Package: Ruido (via r-universe)

October 24, 2026

Title Soundscape Spectral Metrics

Version 1.2.1.9000

Date 2026-10-17

Description Accessible and flexible implementation of spectral ecoacoustic indices. The functions were design to accommodate a variety of sampling designs. Users can tailor calculations by specifying spectrogram time bin size, amplitude thresholds and normality tests. By simplifying computation and standardizing reproducible methods, the package aims to support ecoacoustics studies.

Depends R(>= 4.3.0)

Imports methods, tuneR, matrixStats, nortest, wav

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Suggests ggplot2, patchwork, testthat (>= 3.0.0)

Maintainer Arthur Igor da Fonseca-Freire <arthur.igorr@gmail.com>

BugReports https://github.com/Arthurigorr/Ruido/issues

Config/testthat/edition 3

URL https://github.com/Arthurigorr/Ruido

Language en-US

LazyData true

Config/roxygen2/version 8.1.0

Repository https://arthurigorr.r-universe.dev

Date/Publication 2026-10-24 02:28:30 UTC

RemoteUrl https://github.com/arthurigorr/ruido

RemoteRef HEAD

RemoteSha 4fec9299c07cfc5ef3550e6b1b7caeb1cdd6ee92

Contents

ACIspec . . . 2
activity . . . 5
bgn . . . 7
bgNoise . . . 10
ENTspec . . . 13
EVNspec . . . 16
hamming . . . 18
hanning . . . 19
multActivity . . . 20
noise.matrix-class . . . 23
plot,noise.matrix,ANY-method . . . 24
plotNOISE . . . 25
sampleBGN . . . 27
satBackup . . . 28
singleSat . . . 29
soundMat . . . 33
soundSat . . . 36

Index **41**

`ACIspec` *Spectral Acoustic Complexity Index*

Description

Calculate the Acoustic Complexity Index values of a single audio using the methodology proposed in Pieretti, et al. 2011

Usage

```
ACIspec(
  soundfile,
  channel = "stereo",
  timeBin = 60,
  j = 5,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2)
)
```

Arguments

`soundfile`	wav package numeric matrix, tuneR package Wave object or path to a `.wav` file
`channel`	channel where the metric values will be extracted from. Available channels are: `"stereo"`, `"mono"`, `"left"` or `"right"`. Defaults to `"stereo"`
`timeBin`	size (in seconds) of the time bin. Set to `NULL` to use the entire audio as a single bin. Defaults to `60`
`j`	size (in seconds) of the cluster interval. Set to `NULL` to use the entire bin as a single cluster. Defaults to `5`
`targetSampRate`	desired sample rate of the audios. This argument is only used to down sample the audio. If `NULL`, then audio's sample rate remains the same. Defaults to `NULL`
`wl`	window length of the spectrogram. Defaults to `512`
`window`	window used to smooth the spectrogram. Switch to `hanning(wl)` to use hanning instead. Defaults to `hamming(wl)`
`overlap`	overlap between the spectrogram windows. Defaults to `wl/2` (half the window length)

Details

The Acoustic Complexity Index (`ACI`) quantifies the average proportional change in spectral amplitude between adjacent time steps across frequency bins. Because biological sounds, particularly bird vocalizations, often exhibit rapid and irregular amplitude fluctuations over time, `ACI` captures this temporal variability as a proxy for acoustic complexity.

In `Ruido`, `ACI` is computed independently within each time bin. Within a time bin, the signal is further subdivided into smaller temporal segments, here referred to as cluster intervals j.

For a given frequency bin f_l, acoustic intensity values I_k are evaluated across consecutive time steps k within each cluster interval j. The absolute differences between adjacent time steps are calculated as:

$$d_k = |I_k - I_{k+1}|$$

These differences are summed within each cluster interval:

$$D_j = \sum_{k=1}^{N} d_k$$

where N is the number of time steps Δt_k in interval j. The `ACI` for each cluster interval is then:

$$ACI_j = \frac{D_j}{\sum_{k=1}^{N} I_k}$$

where $\sum_{k=1}^{N} I_k$ is the total acoustic intensity within the same interval.

For each frequency bin f_l, `ACI` values are summed across all cluster intervals within the time bin:

$$ACI_{f_l} = \sum_{j=1}^{m} ACI_j$$

where m is the number of cluster intervals in the time bin.

The result is a frequency-resolved representation of `ACI` for each time bin, rather than a single scalar value for the entire recording.

In the original formulation (Pieretti et al., 2011), `ACI` is further summed across all frequency bins:

$$ACI_{tot} = \sum_{l=1}^{q} ACI_{f_l}$$

where q is the total number of frequency bins. This final aggregation step is not performed in this package.

Value

This function returns a noise.matrix object.

References

Pieretti, N., Farina, A., & Morri, D. (2011). A new methodology to infer the singing activity of an avian community: The Acoustic Complexity Index (ACI). Ecological Indicators, 11(3), 868–873. https://doi.org/10.1016/j.ecolind.2010.11.005

See Also

`ENTspec()` to calculate Spectral Entropy and `bgNoise()` to calculate Background Noise and Soundscape Power.

Examples

```
### This is an secondary example using audio from a real soundscape
### These audios are originated from the Escutadô Project, a project
### that records the soundscapes of the brazilian semiarid
# Getting audiofile from the online Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
rec = paste0("GAL24576_20250401_", sprintf("%06d", 0), ".wav")
recDir = paste(dir, rec , sep = "/")
url = paste0("https://zenodo.org/records/17575795/files/",
             rec,
             "?download=1")

# Downloading the file, might take some time denpending on your internet
download.file(url, destfile = recDir, mode = "wb")

# Running the ACIspec function with all the default arguments
aci = ACIspec(recDir)
```

```
# Here's the result
aci

# Plot ACI values
plot(aci)
```

---

activity *Acoustic Activity Matrix*

Description

Calculate the Acoustic Activity Matrix using the methodology proposed in Burivalova 2018

Usage

```
activity(
  soundfile,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  DCfix = TRUE,
  powthr = 10,
  bgnthr = 0.8,
  beta = TRUE
)
```

Arguments

soundfile	wav package numeric matrix, tuneR package Wave object or path to a .wav file
channel	channel where the saturation values will be extracted from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo".
timeBin	size (in seconds) of the time bin. Set to NULL to use the entire audio as a single bin. Defaults to 60
dbThreshold	minimum allowed value of dB for the spectrograms. Set to NULL to leave db values unrestricted Defaults to -90, as set by Towsey 2017
dbRef	reference value used to compute decibel values from the spectrogram. Defaults to 10
targetSampRate	desired sample rate of the audios. This argument is only used to down sample the audio. If NULL, then audio's sample rate remains the same. Defaults to NULL

wl	window length of the spectrogram. Defaults to 512
window	window used to smooth the spectrogram. Switch to hanning(wl) to use hanning instead. Defaults to hamming(wl)
overlap	overlap between the spectrogram windows. Defaults to wl/2 (half the window length)
histbreaks	breaks used to calculate Background Noise. Available breaks are: "FD", "Sturges", "scott" and 100. Defaults to "FD". Can also be set to any numerical value to limit or increase the amount of breaks.
DCfix	if the DC offset should be removed before the metrics are calculated. Defaults to TRUE
powthr	single numeric value to calculate the activity matrix for soundscape power (in dB). Defaults to 10
bgnthr	single numeric value to calculate the activity matrix for background noise (in %). Defaults to 0.8
beta	how BGN thresholds are calculated. If TRUE, BGN thresholds are calculated using all recordings combined. If FALSE, BGN thresholds are calculated separately for each recording. Defaults to TRUE

Details

To calculate the activity matrix, we use the methodology proposed by Burivalova 2018. We begin by applying the following formula to each time bin of the recording:

$$a_{m,f} = \begin{cases} 1, & \text{if } BGN_{m,f} > \theta_1 \text{ or } POW_{m,f} > \theta_2 \\ 0, & \text{otherwise} \end{cases}$$

where θ is a user-defined threshold applied uniformly to both BGN and POW. We set 1 to active and 0 to inactive frequency windows.

