Package ‘soundmeteR’

June 19, 2022

Type Package

Title R package for sound meter alike analysis

Version 0.1.0

Author Cássio Rachid Simões, Shaka Nóbrega Marinho Furtado, Ingrid Maria Denóbile Torres, Marcelo Felgueiras Napoli and Carlos Barros de Araújo

Maintainer Cássio Rachid Simões <cassiorachid@gmail.com>

Description An R package designed for sound analysis through spectral analysis in octaves and one-thirds octaves, making possible calibration to obtain sound pressure level (SPL) and Sound Intensity Level (SIL).

URL https://github.com/cassiorachid/soundmeteR

BugReports https://github.com/cassiorachid/soundmeteR/issues

License GPL (>=3)

Depends tuneR, seewave, dplyr, progress

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.1.2

R topics documented:

`dBtoLinear` *Convert deciBels scales to linear*

Description

Function to convert dB scales. The conversion can be made either from dB to µPa (`dBtoLinear`) or from µPa to dB (`LineartodB`).

Usage

```
dBtoLinear(x, factor="IL", ref=1)
```

Arguments

`x`	Numerical. A numeric vector or a numeric matrix with dB values.
`factor`	Character. Specify in what factor the function should use to convert your data. `SPL` (Sound Pressure Level) for amplitude like data (factor `20`) or `IL` (Intensity Level) for power like (fator `10`). (By default `IL`)
`ref`	Numerical. Reference value for conversion. For Sound in water the ref is 1 µPa and on air 20 µPa. (By default 1)

Details

For details about the factor choice, we recommend the reading of this web page.

Value

The same object of the input with the converted values.

See Also

`rms.dB`, `LineartodB`, `convSPL`

Examples

```
dBtoLinear(c(80,60,65,62))
LineartodB(dBtoLinear(c(80,60,65,62)))
```

`freq.bands` *Interval Limits*

Description

Function to compute intervals from patterns defined by the user. This function was adapted from `octaves` functions from `seewave` package.

Usage

```
freq.bands(x, interval = 2, below = 3, above = 3)
```

Arguments

`x`	Numerical. The frequency values to start the bands' calculation.
`interval`	Numeric. The interval pattern to be applied. See details for more.
`below`	Numerical. Number of intervals below x.
`above`	Numerical. Number of intervals above x.

Details

The interval specified is applied by x/interval (for bellow) of x*interval (for upper). Some examples of intervals that can be applied are:

- Octaves = 2
- Third of octaves = 2^(1/3)
- Perfect fifth (music theory) = 3/2
- Major third (music theory) = 5/4
- Minor third (music theory) = 6/5
- This link shows other values that can be used.

Value

A numeric vector with the frequency limits of each interval.

See Also

`octaves`

Examples

```
freq.bands(1000, interval=2, below = 1, above = 1) #octaves
freq.bands(1000, interval=2^(1/3), below = 3, above = 3) #Third of octaves

#https://academics.hamilton.edu/music/spellman/class_notes/music_theory.htm
freq.bands(440, interval=3/2, below = 0, above = 1) #Perfect fifth (music theory) of A 440
freq.bands(440, interval=5/4, below = 0, above = 1) #Major third (music theory) of A 440
freq.bands(440, interval=6/5, below = 0, above = 1) #Minor third (music theory) of A 440
```

`LineartodB` *Convert linear scales to deciBels*

Description

Function to convert dB scales. The conversion can be made either from dB to µPa (`dBtoLinear`) or from µPa to dB (`LineartodB`).

Usage

```
LineartodB(x, factor="IL", ref=1)
```

Arguments

x	Numerical. A numeric vector or a numeric matrix with dB values linear values (µPa).
factor	Character. Specify in what factor the function should use to convert your data. SPL (Sound Pressure Level) for amplitude like data (factor 20) or IL (Intensity Level) for power like (fator 10). (By default IL)
ref	Numerical. Reference value for conversion. For Sound in water the ref is 1 µPa and on air 20 µPa. (By default 1)

Details

For details about the factor choice, we recommend the reading of this web page.

Value

The same object of the input with the converted values.

