Package: ohun (via r-universe)

October 22, 2024

Title Optimizing Acoustic Signal Detection

Version 1.0.2

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

Description Facilitates the automatic detection of acoustic signals, providing functions to diagnose and optimize the performance of detection routines. Detections from other software can also be explored and optimized. This package has been peer-reviewed by rOpenSci. Araya-Salas et al. (2022) <doi:10.1101/2022.12.13.520253>.

License GPL (>= 2)

Encoding UTF-8

URL https://docs.ropensci.org/ohun/, https://github.com/ropensci/ohun/

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

VignetteBuilder knitr

RoxygenNote 7.3.2

Language en-US

Imports tuneR, warbleR (>= 1.1.32), cli, methods, stats, utils, seewave (>= 2.0.1), fftw, rlang, sf, igraph, checkmate, ggplot2

Depends R (>= 3.2.1)

Suggests knitr, rmarkdown, testthat, viridis, Sim.DiffProc, vdiffr

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

RemoteUrl https://github.com/ropensci/ohun

RemoteRef master

RemoteSha 2be88b518b841b0e996ce7765c218314a967b035

Contents

`consensus_detection` *Remove ambiguous detections*

Description

`consensus_detection` removes ambiguous detections

Usage

```
consensus_detection(detection, by = "overlap", filter = "max", cores = 1, pb = TRUE)
```

Arguments

`detection`	Data frame or selection table (using the warbleR package's format, see `selection_table`) with the output of `label_detection` containing the start and end of the signals. Must contained at least the following columns: "sound.files", "selec", "start", "end" and "detection.class" (the last one is generated by `label_detection`). It must also contained the column indicated in the 'by' argument (which is 'overlap' by default).
`by`	Character vector of length 1 indicating a column in 'detection' that will be used to filter detections. Must refer to a numeric column. Default is 'overlap', which is return by `label_detection`.
`filter`	Character vector of length 1 indicating the criterium used to filter the column refer to by the 'by' argument. Current options are 'max' (maximum) and 'min' (minimum). Default is 'max'.

cores	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 removes ambiguous detections keeping only the one that maximizes a criterium given by 'filter'. By default it keeps the detection with the highest overlap to the reference signal. It works on the output of label_detection. Useful when several detections match the same reference as in the case of template detection with multiple templates (see template_detector).

Value

A data frame or selection table (if 'detection' was also a selection table, warbleR package's format, see selection_table) as in 'X' but removing ambiguous detections (split and merged positives).

Author(s)

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

References

#' Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

label_detection, template_detector

Examples

```
{
  # load example data
  data("lbh1", "lbh_reference")

  # save sound files
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh2.wav"))

  # template for the first sound file in 'lbh_reference'
  templ1 <- lbh_reference[1, ]

  # generate template correlations
  tc <- template_correlator(
    templates = templ1, path = tempdir(),
    files = "lbh2.wav"
  )

  # template detection
  td <- template_detector(template.correlations = tc, threshold = 0.12)
```

```
  # this detection generates 2 split positives
  diagnose_detection(
    reference = lbh_reference[lbh_reference == "lbh2.wav", ],
    detection = td
  )

  # label detection
  ltd <- label_detection(
    reference = lbh_reference[lbh_reference == "lbh2.wav", ],
    detection = td
  )

  # now they can be filter to keep the detection with the highest score for each split
  ftd <- consensus_detection(ltd, by = "scores")

  # splits must be 0
  diagnose_detection(
    reference = lbh_reference[lbh_reference == "lbh2.wav", ],
    detection = ftd
  )
}
```

`diagnose_detection` *Evaluate the performance of a sound event detection procedure*

Description

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

Usage

```
diagnose_detection(reference, detection, by.sound.file = FALSE,
time.diagnostics = FALSE, cores = 1, pb = TRUE, path = NULL, by = NULL,
 macro.average = FALSE, min.overlap = 0.5)
```

Arguments

`reference`	Data frame or 'selection.table' (following the warbleR package format) with the reference selections (start and end of the sound events) 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". **It must contain the reference selections that will be used for detection optimization**.
`detection`	Data frame or 'selection.table' with the detections (start and end of the sound events) that will be compared against the 'reference' selections. Must contained at least the following columns: "sound.files", "selec", "start" and "end". It can contain data for additional sound files not found in 'references'. In this case the

routine assumes that no sound events are found in those files, so detection from those files are all false positives.

`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 sound events ("mean.duration.true.positives", "mean.duration.false.positives", "mean.duration.false.negatives" and "proportional.duration.true.positives") are returned (if `TRUE`). Default is `FALSE`.

`cores` 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`.