Value

This function returns a 0 and 1 matrix containing the activity for all time bins of the inputted file. The matrix's number of rows will equal to half the set window length (wl) and number of columns will equal the number of bins. Cells with the value of 1 represent the acoustically active frequency of a bin.

References

Burivalova, Z., Towsey, M., Boucher, T., Truskinger, A., Apelis, C., Roe, P., & Game, E. T. (2018). Using soundscapes to detect variable degrees of human influence on tropical forests in Papua New Guinea. Conservation Biology, 32(1), 205-215. https://doi.org/10.1111/cobi.12968

See Also

multActivity() to run this over multiple audio files and singleSat() to get the full saturation values this activity matrix is derived from.

Examples

```
if (require("ggplot2")) {
library(ggplot2)
# We are going to load a sample noise.matrix object to demonstrate the basic usage of singleSat()
# To understand about the origin of this noise.matrix, check: ?sampleBGN
data("sampleBGN")

# View the sample noise.matrix object
sampleBGN

# Run the function
act = activity(sampleBGN)

# Now we can plot the results for the left channel
actLeft = act@values$left$ACT
actDim = dim(actLeft)
numericTime = seq(0, sum(sampleBGN@timeBins), by = sampleBGN@timeBins[1])
labels = paste0(numericTime[-length(numericTime)], "-", numericTime[-1], "s")

actDF = data.frame(BIN = rep(paste0("BIN", seq(actDim[2])), each = actDim[1]),
                   WIN = rep(seq(actDim[1]), actDim[2]),
                   ACT = factor(unlist(actLeft), levels = c(0, 1)))

ggplot(actDF, aes(x = BIN, y = WIN, fill = ACT)) +
  geom_tile() +
  theme_bw() +
  scale_fill_manual(values = c("white", "black")) +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_discrete(expand = c(0,0), labels = labels) +
  labs(x = "Time Bin", y = "Spectral Window") +
  guides(fill = guide_legend(title = "Activity"))

}
```

---

bgn *Summarized Background Noise and Soundscape Power Index*

Description

Calculate the summarized Background Noise and Soundscape Power values of a single audio using the methodology proposed in Towsey 2017

Usage

```
bgn(
  soundfile,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
```

```
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  histbreaks = "FD",
  DCfix = TRUE
)
```

Arguments

soundfile	wav package numeric matrix, tuneR package Wave object or path to a .wav file
channel	channel where the metric values will be extracted from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo"
timeBin	size (in seconds) of the time bin. Set to NULL to use the entire audio as a single bin. Defaults to 60
dbThreshold	minimum allowed value of dB for the spectrograms. Set to NULL to leave db values unrestricted Defaults to -90, as set by Towsey 2017
dbRef	reference value used to compute decibel values from the spectrogram. Defaults to 10
targetSampRate	desired sample rate of the audios. This argument is only used to down sample the audio. If NULL, then audio's sample rate remains the same. Defaults to NULL
wl	length of each waveform frame in samples. The waveform is divided into non-overlapping frames of this length before the envelope is calculated. Defaults to 512
histbreaks	breaks used to calculate Background Noise. Available breaks are: "FD", "Sturges", "scott" and 100. Defaults to "FD". Can also be set to any numerical value to limit or increase the amount of breaks.
DCfix	if the DC offset should be removed before the metrics are calculated. Defaults to TRUE

Details

Background Noise (BGN) is an acoustic metric that estimates the dominant background sound level within a time bin from the waveform amplitude envelope. Following the approach described by Towsey (2017), each time bin is divided into non-overlapping waveform frames, and the maximum absolute amplitude of each frame is used to construct the waveform envelope.

For each waveform frame i, the envelope amplitude A_i is converted to decibels as:

$$dB_i = dbRef. \log_{10}(A_i)$$

where A_i is the maximum absolute amplitude within the frame. Incomplete frames at the end of a time bin are discarded.

The resulting dB values are grouped into histogram bins, and BGN is estimated from the modal histogram bin, representing the most frequently occurring sound level within the time bin:

$$BGN = \text{mode}(dB_i)$$

BGN therefore represents an estimate of the dominant background level of the recording segment, rather than the background level of a particular frequency band.

Soundscape Power (POW) quantifies the difference between the maximum observed sound level and the estimated background level within the same time bin. It is defined as:

$$POW = \max(dB_i) - BGN$$

Higher POW values indicate a greater difference between the strongest acoustic event and the estimated background level.

Value

A list containing the BGN and POW values calculated for each time bin and selected channel.

References

Towsey, M. W. (2017). The calculation of acoustic indices derived from long-duration recordings of the natural environment. In eprints.qut.edu.au. https://eprints.qut.edu.au/110634/ Lamel, L., Rabiner, L., Rosenberg, A., & Wilpon, J. (1981). An improved endpoint detector for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(4), 777-785 https://doi.org/10.1109/TASSP.1981.1163642

See Also

bgNoise() to calculate Spectral Background Noise and Soundscape Power.

Examples

```
### For our main example we'll create an artificial audio with
### white noise to test its Background Noise
# We'll use the package tuneR
library(tuneR)

# Define the audio sample rate, duration and number of samples
samprate = 12050
dur = 60
n = samprate * dur

# Then we generate white noise
set.seed(413)
noise = rnorm(n)

# Linear fade-out envelope
fade = seq(1, 0, length.out = n)

# Apply fade
signal = noise * fade

wave = Wave(left = signal, right = signal,
            samp.rate = samprate,
            bit = 16)
```

```
# Heres our artificial audio

wave

# Running the bgNoise function with all the default arguments
BGN = bgn(wave)

# See the results
BGN
```

---

bgNoise *Spectral Background Noise and Soundscape Power Index*

Description

Calculate the spectral Background Noise and Soundscape Power values of a single audio using the methodology proposed in Towsey 2017

Usage

```
bgNoise(
  soundfile,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  DCfix = TRUE
)
```

Arguments

soundfile	wav package numeric matrix, tuneR package Wave object or path to a .wav file
channel	channel where the metric values will be extracted from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo"
timeBin	size (in seconds) of the time bin. Set to NULL to use the entire audio as a single bin. Defaults to 60
dbThreshold	minimum allowed value of dB for the spectrograms. Set to NULL to leave db values unrestricted Defaults to -90, as set by Towsey 2017
dbRef	reference value used to convert spectrogram amplitude values to decibels. Defaults to 10

`targetSampRate`	desired sample rate of the audios. This argument is only used to down sample the audio. If `NULL`, then audio's sample rate remains the same. Defaults to `NULL`
`wl`	window length of the spectrogram. Defaults to `512`
`window`	window used to smooth the spectrogram. Switch to `hanning(wl)` to use hanning instead. Defaults to `hamming(wl)`
`overlap`	overlap between the spectrogram windows. Defaults to `wl/2` (half the window length)
`histbreaks`	breaks used to calculate Background Noise. Available breaks are: `"FD"`, `"Sturges"`, `"scott"` and `100`. Defaults to `"FD"`. Can also be set to any numerical value to limit or increase the amount of breaks.
`DCfix`	if the DC offset should be removed before the metrics are calculated. Defaults to `TRUE`

Details

Background Noise (`BGN`) is an acoustic metric that estimates the dominant baseline level of acoustic energy within a frequency window and time bin. It was described by Towsey (2017) based on the approach of Lamel et al. (1981).

Spectrogram amplitude values are converted to decibels according to:

$$dB_i = dbRef. \log_{10}(A_i)$$

where A_i is the amplitude of the i-th spectrogram value and $dbRef$ is the coefficient used for the conversion. By default, $dbRef = 10$.

