

# Package ‘Ruido’

May 7, 2026

**Title** Soundscape Background Noise, Power, and Saturation

**Version** 1.0.3

**Date** 2026-4-18

**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 (2017) <<https://eprints.qut.edu.au/110634/>> and Burivalova (2017) <[doi:10.1111/cobi.12968](https://doi.org/10.1111/cobi.12968)>.

**Depends** R(>= 4.3.0)

**Imports** methods, tuneR, signal, noratest

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Suggests** ggplot2, patchwork, testthat (>= 3.0.0)

**Maintainer** Arthur Igor da Fonseca-Freire <[arthur.igorr@gmail.com](mailto: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

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2026-04-18 15:20:02 UTC

## Contents

activity	2
bgNoise	4
multActivity	7
noise.matrix-class	11
plot,noise.matrix,ANY-method	12
plotBGN	13
sampleBGN	14
satBackup	15
singleSat	16
soundMat	20
soundSat	24
<b>Index</b>	<b>28</b>

---

activity	<i>Acoustic Activity Matrix</i>
----------	---------------------------------

---

### Description

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

### Usage

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

### Arguments

soundfile	tuneR Wave object or path to a valid audio
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. 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
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 <code>signal::hanning(wl)</code> to use hanning instead. Defaults to <code>signal::hamming(wl)</code>
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:

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

Where  $\theta_1$  equals the threshold of BGN values and  $\theta_2$  equals the threshold of dB values. 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., Trusking, 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>

**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
  sat <- activity(sampleBGN)

  # Now we can plot the results for the left channel
  satLeft <- sat[,1:3]
  satDim <- dim(satLeft)
  numericTime <- seq(0, sum(sampleBGN@timeBins), by = sampleBGN@timeBins[1])
  labels <- paste0(numericTime[-length(numericTime)], "-", numericTime[-1], "s")

  satDF <- data.frame(BIN = rep(paste0("BIN", seq(satDim[2])), each = satDim[1]),
                     WIN = rep(seq(satDim[1]), satDim[2]),
                     ACT = factor(c(sat), levels = c(0,1)))

  ggplot(satDF, 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"))
}

```

---

bgNoise

*Background Noise and Soundscape Power Index*


---

**Description**

Calculate the 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,

```

```

    targetSampRate = NULL,
    wl = 512,
    window = signal::hamming(wl),
    overlap = ceiling(length(window)/2),
    histbreaks = "FD",
    DCfix = TRUE
)

```

### Arguments

soundfile	tuneR Wave object or path to a valid audio file
channel	channel where the metric values will be extract 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
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 signal::hanning(wl) to use hanning instead. Defaults to signal::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 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. The metric is calculated by taking the modal value of intensity values in temporal bin  $c$  in frequency window  $f$  of a recording:

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

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  $f$ :

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

This mean we'll have a value of BGN and POW to each frequency window of a recording.

**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>

**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 depending 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")

```

---

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,
  targetSampRate = NULL,
  w1 = 512,

```

```

window = signal::hamming(wl),
overlap = ceiling(length(window)/2),
histbreaks = "FD",
DCfix = TRUE,
powthr = 10,
bgnthr = 0.8,
beta = TRUE,
backup = NULL
)

```

### Arguments

soundpath	single or multiple directories to your audio files
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. 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
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 signal::hanning(wl) to use hanning instead. Defaults to signal::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
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:



```

        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

`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 calculate 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 = c("BGN", "POW"),
  nbreaks = 5,
  yunit = c("hz", "khz"),
  main = NULL,
  xlab = "dB",
  ylab = "Frequency",
  col = c("blue", "red"),
  type = "p",
  draw0 = TRUE,
  box = TRUE,
  axes = TRUE,
  annotate = TRUE,
  ...
)
```

## Arguments

x	an noise.matrix object generated by <a href="#">bgNoise</a>
channel	channel or channels to be plotted. 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 indices to be plotted. Available indices are: BGN and POW. Defaults to c("BGN", "POW")
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. Defaults to "dB"

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 <a href="#">base::plot</a>
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 <a href="#">graphical parameters</a> 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.

---

plotBGN	<i>Helper function to plot noise.matrix objects</i>
---------	---

---

### Description

Helper function to plot `noise.matrix` objects

### Usage

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

**Arguments**

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

**Details**

This is a helper function called by `plot, Wave, missing-method`. This is not intended to be seen or used 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 4'11" inches 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 `.RData` file. However you may manually change them by editing the file (not recommended).