`path` Character string containing the directory path where the sound files are located. If supplied then duty cycle (fraction of a sound file in which sounds were detected)is also returned. This feature is more helpful for tuning an energy-based detection. Default is `NULL`.

`by` Character vector with the name of a column in 'reference' for splitting diagnostics. Diagnostics will be returned separated for each level in 'by'. Default is `NULL`.

`macro.average` Logical argument to control if diagnostics are first calculated for each sound file and then averaged across sound files, which can minimize the effect of unbalanced sample sizes between sound files. If `FALSE` (default) diagnostics are based on aggregated statistics irrespective of sound files. The following indices can be estimated by macro-averaging: overlap, mean.duration.true.positives, mean.duration.false.positives, mean.duration.false.positives, mean.duration.false.negatives, proportional.duration.true.positives, recall and precision (f.score is always derived from recall and precision). Note that when applying macro-averaging, recall and precision are not derived from the true positive, false positive and false negative values returned by the function.

`min.overlap` Numeric. Controls the minimum amount of overlap required for a detection and a reference sound for it to be counted as true positive. Default is 0.5. Overlap is measured as intersection over union.

Details

The function evaluates the performance of a sound event detection procedure by comparing its output selection table to a reference selection table in which all sound events of interest have been selected. The function takes any overlap between detected sound events and target sound events as true positives. Note that all sound files located in the supplied 'path' will be analyzed even if not all of them are listed in 'reference'. When several possible matching pairs of sound event and detections are found, the optimal set of matching pairs is found through maximum bipartite matching (using the R package igraph). Priority for assigning a detection to a reference is given by the amount of time overlap. 'splits' and 'merge.positives' are also counted (i.e. counted twice) as 'true.positives'. Therefore "true.positives + false.positives = detections".

Value

A data frame including the following detection performance diagnostics:

- `detections`: total number of detections
- `true.positives`: number of sound events in 'reference' that correspond to any detection. 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 (i.e. don't overlap with) any of the sound events in 'reference'. In a perfect detection routine it should be 0.
- `false.negatives`: number of sound events in 'reference' that were not detected (not found in 'detection'. In a perfect detection routine it should be 0.
- `splits`: number of detections overlapping reference sounds that also overlap with other detections. In a perfect detection routine it should be 0.
- `merges`: number of detections that overlap with two or more reference sounds. In a perfect detection routine it should be 0.
- `mean.duration.true.positives`: mean duration of true positives (in ms). Only included when `time.diagnostics = TRUE`.
- `mean.duration.false.positives`: mean duration of false positives (in ms). Only included when `time.diagnostics = TRUE`.
- `mean.duration.false.negatives`: mean duration of false negatives (in ms). Only included when `time.diagnostics = TRUE`.
- `overlap`: mean intersection over union overlap of true positives.
- `proportional.duration.true.positives`: ratio of duration of true positives to the duration of sound events in 'reference'. In a perfect detection routine it should be 1. Based only on true positives that were not split or merged.
- `duty.cycle`: proportion of a sound file in which sounds were detected. Only included when `time.diagnostics = TRUE` and `path` is supplied. Useful when conducting energy-based detection as a perfect detection can be obtained with a very low amplitude threshold, which will detect everything, but will produce a duty cycle close to 1.
- `recall`: Proportion of sound events in 'reference' that were detected. In a perfect detection routine it should be 1.
- `precision`: Proportion of detections that correspond to sound events in 'reference'. In a perfect detection routine it should be 1.
- `f.score`: Combines recall and precision as the harmonic mean of these two. Provides a single value for evaluating performance. In a perfect detection routine it should be 1.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

optimize_energy_detector, optimize_template_detector

Examples

```
{
  # load data
  data("lbh_reference")

  # perfect detection
  diagnose_detection(reference = lbh_reference, detection = lbh_reference)

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

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

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

  # and extra sound file in reference
  diagnose_detection(
    reference = lbh_reference,
    detection =
      lbh_reference[lbh_reference$sound.files != "lbh1", ]
  )

  # and extra sound file in detection
  diagnose_detection(
    reference =
      lbh_reference[lbh_reference$sound.files != "lbh1", ],
    detection = lbh_reference
  )

  # and extra sound file in detection by sound file
  dd <- diagnose_detection(
    reference =
      lbh_reference[lbh_reference$sound.files != "lbh1", ],
    detection = lbh_reference, time.diagnostics = TRUE, by.sound.file = TRUE
  )

  # get summary
  summarize_diagnostic(dd)
}
```

energy_detector *Detects the start and end of sound events*

Description

energy_detector detects the start and end of sound events based on energy and time attributes