For each frequency window f and time bin c, `BGN` is defined as the modal value of the intensity distribution (in dB), representing the most frequently occurring sound level:

$$BGN_f = \text{mode}(dB_{c,f})$$

This value approximates the continuous background component of the soundscape, filtering out transient acoustic events such as bird calls or other short-duration signals.

Soundscape Power (`POW`) quantifies the contrast between this baseline level and the strongest acoustic events within the same frequency window and time bin. It is defined as:

$$POW_f = \max(dB_{c,f}) - BGN_f$$

where $\max(dB_{c,f})$ is the maximum intensity observed. `POW` can be interpreted as a proxy for signal-to-noise ratio, with higher values indicating stronger or more prominent acoustic events relative to the background level.

Value

This function returns a noise.matrix object

References

Towsey, M. W. (2017). The calculation of acoustic indices derived from long-duration recordings of the natural environment. In eprints.qut.edu.au. https://eprints.qut.edu.au/110634/ Lamel, L., Rabiner, L., Rosenberg, A., & Wilpon, J. (1981). An improved endpoint detector for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(4), 777-785 https://doi.org/10.1109/TASSP.1981.1163642

See Also

`bgn()` to calculate the summarized Background Noise and Soundscape Power. `ACIspec()` to calculate the Acoustic Complexity Index and `ENTspec()` to calculate Spectral Entropy from a single audio file. Also, check `activity()` and `singleSat()`, which use this same Background Noise and Soundscape Power calculation to determine acoustic activity and saturation.

Examples

```
### For our main example we'll create an artificial audio with
### white noise to test its Background Noise
# We'll use the package tuneR
library(tuneR)

# Define the audio sample rate, duration and number of samples
samprate = 12050
dur = 60
n = samprate * dur

# Then we generate white noise
set.seed(413)
noise = rnorm(n)

# Linear fade-out envelope
fade = seq(1, 0, length.out = n)

# Apply fade
signal = noise * fade

wave = Wave(left = signal, right = signal,
            samp.rate = samprate,
            bit = 16)

# Heres our artificial audio

wave

# Running the bgNoise function with all the default arguments
BGN = bgNoise(wave)

# See the results
BGN

# Plot background noise and soundscape power
```

```
plot(BGN)


### This is a secondary example using audio from a real soundscape
### These audios are originated from the Escutadô Project, a project
### that records the soundscapes of the brazilian semiarid
# Getting audiofile from the online Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
rec = paste0("GAL24576_20250401_", sprintf("%06d", 0), ".wav")
recDir = paste(dir, rec , sep = "/")
url = paste0("https://zenodo.org/records/17575795/files/",
             rec,
             "?download=1")

# Downloading the file, might take some time denpending on your internet
download.file(url, destfile = recDir, mode = "wb")

# Running the bgNoise function with all the default arguments
BGN = bgNoise(recDir)

# Here's the result
BGN

# Plot background noise and soundscape power values
plot(BGN)

# Plot the two indices against each other
plot(BGN@values$left$BGN$BGN1, BGN@values$left$POW$POW1,
     xlab = "BGN (dB)", ylab = "POW (dB)", pch = 16)

# Now lets test and plot their correlation
BGNPOWlm = lm(BGN@values$left$BGN$BGN1~BGN@values$left$POW$POW1)
summary(BGNPOWlm)
abline(lm(BGN@values$left$BGN$BGN1~BGN@values$left$POW$POW1), col = "red")
```

ENTspec *Spectral Temporal Entropy Index*

Description

Calculate the Temporal Entropy values of a single audio using the methodology proposed in Towsey, et al. 2017

Usage

```
ENTspec(
  soundfile,
  channel = "stereo",
```

```
  timeBin = 60,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2)
)
```

Arguments

soundfile	wav package numeric matrix, tuneR package Wave object or path to a .wav file
channel	channel where the metric values will be extracted from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo"
timeBin	size (in seconds) of the time bin. Set to NULL to use the entire audio as a single bin. Defaults to 60
targetSampRate	desired sample rate of the audios. This argument is only used to down sample the audio. If NULL, then audio's sample rate remains the same. Defaults to NULL
wl	window length of the spectrogram. Defaults to 512
window	window used to smooth the spectrogram. Switch to hanning(wl) to use hanning instead. Defaults to hamming(wl)
overlap	overlap between the spectrogram windows. Defaults to wl/2 (half the window length)

Details

The Temporal Entropy (ENT) quantifies how concentrated or dispersed acoustic energy is over time within each frequency bin. Unlike indices that track frame-to-frame change (e.g. ACI), ENT treats the distribution of energy across an entire time bin as a probability mass function and measures its Shannon entropy, capturing whether energy is spread evenly through time (high entropy, low concentration) or concentrated into brief pulses (low entropy, high concentration).

In Ruido, ENT is computed independently within each time bin, using every time step of the spectrogram in that bin.

For a given frequency bin f, the squared amplitude values I_t across all time steps t within the time bin are normalized to unit area, producing a probability mass function:

$$pmf_{f,t} = \frac{I_t^2}{\sum_{t=1}^{N} I_t^2}$$

where N is the number of time steps in the time bin. The Shannon entropy of this distribution is then calculated as:

$$H[f] = \frac{-\sum_{t=1}^{N} pmf_{f,t} \times \log_2(pmf_{f,t})}{\log_2 N}$$

To express the result as an intuitive measure of energy concentration rather than dispersion, ENT is calculated as the complement of H:

$$ENT[f] = 1 - H[f]$$

The result is a frequency-resolved representation of ENT for each time bin, rather than a single scalar value for the entire recording. Values close to 1 indicate energy concentrated in few time steps (e.g. transient calls or pulses), while values close to 0 indicate energy spread evenly across the time bin (e.g. steady background noise).

Value

This function returns a noise.matrix object.

References

Towsey, M. W. (2017). The calculation of acoustic indices derived from long-duration recordings of the natural environment. In eprints.qut.edu.au. https://eprints.qut.edu.au/110634/

See Also

ACIspec() to calculate the Acoustic Complexity Index and bgNoise() to calculate Background Noise and Soundscape Power.

Examples

```
### This is an example using audio from a real soundscape
### These audios are originated from the Escutadô Project, a project
### that records the soundscapes of the brazilian semiarid
# Getting audiofile from the online Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
rec = paste0("GAL24576_20250401_", sprintf("%06d", 0), ".wav")
recDir = paste(dir, rec , sep = "/")
url = paste0("https://zenodo.org/records/17575795/files/",
             rec,
             "?download=1")

# Downloading the file, might take some time denpending on your internet
download.file(url, destfile = recDir, mode = "wb")

# Running the ENTspec function with all the default arguments
ent = ENTspec(recDir)

# Here's the result
ent

# Plot ENT values
plot(ent)
```

EVNspec *Spectral Acoustic Event Index*

Description

Calculate the number of acoustic events per second in each frequency bin of a single audio using the methodology described by Towsey (2017).

Usage

```
EVNspec(
  soundfile,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  DCfix = TRUE
)
```

Arguments

soundfile	wav package numeric matrix, tuneR package Wave object or path to a .wav file
channel	channel where the metric values will be extracted from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo"
timeBin	size (in seconds) of the time bin. Set to NULL to use the entire audio as a single bin. Defaults to 60
dbThreshold	minimum allowed value of dB for the spectrograms. Set to NULL to leave db values unrestricted Defaults to -90, as set by Towsey 2017
dbRef	reference value used to compute decibel values from the spectrogram. Defaults to 10
targetSampRate	desired sample rate of the audios. This argument is only used to down sample the audio. If NULL, then audio's sample rate remains the same. Defaults to NULL
wl	window length of the spectrogram. Defaults to 512
window	window used to smooth the spectrogram. Switch to hanning(wl) to use hanning instead. Defaults to hamming(wl)
overlap	overlap between the spectrogram windows. Defaults to wl/2 (half the window length)
histbreaks	breaks used to calculate Background Noise. Available breaks are: "FD", "Sturges", "scott" and 100. Defaults to "FD". Can also be set to any numerical value to limit or increase the amount of breaks.

