Package 'Ruido'

December 11, 2025

Title Soundscape Background Noise, Power, and Saturation

Version 1.0.0

Description Accessible and flexible implementation of three ecoacoustic indices that are less commonly available in existing R frameworks: Background Noise, Soundscape Power and Soundscape Saturation. 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. For more details about the indices read Towsey (2014) <doi:10.1016/j.procs.2014.05.063> and Burivalova (2017) <doi:10.1111/cobi.12968>.

Depends R(>= 4.3.0)

Imports methods, tuneR, signal, nortest

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests ggplot2, spelling

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

NeedsCompilation no

Author Arthur Igor da Fonseca-Freire [aut, cre, cph],
Weslley Geremias dos Santos [ctb],
Lucas Rodriguez Forti [ctb]

Repository CRAN

Date/Publication 2025-12-11 13:30:19 UTC

Contents

`bgNoise` *Background Noise and Soundscape Power Index*

Description

Compute the Background Noise and Soundscape Power values of an audio using Towsey 2017 methodology

Usage

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

Arguments

`soundfile`	tuneR Wave object or path to a valid audio
`channel`	channel where the background noise values will be extract from. Available channels are: `"stereo"`, `"mono"`, `"left"` or `"right"`. Defaults to `"stereo"`.
`timeBin`	size (in seconds) of the time bin. Defaults to `60`.
`dbThreshold`	minimum allowed value of dB for the spectrograms. Defaults to `-90`, as set by Towsey 2017.
`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 `signal::hammning(wl)`. Switch to `signal::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 number to limit or increase the amount of breaks.

Details

Background Noise (`BGN`) is an index that measures the most common continuous baseline level of acoustic energy in a frequency window and in a time bin. It was developed by Towsey 2017 using the Lamel et al 1981 algorithm. is calculated by taking the modal value of intensity values in temporal bin c in frequency window f:

$$BGN_f = mode(dB_{cf})$$

Soundscape Power represents a measure of signal-to-noise ratio. It measures the relation of BGN to the loudest intensities in temporal bin c in frequency window f:

$$POW_f = max(dB_{cf}) - BGN_{cf}$$

Value

A list containing three objects: The first and second one contains a matrix with the values of Background Noise and Soundscape Power respectively to each time bin and for each frequency window of your soundfile. The third object is the duration in second of your time bins.

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

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 <- 59
sampN <- sampRate * dur

# Then we Ggenerate the white noise for our audio and apply FFT
set.seed(413)
ruido <- rnorm(sampN)
spec <- fft(ruido)
```

```
# Now we create a random spectral envelope for the audio and apply the spectral envelope
nPoints <- 50
env <- runif(nPoints)
env <- approx(env, n=nPoints)$y
specMod <- spec * env

# Now we invert the FFT back to into a time waveform and normalize and convert to Wave
out <- Re(fft(specMod, inverse=TRUE)) / sampN
wave <- Wave(left = out, samp.rate = sampRate, bit = 16)
wave <- normalize(wave, unit = "16")

# Here's our artificial audio
wave

# Running the bgNoise function with all the default arguments
bgn <- bgNoise(wave)

# Print the results
head(bgn$left$BGN)
head(bgn$left$POW)

# Plotting background noise and soundscape profile for the first minute of the recording
par(mfrow = c(2,2))
plot(x = bgn$left$BGN$BGN1, y = seq(1,bgn$sampRate, length.out = 256),
     xlab = "Background Noise (dB)", ylab = "Frequency (hz)",
     main = "BGN by Frequency",
     type = "l")
plot(x = bgn$left$POW$POW1, y = seq(1,bgn$sampRate, length.out = 256),
     xlab = "Soundscape Power (dB)", ylab = "Frequency (hz)",
     main = "POW by Frequency",
     type = "l")
plot(bgn$left$BGN$BGN1~bgn$left$POW$POW1, pch = 16,
     xlab = "Soundscape Power (dB)", ylab = "Background Noise (dB)",
     main = "BGN~POW in left ear")
hist(bgn$left$BGN$BGN1, main = "Histogram of BGN distribution",
      xlab = "Background Noise (BGN)")


  oldpar <- par(no.readonly = TRUE)
  ### This is a secondary example using audio from a real soundscape
  ### These audios are originated from the Escutad\^o Project
  # Getting audiofile from the online Zenodo library
  dir <- tempdir()
  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)
```