Usage

```
energy_detector(files = NULL, envelopes = NULL, path = ".", hop.size = 11.6, wl = NULL,
thinning = 1, bp = NULL, smooth = 5, threshold = 5, peak.amplitude = 0,
hold.time = 0, min.duration = 0, max.duration = Inf, cores = 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. Supported file formats:'.wav', '.mp3', '.flac' and '.wac'. If not supplied the function will work on all sound files (in the supported format) in 'path'.
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. The current working directory is used as default.
hop.size	A numeric vector of length 1 specifying the time window duration (in ms). Default is 11.6 ms, which is equivalent to 512 wl for a 44.1 kHz sampling rate. Ignored if 'wl' is supplied.
wl	A numeric vector of length 1 specifying the window length of the spectrogram. Default is NULL. If supplied, 'hop.size' is ignored. Used internally for bandpass filtering (so only applied when 'bp' is 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. Bandpass is done using

the function ffilter, which applies a short-term Fourier transformation to first create a spectrogram in which the target frequencies are filtered and then is back transformed into a wave object using a reverse Fourier transformation.

`smooth` A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. It controls the time 'neighborhood' (in ms) in which amplitude samples are smoothed (i.e. averaged with neighboring samples). Default is 5. 0 means no smoothing is applied. Note that smoothing is applied before thinning (see 'thinning' argument). The function envelope is used internally which is analogous to sum smoothing in env. This argument is used internally by get_envelopes. Not used if 'envelopes' are supplied.

`threshold` Numeric vector of length 1 with a value between 0 and 100 specifying the amplitude threshold for detecting sound event occurrences. Amplitude is represented as a percentage so 0 and 100 represent the lowest amplitude and highest amplitude respectively. Default is 5.

`peak.amplitude` Numeric vector of length 1 with the minimum peak amplitude value. Detections below that value are excluded. Peak amplitude is the maximum sound pressure level (in decibels) across the sound event (see sound_pressure_level). This can be useful when expecting higher peak amplitude in the target sound events compared to non-target sound events or when keeping only the best examples of the target sound events. Default is 0.

`hold.time` Numeric vector of length 1. Specifies the time range (in ms) 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 ms) of the sound events to be detected. It removes sound events below that threshold. If 'hold.time' is supplied sound events are first merged and then filtered by duration. Default is 0 (i.e. no filtering based on minimum duration).

`max.duration` Numeric vector of length 1 giving the longest duration (in ms) of the sound events to be detected. It removes sound events above that threshold. If 'hold.time' is supplied sound events are first merged and then filtered by duration. Default is `Inf` (i.e. no filtering based on maximum duration).

`cores` 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 detects the time position of target sound events based on energy and time thresholds. It first detect all sound above a given energy threshold (argument 'energy'). If 'hold.time' is supplied then detected sounds are merged if necessary. Then the sounds detected are filtered based on duration attributes ('min.duration' and 'max.duration'). If 'peak.amplitude' is higher than 0 then only those sound events with higher peak amplitude are kept. Band pass filtering ('bp'), thinning ('thinning') and envelope smoothing ('smooth') are applied (if supplied) before threshold detection.

Value

The function returns a 'selection_table' (warbleR package's formats, see selection_table) or data frame (if sound files can't be found) containing the start and end of each sound event by sound file. If no sound event 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>)

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

optimize_energy_detector

Examples

```
# Save example files into temporary working directory
data("lbh1", "lbh2", "lbh_reference")
tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

# using smoothing and minimum duration
detec <- energy_detector(files = c("lbh1.wav", "lbh2.wav"),
path = tempdir(), threshold = 6, smooth = 6.8,
bp = c(2, 9), hop.size = 3, min.duration = 0.05)

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

# without declaring 'files'
detec <- energy_detector(path = tempdir(), threshold = 60, smooth = 6.8,
bp = c(2, 9), hop.size = 6.8, min.duration = 90)

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

# using hold time
detec <- energy_detector(threshold = 10, hold.time = 150,
bp = c(2, 9), hop.size = 6.8, path = tempdir())

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

# calculate envelopes first
```

```
envs <- get_envelopes(bp = c(2, 9), hop.size = 6.8, path = tempdir())

# then run detection providing 'envelopes' (but no 'files')
detec <- energy_detector(envelopes = envs, threshold = 10, hold.time = 150, min.duration = 50)

# diagnose detection
diagnose_detection(reference = lbh_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_reference
 flac_reference$sound.files <- gsub(".wav", ".flac", flac_reference$sound.files)

 # run detection
detec <- energy_detector(files = c("lbh1.flac", "lbh2.flac"), path = tempdir(), threshold = 60,
 smooth = 6.8, bp = c(2, 9), hop.size = 6.8, min.duration = 90)