DCfix if the DC offset should be removed before the metrics are calculated. Defaults to TRUE

Details

Spectral Acoustic Events (EVNsp) quantifies the number of acoustic events per second within each frequency bin. An acoustic event is defined as an upward crossing of a 3-dB threshold above the estimated background noise level.

The spectrogram is first converted to decibels and the spectral Background Noise (BGN) is calculated independently for each frequency bin and time bin. The resulting background level is then subtracted from the corresponding frequency bin of the spectrogram to produce a noise-reduced spectrogram:

$$D_{f,t} = dB_{f,t} - BGN_f$$

where $D_{f,t}$ is the noise-reduced intensity for frequency bin f at spectral frame t, $dB_{f,t}$ is the corresponding spectrogram intensity, and BGN_f is the estimated background level for that frequency bin.

An acoustic event is counted whenever the noise-reduced intensity crosses the 3-dB threshold from below to at or above the threshold:

$$EVN_f = \sum_t I(D_{f,t-1} < 3; \mathrm{dB}; \mathrm{and}; D_{f,t} \geq 3; \mathrm{dB})$$

where I is an indicator function. Thus, consecutive spectral frames remaining above the threshold are considered part of the same event and are not counted repeatedly. Values are divided by the bin size in seconds.

Event counts are calculated independently for each frequency bin. The resulting values represent the number of detected acoustic events per time bin and are expressed as events per second.

Value

This function returns a noise.matrix object containing the number of acoustic events per second in each frequency bin.

References

Towsey, M. W. (2017). The calculation of acoustic indices derived from long-duration recordings of the natural environment. In eprints.qut.edu.au. https://eprints.qut.edu.au/110634/ Lamel, L., Rabiner, L., Rosenberg, A., & Wilpon, J. (1981). An improved endpoint detector for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(4), 777-785 https://doi.org/10.1109/TASSP.1981.1163642

See Also

bgNoise() to calculate spectral Background Noise and Soundscape Power, bgn() to calculate summarized Background Noise and Soundscape Power, and ACIspec() to calculate the Acoustic Complexity Index from a single audio file.

Examples

```
### These audios are originated from the Escutadô Project, a project
### that records the soundscapes of the brazilian semiarid
# Getting audiofile from the online Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
rec = paste0("GAL24576_20250401_", sprintf("%06d", 0), ".wav")
recDir = paste(dir, rec , sep = "/")
url = paste0("https://zenodo.org/records/17575795/files/",
             rec,
             "?download=1")

# Downloading the file, might take some time denpending on your internet
download.file(url, destfile = recDir, mode = "wb")

# Running the EVNspec function with all the default arguments
evn = EVNspec(recDir)

# Here's the result
evn

# Plot EVN values
plot(evn)
```

---

hamming *Calculate A Hamming Window*

Description

Generates a Hamming window of a specified length.

Usage

```
hamming(n)
```

Arguments

n	single positive integer greater than 1 specifying the length of the window

Details

The Hamming window is a cosine window commonly used to reduce spectral leakage before Fourier transformation.

The window is calculated following the formula described in Harris, 1978:

$$w_k = 0.54 - 0.46 \cos\left(\frac{2\pi k}{n-1}\right)$$

for $k = 0, \ldots, n-1$.

Value

A numeric vector containing the Hamming window.

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. https://doi.org/10.1109/proc.1978.10837

Examples

```
hammingWindow = hamming(256)
plot(hammingWindow, type = "l")
```

`hanning` *Calculate A Hann Window*

Description

Generates a Hann window of a specified length.

Usage

```
hanning(n)
```

Arguments

`n`	single positive integer greater than 1 specifying the length of the window

Details

The Hann window is a cosine window commonly used to reduce spectral leakage before Fourier transformation.

The window is calculated following the formula described in Harris, 1978:

$$w_k = 0.5 - 0.5 \cos\left(\frac{2\pi k}{n-1}\right)$$

for $k = 0, \ldots, n-1$.

Value

A numeric vector containing the Hann window.

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. https://doi.org/10.1109/proc.1978.10837

Examples

```
hannWindow = hanning(256)
plot(hannWindow, type = "l")
```

`multActivity` *Multiple Acoustic Activity Matrix*

Description

Calculate the Acoustic Activity Matrix used in the the calculation of Soundscape Saturation using Burivalova 2018 methodology for a set of recordings

Usage

```
multActivity(
  soundpath,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  DCfix = TRUE,
  powthr = 10,
  bgnthr = 0.8,
  beta = TRUE,
  saveNoise = FALSE,
  backup = NULL
)
```

Arguments

`soundpath`	single or multiple directories to your `.wav` audio files
`channel`	channel where the saturation values will be extracted from. Available channels are: `"stereo"`, `"mono"`, `"left"` or `"right"`. Defaults to `"stereo"`.
`timeBin`	size (in seconds) of the time bin. Set to `NULL` to use the entire audio as a single bin. Defaults to `60`
`dbThreshold`	minimum allowed value of dB for the spectrograms. Set to `NULL` to leave db values unrestricted Defaults to `-90`, as set by Towsey 2017
`dbRef`	reference value used to compute decibel values from the spectrogram.
`targetSampRate`	desired sample rate of the audios. This argument is only used to down sample the audio. If `NULL`, then audio's sample rate remains the same. Defaults to `NULL`

wl	window length of the spectrogram. Defaults to 512
window	window used to smooth the spectrogram. Switch to hanning(wl) to use hanning instead. Defaults to hamming(wl)
overlap	overlap between the spectrogram windows. Defaults to wl/2 (half the window length)
histbreaks	breaks used to calculate Background Noise. Available breaks are: "FD", "Sturges", "scott" and 100. Defaults to "FD". Can also be set to any numerical value to limit or increase the amount of breaks.
DCfix	if the DC offset should be removed before the metrics are calculated. Defaults to TRUE
powthr	single numeric value to calculate the activity matrix for soundscape power (in dB). Defaults to 10
bgnthr	single numeric value to calculate the activity matrix for background noise (in %). Defaults to 0.8
beta	how BGN thresholds are calculated. If TRUE, BGN thresholds are calculated using all recordings combined. If FALSE, BGN thresholds are calculated separately for each recording. Defaults to TRUE
saveNoise	whether the noise.matrix objects generated internally should also be included in the output. Defaults to TRUE
backup	directory to save the backup. Defaults to NULL

Details

In this function, we only generate activity matrices for an directory using Burivalova 2018 methodology. For each time bin of the recording we apply the following formula:

$$a_{m,f} = \begin{cases} 1, & \text{if } BGN_{m,f} > \theta_1 \text{ or } POW_{m,f} > \theta_2 \\ 0, & \text{otherwise} \end{cases}$$

where θ is a user-defined threshold applied uniformly to both BGN and POW. We set 1 to active and 0 to inactive frequency windows.

If backup is set to a valid directory, a file named "SATBACKUP.rds" is automatically saved after every batch of five processed files. This file stores the current processing state and allows interrupted runs (e.g., due to manual termination, session crashes, or system shutdowns) to be resumed using satBackup().

To resume processing, pass the saved file (e.g., "path/SATBACKUP.rds") to satBackup(). Once a backup has been created, all original arguments and file paths must remain unchanged, unless they are explicitly modified within the saved .rds object.

Value

A list containing five objects. The first and second objects (powthresh and bgnthresh) are the threshold values inputted as arguments into the function. The third (info) contains the following variables from every audio file: PATH, AUDIO, CHANNEL, DURATION, BIN, SAMPRATE.. The fourth object (values) contains a matrix with the the values of activity for each bin of each recording and the size of the bin in seconds. The fifth contains a list with errors that occurred with specific files during the function.