See Also

rms.dB, dBtoLinear, convSPL

Examples

```
dBtoLinear(c(80,60,65,62))
LineartodB(dBtoLinear(c(80,60,65,62)))
```

---

pwrspec *Power Spectrum of a sound file*

Description

This functions computes a power spectrum from a sound file.

Usage

```
pwrspec(
  file,
  channel = "left",
  from = 0,
  to = Inf,
  bandpass = c(0, Inf),
  res.scale = "microPa",
  ref = 1
)
```

Arguments

`file`	A wave file path in your computer or a class Wave object already loaded into R environment.
`channel`	Argument passed to mono function from tuneR to extract the desired channel.
`from`	Numeric. The start time in seconds of the sample you want to analyze. Could also be relative to the end of the file (in negative values), see examples.
`to`	Numeric. The end time in seconds of the sample you want to analyze. Could also be relative to the end of the file (in negative values), see examples.
`bandpass`	A vector with length two with lower and upper limits of the band pass interval in Hz.
`res.scale`	Character. Specify the kind of scale the power spectrum amplitude should be adjusted. `microPa` for linear values in µPa, and `dB` fo decibells values in dB-SPL. (By default "microPa")
`ref`	Numerical. Reference value for dB conversion. For Sound in water the ref is 1 µPa and on air 20 µPa. (By default 1)

Value

This function returns a data.frame with Frequency (Hz) and Intensity (microPa) of the file.

References

Power spectrum adapted from: Carcagno, S. 2013. Basic Sound Processing with R [Blog post]. Retrieved from http://samcarcagno.altervista.org/blog/basic-sound-processing-r/

Miyara, F. 2017. Software-Based Acoustical Measurements. Springer. 429 pp. DOI: 10.1007/978-3-319-55871-4

Examples

```
data(tham)
pwrspec(tham)

#with from, to and bandpass
pwrspec(tham, from=3.8, to=7.1, bandpass=c(900,2000))

#from and to relative to duration (negatives values from the end of the soundfiles)
pwrspec(tham, from=-3.49, to=-0.9, bandpass=c(900,2000))
```

`rms.dB` *Root Mean Square with dB values*

Description

Function to compute the root mean square (RMS) of values in decibels (dB).

Usage

```
rms.dB(x, level="SPL", ref=1, ...)
```

Arguments

`x`	Numerical. A numeric vector or a numeric matrix with dB values.
`level`	Character. Specify in what scale your data is. `SPL` for Sound Pressute Level or `IL` for Intensity Level. (By default `SPL`)
`na.rm`	Logical. Argument passed to `mean`. Should NA be removed? (By default FALSE)
`ref`	Numerical. Reference value for conversion. For Sound in water the ref is 1microPa and on air 20 microPa. (By default 1)

Details

This function converts your dB data to linear values (through `dBtoLinear` function), compute the Root Mean Square (rms), and converts the result back to dB (through `LineartodB` function).

This function was adapted from `meandB` and `rms` functions from `seewave` package. See their help for more details.

Value

A numeric value that represents the root mean square of x.

See Also

`meandB`, `rms`

Examples

```
rms.dB(c(80,60,65,62))
```

---

song.level *RMS from a sample of a sound file*

Description

RMS from a sample of a sound file

Usage

```
song.level(
  files = "wd",
  channel = "left",
  from = 0,
  to = Inf,
  freq.interval = c(0, Inf),
  fdom.int = c(0, Inf),
  wl = 512,
  ovlp = 50,
  CalibPosition = NULL,
  CalibValue = NULL,
  freq.weight = "none",
  ref = 20
)
```