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

---

get_envelopes *Extract absolute amplitude envelopes*

Description

get_envelopes extracts absolute amplitude envelopes to speed up energy detection

Usage

```
get_envelopes(path = ".", files = NULL, bp = NULL, hop.size = 11.6, wl = NULL,
cores = 1, thinning = 1, pb = TRUE, smooth = 5, normalize = TRUE)
```

Arguments

path	Character string containing the directory path where the sound files are located. The current working directory is used as default.
files	character vector or indicating the sound files that will be analyzed. Supported file formats:'.wav', '.mp3', '.flac' and '.wac'. If not supplied the function will work on all sound files (in the supported format) in 'path'.
bp	Numeric vector of length 2 giving the lower and upper limits of a frequency bandpass filter (in kHz). Default is NULL. Bandpass is done using the function ffilter, which applies a short-term Fourier transformation to first create a spectrogram in which the target frequencies are filtered and then is back transformed into a wave object using a reverse Fourier transformation.

`hop.size` A numeric vector of length 1 specifying the time window duration (in ms). Default is 11.6 ms, which is equivalent to 512 wl for a 44.1 kHz sampling rate. Ignored if 'wl' is supplied.

`wl` A numeric vector of length 1 specifying the window length of the spectrogram. Default is `NULL`. If supplied, 'hop.size' is ignored. Used internally for bandpass filtering (so only applied when 'bp' is supplied).

`cores` 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 ('smooth' argument) is applied.

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

`smooth` A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. It controls the time 'neighborhood' (in ms) in which amplitude samples are smoothed (i.e. averaged with neighboring samples). Default is 5. 0 means no smoothing is applied. Note that smoothing is applied before thinning (see 'thinning' argument). The function `envelope` is used internally which is analogous to sum smoothing in `env`. This argument is used internally by `get_envelopes`.

`normalize` Logical argument to control if envelopes are normalized to a 0-1 range.

Details

This function extracts the absolute amplitude envelopes of sound files. Can be used to manipulate envelopes before running `energy_detector`.

Value

An object of class 'envelopes'.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

energy_detector

Examples

```
{
  # Save to temporary working directory
  data(list = c("lbh1", "lbh2"))
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
  tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

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

  # extract segment for the first sound event 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(), smooth = 6.8)
  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, smooth = 6.8)
  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, smooth = 6.8)
  x <- envs[[1]]$envelope
  plot(x[(length(x) / 9):(length(x) / 4)],
    type = "l", xlab = "samples", ylab = "amplitude",
    normalize = FALSE
  )
}
```

---

get_templates *Find templates representative of the structural variation of sound events*

Description

get_templates find the sound events that are closer to the acoustic space centroid (i.e. close to the average acoustic structure) in a reference table.

Usage

```
get_templates(reference, acoustic.space = NULL, path = ".",
n.sub.spaces = 1, plot = TRUE, color = "#21908C4D", ...)
```

Arguments

reference	Selection table (using the warbleR package's format, see selection_table) or data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of sound event (start and end).
acoustic.space	Numeric matrix or data frame with the two dimensions of a custom acoustic space to be used for finding templates. if not supplied the acoustic space is calculated internally (default). Optional. Note that the function assumes that 'reference' and 'acoustic.space' refer to the same sound events and similarly ordered.
path	Character string containing the directory path where the sound files are located. The current working directory is used as default.
n.sub.spaces	Integer vector of length 1 with the number of sub-spaces to split the total acoustic space. If n.sub.spaces = 1, only the sound event closer to the centroid is returned. If n.sub.spaces > 1 the function returns additional sound events, corresponding to those closer to the centroids of the sub-spaces. To do this, the function defines sub-spaces as equal-size slices of a circle centered at the centroid of the acoustic space.
plot	Logical to control if the plot is created. Default is TRUE.
color	Character string with the point color. Default is '#21908C4D'.
...	Additional arguments to be passed to spectro_analysis for further customization when measuring parameters to calculate the acoustic space.

Details

This function finds sound events (from a reference table) that are representative of the acoustic structure variation of all sound events. This is done by finding the events closer to the centroid of the acoustic space. If the acoustic space is not supplied ('acoustic.space' argument) then the function will estimate it by measuring several acoustic features using the function spectro_analysis (features related to energy distribution in the frequency and time domain as well as features of the dominant frequency contours, see spectro_analysis for more details) and summarizing it with Principal Component Analysis (after z-transforming parameters) using the function prcomp. Acoustic features with missing values are removed before estimating Principal Component Analysis. The rationale is that a sound event close to the average structure is more likely to share structural features with most events across the acoustic space than a sound event in the periphery of the space. If only 1 template is required the function returns the sound event closest to the acoustic space centroid. If more than 1 template is required additional sound events are returned that are representative of the acoustic space. To do this, the function defines sub-spaces as equal-size slices of a circle centered at the centroid of the acoustic space. A column 'template' is included in the output selection table that identifies each template. Custom acoustic spaces can be supplied with argument 'acoustic.space'. Notice that the function aims to partition spaces in which sounds are somehow homogeneously distributed. When clear clusters are found in the distribution of the acoustic space thus clusters might not match the sub-spaces defined by the function.