References

Burivalova, Z., Towsey, M., Boucher, T., Truskinger, A., Apelis, C., Roe, P., & Game, E. T. (2018). Using soundscapes to detect variable degrees of human influence on tropical forests in Papua New Guinea. Conservation Biology, 32(1), 205-215. https://doi.org/10.1111/cobi.12968

See Also

`activity()` to run this on a single audio file, `soundSat()` and `soundMat()` to get saturation values instead. Also, check `satBackup()` if you are working with larger datasets and want some safety.

Examples

```
if (require("ggplot2") & require("patchwork")) {
  ### Generating an artificial audio for the example
  ## For this example we'll generate a sweep in a noisy soundscape
  library(ggplot2)
  library(patchwork)

  ### Downloading audiofiles from public Zenodo library
  dir = paste0(tempdir(), "/forExamples")
  dir.create(dir)
 recName = paste0("GAL24576_20250401_", sprintf("%06d", seq(0, 200000, by = 50000)), ".wav")
  recDir = paste(dir, recName, sep = "/")

  for (rec in recName) {
    print(rec)
    url = paste0("https://zenodo.org/records/17575795/files/",
                 rec,
                 "?download=1")
    download.file(url,
                  destfile = paste(dir, rec, sep = "/"),
                  mode = "wb")
  }

  time = sapply(strsplit(recName, "_"), function(x)
    paste(substr(x[3], 1, 2), substr(x[3], 3, 4), substr(x[3], 5, 6), sep = ":"))
  date = sapply(strsplit(recName, "_"), function(x)
    paste(substr(x[2], 1, 4), substr(x[2], 5, 6), substr(x[2], 7, 8), sep = "-"))

  dateTime = as.POSIXct(paste(date, time))

  timeLabels = time[c(1, 7, 13, 19, 24)]
  timeBreaks = as.character(dateTime[c(1, 7, 13, 19, 24)])

  breaks = round(c(1, cumsum(rep(256 / 6, 6))))

  ### Running the function
  act = multActivity(dir)

  plotN = 1

  sDim = dim(act$values)
```

```
sampRate = act$info$SAMPRATE[1]
kHz = cumsum(c(0, rep(sampRate / 6, 6))) / 1000

plotList = list()

for (cha in c("left", "right")) {
  actCurrent = act$values[, act$info$CHANNEL == cha]
  actCurrentDF = data.frame(
    TIME = as.character(rep(dateTime, each = sDim[1])),
    SPEC = rep(seq(sDim[1]), sDim[2]),
    VAL = factor(c(unlist(actCurrent)), levels = c(0, 1))
  )

  plotList[[plotN]] = ggplot(actCurrentDF, aes(x = TIME, y = SPEC, fill = VAL)) +
    geom_tile() +
    theme_classic() +
    scale_y_continuous(expand = c(NA, NA),
                       labels = kHz,
                       breaks = breaks) +
    scale_x_discrete(expand = c(0, 0),
                     labels = time) +
    scale_fill_manual(values = c("white", "black"),
                      labels = c("Inactive", "Active")) +
    guides(fill = guide_legend(title = "Acoustic Activity")) +
    labs(
      x = "Time of Day",
      y = "Frequency (kHz)",
      title = paste("Acoustic Activity in the", cha, "channel")
    )

  plotN = plotN + 1

}

plotList[[1]] + plotList[[2]] + plot_layout(guide = "collect")

unlink(recDir)
unlink(dir)

}
```

---

noise.matrix-class *noise.matrix class*

Description

This class object is generated by the bgNoise function. It can be read by soundSat. Methods: show() and plot()

Value

A `noise.matrix` object

Slots

`index` the calculated index

`values` a list containing the values of BGN and POW. Indices are separated in individual data.frames by channel

`timeBins` numeric vector containing the duration of each time bin

`sampRate` single numeric string containing the audio's sampling rate

`wl` single numeric string containing the spectrogram's window length

`channel` single character string containing the channel or channels used to calculated the indices

See Also

`bgNoise()`

`plot,noise.matrix,ANY-method`

Plot noise.matrix objects

Description

Plot noise.matrix objects

Usage

```
## S4 method for signature 'noise.matrix,ANY'
plot(
  x,
  channel = NULL,
  bin = 1,
  index = NULL,
  nbreaks = 5,
  yunit = c("hz", "khz"),
  main = NULL,
  xlab = NULL,
  ylab = "Frequency",
  col = c("blue", "red"),
  type = "p",
  draw0 = TRUE,
  box = TRUE,
  axes = TRUE,
  annotate = TRUE,
  ...
)
```

Arguments

x	an noise.matrix object
channel	channel or channels to be ploted. By default, this set to x@channel, but can be changed to left or right if x@channel = stereo
bin	temporal bin to be plotted. Defaults to 1
index	a character vector of length 1 or 2 with indexes to be plotted. Available indices are: c("BGN", "POW"), "BGN", "POW" and "ACI". Defaults to the indexes listed in x@index
nbreaks	amount of breaks of the y axis. Defaults to 5
yunit	frequency unit to be used in plot. Available units are: "hz" and "khz". Defaults to "hz"
main	title for the plot. Set two strings if you are plotting and stereo noise.matrix. If set to NULL, default title will be left/right/mono channel
xlab	label for the x-axis. Changes depending on x@index
ylab	label for the y-axis. Defaults to "Frequency"
col	plotting color for de indices. Defaults to c("blue","red")
type	desired plot type. For details see base::plot
draw0	if a stripped line should be drawn at 0. Defaults to TRUE
box	if a box should be drawn around the plot. Defaults to TRUE
axes	if axes should be drawn. Defaults to TRUE
annotate	if bin information should be added to the plot. Defaults to TRUE
...	further graphical parameters passed down to plot

Details

This is a method to quickly plot the results of bgNoise. This calls the helper function plotBGN, which is not meant to be used or seen by the user.

---

plotNOISE *Plot noise.matrix objects*

Description

Plot noise.matrix objects

Usage

```
plotNOISE(
  x,
  channel,
  bin,
  index,
  nbreaks,
  yunit,
  main,
  xlab,
  ylab,
  col,
  type,
  draw0,
  box,
  axes,
  annotate,
  ...
)
```

Arguments

x	an noise.matrix object
channel	channel or channels to be ploted. By default, this set to x@channel, but can be changed to left or right if x@channel = stereo
bin	temporal bin to be plotted. Defaults to 1
index	a character vector of length 1 or 2 with indexes to be plotted. Available indices are: c("BGN", "POW"), "BGN", "POW" and "ACI". Defaults to the indexes listed in x@index
nbreaks	amount of breaks of the y axis. Defaults to 5
yunit	frequency unit to be used in plot. Available units are: "hz" and "khz". Defaults to "hz"
main	title for the plot. Set two strings if you are plotting and stereo noise.matrix. If set to NULL, default title will be left/right/mono channel
xlab	label for the x-axis. Changes depending on x@index
ylab	label for the y-axis. Defaults to "Frequency"
col	plotting color for de indices. Defaults to c("blue","red")
type	desired plot type. For details see base::plot
draw0	if a stripped line should be drawn at 0. Defaults to TRUE
box	if a box should be drawn around the plot. Defaults to TRUE
axes	if axes should be drawn. Defaults to TRUE
annotate	if bin information should be added to the plot. Defaults to TRUE
...	further graphical parameters passed down to plot

Details

This is a method to quickly plot the results of bgNoise. This calls the helper function `plotBGN`, which is not meant to be used or seen by the user.

`sampleBGN` *Sample noise.matrix object*

Description

This is a sample noise.matrix object generated by the bgNoise function. Used in the examples of: singleSat and activity The original sound file was obtained from the Escutadô Project, an initiative aimed at recording soundscapes across the Brazilian semiarid region. The audio was recorded by an autonomous recorder (SM4) attached approximately 1.5 meters above ground on a metal rod installed in a Black Jurema (Mimosa tenuiflora) forest in Federal Rural University of the Semi-Arid (UFERSA), located in Mossoró, Rio Grande do Norte, Brazil. The recording was taken at midnight (00:00) and is 3 minutes long, with a sampling rate of 48 kHz and gain levels of 16 dB on the left channel and 10 dB on the right channel.