Arguments

`files`	The audiofile to be analyzed. Can be "wd" to get all ".wav" files on the work directory, a file name (or a character containing a list of filenames) that exist in the work directory (only ".wav" files accepted), or an Wave object (or a list containing more than one Wave object). (By default: "wd")
`channel`	Argument passed to mono function from tuneR to extract the desired channel.
`from`	Numeric. The start time in seconds of the sample you want to analyze. Could also be relative to the end of the file (in negative values), see examples.
`to`	Numeric. The end time in seconds of the sample you want to analyze. Could also be relative to the end of the file (in negative values), see examples.
`freq.interval`	Frequency interval to compute the RMS. Can be a vector with length two with lower and upper interval of frequencies (in Hz), or a pattern to calculate a interval (as octaves). For the last, see freq.bands function for details.
`fdom.int`	Vector with length two. Vector with length two with lower and upper interval of frequencies (in Hz) to find the dominant frequency. This frequency will be used as the center the interval only if a pettern is specified.
`wl`	Explain.
`ovlp`	Explain.
`CalibPosition`	Missing
`CalibValue`	Canbe a value to apply of the ref value from a calib signal (specified by CalibPosition).
`freq.weight`	Character. Argument passed to dBweight to indicate the weighting curve to use on the anlysis. 'A', 'B', 'C', 'D', 'ITU', and 'none' are supported. See dBweight for details. (By default: "none")
`ref`	Numerical. The reference value for dB conversion. For sound in water, the common is 1 microPa, and for sound on air 20 microPa. (By default 20)

Examples

```
song.level(tham, freq.interval=c(22, 20000))

song.level(tham, freq.interval=c(22, 20000), CalibValue = 130.24)

song.level(tham, freq.interval=c(22, 20000), CalibValue = 130.24, freq.weight="A")

song.level(tham, freq.interval=c(22, 20000), CalibValue = 130.24, freq.weight="B")

song.level(tham, freq.interval=c(22, 20000), CalibValue = 130.24, freq.weight="C")

song.level(tham, from = 3.883035, to=7.044417, freq.interval=c(22.09429, 22627.38), CalibValue = 130.24, freq.

#Perfect fifth (music theory)
song.level(tham, fdom.int = c(800, 2000), from = 3.8, to=7, freq.interval=3/2, CalibValue = 130.24, freq.weight

#Perfect fifth (music theory)
song.level(tham, fdom.int = c(800, 2000), from = 3.8, to=7, freq.interval=3/2, CalibValue = 130.24, freq.weight
```

soundmeter — *Function that makes sound meter alike measurements*

Description

Function that makes sound meter alike measurements

Usage

```
soundmeter(
  files = "wd",
  from = 0,
  to = Inf,
  CalibPosition = NULL,
  CalibValue = NULL,
  ref = 20,
  fw = "none",
  bands = "octaves",
  tw = "fast",
  progressbar = T,
  channel = "left",
  saveresults = F,
  outname = NULL
)
```

Arguments

`CalibPosition`	anda de mãos dadas com calib value. Pode ser negativo, positivo ou data.frame com essas combinações
`CalibValue`	Anda de mãos dadas com calib position. Quando tem o position, ele é considerado o valor de referência, quando não tem o position, ele é considerado o valor de calibração.
`fw`	Character. Argument passed to `dBweight` to indicate the frequency weighting curve to use on the anlysis. 'A', 'B', 'C', 'D', 'ITU', and 'none' are supported. See `dBweight` for details. (By default: "none")
`tw`	Time weighting
`progressbar`	Logical. Activate or deactivate a progress bar with elapsed time and the last concluded file number. (By default: `TRUE`)
`channel`	Only "left" or "right" acepted. By default "left"
`saveresults`	Logical. Set `TRUE` if you want to save a txt file with the results of the function execution. (By default: `FALSE`)
`outname`	Character. If `saveresults` is `TRUE`, you can specify a name to appear on the txt file name after the default name. (By default: `NULL`)

Details

If your reference signal is in a separate file, we recommend you to get the `CalibValue` with Tweighting function. It's examples provide more details.

Examples

```
data("tham")
soundmeter(tham, CalibValue = 130.24, tw="slow") #slow time window with calib value
soundmeter(tham, CalibValue = 130.24, tw="fast") #fast

soundmeter(tham, CalibValue = 130.24, tw="fast", fw="A") #fast with frequency weight

soundmeter(tham, CalibValue = NULL, tw="fast", ref=1) #fast time window in dBFS
```

---

sumdB *Sum with dB values*

Description

Sum with dB values

Usage

```
sumdB(x, level = "IL", na.rm = FALSE, ...)
```

Arguments

x	Numerical. A numeric vector or matrix with dB values.
level	Character. Specify in what scale your data is. `SPL` for Sound Pressute Level or `IL` for Intensity Level. (By default `SPL`)
na.rm	Logical. Argument passed to `sum`. Should NA be removed? (By default FALSE)

Details

This function computes the sum of dB values.

