Package ‘ohun’

August 23, 2021

Type Package

Title Automatic detection of acoustic signals

Version 0.1.0

Maintainer Marcelo Araya-Salas <marcelo.araya@ucr.ac.cr>

Description
Facilitates the automatic detection of acoustic signals, providing functions to diagnose and optimize detection routines. Detections from other software can also be explored and optimized.

License GPL (>= 2)

Encoding UTF-8

URL https://marce10.github.io/ohun/

BugReports https://github.com/maRce10/ohun/issues/

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.1.1

Repository CRAN

Language en-US

Date/Publication 2016-04-19 08:12:11

Imports pbapply, crayon, methods, stats, utils, seewave (>= 2.0.1), RCurl, fftw, knitr, rjson

Depends R (>= 3.2.1), tuneR, warbleR

R topics documented:

`diagnose_detection` *Evaluate the performance of a signal detection procedure*

Description

`diagnose_detection` evaluates the performance of a signal detection procedure comparing the output selection table to a reference selection table

Usage

```
diagnose_detection(reference, detection, by.sound.file = FALSE,
time.diagnostics = FALSE)
```

Arguments

`reference` Data frame or 'selection.table' with the reference selections (start and end of the signals) that will be used to evaluate the performance of the detection, represented by those selections in 'detection'. Must contained at least the following columns: "sound.files", "selec", "start" and "end".

`detection` Data frame or 'selection.table' with the detections (start and end of the signals) that will be compared against the 'reference' selections. Must contained at least the following columns: "sound.files", "selec", "start" and "end".

`by.sound.file` Logical argument to control whether performance diagnostics are summarized across sound files (when `by.sound.file` = `FALSE`, when more than 1 sound file is included in 'reference') or shown separated by sound file. Default is `FALSE`.

`time.diagnostics`
Logical argument to control if diagnostics related to the duration of the signals ("mean.duration.true.positives", "mean.duration.false.positives", "mean.duration.false.negatives" and "proportional.duration.true.positives") are returned (if `TRUE`). Default is `FALSE`.

Details

The function evaluates the performance of a signal detection procedure by comparing its output selection table to a reference selection table in which all signals of interest have been selected.

Value

A data frame including the following detection performance diagnostics:

- `true.positives`: number of detections that correspond to signals referenced in 'reference'. Matching is defined as some degree of overlap in time. In a perfect detection routine it should be equal to the number of rows in 'reference'.
- `false.positives`: number of detections that don't match any of the signals referenced in 'reference'. In a perfect detection routine it should be 0.
- `false.negatives`: number of signals in 'reference' that were not detected (not found in 'detection'. In a perfect detection routine it should be 0.

- split.positives: number of signals referenced in 'reference' that were overlapped by more than 1 detection (i.e. detections that were split). In a perfect detection routine it should be 0.
- mean.duration.true.positives: mean duration of true positives (in s). Only included when time.diagnostics = TRUE.
- mean.duration.false.positives: mean duration of false positives (in s). Only included when time.diagnostics = TRUE.
- mean.duration.false.negatives: mean duration of false negatives (in s). Only included when time.diagnostics = TRUE.
- proportional.duration.true.positives: ratio of total duration of true positives to the total duration of signals referenced in 'reference'. In a perfect detection routine it should be 1. Only included when time.diagnostics = TRUE.
- sensitivity: Proportion of signals referenced in 'reference' that were detected. In a perfect detection routine it should be 1.
- specificity: Proportion of detections that correspond to signals referenced in 'reference' that were detected. In a perfect detection routine it should be 1.

Author(s)

Marcelo Araya-Salas <marcelo.araya@ucr.ac.cr>)

References

Araya-Salas, M. (2021), ohun: automatic detection of acoustic signals. R package version 0.1.0.

