Package ‘soundecology’

October 14, 2022

Title Soundscape Ecology

Version 1.3.3

Date 2018-03-04

Author Luis J. Villanueva-Rivera and Bryan C. Pijanowski

Maintainer Luis J. Villanueva-Rivera <ljvillanueva@coquipr.com>

Description Functions to calculate indices for soundscape ecology and other ecology research that uses audio recordings.

Depends R(>= 2.14.0)

Suggests knitr

Imports parallel, pracma, oce, ineq, vegan, tuneR, seewave

License GPL-3

URL http://ljvillanueva.github.io/soundecology/

BugReports http://github.com/ljvillanueva/soundecology/issues

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-03-05 04:10:15 UTC

R topics documented:

acoustic_complexity *Acoustic Complexity Index*

Description

Acoustic Complexity Index (ACI) from Pieretti, *et al.* 2011. The ACI is based on the "observation that many biotic sounds, such as bird songs, are characterized by an intrinsic variability of intensities, while some types of human generated noise (such as car passing or airplane transit) present very constant intensity values" (Pieretti, *et al.* 2011).

The index was tested to the ACItot calculated using SoundscapeMeter v 1.0.14.05.2012, courtesy of A. Farina.

The results given are accumulative. Very long samples will return very large values for ACI. I recommend to divide by number of minutes to get a range of values easier to compare.

Usage

```
acoustic_complexity(soundfile, min_freq = NA, max_freq = NA, j = 5, fft_w = 512)
```

Arguments

- `soundfile` — an object of class `Wave` loaded with the function readWave of the `tuneR` package.
- `min_freq` — miminum frequency to use when calculating the value, in Hertz. The default is `NA`.
- `max_freq` — maximum frequency to use when calculating the value, in Hertz. The default is the maximum for the file.
- `j` — the cluster size, in seconds.
- `fft_w` — FFT window to use.

Value

Returns a list with three objects per channel

- `AciTotAll_left` — the ACI total for the left channel
- `AciTotAll_right` — the ACI total for the right channel
- `AciTotAll_left_bymin` — the ACI total for the left channel divided by the number of minutes
- `AciTotAll_right_bymin` — the ACI total for the right channel divided by the number of minutes
- `aci_fl_left_vals` — values of ACI(fl) for the left channel
- `aci_fl_right_vals` — values of ACI(fl) for the right channel

`aci_left_matrix`
: Matrix of all values before calculating ACI(fl) for the left channel

`aci_right_matrix`
: Matrix of all values before calculating ACI(fl) for the right channel

References

Pieretti, N., A. Farina, and D. Morri. 2011. A new methodology to infer the singing activity of an avian community: The Acoustic Complexity Index (ACI). Ecological Indicators 11: 868-873. doi: 10.1016/j.ecolind.2010.11.005

Examples

```
data(tropicalsound)
ACI <- acoustic_complexity(tropicalsound)
print(ACI$AciTotAll_left)

summary(ACI)
```

`acoustic_diversity` *Acoustic Diversity Index*

Description

Acoustic Diversity Index from Villanueva-Rivera *et al.* 2011. The ADI is calculated by dividing the spectrogram into bins (default 10) and taking the proportion of the signals in each bin above a threshold (default -50 dBFS). The ADI is the result of the Shannon index applied to these bins.

Usage

```
acoustic_diversity(soundfile, max_freq = 10000, db_threshold = -50,
freq_step = 1000, shannon = TRUE)
```

Arguments

`soundfile`	an object of class `Wave` loaded with the function readWave of the `tuneR` package.
`max_freq`	maximum frequency to use when calculating the value, in Hertz.
`db_threshold`	threshold to use in the calculation.
`freq_step`	size of frequency bands.
`shannon`	TRUE to use the Shannon's diversity index to calculate the ADI (default).

Value

Returns a list with five objects per channel

- `adi_left` ADI value for the left channel
- `adi_right` ADI value for the right channel
- `left_band_values` vector of proportion values for each band for the left channel
- `right_band_values` vector of proportion values for each band for the right channel
- `left_bandrange_values` vector of frequency values for each band for the left channel
- `right_bandrange_values` vector of frequency values for each band for the right channel

Note

The code to calculate the ADI has changed due to an error we detected in the original scripts in which the value was calculated using a different equation. In a test of ~38k files, both ways to calculate were highly correlated. This version of the function uses the Shannon's Diversity Index. To obtain a result using the old calculation, set the argument `shannon` to `FALSE`. Please check the vignette "Changes in the Acoustic Diversity Index", included in the package, for more details.