```
# Print the results
head(bgn$left$BGN)
head(bgn$left$POW)

# Plotting background noise and soundscape profile for the first minute of the recording
par(mfrow = c(2,2))
plot(x = bgn$left$BGN$BGN1, y = seq(1,bgn$sampRate, length.out = 256),
     xlab = "Background Noise (dB)", ylab = "Frequency (hz)",
     main = "BGN by Frequency",
     type = "l")
plot(x = bgn$left$POW$POW1, y = seq(1,bgn$sampRate, length.out = 256),
     xlab = "Soundscape Power (dB)", ylab = "Frequency (hz)",
     main = "POW by Frequency",
     type = "l")
plot(bgn$left$BGN$BGN1~bgn$left$POW$POW1, pch = 16,
     xlab = "Soundscape Power (dB)", ylab = "Background Noise (dB)",
     main = "BGN~POW in left ear")
plot(bgn$right$BGN$BGN1~bgn$right$POW$POW1, pch = 16,
     xlab = "Soundscape Power (dB)", ylab = "Background Noise (dB)",
     main = "BGN~POW in right ear")

unlink(recDir)
par(oldpar)
```

`satBackup` *Backup for Soundscape Saturation Index*

Description

This function is a way to continue an unfinished process of the `soundSat()` function through a backup file. Arguments can't be inputted nor changed since the function will automatically load them from the original `soundSat()` run.

Usage

```
satBackup(backupPath, od)
```

Arguments

`backupPath`	path you set in your backup in the `soundSat()` function. Audiofiles already finished will be drawn from this path
`od`	path or paths containing your original audiofiles

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 statitics values of the normality test that yielded the most normal distribution. The fourth object (values) contains a data.frame with the 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.

Examples

```
## Not run:
# It's impossible to create a functioning example since you would have to manually stop the process
# 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 <- tempdir()
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 = "C:/Users/OAS/Desktop/Arthur/17243660/BACK", wl = 256)

# Now pretend the process was interrupted (manually/your R crashed/your computer turned off)
# To recall the backup you simply:

satB <- satBackup("C:/Users/OAS/Desktop/Arthur/17243660/BACK", dir)

unlink(recDir)

## End(Not run)
```

`singleSat` *Single Soundscape Saturation Index*

Description

Single Soundscape Saturation Index

Usage

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

```
  timeBin = 60,
  dbThreshold = -90,
  targetSampRate = NULL,
  wl = 512,
  window = signal::hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  powthr = 10,
  bgnthr = 0.8
)
```

Arguments

`soundfile`	tuneR Wave object or path to a valid audio
`channel`	channel where the background noise values will be extract from. Available channels are: `"stereo"`, `"mono"`, `"left"` or `"right"`. Defaults to `"stereo"`.
`timeBin`	size (in seconds) of the time bin. Defaults to `60`.
`dbThreshold`	minimum allowed value of dB for the spectrograms. Defaults to `-90`, as set by Towsey 2017.
`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 `signal::hammning(wl)`. Switch to `signal::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 number to limit or increase the amount of breaks.
`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`

Details

Soundscape Saturation (`SAT`) is a measure of the proportion of frequency bins that are acoustically active in a determined window of time. It was developed by Burivalova et al. 2017 as an index to test the acoustic niche hypothesis. To calculate this function, first we need to generate an activity matrix for each time bin of your recording with the following formula:

$$a_{mf} = 1 \; if (BGN_{mf} > \theta_1) \; or \; (POW_{mf} > \theta_2); \; otherwise, \; a_{mf} = 0,$$

Where θ_1 is the threshold of BGN values and θ_2 is a threshold of dB values. Since we define a single threshold for both in this function, we don't have to worry about generating a saturation value for

many different combinations. For the selected threshold a soundscape saturation measure will be taken with the following formula:

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

Since this is analyzing the soundscape saturaion of a single file, no normality tests will be done.

Value

A data frame containing the saturation values for all time bins of the inputed file

References

Burivalova, Z., Towsey, M., Boucher, T., Truskinger, A., Apelis, C., Roe, P., & Game, E. T. (2017). 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

Examples

```
### Generating an artificial audio for the example
## For this example we'll generate a sweep in a noisy soundscape
library(tuneR)