See Also

optimize_auto_detec, optimize_find_peaks

Examples

```
{
# perfect detection
diagnose_detection(reference = lbh_selec_reference, detection = lbh_selec_reference)

# missing one in detection
diagnose_detection(reference = lbh_selec_reference, detection = lbh_selec_reference[-1, ])

# an extra one in detection
diagnose_detection(reference = lbh_selec_reference[-1, ], detection = lbh_selec_reference)

# with time diagnostics
diagnose_detection(reference = lbh_selec_reference[-1, ],
detection = lbh_selec_reference, time.diagnostics = TRUE)

# and extra sound file in reference
diagnose_detection(reference = lbh_selec_reference,
detection =
lbh_selec_reference[lbh_selec_reference$sound.files != "Phae.long1.wav", ])
# and extra sound file in detection
diagnose_detection(reference =
lbh_selec_reference[lbh_selec_reference$sound.files != "Phae.long1.wav", ],
detection = lbh_selec_reference)

# and extra sound file in detection by sound file
dd <- diagnose_detection(reference =
lbh_selec_reference[lbh_selec_reference$sound.files != "Phae.long1.wav", ],
detection = lbh_selec_reference, time.diagnostics = TRUE, by.sound.file = TRUE)

# get summary
summarize_diagnostic(dd)


}
```

energy_detector   energy_detector *detects the start and end of acoustic signals*

Description

energy_detector detects the start and end of acoustic signals

Usage

```
energy_detector(files = NULL, envelopes = NULL, path = NULL, wl = 512, power = 1,
thinning = 1, bp = NULL, ssmooth = 0, filter = "ffilter", threshold = 15, hold.time = 0,
min.duration = NULL, max.duration = NULL, parallel = 1, pb = TRUE)
```

Arguments

files   Character vector indicating the sound files that will be analyzed. Optional. If 'files' and 'envelopes' are not supplied then the function will work on all supported format sound files in the working directory.

envelopes   An object of class 'envelopes' (generated by get_envelopes) containing the amplitude envelopes of the sound files to be analyzed. If 'files' and 'envelopes' are not supplied then the function will work on all supported format sound files in the working directory.

path   Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.

wl   A numeric vector of length 1 specifying the window used internally by ffilter for bandpass filtering (so only applied when 'bp' is supplied). Default is 512.

power   A numeric vector of length 1 indicating a power factor applied to the amplitude envelope. Increasing power will reduce low amplitude modulations and increase high amplitude modulations, in order to reduce background noise. Default is 1 (no change). This argument is used internally by get_envelopes. Not used if 'envelopes' are supplied.

`thinning` Numeric vector of length 1 in the range 0~1 indicating the proportional reduction of the number of samples used to represent amplitude envelopes (i.e. the thinning of the envelopes). Usually amplitude envelopes have many more samples than those needed to accurately represent amplitude variation in time, which affects the size of the output (usually very large R objects / files). Default is `1` (no thinning). Higher sampling rates can afford higher size reduction (e.g. lower thinning values). Reduction is conducted by interpolation using `approx`. Note that thinning may decrease time precision, and the higher the thinning the less precise the time detection. This argument is used internally by `get_envelopes`. Not used if 'envelopes' are supplied.

`bp` Numeric vector of length 2 giving the lower and upper limits of a frequency bandpass filter (in kHz). Default is `NULL`. This argument is used internally by `get_envelopes`. Not used if 'envelopes' are supplied.

`ssmooth` A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. Default is 0 (no smoothing). Note that smoothing is applied before thinning (see 'thinning' argument). This argument is used internally by `get_envelopes`. Not used if 'envelopes' are supplied. #' @param ssmooth A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. Default is 0. Note that smoothing is applied before thinning (see 'thinning' argument).

`filter` Character vector of length 1 indicating the bandpass filter to be applied (only used if 'bp' is supplied). Three options available, (corresponding to the frequency filter functions in the 'seewave' package): ffilter (`ffilter`), bwfilter (`bwfilter`) and fir (`fir`). Not used if 'envelopes' are supplied.

`threshold` A numeric vector of length 1 specifying the amplitude threshold for detecting signals (in %).

`hold.time` Numeric vector of length 1. Specifies the time range at which selections will be merged (i.e. if 2 selections are separated by less than the specified hold.time they will be merged in to a single selection). Default is `0` (no hold time applied).

`min.duration` Numeric vector of length 1 giving the shortest duration (in seconds) of the signals to be detected. It removes signals below that threshold. If 'hold.time' is supplied signals are first merged and then filtered by duration.

`max.duration` Numeric vector of length 1 giving the longest duration (in seconds) of the signals to be detected. It removes signals above that threshold. If 'hold.time' is supplied signals are first merged and then filtered by duration.

`parallel` Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).