For audio files with one channel, the results are showed as the left channel, the right channel returns `NA`.

References

Villanueva-Rivera, L. J., B. C. Pijanowski, J. Doucette, and B. Pekin. 2011. A primer of acoustic analysis for landscape ecologists. Landscape Ecology 26: 1233-1246. doi: 10.1007/s10980-011-9636-9.

Examples

```
data(tropicalsound)
result <- acoustic_diversity(tropicalsound)

print(result$adi_left)

summary(result)
```

acoustic_evenness *Acoustic Evenness Index*

Description

Acoustic Evenness Index from Villanueva-Rivera *et al.* 2011 (band evenness using the Gini index). The AEI is calculated by dividing the spectrogram into bins (default 10) and taking the proportion of the signals in each bin above a threshold (default -50 dBFS). The AEI is the result of the Gini index applied to these bins.

Usage

```
acoustic_evenness(soundfile, max_freq = 10000, db_threshold = -50, freq_step = 1000)
```

Arguments

`soundfile`	an object of class `Wave` loaded with the function readWave of the `tuneR` package.
`max_freq`	maximum frequency to use when calculating the value, in Hertz.
`db_threshold`	threshold to use in the calculation.
`freq_step`	size of frequency bands.

Value

Returns a list with five objects per channel

`aei_left`	AEI for the left channel
`aei_right`	AEI for the right channel

Note

For audio files with one channel, the results are showed as the left channel, the right channel returns `NA`.

References

Villanueva-Rivera, L. J., B. C. Pijanowski, J. Doucette, and B. Pekin. 2011. A primer of acoustic analysis for landscape ecologists. Landscape Ecology 26: 1233-1246. doi: 10.1007/s10980-011-9636-9.

Examples

```
data(tropicalsound)
result <- acoustic_evenness(tropicalsound)
print(result$aei_left)

summary(result)
```

bioacoustic_index *Bioacoustic Index*

Description

Bioacoustic Index from Boelman, *et al.* 2007. Inspired on Matlab code courtesy of NT Boelman. Several parts where changed, in particular log math, so this won't be directly comparable to the original code in the paper.

The Bioacoustic Index is calculated as the "area under each curve included all frequency bands associated with the dB value that was greater than the minimum dB value for each curve. The area values are thus a function of both the sound level and the number of frequency bands used by the avifauna" (Boelman, *et al.* 2007).

Usage

```
bioacoustic_index(soundfile, min_freq = 2000, max_freq = 8000, fft_w = 512)
```

Arguments

`soundfile`	an object of class `Wave` loaded with the function readWave of the `tuneR` package.
`min_freq`	minimum frequency to use when calculating the value, in Hertz.
`max_freq`	maximum frequency to use when calculating the value, in Hertz.
`fft_w`	FFT window size.

Value

Returns a list with one object per channel

`left_area`	area under the curve for the left channel
`right_area`	area under the curve for the right channel

References

Boelman NT, Asner GP, Hart PJ, Martin RE. 2007. Multi-trophic invasion resistance in Hawaii: bioacoustics, field surveys, and airborne remote sensing. Ecological Applications 17: 2137-2144.

Examples

```
data(tropicalsound)
bioindex <- bioacoustic_index(tropicalsound)
print(bioindex$left_area)

summary(bioindex)
```

`measure_signals` *Measure a signal or song in a wavefile*

Description

This function lets the user select bounding boxes to get statistics of the signals of interest in a sound file.

Usage

```
measure_signals(wavfile, wl = 512, min_freq = NA, max_freq = NA, min_time = NA,
  max_time = NA, plot_range = 50, dBFS_range = 30, sample_size = 1,
  resultfile = NA, channel = "left")
```

Arguments

`wavfile`	a sound file in wav format.
`wl`	window length for the spectrogram.
`min_freq`	minimum frequency to draw the spectrogram, in kiloHertz.
`max_freq`	maximum frequency to draw the spectrogram, in kiloHertz.
`min_time`	minimum time to draw the spectrogram, in seconds.
`max_time`	maximum time to draw the spectrogram, in seconds.
`plot_range`	lower limit of values to plot the spectrogram.
`dBFS_range`	range of values that is considered a signal, based on the maximum that is calculated. See notes below.
`sample_size`	number of samples to measure in the spectrogram.
`resultfile`	name of the file to save the results.
`channel`	which channel to plot.