**Usage**

```
satBackup(backup)
```

**Arguments**

backup            path to the `.RData` 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.RData", 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,
targetSampRate = NULL,
wl = 512,
window = signal::hamming(wl),
overlap = ceiling(length(window)/2),
histbreaks = "FD",
DCfix = TRUE,
powthr = 10,
bgnthr = 0.8,
beta = TRUE
)

```

### Arguments

soundfile	tuneR Wave object, Ruido noise.matrix 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. 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
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::hamming(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 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) 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. 2018 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:

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

Where  $\theta_1$  is the threshold of BGN values and  $\theta_2$  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:

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

### 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: 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 depending 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,
  targetSampRate = NULL,
  wl = 512,
  window = signal::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,

```

```

    backup = NULL
  )

```

### Arguments

soundpath	single or multiple directories to your audio files
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. 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
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 <code>signal::hanning(wl)</code> to use hanning instead. Defaults to <code>signal::hamming(wl)</code>
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 <code>c(5, 20, 1)</code> , 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 <code>c(0.5, 0.9, 0.05)</code> , 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
backup	path to save the backup. Defaults to NULL

### Details

Check [soundSat\(\)](#) to see how the indices are calculated.

If backup is set to a valid directory, a file named "SATBACKUP.RData" is saved after every batch of five processed files. If the function execution is interrupted (e.g., manual termination, an R session crash, or a system shutdown), this backup file can be passed to `satBackup()` (e.g., as `~path/SATBACKUP.RData`) to resume the original process. Once a backup is created, all arguments and file paths must remain unchanged, unless they are manually modified within the .RData object.

**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$values[sides == "left", 1],
  main = paste0("POW = ", split[[1]][1], "dB | BGN = ", split[[1]][2], "%"),
  type = "b",
  ylim = c(0,1),
  xlab = "Time Index", ylab = "Soundscape Saturation (%)", col = "goldenrod"
)
points(sat$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$values[sides == "left", 144],
  main = paste0("POW = ", split[[144]][1], "dB | BGN = ", split[[144]][2], "%"),
  type = "b",
  ylim = c(0,1),
  xlab = "Time Index", ylab = "Soundscape Saturation (%)", col = "goldenrod"
)
points(sat$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$values[sides == "left", shapPos],
  main = paste0(
    "POW = ",
    split[[shapPos]][1],
    "dB | BGN = ",
    split[[shapPos]][2],
    "%",
    "\nshapiro.test. statistic (W): ",
    which.max(shapNorm)
  ),
  type = "b",
  ylim = c(0,1),
  xlab = "Time Index", ylab = "Soundscape Saturation (%)", col = "goldenrod"
)
points(sat$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,
  targetSampRate = NULL,
  wl = 512,
  window = signal::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 = "ad.test",
  beta = TRUE,
  backup = NULL
)

```

### Arguments

soundpath	single or multiple directories to your audio files
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. 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
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 signal::hanning(wl) to use hanning instead. Defaults to signal::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 containing the normality test used to determine which threshold combination has the most normal distribution of values. We recommend to pick any test from the nortest package. Defaults to "ad.test". "ks.test" is not available. "shapiro.test" can be used, however we recommend using only when analyzing very few recordings
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
backup	path to save the backup. Defaults to NULL

## 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. 2018 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:

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

Where  $\theta_1$  is the threshold of BGN values and  $\theta_2$  is a threshold of dB values. Since we define an 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 backup is set to a valid directory, a file named "SATBACKUP.RData" is saved after every batch of five processed files. If the function execution is interrupted (e.g., manual termination, an R

session crash, or a system shutdown), this backup file can be passed to `satBackup()` (e.g., as `~path/SATBACKUP.RData`) to resume the original process. Once a backup is created, all arguments and file paths must remain unchanged, unless they are manually modified within the `.RData` object.

## Value

A list containing five objects. The first and second objects (`powthresh` and `bgntresh`) 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., Trusking, 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, 14

activity, 2, 14

activity(), 18

base::plot, 13, 14

bgNoise, 4, 11–14

bgNoise(), 11

graphical parameters, 13, 14

multActivity, 7

multActivity(), 15, 22, 26

noise.matrix, 6

noise.matrix-class, 11

plot(), 11

plot, noise.matrix, ANY-method, 12

plotBGN, 13

sampleBGN, 14

satBackup, 15

satBackup(), 9, 22, 26

show(), 11

singleSat, 14, 16

soundMat, 20

soundMat(), 9, 15, 18, 26

soundSat, 11, 24

soundSat(), 9, 15, 18, 21, 22