This function converts your dB data to linear values (through `dBtoLinear` function), compute the sum, and converts the result back to dB (through `LineartodB` function).

Value

A numeric value that represents the sum of x.

See Also

`moredB`

Examples

```
sumdB(c(80,60,65,62))
sumdB(c(30,30), level="IL")
sumdB(c(30,30), level="SPL")
```

`tham` — Thamnophilus stictocephalus *song in a landscape*

Description

Thamnophilus stictocephalus song in a landscape

Usage

```
data(tham)
```

Format

An object of class "Wave"; see readWave.

Details

A landscape recording presenting *Thamnophilus stictocephalus* song between 3.883035 and 7.044417 seconds. The "calib.value" for this record is 130.24.

Source

Recording by Ingrid Maria Denóbile Torres

`timbre` — *Level per octaves or one-thirds octaves*

Usage

```
timbre(files="wd", weighting="none", bands="thirds", ref=20, saveresults=F,
       outname=NULL, Leq.calib=NULL, Calib.value=NULL, time.mess=T, stat.mess=T)
```

Arguments

`files`	The audiofile to be analyzed. Can be "wd" to get all ".wav" files on the work directory, a file name (or a character containing a list of filenames) that exist in the work directory (only ".wav" files accepted), or an Wave object (or a list containing more than one Wave object). (By default: "wd")
`channel`	Argument passe to mono function from tuneR to extract the desired channel.
`from`	Numeric. The start time in seconds of the sample you want to analyze. Could also be relative to the end of the file (in negative values), see examples.
`to`	Numeric. The end time in seconds of the sample you want to analyze. Could also be relative to the end of the file (in negative values), see examples.
`weighting`	Character. Argument passed to `dBweight` to indicate the weighting curve to use on the anlysis. 'A', 'B', 'C', 'D', 'ITU', and 'none' are supported. See `dBweight` for details. (By default: "none")
`bands`	Character. Choose the type of frequency band of the output. "octaves" to octaves bands intervals or "thirds" to one-third octaves bands intervals. (by deafault: "thirds")

`ref`	Numerical. The reference value for dB conversion. For sound in water, the common is 1 microPa, and for sound on air 20 microPa. (By default 20)
`saveresults`	Logical. Set `TRUE` if you want to save a txt file with the results of the function execution. (By default: `FALSE`)
`outname`	Character. If `saveresults` is `TRUE`, you can specify a name to appear on the txt file name after the default name. (By default: `NULL`)
`Leq.calib`	Numerical. The sound pressure level (in dB SPL) that the signal in the audio file must have (by default: `NULL`). Can not be set if `Calib.value` is also set.
`Calib.value`	Numerical. The calibration value returned from the analysis of a reference signal using `Leq.calib` (by default: `NULL`). Can not be set if `Leq.calib` is also set.
`time.mess`	Logical. Activate or deactivate message of time to complete the function execution. (By default: `TRUE`)
`stat.mess`	Logical. Activate or deactivate status message of the function execution. (By default: `TRUE`)

Details

Caution: You need to use an audiofile with entire values of seconds of duration to avoid bugs. Example: 35s, 60s, 19s. By default, the function will trunc your audiofile to the next entire value of seconds.

These function works only with mono audiofiles.

The audio files need to have at least 44100Hz of sampling rate.

If you intend to work with decibels at full scale (dBFS), we recommend setting `ref=1`. With this, your results will be relative to 0 dBFS.