`pb` Logical argument to control progress bar. Default is `TRUE`.

Details

This function determines the start and end of signals in sound files. Sound files should be located in the working directory or the path to the sound files should be provided using the 'path' argument. The routine steps are: 1) calculating/modfiying amplitude envelopes (unless 'envelopes' is supplied), 2) detecting signals above the threshold, 3) merging signals (if 'hold.time' is supplied) and 4) filtering signals based on duration (if 'min.duration' and/or 'max.duration' are supplied).

Note that when envelopes are not supplied they are calculated on the fly which is more efficient in terms of memory usage (better option if R crashes!).

Value

A data frame containing the start and end of each signal by sound file. If no signal was detected for a sound file it is not included in the output data frame.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>). Implements a modified version of the timer function from seewave.

References

Araya-Salas, M. (2021), ohun: automatic detection of acoustic signals. R package version 0.1.0.

See Also

optimize_energy_detector

Examples

```
{
 ## Not run:
# Save example files into temporary working directory
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))
writeWave(Phae.long3, file.path(tempdir(), "Phae.long3.wav"))
writeWave(Phae.long4, file.path(tempdir(), "Phae.long4.wav"))

# using smoothing and minimum duration
detec <- energy_detector(files = c("Phae.long1.wav", "Phae.long2.wav",
"Phae.long3.wav", "Phae.long4.wav"),
path = tempdir(), threshold = 6, ssmooth = 300,
bp = c(2, 9), wl = 300, min.duration = 0.09)

# diagnose detection
diagnose_detection(reference = lbh_selec_reference,
detection = detec)

# without declaring 'files'
detec <- energy_detector(path = tempdir(), threshold = 6, ssmooth = 300,
bp = c(2, 9), wl = 300, min.duration = 0.09)

# diagnose detection
diagnose_detection(reference = lbh_selec_reference,
detection = detec)

# using hold time
detec <- energy_detector(threshold = 10, hold.time = 0.15,
bp = c(2, 9), wl = 300, path = tempdir())

# diagnose detection
diagnose_detection(reference = lbh_selec_reference, detection = detec)

# calculate envelopes first
envs <- get_envelopes(bp = c(2, 9), wl = 300, path = tempdir())

# then run detection providing 'envelopes' (but no 'files')
detec <- energy_detector(envelopes = envs, threshold = 10, hold.time = 0.15, min.duration = 0.05)

# diagnose detection
diagnose_detection(reference = lbh_selec_reference, detection = detec, time.diagnostics = TRUE)

# USIN OTHER SOUND FILE FORMAT (flac program must be installed)
 # fisrt convert files to flac
 warbleR::wav_2_flac(path = tempdir())

 # change sound file extension to flac
 flac_reference <- lbh_selec_reference
 flac_reference$sound.files <- gsub(".wav", ".flac", flac_reference$sound.files)

 # run detection
 detec <- energy_detector(files = c("Phae.long1.flac", "Phae.long2.flac",
 "Phae.long3.flac", "Phae.long4.flac"), path = tempdir(), threshold = 6,
 ssmooth = 300, bp = c(2, 9), wl = 300, min.duration = 0.09)

 # diagnose detection
 diagnose_detection(reference = flac_reference, detection = detec)

## End(Not run)
}
```

get_envelopes *Extract absolute amplitude envelopes*

Description

get_envelopes extracts absolute amplitude envelopes to speed up energy detection

Usage

```
get_envelopes(path = NULL, files = NULL, bp = NULL, wl = 512, power = 1,
parallel = 1, thinning = 1, pb = TRUE, ssmooth = 0, normalize = TRUE,
filter = "ffilter")
```

Arguments

`path` Character string containing the directory path where the sound files are located. If `NULL` (default) then the current working directory is used.

`files` character vector or indicating the sound files that will be analyzed.

`bp` Numeric vector of length 2 giving the lower and upper limits of a frequency bandpass filter (in kHz). Default is `NULL`.

`wl` A numeric vector of length 1 specifying the window used internally for bandpass filtering (so only applied when 'bp' is supplied). Default is 512.

`power` A numeric vector of length 1 indicating a power factor applied to the amplitude envelope. Increasing power will reduce low amplitude modulations and increase high amplitude modulations, in order to reduce background noise. Default is 1 (no change).

`parallel` Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).