Value

The function will open a spectrogram plot to allow the user to click on the regions of interest. Once all the samples are selected, the function saves a file with the values measured in each sample. In addition, the results of the function `dfreq` of the package `seewave` are saved on a folder named the same as the wavfile, without the .wav extension.

Note

For the `dBFS_range` argument, the code uses the maximum of the values inside the selected region and saves as a resulting signal the values that fall between `(maximum - dBFS_range)` and the maximum. A selected region with a maximum value of -5 and `dBFS_range` set to 30 will consider the area with values between -35 and -5 dBFS as a signal.

The function creates a folder `dfreq` where it saves csv files with the results of the function `dfreq` from `seewave`. The name of each file is coded as: wavfile.samplenumber.csv

Examples

```
## Not run:
#Take 5 samples of the file file.wav between 1 - 4 kHz, from 10 to 30 seconds.
measure_signals(wavfile="file.wav", wl=2048, min_freq=1, max_freq=4,
  dBFS_range=30, min_time=10, max_time=30, sample_size=5,
  resultfile="results.csv", plot_range=70)

## End(Not run)
```

`multiple_sounds` *Multiple sound files*

Description

Function to extract the specified index from all the wav or flac files in a directory. The results, including the filename and wave technical details, are saved to a csv file. If the computer has multiple cores, it can run files in parallel.

Usage

```
multiple_sounds(directory, resultfile, soundindex, no_cores = 1,
flac = FALSE, from = NA, to = NA, units = NA, ...)
```

Arguments

`directory`	a valid directory, readable by the user, that contains the wav files.
`resultfile`	name of the text file to which write the results in comma-separated values format.
`soundindex`	which index to calculate: • `ndsi` • `acoustic_complexity` • `acoustic_diversity` • `acoustic_evenness` • `bioacoustic_index` • `H` from the `seewave` package
`no_cores`	number of cores to use when calculating the indices. Can be `max` to use all cores, `-1` to use all but one core, or any positive integer. Default is 1. Uses the `parallel` package.
`flac`	logical variable to indicate that the files are in FLAC format. FLAC must be installed in the system (see note below). Uses the function `wav2flac` of `seewave`.
`from`	tells `readWave` where to start loading the files. All three arguments `from`, `to`, and `units` must be specified at the same time, if used.
`to`	tells `readWave` where to stop loading the files. All three arguments `from`, `to`, and `units` must be specified at the same time, if used.

`units`	tells `readWave` which units to use to determine the start and stop points to load the files. The options are `"samples"`, `"seconds"`, `"minutes"`, or `"hours"`. All three arguments `from`, `to`, and `units` must be specified at the same time, if used.
`...`	additional variables to pass to the selected function. See each function's help for details.

Note

FLAC stands for Free Lossless Audio Codec. Files in FLAC format have been compressed without destruction of data, which happens in lossy compression codecs like the popular MP3. Files can be between 40-60% of the size of the original wav file, although this value depends on the contents. For more information and to download FLAC, visit http://xiph.org/flac/

Examples

```
## Not run:
#Calculate the ACI of all the wav
# files in the directory "/home/user/wavs/"
# using the function acoustic_complexity
multiple_sounds(directory = "/home/user/wavs/",
resultfile = "/home/user/results.csv",
soundindex = "acoustic_complexity")

#Calculate the same as above using 12000Hz as the
# maximum frequency instead of the default.
multiple_sounds(directory = "/home/user/wavs/",
resultfile = "/home/user/results.csv",
soundindex = "acoustic_complexity", max_freq = 12000)

#Calculate the same as above using two cores
multiple_sounds(directory = "/home/user/wavs/",
resultfile = "/home/user/results.csv",
soundindex = "acoustic_complexity", no_cores = 2)

#Calculate the same as above using all the cores
# the computer has
multiple_sounds(directory="/home/user/wavs/",
resultfile = "/home/user/results.csv",
soundindex = "acoustic_complexity", no_cores = "max")

#Calculate the same as above using all but one cores
multiple_sounds(directory = "/home/user/wavs/",
resultfile = "/home/user/results.csv",
soundindex = "acoustic_complexity", no_cores = -1)

## End(Not run)
```

ndsi *Normalized Difference Soundscape Index*

Description

Normalized Difference Soundscape Index (NDSI) from REAL and Kasten, *et al.* 2012. The NDSI seeks to "estimate the level of anthropogenic disturbance on the soundscape by computing the ratio of human-generated (anthrophony) to biological (biophony) acoustic components found in field collected sound samples" (Kasten, *et al.* 2012).