Usage

```
sampleBGN
```

Format

`sampleBGN`**:**

A noise.matrix object with:

@values Background Noise and Soundscape Power values

@timeBins Duration in seconds of each time bin

@sampRate The recording's sample rate

@wl The window length used to generate the spectrogram used to calculate the metrics

@channel The channel or channels of the recording

Source

https://zenodo.org/records/17243660

satBackup *Backup for Ruido's functions*

Description

This function offers a way to continue an unfinished process of the soundSat(), soundMat() or multActivity() functions through a backup file. Arguments can't be inputted nor changed since the function will automatically load them from the .rds file. However you may manually change them by editing the file (not recommended).

Usage

```
satBackup(backup)
```

Arguments

backup	path to the .rds file create by the backup of soundSat, soundMat or multActivity

Value

This functions returns the same output of soundSat(), soundMat() or multActivity()

Examples

```
## Not run:
# It's impossible to demonstrate this function's intended use due to it's nature
# However, here is how this function is used:
## This example will load an entire day of audios to your computer, so beware.

### Downloading audiofiles from public Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
recName = paste0("GAL24576_20250401_", sprintf("%06d", seq(0, 230000, by = 10000)),".wav")
recDir = paste(dir, recName, sep = "/")

for(rec in recName) {
  print(rec)
  url = paste0("https://zenodo.org/records/17575795/files/", rec, "?download=1")
  download.file(url, destfile = paste(dir, rec, sep = "/"), mode = "wb")
}

sat = soundSat(dir, backup = dir)

# Now pretend the process was interrupted (manually/your R crashed/your computer turned off)
# We get the backup file

list.files(dir)
backupDir = paste(dir, "SATBACKUP.rds", sep = "/")
```

```
# To recall the backup you simply:

satB = satBackup(backupDir)

head(satB$values)

unlink(dir, recursive = TRUE)

## End(Not run)
```

`singleSat` *Single Soundscape Saturation Index*

Description

Single Soundscape Saturation Index

Usage

```
singleSat(
  soundfile,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  DCfix = TRUE,
  powthr = 10,
  bgnthr = 0.8,
  beta = TRUE
)
```

Arguments

`soundfile`	wav package numeric matrix, tuneR package Wave object, Ruido noise.matrix object or path to a `.wav` file
`channel`	channel where the background noise values will be extracted from. Available channels are: `"stereo"`, `"mono"`, `"left"` or `"right"`. Defaults to `"stereo"`.
`timeBin`	size (in seconds) of the time bin. Set to `NULL` to use the entire audio as a single bin. Defaults to `60`
`dbThreshold`	minimum allowed value of dB for the spectrograms. Set to `NULL` to leave db values unrestricted Defaults to `-90`, as set by Towsey 2017
`dbRef`	reference value used to compute decibel values from the spectrogram.

targetSampRate	sample rate of the audios. Defaults to NULL to not change the sample rate. This argument is only used to down sample the audio.
wl	window length of the spectrogram. Defaults to 512.
window	window used to smooth the spectrogram. Defaults to hamming(wl). Switch to hanning(wl) if to use hanning instead.
overlap	overlap between the spectrogram windows. Defaults to wl/2 (half the window length)
histbreaks	breaks used to calculate Background Noise. Available breaks are: "FD", "Sturges", "scott" and 100. Defaults to "FD". Can also be set to any numerical value to limit or increase the amount of breaks.
DCfix	if the DC offset should be removed before the metrics are calculated. Defaults to TRUE
powthr	a single value to evaluate the activity matrix for Soundscape Power (in %dB). Defaults to 10.
bgnthr	a single value to evaluate the activity matrix for Background Noise (in %). Defaults to 0.8
beta	how BGN thresholds are calculated. If TRUE, BGN thresholds are computed using all recordings combined.

Details

Soundscape Saturation (SAT) quantifies the proportion of frequency bins that are acoustically active within a given time bin. It wasproposed by Burivalova et al. (2018) as a metric to evaluate the acoustic niche hypothesis.

For each time bin m, an activity matrix $a_{m,f}$ is first constructed across frequency bins f. A frequency bin is considered active if either its background level (BGN) or its soundscape power (POW) exceeds a defined threshold:

$$a_{m,f} = \begin{cases} 1, & \text{if } BGN_{m,f} > \theta_1 \text{ or } POW_{m,f} > \theta_2 \\ 0, & \text{otherwise} \end{cases}$$

where θ is a user-defined threshold applied uniformly to both BGN and POW.

Soundscape saturation for time bin m is then calculated as the proportion of active frequency bins:

$$S_m = \frac{\sum_{f=1}^{N} a_{m,f}}{N}$$

where N is the total number of frequency bins. Higher values of SAT indicate a greater fraction of the frequency spectrum being occupied by acoustic activity.

Value

A list containing the saturation values for all time bins of the inputted file

References

Burivalova, Z., Towsey, M., Boucher, T., Truskinger, A., Apelis, C., Roe, P., & Game, E. T. (2018). Using soundscapes to detect variable degrees of human influence on tropical forests in Papua New Guinea. Conservation Biology, 32(1), 205-215. https://doi.org/10.1111/cobi.12968

See Also

soundSat() and soundMat() to work with multiple audio files and activity() to get only the activity matrix.

Examples

```
# First example: Using a Ruido noise.matrix object
# We are going to load a sample noise.matrix object to demonstrate the basic usage of singleSat()
# To understand about the origin of this noise.matrix, check: ?sampleBGN
data("sampleBGN")

# View the sample noise.matrix object
sampleBGN

# Run the function
SAT = singleSat(sampleBGN)

# View the results
SAT

# Now lets plot our results to see the dynamics of soundscape saturation by minute
maxV = max(unlist(SAT))
minV = min(unlist(SAT))

plot(x = c(1, 3), y = c(minV, maxV), type = "n",
     xlab = "Minute", ylab = "Soundscape Saturation (%)", xaxt = "n")
lines(x = 1:3, SAT$left, col = "#1ECBE1", type = "b", pch = 16)
axis(1, at = 1:3)
lines(x = 1:3, SAT$right, col = "#E1341E", type = "b", pch = 16)
legend("topright", legend = c("Left", "Right"), col = c("#1ECBE1", "#E1341E"), lty = 1, pch = 16)

# Second example: Using a tuneR Wave-class object
# Lets produce an artificial audio with the tuneR package to demonstrate that
# the function can also read Wave-class objects (This is the same object used in
# the example of bgNoise!)

library(tuneR)

oldpar = par(no.readonly = TRUE)

# Define parameters for the artificial audio
samprate = 12050
dur = 59
n = samprate * dur

# White noise
```

```
set.seed(413)
noise = rnorm(n)

# Linear fade-out envelope
fade = seq(1, 0, length.out = n)

# Apply fade
signal = noise * fade

# Create Wave object
wave = Wave(
  left = signal,
  samp.rate = samprate,
  bit = 16
)

# Running singleSat() on the artificial audio
sat = singleSat(wave, timeBin = 10)

# Now we can plot the results
# In the left we have a periodogram and in the right saturation values
# along one minute
par(mfrow = c(1,2))
image(periodogram(wave, width = 8192, normalize = FALSE), xlab = "Time (s)",
ylab = "Frequency (hz)", axes = FALSE)
axis(1, labels = seq(0,60, 10), at = seq(0,7e5,length.out = 7))
axis(2)
plot(sat$mono, xlab = "Time (s)", ylab = "Soundscape Saturation (%)",
type = "b", pch = 16, axes = FALSE)
axis(1, labels = paste0(c("0-10","10-20","20-30","30-40","40-50","50-59"),
"s"), at = 1:6)
axis(2)

par(oldpar)

# Third example: Calculating activity beforehand
## You can use a noise.matrix with activity values to calculate saturation.

data("sampleBGN")
sampleBGN
singleSat(activity(sampleBGN))


# Fourth example: Reading a file directly
# Lets begin by loading an audio from the online Zenodo library and
# read it directly with the function
# Getting audiofile from the online Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
rec = paste0("GAL24576_20250401_", sprintf("%06d", 0),".wav")
recDir = paste(dir,rec , sep = "/")
url = paste0("https://zenodo.org/records/17575795/files/", rec, "?download=1")
```

```
# Downloading the file, might take some time denpending on your internet
download.file(url, destfile = recDir, mode = "wb")

# Now we calculate soundscape saturation for both sides of the recording
sat = singleSat(recDir)

# Printing the results
print(sat)

barplot(unlist(sat), col = c("darkgreen", "red"),
       names.arg = c("Left", "Right"), ylab = "Soundscape Saturation (%)")

unlink(dir, recursive = TRUE)
```

---

soundMat *Soundscape Saturation Matrix*

Description

Get the Soundscape Saturation matrix with all threshold combinations instead of the combination with the most normal distribution.