References

Power spectrum adapted from: Carcagno, S. 2013. Basic Sound Processing with R [Blog post]. Retrieved from http://samcarcagno.altervista.org/blog/basic-sound-processing-r/

Miyara, F. 2017. Software-Based Acoustical Measurements. Springer. 429 pp. DOI: 10.1007/978-3-319-55871-4

Examples

```
data(tham)
timbre(tham)
timbre(tham, Calib.value=130.24)
timbre(tham, Calib.value=130.24, weighting="A")
timbre(tham, Calib.value=130.24, weighting="A", bands="octaves")

timbre(tham, Leq.calib=48.17, weighting="A")

timbre(tham, from=-3.49, to=-0.9, ref=1) #dB at Full Scale
timbre(tham, from=-3.49, to=-0.9, Calib.value=130.24) #song Sound Pressure Level
timbre(tham, from=0, to=3.8, Calib.value=130.24) #background Sound Pressure Level
```

timbreCal *Timbre analysis for audiofiles with reference signal*

Description

This function passes the parameters to timbre to automatize the calibration and return spectral analysis with dB SPL results.

Usage

```
timbreCal(files="wd", SignalDur=NULL, RefValue=NULL, ref=20, weighting="none",
       bands="thirds", saveresults=F, outname=NULL, time.mess=T, stat.mess=T)
```

Arguments

files	The audiofile to be analyzed. Can be "wd" to get all ".wav" files on the work directory, a file name (or a character containing a list of filenames) that exist in the work directory (only ".wav" files accepted), or an Wave object (or a list containing more than one Wave object). (By default: "wd")
channel	Argument passed to mono function from tuneR to extract the desired channel.
SignalDur	Numerical. Specify the reference signal duration (in seconds) on the beggining of the audiofile. (By default: NULL)
RefValue	Numerical. Specify the reference signal sound pressure level (in deciBells SPL) on the beggining of the audiofile. (By default: NULL)
ref	Numerical. The reference value for dB conversion. For sound in water, the common is 1 microPa, and for sound on air 20 microPa. (By default 20)
weighting	Character. Indicate the weighting curve to use on the anlysis. A, B, C and none are supported. (By default: "none")
bands	Character. Choose the type of frequency band of the output. "octaves" to octaves bands intervals or "thirds" to one-third octaves bands intervals. (by deafault: "thirds")
saveresults	Logical. Set TRUE if you want to save a txt file with the results of the function execution. (By default: FALSE)
outname	Character. If saveresults is TRUE, you can specify a name to appear on the txt file name after the default name. (By defaulf: NULL)
time.mess	Logical. Activate or deactivate message of time to complete the function execution. (By default: TRUE)
stat.mess	Logical. Activate or deactivate status message of the function execution. (By default: TRUE)

Details

To use this function, the audio file must begin with 2 seconds of silence, followed by a reference signal with known SPL, followed by another 2 seconds of silence, and the following sound to analyze.

The duration of the reference signal must be specified (in seconds) on the SignalDur argument and his value (in dB SPL) on the refValue argument.

See Also

timbre

Tweighting *Time Weighting of a Audiofile*

Description

Escrever descrição

Usage

```
Tweighting(file, window = "fast", Leq.calib = NULL, ...)
```

Arguments

`file`	Wave object
`window`	Character. Wich time window should be used. 'fast' or 'slow' are accepted. (by default: "fast")
`Leq.calib`	Numeric. The sound pressure level (in dB SPL) that the signal in the audio file must have (by default: NULL). This parameter is passed to `timbre` function.
`...`	Further arguments passed to `timbre`.

Details

This function split your audiofile in smaller files defined as `fast` (0.125s) and `slow` (1s) and analyze each one with `timbre` function.

Value

A numeric vector

See Also

timbre, soundmeter

Examples

```
#creating an example sound file
som=sine(1000, duration = 44500)

#default options without calibration (results in dBFS)
Tweighting(som)

#Simulation of a calib signal with a Leq of 94dB in the field ####
#fast
Tweighting(som, window = "fast", bands="octaves", Leq.calib=94)
#slow
Tweighting(som, window = "slow", bands="octaves", Leq.calib=94)

#Using the result of the simulation above to calibrate the sound and output
#fast
```

```
Tweighting(som, window = "fast", bands="octaves", Calib.value=309.67)
#slow
Tweighting(som, window = "slow", bands="octaves", Calib.value=309.67)

#With tham data
data(tham)
Tweighting(tham, window = "fast", bands="octaves", Calib.value=130.24) #fast
Tweighting(tham, window = "slow", bands="octaves", Calib.value=130.24) #slow
```

Index