`thinning` Numeric vector of length 1 in the range 0~1 indicating the proportional reduction of the number of samples used to represent amplitude envelopes (i.e. the thinning of the envelopes). Usually amplitude envelopes have many more samples than those needed to accurately represent amplitude variation in time, which affects the size of the output (usually very large R objects / files). Default is `1` (no thinning). Higher sampling rates can afford higher size reduction (e.g. lower thinning values). Reduction is conducted by linear interpolation using `approx`. Note that thinning may decrease time precision and that the higher the thinning the less precise the time detection. It's generally not advised if no smoothing ('ssmooth' argument) is applied.

`pb` Logical argument to control progress bar. Default is `TRUE`.

`ssmooth` A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. Default is 0. Note that smoothing is applied before thinning (see 'thinning' argument).

`normalize` Logical argument to control if

`filter` Character vector of length 1 indicating the bandpass filter to be applied (only used if 'bp' is supplied). Three options available, (corresponding to the frequency filter functions in the 'seewave' package): ffilter (`ffilter`), bwfilter (`bwfilter`) and fir (`fir`).

Details

This function determines the start and end of signals in the sound file selections listed in the input data frame ('X'). Alternatively, if no data frame is provided, the function detects signals across each entire sound file. It can also create long spectrograms highlighting the start and of the detected signals for all sound files in the working directory (if `img = TRUE`). Sound files should be located in the working directory or the path to the sound files should be provided using the 'path' argument. The input data frame should have the following columns: c("sound.files","selec","start","end"). This function uses a modified version of the `timer` function from seewave package to detect signals.

Value

An object of class 'envelopes'.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>). Implements a modified version of the timer function from seewave.

References

Araya-Salas, M., & Smith-Vidaurre, G. (2017). warbleR: An R package to streamline analysis of animal acoustic signals. Methods in Ecology and Evolution, 8(2), 184-191.

See Also

energy_detector

Examples

```
{
# Save to temporary working directory
data(list = c("Phae.long1", "Phae.long2"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))

# get raw absolute amplitude envelopes
envs <- get_envelopes(path = tempdir())

# extract segment for the first signal in the first sound file
x <- envs[[1]]$envelope

# and plot it
plot(x[(length(x)/9):(length(x)/4)], type = "l", xlab = "samples", ylab = "amplitude")

# smoothing envelopes
envs <- get_envelopes(path = tempdir(), ssmooth = 300)
x <- envs[[1]]$envelope
plot(x[(length(x)/9):(length(x)/4)], type = "l", xlab = "samples", ylab = "amplitude")

# smoothing and thinning
envs <- get_envelopes(path = tempdir(), thinning = 1/10, ssmooth = 300)
x <- envs[[1]]$envelope
plot(x[(length(x)/9):(length(x)/4)], type = "l", xlab = "samples", ylab = "amplitude")

# no normalization
envs <- get_envelopes(path = tempdir(), thinning = 1/10, ssmooth = 300)
x <- envs[[1]]$envelope
plot(x[(length(x)/9):(length(x)/4)], type = "l", xlab = "samples", ylab = "amplitude",
normalize = FALSE)
}
```

ohun *ohun: Automatic detection of acoustic signals*

Description

ohun is intended to facilitate the automatic detection of acoustic signals, providing functions to diagnose and optimize detection routines. Detections from other software can also be explored and optimized.

Details

The main features of the package are:

- The batch processing of sound files for signal detection
- The usage of annotations for detection optimization and diagnostic

The package offers functions to:

- Energy-based detection
- Template-based detection
- Diagnose detection precision
- Optimize detection routines based on reference annotations

Most of the functions allow the parallelization of tasks, which distributes the tasks among several processors to improve computational efficiency.

License: GPL (>= 2)

Author(s)

Marcelo Araya-Salas & Grace Smith Vidaurre

Maintainer: Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

optimize_energy_detector *Optimize the detection of signals based on a-priori detections*

Description

Optimize the detection of signals based on a-priori detections

Usage

```
optimize_energy_detector(reference, files = NULL, threshold = 15, power = 1,
 wl = 512, ssmooth = 0, hold.time = 0, min.duration = NULL, max.duration = NULL,
  thinning = 1, filter = "ffilter", parallel = 1, pb = TRUE, by.sound.file = FALSE,
   bp = NULL, path = NULL, previous.output = NULL)
```

Arguments

`reference` ’selection_table’ object or a data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end). **It should contain the reference selections that will be used for detection optimization**.