# Define the audio sample rate, duration and number of samples
samprate <- 12050
dur <- 59

# Create a time vector
t <- seq(0, dur, by = 1/samprate)

# It starts at the frequency 50hz and goes all the way up to 4000hz
# The sweep is exponnential
freqT <- 50 * (4000/50)^(t/dur)

# Generate the signal
signal1 <- sin(10 * pi * cumsum(freqT) / samprate)
# We create an envelope to give the sweep a fade away
envelope <- exp(-4 * t / dur)
# Generating low noise for the audio
set.seed(413)
noise <- rnorm(length(t), sd = 0.3)
# Adding everything together in our signal
signal <- signal1 * envelope + noise

# Normalize to 16-bit WAV range to create our Wave object
signalNorm <- signal / max(abs(signal))
wave_obj <- Wave(left = signalNorm, samp.rate = samprate, bit = 16)

# Now we calculate soundscape saturation for our audio
# Here we are using timeBin = 10 so we get Soundscape Saturation values
# every 10 seconds on the audio
```

```
SAT <- singleSat(wave_obj, timeBin = 10)

# Now we can plot the results
# In the left we have a periodogram and in the right saturaion values
# along one minute
par(mfrow = c(1,2))
image(periodogram(wave_obj, width = 64), xlab = "Time (s)",
ylab = "Frequency (hz)", log = "y", axes = FALSE)
axis(1, labels = seq(0,60, 10), at = seq(0,7e5,length.out = 7))
axis(2)
plot(SAT$left, 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)



oldpar <- par(no.readonly = TRUE)

# Getting audiofile from the online Zenodo library
dir <- tempdir()
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, wl = 256)

# Printing the results
print(sat)

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

unlink(recDir)
par(oldpar)
```

---

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,
  targetSampRate = NULL,
  wl = 512,
  window = signal::hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  powthr = c(5, 20, 1),
  bgnthr = c(0.5, 0.9, 0.05),
  backup = NULL
)
```

Arguments

soundpath	single directory or multiple directory to audio files. The directory must lead to a single folder or a combination of folders.
channel	channel where the saturation values will be extract from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo".
timeBin	size (in seconds) of the time bin. Defaults to 60.
dbThreshold	minimum allowed value of dB for the spectrograms. Defaults to -90, as set by Towsey.
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 signal::hammning(wl). Switch to signal::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 number to limit or increase the amount of breaks.
powthr	vector of values to evaluate the activity matrix for Soundscape Power (in dB). The first value corresponds to the lowest dB value and the second corresponds to the highest, the third value is the step. Defaults to c(5, 20, 1), which means the first thresholds starts at 5dB and jumps a whole number till 20dB.
bgnthr	vector of values to evaluate the activity matrix for Background Noise (in %). The first value corresponds to the lowest quantile value and the second corresponds to the highest, the third value is the step. Defaults to c(0.5, 0.9, 0.05), which means the first thresholds starts at 50% and jumps 5% till 90%.

backup — path to backup the saturation values in case the computer is turned off during processing or in case you cannot be sure the computer will be on for the entire process. Defaults to NULL. The backup will be updated every 5 recordings processed.

Details

Soundscape Saturation (SAT) is a measure of the proportion of frequency bins that are acoustically active in a determined window of time. It was developed by Burivalova et al. 2017 as an index to test the acoustic niche hypothesis. To calculate this function, first we need to generate an activity matrix for each time bin of your recording with the following formula:

$$a_{mf} = 1 \; if(BGN_{mf} > \theta_1) \; or \; (POW_{mf} > \theta_2); \; otherwise, \; a_{mf} = 0,$$

Where θ_1 is the threshold of BGN values and θ_2 is a threshold of dB values. Since we define a interval for both the threshold, this means that an activity matrix will be generated for each bin of each recording. For each combination of threshold a SAT measure will be taken with the following formula:

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

After these equations are done, we check every threshold combination for normality and pick the combination that yields the most normal distribution of saturation values.

If you set a path for the "path" argument, a single rds file will be written in your path. This objects can be loaded again through the "satBack" function to continue the calculation of saturation in case the process is suddenly stopped.