Usage

```
soundMat(
  soundpath,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  DCfix = TRUE,
  powthr = c(5, 20, 1),
  bgnthr = c(0.5, 0.9, 0.05),
  beta = TRUE,
  saveNoise = FALSE,
  backup = NULL
)
```

Arguments

soundpath	single or multiple directories to your .wav audio files

channel	channel where the saturation values will be extracted from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo".
timeBin	size (in seconds) of the time bin. Set to NULL to use the entire audio as a single bin. Defaults to 60
dbThreshold	minimum allowed value of dB for the spectrograms. Set to NULL to leave db values unrestricted Defaults to -90, as set by Towsey 2017
dbRef	reference value used to compute decibel values from the spectrogram.
targetSampRate	desired sample rate of the audios. This argument is only used to down sample the audio. If NULL, then audio's sample rate remains the same. Defaults to NULL
wl	window length of the spectrogram. Defaults to 512
window	window used to smooth the spectrogram. Switch to hanning(wl) to use hanning instead. Defaults to hamming(wl)
overlap	overlap between the spectrogram windows. Defaults to wl/2 (half the window length)
histbreaks	breaks used to calculate Background Noise. Available breaks are: "FD", "Sturges", "scott" and 100. Defaults to "FD". Can also be set to any numerical value to limit or increase the amount of breaks.
DCfix	if the DC offset should be removed before the metrics are calculated. Defaults to TRUE
powthr	numeric vector of length three containing the the range of thresholds used to evaluate the Soundscape Power of the Activity Matrix (in dB). The values correspond to the minimum threshold, maximum threshold and step size respectively. Defaults to c(5, 20, 1), which evaluates thresholds from 5 dB to 20 dB in increments of 1 dB
bgnthr	numeric vector of length three containing the the range of thresholds used to evaluate the Background Noise of the Activity Matrix (in %). The values correspond to the minimum threshold, maximum threshold and step size respectively. Defaults to c(0.5, 0.9, 0.05), which evaluates thresholds from 50% to 90% in increments of 5%
beta	how BGN thresholds are calculated. If TRUE, BGN thresholds are calculated using all recordings combined. If FALSE, BGN thresholds are calculated separately for each recording. Defaults to TRUE
saveNoise	whether the noise.matrix objects generated internally should also be included in the output. Defaults to TRUE
backup	path to save the backup. Defaults to NULL

Details

Check soundSat() to see how the indices are calculated.

Value

A list containing three objects. The first (info) contains the following variables from every audio file: PATH, AUDIO, CHANNEL, DURATION, BIN, SAMPRATE. The second (values) contains saturation values from all possible threshold combinations. The third (errors) contains the error messages and the paths to the files that returned an error during processing.

References

Burivalova, Z., Towsey, M., Boucher, T., Truskinger, A., Apelis, C., Roe, P., & Game, E. T. (2018). Using soundscapes to detect variable degrees of human influence on tropical forests in Papua New Guinea. Conservation Biology, 32(1), 205-215. https://doi.org/10.1111/cobi.12968

See Also

`soundSat()` to get only the threshold with the most normal distribution and `multActivity()` to generate only activity matrices. Also, check `satBackup()` if you are working with larger datasets and want some safety.

Examples

```
oldpar = par(no.readonly = TRUE)
### Downloading audiofiles from public Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
recName = paste0("GAL24576_20250401_", sprintf("%06d", seq(0, 200000, by = 50000)), ".wav")
recDir = paste(dir, recName, sep = "/")

for (rec in recName) {
  print(rec)
  url = paste0("https://zenodo.org/records/17575795/files/",
               rec,
               "?download=1")
  download.file(url, destfile = paste(dir, rec, sep = "/"), mode = "wb")
}

### Running the function
sat = soundMat(dir)

### Plotting results
sides = sat$info$CHANNEL

thresholds = colnames(sat$values)
split = strsplit(thresholds, "/")

shapNorm = apply(sat$values, 2, function(x)

  if (var(x) == 0) {
    0
  } else {
    shapiro.test(x)$statistic
  })

shapPos = which.max(shapNorm)

par(mfrow = c(3, 2))

plot(
  sat$values[sides == "left", 1],
  main = paste0("POW = ", split[[1]][1], "dB | BGN = ", split[[1]][2], "%"),
```

```
  type = "b",
  ylim = c(0,1),
  xlab = "Time Index", ylab = "Soundsacpe Saturation (%)", col = "goldenrod"
)
points(sat$values[sides == "right", 1], col = "maroon", type = "b")

hist(sat$values[,1], main = paste("Histogram of POW = ", split[[1]][1],
"dB | BGN = ", split[[1]][2], "%"), xlab = "Soundscape Saturation (%)")

plot(
sat$values[sides == "left", 144],
main = paste0("POW = ", split[[144]][1], "dB | BGN = ", split[[144]][2], "%"),
type = "b",
ylim = c(0,1),
xlab = "Time Index", ylab = "Soundsacpe Saturation (%)", col = "goldenrod"
)
points(sat$values[sides == "right", 144], col = "maroon", type = "b")

hist(sat$values[,144], main = paste("Histogram of POW = ", split[[144]][1],
"dB | BGN = ", split[[144]][2], "%"), xlab = "Soundscape Saturation (%)")

plot(
  sat$values[sides == "left", shapPos],
  main = paste0(
    "POW = ",
    split[[shapPos]][1],
    "dB | BGN = ",
    split[[shapPos]][2],
    "%",
    "\nshapiro.test. statistic (W): ",
    which.max(shapNorm)
  ),
  type = "b",
  ylim = c(0,1),
  xlab = "Time Index", ylab = "Soundsacpe Saturation (%)", col = "goldenrod"
)
points(sat$values[sides == "right", shapPos], col = "maroon", type = "b")
hist(sat$values[,shapPos], main = paste("Histogram of POW = ",
split[[shapPos]][1], "dB | BGN = ", split[[shapPos]][2], "%"),
xlab = "Soundscape Saturation (%)")

unlink(dir, recursive = TRUE)
par(oldpar)
```

---

soundSat *Soundscape Saturation Index*

Description

Calculate Soundscape Saturation for a combination of recordings using the methodology proposed in Burivalova 2018.

Usage

```
soundSat(
  soundpath,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  dbRef = 10,
  targetSampRate = NULL,
  wl = 512,
  window = hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  DCfix = TRUE,
  powthr = c(5, 20, 1),
  bgnthr = c(0.5, 0.9, 0.05),
  normality = "auto",
  beta = TRUE,
  saveNoise = FALSE,
  backup = NULL
)
```

Arguments

soundpath	single or multiple directories to your .wav audio files
channel	channel where the saturation values will be extracted from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo".
timeBin	size (in seconds) of the time bin. Set to NULL to use the entire audio as a single bin. Defaults to 60
dbThreshold	minimum allowed value of dB for the spectrograms. Set to NULL to leave db values unrestricted Defaults to -90, as set by Towsey 2017
dbRef	reference value used to compute decibel values from the spectrogram.
targetSampRate	desired sample rate of the audios. This argument is only used to down sample the audio. If NULL, then audio's sample rate remains the same. Defaults to NULL
wl	window length of the spectrogram. Defaults to 512
window	window used to smooth the spectrogram. Switch to hanning(wl) to use hanning instead. Defaults to hamming(wl)
overlap	overlap between the spectrogram windows. Defaults to wl/2 (half the window length)
histbreaks	breaks used to calculate Background Noise. Available breaks are: "FD", "Sturges", "scott" and 100. Defaults to "FD". Can also be set to any numerical value to limit or increase the amount of breaks.