`files` Character vector indicating the sound files that will be analyzed. Optional. If not supplied the function will work on the sound files in ’reference’. It can be used to include signals with no signals.

`threshold` A numeric vector specifying the amplitude threshold for detecting signals (in %). Default is 15. **Several values can be supplied for optimization**.

`power` A numeric vector indicating a power factor applied to the amplitude envelope. Increasing power will reduce low amplitude modulations and increase high amplitude modulations, in order to reduce background noise. Default is 1 (no change). **Several values can be supplied for optimization**.

`wl` A numeric vector of length 1 specifying the window used internally by `ffilter` for bandpass filtering (so only applied when ’bp’ is supplied). Default is 512.

`ssmooth` A numeric vector to smooth the amplitude envelope with a sum smooth function. Default is 0 (no smoothing). **Several values can be supplied for optimization**.

`hold.time` Numeric vector of length 1. Specifies the time range at which selections will be merged (i.e. if 2 selections are separated by less than the specified hold.time they will be merged in to a single selection). Default is `0` (no hold time applied). **Several values can be supplied for optimization**.

`min.duration` Numeric vector giving the shortest duration (in seconds) of the signals to be detected. It removes signals below that threshold. **Several values can be supplied for optimization**.

`max.duration` Numeric vector giving the longest duration (in seconds) of the signals to be detected. It removes signals above that threshold. **Several values can be supplied for optimization**.

`thinning` Numeric vector in the range 0~1 indicating the proportional reduction of the number of samples used to represent amplitude envelopes (i.e. the thinning of the envelopes). Usually amplitude envelopes have many more samples than those needed to accurately represent amplitude variation in time, which affects the size of the output (usually very large R objects / files). Default is 1 (no thinning). Higher sampling rates may afford higher size reduction (e.g. lower thinning values). Reduction is conducted by interpolation using `approx`. Note that thinning may decrease time precision, and the higher the thinning the less precise the time detection. **Several values can be supplied for optimization**.

`filter` Character vector of length 1 indicating the bandpass filter to be applied (only used if ’bp’ is supplied). Three options available, (corresponding to the frequency filter functions in the ’seewave’ package): ffilter (`ffilter`), bwfilter (`bwfilter`) and fir (`fir`).

`parallel` Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).

`pb` Logical argument to control progress bar and messages. Default is `TRUE`.

`by.sound.file` Logical argument to control whether performance diagnostics are summarized across sound files (when `by.sound.file = FALSE`, when more than 1 sound file is included in 'reference') or shown separated by sound file. Default is `FALSE`.

`bp` Numeric vector of length 2 giving the lower and upper limits of a frequency bandpass filter (in kHz). Default is `NULL`. This argument is used internally by `get_envelopes`. Not used if 'envelopes' are supplied.

`path` Character string containing the directory path where the sound files are located. If `NULL` (default) then the current working directory is used.

`previous.output` Data frame with the output of a previous run of this function. This will be used to include previous results in the new output and avoid recalculating detection performance for parameter combinations previously evaluated.

Details

This function takes a selections data frame or 'selection_table' ('reference') estimates the detection performance under different detection parameter combinations. This is done by comparing the position in time of the detection to those of the reference selections in 'reference'. The function returns several diagnostic metrics to allow user to determine which parameter values provide a detection that more closely matches the selections in 'reference'. Those parameters can be later used for performing a more efficient detection using `energy_detector`.

Value

A data frame in which each row shows the result of a detection job with a particular combination of tuning parameters (including in the data frame). It also includes the following diagnostic metrics:

- `true.positives`: number of detections that correspond to signals referenced in 'reference'. Matching is defined as some degree of overlap in time. In a perfect detection routine it should be equal to the number of rows in 'reference'.
- `false.positives`: number of detections that don't match any of the signals referenced in 'reference'. In a perfect detection routine it should be 0.
- `false.negatives`: number of signals in 'reference' that were not detected (not found in 'detection'. In a perfect detection routine it should be 0.
- `split.positives`: number of signals referenced in 'reference' that were overlapped by more than 1 detection (i.e. detections that were split). In a perfect detection routine it should be 0.
- `mean.duration.true.positives`: mean duration of true positives (in s).
- `mean.duration.false.positives`: mean duration of false positives (in s).
- `mean.duration.false.negatives`: mean duration of false negatives (in s). Only included when `time.diagnostics = TRUE`.
- `proportional.duration.true.positives`: ratio of total duration of true positives to the total duration of signals referenced in 'reference'. In a perfect detection routine it should be 1.
- `sensitivity`: Proportion of signals referenced in 'reference' that were detected. In a perfect detection routine it should be 1.
- `specificity`: Proportion of detections that correspond to signals referenced in 'reference' that were detected. In a perfect detection routine it should be 1.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>).

References

Araya-Salas, M. (2021), ohun: automatic detection of acoustic signals. R package version 0.1.0.

Examples

```
{
# Save example files into temporary working directory
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_reference"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))
writeWave(Phae.long3, file.path(tempdir(), "Phae.long3.wav"))
writeWave(Phae.long4, file.path(tempdir(), "Phae.long4.wav"))

# using smoothing and minimum duration
optimize_energy_detector(reference = lbh_selec_reference, path = tempdir(),
threshold = c(6, 10), ssmooth = 300, bp = c(2, 9), wl = 300, min.duration = 0.09)

#  2 different filters
optimize_energy_detector(reference = lbh_selec_reference, path = tempdir(),
threshold = c(6, 10, 15), ssmooth = 300, filter = c("ffilter", "fir"),
bp = c(2, 9), wl = 300, min.duration = 0.09)

# with thinning and smoothing
optimize_energy_detector(reference = lbh_selec_reference, path = tempdir(),
 threshold = c(6, 10, 15), ssmooth = c(300, 1000), thinning = c(0.1, 0.01),
 bp = c(2, 9), wl = 300, min.duration = 0.09)

# by sound file
(opt_ed <- optimize_energy_detector(reference = lbh_selec_reference, path = tempdir(),
threshold = c(6, 10, 15), ssmooth = 300, filter = c("ffilter", "fir"),
bp = c(2, 9), wl = 300, min.duration = 0.09, by.sound.file = TRUE))

# summarize
summarize_diagnostic(opt_ed)

# using hold time
(op_ed <- optimize_energy_detector(reference = lbh_selec_reference, threshold = 10,
hold.time = c(0.1, 0.15), bp = c(2, 9), wl = 300, path = tempdir()))

# including previous output in new call
optimize_energy_detector(reference = lbh_selec_reference, threshold = 10,
hold.time = c(0.05, 0.2), previous.output = op_ed,
bp = c(2, 9), wl = 300, path = tempdir())

# having and extra file in files (simulating a file that should have no detetions)
sub_reference <- lbh_selec_reference[lbh_selec_reference$sound.files != "Phae.long1.wav", ]

optimize_energy_detector(reference = sub_reference, files = unique(lbh_selec_reference$sound.files),
threshold = 10, hold.time = c(0.1, 0.15), bp = c(2, 9), wl = 300, path = tempdir())
}
```

`split_sound_files` *Splits sound files*

Description

`split_sound_files` splits sound files in shorter segments

Usage

```
split_sound_files(path = NULL, sgmt.dur = 10, sgmts = NULL, files = NULL,
 parallel = 1, pb = TRUE, only.sels = FALSE, X = NULL)
```

Arguments

`path` Directory path where sound files are found. If `NULL` (default) then the current working directory is used.

`sgmt.dur` Numeric. Duration (in s) of segments in which sound files would be split. Sound files shorter than 'sgmt.dur' won't be split. Ignored if 'sgmts' is supplied.

`sgmts` Numeric. Number of segments in which to split each sound file. If supplied 'sgmt.dur' is ignored.

`files` Character vector indicating the subset of files that will be split.

`parallel` Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).

`pb` Logical argument to control progress bar. Default is `TRUE`. Only used when

`only.sels` Logical argument to control if only the data frame is returned (no wave files are saved). Default is `FALSE`.

`X` 'selection_table' object or a data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end). If supplied the data frame/selection table is modified to reflect the position of the selections in the new sound files. Note that some selections could split between 2 segments. To deal with this, a 'split.sels' column is added to the data frame in which those selection are labeled as 'split'. Default is `NULL`.