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 statitics values of the normality test that yielded the most normal distribution. The fourth object (values) contains a data.frame with the 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. (2017). 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

Examples

```
oldpar <- par(no.readonly = TRUE)
### Downloading audiofiles from public Zenodo library
dir <- tempdir()
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 <- ifelse(grepl("left", sat$info$BIN), "left", "right")

thresholds <- colnames(sat$matrix)
split <- strsplit(thresholds, "/")

shapNorm <- apply(sat$matrix, 2, function(x)

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

shapPos <- which.max(shapNorm)

par(mfrow = c(3, 2))

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

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

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

hist(sat$matrix[,144], main = paste("Histogram of POW = ", split[[144]][1],
```

```
"dB | BGN = ", split[[144]][2], "%"), xlab = "Soundscape Saturation (%)")

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

unlink(recDir)
par(oldpar)
```

---

soundSat *Soundscape Saturation Index*

Description

Calculate Soundscape Saturation for a combination of recordings using Burivalova 2018 methodology

Usage

```
soundSat(
  soundpath,
  channel = "stereo",
  timeBin = 60,
  dbThreshold = -90,
  targetSampRate = NULL,
  wl = 512,
  window = signal::hamming(wl),
  overlap = ceiling(length(window)/2),
  histbreaks = "FD",
  powthr = c(5, 20, 1),
  bgnthr = c(0.5, 0.9, 0.05),
  normality = "ad.test",
```

```
  backup = NULL
)
```

Arguments

soundpath	single directory or multiple directory to audio files. The directory must lead to a single folder or a combination of folders.
channel	channel where the saturation values will be extract from. Available channels are: "stereo", "mono", "left" or "right". Defaults to "stereo".
timeBin	size (in seconds) of the time bin. Defaults to 60.
dbThreshold	minimum allowed value of dB for the spectrograms. Defaults to -90, as set by Towsey.
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 signal::hammning(wl). Switch to signal::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 number to limit or increase the amount of breaks.
powthr	vector of values to evaluate the activity matrix for Soundscape Power (in dB). The first value corresponds to the lowest dB value and the second corresponds to the highest, the third value is the step. Defaults to c(5, 20, 1), which means the first thresholds starts at 5dB and jumps a whole number till 20dB.
bgnthr	vector of values to evaluate the activity matrix for Background Noise (in %). The first value corresponds to the lowest quantile value and the second corresponds to the highest, the third value is the step. Defaults to c(0.5, 0.9, 0.05), which means the first thresholds starts at 50% and jumps 5% till 90%.
normality	normality test to determine which threshold combination has the most normal distribution of values. We recommend to pick any test from the nortest package. Input the test as a character. Defaults to "ad.test". "ks.test" is not available. "shapiro.test" can be used, however we recommend using only when analyzing very few recordings.
backup	path to backup the saturation values in case the computer is turned off during processing or in case you cannot be sure the computer will be on for the entire process. Defaults to NULL. The backup will be updated every 5 recordings processed.

Details

Soundscape Saturation (SAT) is a measure of the proportion of frequency bins that are acoustically active in a determined window of time. It was developed by Burivalova et al. 2017 as an index to test the acoustic niche hypothesis. To calculate this function, first we need to generate an activity matrix for each time bin of your recording with the following formula:

$$a_{mf} = 1\ if(BGN_{mf} > \theta_1)\ or\ (POW_{mf} > \theta_2);\ otherwise,\ a_{mf} = 0,$$

Where θ_1 is the threshold of BGN values and θ_2 is a threshold of dB values. Since we define a interval for both the threshold, this means that an activity matrix will be generated for each bin of each recording. For each combination of threshold a SAT measure will be taken with the following formula:

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

After these equations are done, we check every threshold combination for normality and pick the combination that yields the most normal distribution of saturation values.

If you set a path for the "path" argument, a single rds file will be written in your path. This objects can be loaded again through the "satBack" function to continue the calculation of saturation in case the process is suddenly stopped.

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 statitics values of the normality test that yielded the most normal distribution. The fourth object (values) contains a data.frame with the 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. (2017). 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

Examples

```
### Downloading audiofiles from public Zenodo library
dir <- tempdir()
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 <- soundSat(dir, wl = 256)

### Preparing the plot
timeSplit <- strsplit(sat$values$AUDIO, "_")
sides <- ifelse(grepl("left", sat$values$BIN), "left", "right")
```

```
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("topright", legend = c("Left Ear", "Right Ear"),
       col = c("darkgreen", "red"), lty = 1)

unlink(recDir)
```

Index