Tested with Matlab code courtesy of S. Gage.

Usage

```
ndsi(soundfile, fft_w = 1024, anthro_min = 1000, anthro_max = 2000,
bio_min = 2000, bio_max = 11000)
```

Arguments

`soundfile`	an object of class `Wave` loaded with the function readWave of the `tuneR` package.
`fft_w`	FFT window size.
`anthro_min`	minimum value of the range of frequencies of the anthrophony.
`anthro_max`	maximum value of the range of frequencies of the anthrophony.
`bio_min`	minimum value of the range of frequencies of the biophony.
`bio_max`	maximum value of the range of frequencies of the biophony.

Details

The bin size is determined as the difference between anthro_max and anthro_min, by default 1000 Hz.

Value

Returns a list with one object per channel

`ndsi_left`	NDSI value for the left channel
`ndsi_right`	NDSI value for the right channel
`biophony_left`	value for the biophony for the left channel
`anthrophony_left`	value for the anthrophony for the left channel
`biophony_right`	value for the biophony for the right channel
`anthrophony_right`	value for the anthrophony for the right channel

References

Remote Environmental Assessment Laboratory. http://www.real.msu.edu

Kasten, Eric P., Stuart H. Gage, Jordan Fox, and Wooyeong Joo. 2012. The Remote Environmental Assessment Laboratory's Acoustic Library: An Archive for Studying Soundscape Ecology. Ecological Informatics 12: 50-67. doi: 10.1016/j.ecoinf.2012.08.001

Examples

```
data(tropicalsound)
NDSI <- ndsi(tropicalsound)
print(NDSI$ndsi_left)

summary(NDSI)
```

---

soundecology *Soundscape Ecology*

Description

Functions to calculate indices for soundscape ecology and other ecology research that uses audio recordings.

Details

Package:	soundecology
Type:	Package
Version:	1.3.3
Date:	2018-03-04
License:	GPLv3

Author(s)

Luis J. Villanueva-Rivera and Bryan C. Pijanowski

---

sound_raster *ASCII raster from sound file*

Description

This function creates a raster file in ASCII format from the spectrogram of a soundfile. This file can be opened in ArcGIS or any other GIS software. For more details see the tutorial of Villanueva-Rivera *et al.* 2011.

Usage

```
sound_raster(wavfile = NA, wav_directory = NA, max_freq = 10000, no_cores = 1)
```

Arguments

wavfile	a single sound file in wav format.
max_freq	maximum frequency to draw the spectrogram, in Hertz.
wav_directory	a directory that contains wav files. To specify the working directory, use wav_directory="."
no_cores	number of cores to use when working in a directory. Can be max to use all cores, -1 to use all but one core, or any positive integer. Default is 1. Uses the parallel package.

Value

The function will save a file for each channel, in the same directory where the files are at, with the extension .asc.

Note

To get a raster file for a single file, use the argument wavfile. For many files, use the argument wav_directory. Do not use both at the same time or the function will return an error.

This function was released with the version 1.3 of the tutorial of the primer paper, available at:

http://ltm.agriculture.purdue.edu/soundscapes/primer/

and at the website of the package:

http://ljvillanueva.github.io/soundecology/

References

Villanueva-Rivera, L. J., B. C. Pijanowski, J. Doucette, and B. Pekin. 2011. A primer of acoustic analysis for landscape ecologists. Landscape Ecology 26: 1233-1246. doi: 10.1007/s10980-011-9636-9.

Examples

```
## Not run:
sound_raster(wavfile = "file1.wav")

sound_raster(wav_directory = "/home/user/wavdirectory")

sound_raster(wav_directory = "/home/user/wavdirectory", no_cores = 4)

## End(Not run)
```

`tropicalsound` *tropicalsound sound example*

Description

Sample sound of a digital recording of a chorus of tropical frogs.

Usage

```
data(tropicalsound)
```

Format

A `Wave` object.

Details

Duration = 20 sec. Sampling rate = 22050 Hz.

Source

Recording made at a tropical rainforest in Puerto Rico by Luis J. Villanueva-Rivera.

Examples

```
data(tropicalsound)

tropicalsound
```

Index