DCfix	if the DC offset should be removed before the metrics are calculated. Defaults to TRUE
powthr	numeric vector of length three containing the the range of thresholds used to evaluate the Soundscape Power of the Activity Matrix (in dB). The values correspond to the minimum threshold, maximum threshold and step size respectively. Defaults to c(5, 20, 1), which evaluates thresholds from 5 dB to 20 dB in increments of 1 dB
bgnthr	numeric vector of length three containing the the range of thresholds used to evaluate the Background Noise of the Activity Matrix (in %). The values correspond to the minimum threshold, maximum threshold and step size respectively. Defaults to c(0.5, 0.9, 0.05), which evaluates thresholds from 50% to 90% in increments of 5%
normality	character string with the normality test used to determine which threshold combination has the most normal distribution of values. One of "auto", "shapiro.test", "sf.test", "lillie.test", "ad.test", "cvm.test" or "pearson.test". Defaults to "auto". "ks.test" is not available. Check details for more information.
beta	how BGN thresholds are calculated. If TRUE, BGN thresholds are calculated using all recordings combined. If FALSE, BGN thresholds are calculated separately for each recording. Defaults to TRUE
saveNoise	whether the noise.matrix objects generated internally should also be included in the output. Defaults to TRUE
backup	path to save the backup. Defaults to NULL

Details

Soundscape Saturation (SAT) quantifies the proportion of frequency bins that are acoustically active within a given time bin. It waspropposed by Burivalova et al. (2018) as a metric to evaluate the acoustic niche hypothesis.

For each time bin m, an activity matrix $a_{m,f}$ is first constructed across frequency bins f. A frequency bin is considered active if either its background level (BGN) or its soundscape power (POW) exceeds a defined threshold:

$$a_{m,f} = \begin{cases} 1, & \text{if } BGN_{m,f} > \theta_1 \text{ or } POW_{m,f} > \theta_2 \\ 0, & \text{otherwise} \end{cases}$$

where θ is a user-defined threshold applied uniformly to both BGN and POW.

Soundscape saturation for time bin m is then calculated as the proportion of active frequency bins:

$$S_m = \frac{\sum_{f=1}^{N} a_{m,f}}{N}$$

where N is the total number of frequency bins. Higher values of SAT indicate a greater fraction of the frequency spectrum being occupied by acoustic activity.

After computing S_m, the function evaluates all tested threshold values and selects the one that yields the most normally distributed set of saturation values across time bins. Normality is assessed to identify a threshold that best stabilizes the distribution of SAT.

If backup is set to a valid directory, a file named "SATBACKUP.rds" is automatically saved after every batch of five processed files. This file stores the current processing state and allows interrupted runs (e.g., due to manual termination, session crashes, or system shutdowns) to be resumed using satBackup().

To resume processing, pass the saved file (e.g., "path/SATBACKUP.rds") to satBackup(). Once a backup has been created, all original arguments and file paths must remain unchanged, unless they are explicitly modified within the saved .rds object.

For the normality argument, each test only works within a range of sample sizes (3-5000 for "shapiro.test", 5-5000 for "sf.test", 5 or more for "lillie.test", 8 or more for "ad.test" and "cvm.test", 2 or more for "pearson.test"). If the chosen test does not fit the number of values, a warning is issued and "shapiro.test" is used (or "ad.test" if there are more than 5000 values). With "auto", "shapiro.test" is used for up to 5000 values and "ad.test" for more.

Value

A list containing five objects. The first and second objects (powthresh and bgnthresh) are the threshold values that yielded the most normal distribution of saturation values using the normality test set by the user. The third (normality) contains the statistics values of the normality test that yielded the most normal distribution. The fourth object (values) contains a data.frame with the values of saturation for each bin of each recording and the size of the bin in seconds. The fifth contains a data.frame with errors that occurred with specific files during the function.

References

Burivalova, Z., Towsey, M., Boucher, T., Truskinger, A., Apelis, C., Roe, P., & Game, E. T. (2018). Using soundscapes to detect variable degrees of human influence on tropical forests in Papua New Guinea. Conservation Biology, 32(1), 205-215. https://doi.org/10.1111/cobi.12968

See Also

soundMat() to get saturation for ALL thresholds and multActivity() to get only activity values. Also, check satBackup() if you are working with bigger datasets.

Examples

```
### Downloading audiofiles from public Zenodo library
dir = paste(tempdir(), "forExample", sep = "/")
dir.create(dir)
recName = paste0("GAL24576_20250401_", sprintf("%06d", seq(0, 200000, by = 50000)),".wav")
recDir = paste(dir, recName, sep = "/")

for(rec in recDir) {
 print(rec)
 url = paste0("https://zenodo.org/records/17575795/files/", basename(rec), "?download=1")
 download.file(url, destfile = rec, mode = "wb")
}
```

```
### Running the function
sat = soundSat(dir)

### Preparing the plot
timeSplit = strsplit(sat$values$AUDIO, "_")
sides = sat$values$CHANNEL
date = sapply(timeSplit, function(x)
  x[2])
time = sapply(timeSplit, function(x)
  substr(x[3],1,6))
datePos = paste(substr(date,1,4), substr(date,5,6), substr(date,7,8), sep = "-")
timePos = paste(substr(time,1,2), substr(time,3,4), substr(time,5,6), sep = ":")
dateTime = as.POSIXct(paste(datePos, timePos), format = "%Y-%m-%d %H:%M:%OS")
leftEar = data.frame(SAT = sat$values$SAT[sides == "left"], HOUR = dateTime[sides == "left"])
rightEar = data.frame(SAT = sat$values$SAT[sides == "right"], HOUR = dateTime[sides == "right"])

### Plotting results

plot(SAT~HOUR, data = leftEar, ylim = c(range(sat$values$SAT)),
col = "darkgreen", pch = 16,
     ylab = "Soundscape Saturation (%)", xlab = "Time of Day", type = "b")
points(SAT~HOUR, data = rightEar, ylim = c(range(sat$values$SAT)),
col = "red", pch = 16, type = "b")
legend("bottomright", legend = c("Left Ear", "Right Ear"),
       col = c("darkgreen", "red"), lty = 1)

unlink(dir, recursive = TRUE)
```

Index

∗ **datasets**
sampleBGN, 27

ACIspec, 2
ACIspec(), *12*, *15*, *17*
activity, 5, *27*
activity(), *12*, *22*, *31*

base::plot, *25, 26*
bgn, 7
bgn(), *12*, *17*
bgNoise, 10, *23*, *25*, *27*
bgNoise(), *4*, *9*, *15*, *17*, *24*

ENTspec, 13
ENTspec(), *4*, *12*
EVNspec, 16

graphical parameters, *25, 26*

hamming, 18
hanning, 19

multActivity, 20
multActivity(), *6*, *28*, *35*, *39*

noise.matrix, *4*, *11*, *15*, *17*
noise.matrix-class, 23

plot(), *23*
plot,noise.matrix,ANY-method, 24
plotNOISE, 25

sampleBGN, 27
satBackup, 28
satBackup(), *21, 22*, *35*, *39*
show(), *23*
singleSat, *27*, 29
singleSat(), *6*, *12*
soundMat, 33
soundMat(), *22*, *28*, *31*, *39*
soundSat, *23*, 36
soundSat(), *22*, *28*, *31*, *34, 35*