Details

This function aims to reduce the size of sound files in order to simplify some processes that are limited by sound file size (big files can be manipulated, e.g. `auto_detec`).

Value

Wave files for each segment in the working directory (if `only.sels = FALSE`, named as 'sound.file.name-#.wav') and a data frame in the R environment containing the name of the original sound files (org.sound.files), the name of the clips (sound.files) and the start and end of clips in the original files. Clips are saved in .wav format. If 'X' is supplied then a data frame with the position of the selections in the newly created clips is returned instead.

Author(s)

Marcelo Araya-Salas (<`marcelo.araya@ucr.ac.cr`>)

References

Araya-Salas, M. (2021), ohun: automatic detection of acoustic signals. R package version 0.1.0.

See Also

`cut_sels`

Examples

```
{
#load data and save to temporary working directory
data(list = c("Phae.long1", "Phae.long2", "Phae.long3"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))
writeWave(Phae.long3, file.path(tempdir(), "Phae.long3.wav"))

#split files in 1 s files
split_sound_files(sgmt.dur = 1, path = tempdir())

# Check this folder
tempdir()
}
```

`summarize_diagnostic` *Summarize detection diagnostics*

Description

`summarize_diagnostic` summarizes detection diagnostics

Usage

```
summarize_diagnostic(diagnostic, time.diagnostics = FALSE)
```

Arguments

`diagnostic` A data frame with the reference selections (start and end of the signals) that will be used to evaluate the performance of the detection, represented by those selections in 'detection'. Must contained at least the following columns: "sound.files", "selec", "start" and "end".

`time.diagnostics`
Logical argument to control if diagnostics related to the duration of the signals ("mean.duration.true.positives", "mean.duration.false.positives", "mean.duration.false.negatives" and "proportional.duration.true.positives") are returned (if `TRUE`). Default is `FALSE`.

Details

The function summarizes a detection diagnostic data frame in which diagnostic parameters are shown split by (typically) a categorical column, usually sound files. This function is used internally by `diagnose_detection`.

Value

A data frame, typically the output of a detection optimization function (`diagnose_detection`, `optimize_find_peaks`, `optimize_auto_detec`) including the following detection performance diagnostics:

- `true.positives`: number of detections that correspond to signals referenced in 'reference'. Matching is defined as some degree of overlap in time. In a perfect detection routine it should be equal to the number of rows in 'reference'.
- `false.positives`: number of detections that don't match any of the signals referenced in 'reference'. In a perfect detection routine it should be 0.
- `false.negatives`: number of signals in 'reference' that were not detected (not found in 'detection'. In a perfect detection routine it should be 0.
- `split.positives`: number of signals referenced in 'reference' that were overlapped by more than 1 detection (i.e. detections that were split). In a perfect detection routine it should be 0.
- `mean.duration.true.positives`: mean duration of true positives (in s). Optional.
- `mean.duration.false.positives`: mean duration of false positives (in s). Optional.
- `mean.duration.false.negatives`: mean duration of false negatives (in s). Optional.
- `proportional.duration.true.positives`: ratio of total duration of true positives to the total duration of signals referenced in 'reference'. In a perfect detection routine it should be 1. Optional.
- `sensitivity`: Proportion of signals referenced in 'reference' that were detected. In a perfect detection routine it should be 1.
- `specificity`: Proportion of detections that correspond to signals referenced in 'reference' that were detected. In a perfect detection routine it should be 1.

Author(s)

Marcelo Araya-Salas <`marcelo.araya@ucr.ac.cr`>)

References

Araya-Salas, M. (2021), ohun: automatic detection of acoustic signals. R package version 0.1.0.

See Also

diagnose_detection

Examples

```
{
# load example selection tables

data(list = c("lbh_selec_reference", "lbh_selec_table"))

# 'lbh_selec_reference' has all signals annotated while 'lbh_selec_table' is missing a few

# run diagnose_detection() by sound file
diag <- diagnose_detection(reference = lbh_selec_reference,
detection = lbh_selec_table, by.sound.file = TRUE)

# summarize
summarize_diagnostic(diagnostic = diag)

# should be the same as this:
diagnose_detection(reference = lbh_selec_reference,
detection = lbh_selec_table, by.sound.file = FALSE)
}
```

Index