Value

The function returns a 'selection_table' (warbleR package's formats, see selection_table) or data frame (if sound files can't be found) containing the start and end of each sound event by sound file.

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., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

template_detector

Examples

```
{
  # Save example files into temporary working directory
  data("lbh1", "lbh2", "lbh_reference")
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
  tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

  # get a single mean template
  template <- get_templates(reference = lbh_reference, path = tempdir())

  # get 3 templates
  template <- get_templates(reference = lbh_reference, n.sub.spaces = 3, path = tempdir())
}
```

---

label_detection *Label detections from a sound event detection procedure*

Description

label_detection labels the performance of a sound event detection procedure comparing the output selection table to a reference selection table

Usage

```
label_detection(reference, detection, cores = 1, pb = TRUE, min.overlap = 0.5,
 by = NULL)
```

Arguments

reference	Data frame or 'selection.table' (following the warbleR package format) with the reference selections (start and end of the sound events) 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". **It must contain the reference selections that will be used for detection optimization**.
detection	Data frame or 'selection.table' with the detections (start and end of the sound events) that will be compared against the 'reference' selections. Must contained at least the following columns: "sound.files", "selec", "start" and "end". It can contain data for additional sound files not found in 'references'. In this case the routine assumes that no sound events are found in those files, so detection from those files are all false positives.
cores	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.
min.overlap	Numeric. Controls the minimum amount of overlap required for a detection and a reference sound for it to be counted as true positive. Default is 0.5. Overlap is measured as intersection over union.
by	Character vector with the name of a categorical column in 'reference' for running a stratified. Labels will be returned separated for each level in 'by'. Default is NULL.

Details

The function identifies the rows in the output of a detection routine as true or false positives. This is achieved by comparing the data frame to a reference selection table in which all sound events of interest have been selected.

Value

A data frame or selection table (if 'detection' was also a selection table, warbleR package's format, see selection_table) including three additional columns, 'detection.class', which indicates the class of each detection, 'reference' which identifies the event in the 'reference' table that was detected and 'overlap' which refers to the amount overlap to the reference sound. See diagnose_detection for a description of the labels used in 'detection.class'. The output data frame also contains an additional data frame with the overlap for each pair of overlapping detection/reference. Overlap is measured as intersection over union.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

diagnose_detection, summarize_diagnostic

Examples

```
{
  # load data
  data("lbh_reference")

  # an extra one in detection (1 false positive)
  label_detection(reference = lbh_reference[-1, ], detection = lbh_reference)

  # missing one in detection (all true positives)
  label_detection(reference = lbh_reference, detection = lbh_reference[-1, ])

  # perfect detection (all true positives)
  label_detection(reference = lbh_reference, detection = lbh_reference)

  # and extra sound file in reference (all true positives)
  label_detection(
    reference = lbh_reference, detection =
      lbh_reference[lbh_reference$sound.files != "lbh1.wav", ]
  )

  # and extra sound file in detection (some false positives)
  label_detection(
    reference =
      lbh_reference[lbh_reference$sound.files != "lbh1.wav", ],
    detection = lbh_reference
  )

  # duplicate 1 detection row (to get 2 splits)
  detec <- lbh_reference[c(1, seq_len(nrow(lbh_reference))), ]
  detec$selec[1] <- 1.2
  label_detection(
    reference = lbh_reference,
    detection = detec
  )

  # merge 2 detections (to get split and merge)
  Y <- lbh_reference
  Y$end[1] <- 1.2
  label_detection(reference = lbh_reference, detection = Y)

  # remove split to get only merge
  Y <- Y[-2, ]
  label_detection(reference = lbh_reference, detection = Y)
}
```

label_spectro *Plot a labeled spectrogram*

Description

label_spectro plot a spectrogram along with amplitude envelopes or cross-correlation scores

Usage

```
label_spectro(wave, reference = NULL, detection = NULL,
 envelope = FALSE, threshold = NULL, smooth = 5, collevels = seq(-100, 0, 5),
 palette = viridis::viridis, template.correlation = NULL,
 line.x.position = 2, hop.size = NULL, ...)
```

