler, 53
- ffilter, 6, 54, 58, 74
- fft, 13, 34, 45, 55, 58, 74, 80, 81, 100, 122, 123, 128, 131
- field, 56
- filled.contour, 11, 30, 128
- filter, 120
- fir, 55, 57, 120
- fma, 7, 59
- fpeaks, 13, 60, 76, 77, 80, 81, 122, 123
- ftwindow, 23, 26, 33, 44, 55, 58, 63, 74, 79, 121, 126, 130, 143
- fund, 14, 16, 64
- H, 65, 109, 110, 116, 136
- hilbert, 48, 60, 66, 68, 69, 105
- ifreq, 60, 67, 67, 147
- IQR, 124
- itakura.dist, 38, 69, 78, 119
- kernel, 19, 35, 39, 45, 48, 65, 113, 138
- kl.dist, 38, 70, 70, 73, 78, 119
- ks.dist, 38, 70, 71, 72, 78, 119
- ks.test, 73
- lfs, 55, 58, 73
- listen, 75
- localpeaks, 62, 76, 81, 123
- locator, 11, 42, 128
- log, 71
- logspec.dist, 38, 70, 71, 77, 119
- mean, 107, 109, 124
- meandB, 78
- meanspec, 6, 7, 20, 21, 27, 34, 37, 38, 51, 52, 55, 58, 60, 62, 63, 69, 71, 72, 76, 77, 79, 99, 100, 114, 115, 118, 119, 123, 124
- median, 124
- mel, 82
- micsens, 83
- Mod, 48
- moredB, 8, 18, 31, 84
- mutew, 5, 29, 33, 50, 85, 95, 101, 103
- na.omit, 146
- noisew, 86, 98, 106, 135
- normalize, 110
- notefreq, 87, 89
- octaves, 87, 88
- orni, 89
- oscillo, 4–6, 11, 25, 28, 29, 32, 33, 43, 45, 46, 48, 50, 67, 85, 90, 93–95, 101, 103, 104, 106, 107, 113, 127, 128, 145
- oscilloST, 43, 91, 92

- par, 91
- pastew, 5, 29, 33, 50, 85, 91, 94, 101, 103, 139
- peewit, 95
- pellucens, 96
- phaseplot, 97
- play, 75
- plot, 6, 8, 9, 13, 16, 19, 21, 23, 26, 34, 35, 37, 43, 44, 51, 61, 64, 68, 72, 73, 76, 80, 97–99, 118, 122, 124, 132, 138, 144, 146
- plot3d, 97
- polygon, 144
- pulse, 86, 98, 135
  
- Q, 99
- quantile, 124
  
- repw, 95, 100
- resamp, 102
- revw, 5, 29, 33, 50, 85, 95, 101, 103
- rmam, 104
- rmnoise, 105
- rmoffset, 106
- rms, 24, 25, 107, 109, 110
- rnorm, 86
- roughness, 108, 110
- rugo, 109, 109
- runif, 86
  
- Sample, 40
- savewav, 110, 142
- sd, 124
- seedata, 111
- seewave, 112
- seewave-package (seewave), 112
- setenv, 42, 112
- sfm, 27, 114, 116, 124
- sh, 26, 27, 65, 66, 109, 110, 114, 115, 115, 124, 136
- sheep, 117
- simspec, 38, 70, 71, 73, 78, 81, 118
- smooth.spline, 105
- smoothw, 120
- spec, 20, 21, 27, 34, 37, 38, 45, 51, 52, 55, 58–60, 62, 63, 69, 71, 72, 76, 77, 80, 81, 99, 100, 114, 115, 118, 119, 121, 124
- spec.pgram, 11, 12, 17
- specprop, 123
- spectro, 11, 12, 15, 17, 24, 29, 30, 34, 45, 55, 63, 74, 126, 131, 134, 144
- spectro3D, 45, 63, 128, 129, 144
- symba, 41, 131
- synth, 42, 47, 86, 98, 113, 133
  
- th, 26, 66, 109, 110, 116, 135
- tico, 137
- timer, 5, 91, 138
  
- wasp, 54, 139
- wav2flac, 141
- Wave, 40, 110
- wf, 45, 128, 131, 143
- write.table, 49
- writeWave, 110
  
- zapsilw, 5, 29, 33, 50, 85, 95, 101, 145
- zc, 5, 69, 146