Arguments

wave	A 'wave' class object.
reference	Data frame or 'selection.table' (following the warbleR package format) with the reference selections (start and end of the sound events). Must contained at least the following columns: "sound.files", "selec", "start" and "end".
detection	Data frame or 'selection.table' with the detection (start and end of the sound events) Must contained at least the following columns: "sound.files", "selec", "start" and "end".
envelope	Logical to control whether the amplitude envelope is plotted. Default is FALSE.
threshold	A numeric vector on length 1 indicated the amplitude or correlation threshold to plot on the envelope or correlation scores respectively. Default is NULL. Note that for amplitude the range of valid values is 0-1, while for correlations the range is 0-100.
smooth	A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. It controls the time range (in ms) in which amplitude samples are smoothed (i.e. averaged with neighboring samples). Default is 5. 0 means no smoothing is applied.
collevels	Numeric sequence of negative numbers to control color partitioning and amplitude values that are shown (as in spectro).
palette	Function with the color palette to be used on the spectrogram (as in spectro)
template.correlation	List extracted from the output of template_correlator containing the correlation scores and metadata for an specific sound file/template dyad. For instance 'correlations[[1]]' where 'correlations' is the output of a template_correlator call. If supplied the correlation is also plotted. Default is NULL.
line.x.position	Numeric vector of length 1 with the position in the frequency axis (so in kHz) of the lines highlighting sound events. Default is 2.

hop.size	A numeric vector of length 1 specifying the time window duration (in ms). Default is 11.6 ms, which is equivalent to 512 'wl' for a 44.1 kHz sampling rate.
...	Additional arguments to be passed to spectro for further spectrogram customization.

Details

This function plots spectrograms annotated with the position of sound events. **Created for graphs included in the vignette, and probably only useful for that or for very short recordings**. Only works on a single 'wave' object at the time.

Value

A spectrogram along with lines highlighting the position of sound events in 'reference' and/or 'detection'. If supplied it will also plot the amplitude envelope or corelation scores below the spectrogram.

Author(s)

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

References

#' Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

energy_detector, template_correlator, template_detector

Examples

```
{
  # load example data
  data(list = "lbh1", "lbh_reference")

  # adding labels
  label_spectro(
    wave = lbh1,
    reference = lbh_reference[lbh_reference$sound.files == "lbh1.wav", ],
    wl = 200, ovlp = 50, flim = c(1, 10)
  )

  # adding envelope
  label_spectro(
    wave = lbh1,
    detection = lbh_reference[lbh_reference$sound.files == "lbh1.wav", ],
    wl = 200, ovlp = 50, flim = c(1, 10)
  )
```

```
  # see the package vignette for more examples
}
```

---

lbh1 *Long-billed hermit recording*

Description

lbh1 a wave object with long-billed hermit (*Phaethornis longirostris*) songs extracted from xeno-canto's '154138' recording.

Usage

```
data(lbh1)
```

Format

An object of class Wave of length 110250.

Source

Marcelo Araya-Salas

---

lbh2 *Long-billed hermit recording*

Description

lbh2 a wave object with long-billed hermit (*Phaethornis longirostris*) songs extracted from xeno-canto's '154129' recording.

Usage

```
data(lbh2)
```

Format

An object of class Wave of length 110250.

Source

Marcelo Araya-Salas

`lbh_reference` *Example data frame of a selection table including all sound events of interests*

Description

`lbh_reference` is a data frame containing the start, end, bottom and top frequency of all songs in 'lbh_1.wav' and 'lbh_2.wav' recordings. #'

Usage

```
data(lbh_reference)
```

Format

A data frame with 19 rows and 6 variables:

sound.files recording names

selec selection numbers within recording

start start times of selected sound event

end end times of selected sound event

bottom.freq lower limit of frequency range

top.freq upper limit of frequency range

Details

A data frame containing the start, end, low and high frequency of *Phaethornis longirostris* (Long-billed Hermit) songs from the 2 example sound files included in this package ('lbh_1' and 'lbh_2'). These two files are clips extracted from the xeno-canto's '154138' and '154129' recordings respectively.

Source

Marcelo Araya-Salas, ohun

`merge_overlaps` *Merge overlapping selections*

Description

`merge_overlaps` merges several overlapping selections into a single selection

Usage

```
merge_overlaps(X, pb = TRUE, cores = 1)
```

Arguments

`X`	Data frame or 'selection.table' (following the warbleR package format) with selections (start and end of the sound events). Must contained at least the following columns: "sound.files", "selec", "start" and "end".
`pb`	Logical argument to control progress bar. Default is `TRUE`.
`cores`	Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).

Details

The function finds time-overlapping selection in reference tables and collapses them into a single selection. It can be useful to prepare reference tables to be used in an energy detection routine. In such cases overlapping selections are expected to be detected as a single sound. Therefore, merging them can be useful to prepare references in a format representing a more realistic expectation of how a perfect energy detection routine would look like.

Value

If any time-overlapping selection is found it returns a data frame in which overlapping selections are collapse into a single selection.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

`summarize_diagnostic`, `label_detection`

Examples

```
{
  # load data
  data("lbh_reference")

  # nothing to merge
  merge_overlaps(lbh_reference)

  # create artificial overlapping selections
  lbh_ref2 <- rbind(as.data.frame(lbh_reference[c(3, 10), ]), lbh_reference[c(3, 10), ])

  lbh_ref2$selec <- seq_len(nrow(lbh_ref2))

  merge_overlaps(lbh_ref2)
}
```

`optimize_energy_detector` *Optimize energy-based sound event detection*

Description

Optimize energy-based sound event detection under different correlation threshold values

Usage

```
optimize_energy_detector(reference, files = NULL, threshold = 5,
peak.amplitude = 0, hop.size = 11.6, wl = NULL, smooth = 5, hold.time = 0,
min.duration = NULL, max.duration = NULL, thinning = 1, cores = 1, pb = TRUE,
 by.sound.file = FALSE, bp = NULL, path = ".", previous.output = NULL, envelopes = NULL,
 macro.average = FALSE, min.overlap = 0.5)
```

Arguments

`reference`	Selection table (using the warbleR package's format, see `selection_table`) or data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of sound event (start and end). **It must 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 sound files with no target sound events. Supported file formats:'.wav', '.mp3', '.flac' and '.wac'. If not supplied the function will work on all sound files (in the supported format) in 'path'.
`threshold`	A numeric vector specifying the amplitude threshold for detecting sound events (in %). Default is 5. **Several values can be supplied for optimization**.

`peak.amplitude` Numeric vector of length 1 with the minimum peak amplitude value. A detection below that value would be excluded. Peak amplitude is the maximum sound pressure level (in decibels) across the sound event (see `sound_pressure_level`). This can be useful when expecting higher peak amplitude in the target sound events compared to non-target sound events or when keeping only the best examples of the target sound events (i.e. high precision and low recall). Default is 0. **Several values can be supplied for optimization**.

`hop.size` A numeric vector of length 1 specifying the time window duration (in ms). Default is 11.6 ms, which is equivalent to 512 wl for a 44.1 kHz sampling rate. Ignored if 'wl' is supplied.

`wl` A numeric vector of length 1 specifying the window length of the spectrogram. Default is `NULL`. If supplied, 'hop.size' is ignored. Used internally for bandpass filtering (so only applied when 'bp' is supplied).

`smooth` A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. It controls the time 'neighborhood' (in ms) in which amplitude samples are smoothed (i.e. averaged with neighboring samples). Default is 5. 0 means no smoothing is applied. Note that smoothing is applied before thinning (see 'thinning' argument). The function `envelope` is used internally which is analogous to sum smoothing in `env`. This argument is used internally by `get_envelopes`. **Several values can be supplied for optimization**.

`hold.time` Numeric vector of length 1. Specifies the time range (in ms) 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 ms) of the sound events to be detected. It removes sound events below that threshold. **Several values can be supplied for optimization**.

`max.duration` Numeric vector giving the longest duration (in ms) of the sound events to be detected. It removes sound events 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**.

`cores` 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` and 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. Bandpass is done using the function ffilter, which applies a short-term Fourier transformation to first create a spectrogram in which the target frequencies are filtered and then is back transformed into a wave object using a reverse Fourier transformation.

path — Character string containing the directory path where the sound files are located. The current working directory is used as default.

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.

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.

macro.average — Logical argument to control if diagnostics are first calculated for each sound file and then averaged across sound files, which can minimize the effect of unbalanced sample sizes between sound files. If FALSE (default) diagnostics are based on aggregated statistics irrespective of sound files. The following indices can be estimated by macro-averaging: overlap, mean.duration.true.positives, mean.duration.false.positives, mean.duration.false.positives, mean.duration.false.negatives, proportional.duration.true.positives, recall and precision (f.score is always derived from recall and precision). Note that when applying macro-averaging, recall and precision are not derived from the true positive, false positive and false negative values returned by the function.

min.overlap — Numeric. Controls the minimum amount of overlap required for a detection and a reference sound for it to be counted as true positive. Default is 0.5. Overlap is measured as intersection over union.

Details

This function takes a selections data frame or 'selection_table' ('reference') estimates the detection performance of a energy detector 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 sound events in 'reference' that correspond to any detection. 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 sound events in 'reference'. In a perfect detection routine it should be 0.
- false.negatives: number of sound events in 'reference' that were not detected (not found in 'detection'. In a perfect detection routine it should be 0.
- splits: number of detections overlapping reference sounds that also overlap with other detections. In a perfect detection routine it should be 0.
- merges: number of detections that overlap with two or more reference sounds. In a perfect detection routine it should be 0.
- mean.duration.true.positives: mean duration of true positives (in ms). Only included when time.diagnostics = TRUE.
- mean.duration.false.positives: mean duration of false positives (in ms). Only included when time.diagnostics = TRUE.
- mean.duration.false.negatives: mean duration of false negatives (in ms). Only included when time.diagnostics = TRUE.
- overlap: mean intersection over union overlap of true positives.
- proportional.duration.true.positives: ratio of duration of true positives to th duration of sound events in 'reference'. In a perfect detection routine it should be 1. Based only on true positives that were not split or merged. Only included when time.diagnostics = TRUE.
- duty.cycle: proportion of a sound file in which sounds were detected. Only included when time.diagnostics = TRUE and path is supplied.
- recall: Proportion of sound events in 'reference' that were detected. In a perfect detection routine it should be 1.
- precision: Proportion of detections that correspond to sound events in 'reference'. In a perfect detection routine it should be 1.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

Examples

```
# Save example files into temporary working directory
data("lbh1", "lbh2", "lbh_reference")
tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

# using smoothing and minimum duration
optimize_energy_detector(
  reference = lbh_reference, path = tempdir(),
  threshold = c(6, 10), smooth = 6.8, bp = c(2, 9), hop.size = 6.8,
  min.duration = 90
```

```
)

# with thinning and smoothing
optimize_energy_detector(
  reference = lbh_reference, path = tempdir(),
  threshold = c(6, 10, 15), smooth = c(7, 10), thinning = c(0.1, 0.01),
  bp = c(2, 9), hop.size = 6.8, min.duration = 90
)

# by sound file
(opt_ed <- optimize_energy_detector(
  reference = lbh_reference,
  path = tempdir(), threshold = c(6, 10, 15), smooth = 6.8, bp = c(2, 9),
  hop.size = 6.8, min.duration = 90, by.sound.file = TRUE
))

# summarize
summarize_diagnostic(opt_ed)

# using hold time
(op_ed <- optimize_energy_detector(
  reference = lbh_reference,
  threshold = 10, hold.time = c(100, 150), bp = c(2, 9), hop.size = 6.8,
  path = tempdir()
))

# including previous output in new call
optimize_energy_detector(
  reference = lbh_reference, threshold = 10,
  hold.time = c(50, 200), previous.output = op_ed, smooth = 6.8,
  bp = c(2, 9), hop.size = 7, path = tempdir()
)

# having and extra file in files (simulating a file that should have no detetions)
sub_reference <- lbh_reference[lbh_reference$sound.files != "lbh1.wav", ]

optimize_energy_detector(
  reference = sub_reference, files = unique(lbh_reference$sound.files),
  threshold = 10, hold.time = c(1, 150), bp = c(2, 9), smooth = 6.8,
  hop.size = 7, path = tempdir()
)
```

`optimize_template_detector`

Optimize acoustic template detection

Description

`optimize_template_detector` optimizes acoustic template detection

Usage

```
optimize_template_detector(template.correlations, reference, threshold,
cores = 1, pb = TRUE, by.sound.file = FALSE, previous.output = NULL,
macro.average = FALSE, min.overlap = 0.5)
```

Arguments

template.correlations	An object of class 'template_correlations' (generated by template_correlator) in which to optimize detections. Must contain data for all sound files as in 'reference'. It can also contain data for additional sound files. In this case the routine assumes that no sound events are found in those files, so detection from those files are all false positives.
reference	Data frame or 'selection.table' (following the warbleR package format) with the reference selections (start and end of the sound events) 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". **It must contain the reference selections that will be used for detection optimization**.
threshold	Numeric vector of length > 1 with values between 0 and 1 specifying the correlation threshold for detecting sound event occurrences (i.e. correlation peaks). Must be supplied. **Several values should be supplied for optimization**.
cores	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 to control if diagnostics are calculated for each sound file independently (TRUE) or for all sound files combined (FALSE, default).
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.
macro.average	Logical argument to control if diagnostics are first calculated for each sound file and then averaged across sound files, which can minimize the effect of unbalanced sample sizes between sound files. If FALSE (default) diagnostics are based on aggregated statistics irrespective of sound files. The following indices can be estimated by macro-averaging: overlap, mean.duration.true.positives, mean.duration.false.positives, mean.duration.false.positives, mean.duration.false.negatives, proportional.duration.true.positives, recall and precision (f.score is always derived from recall and precision). Note that when applying macro-averaging, recall and precision are not derived from the true positive, false positive and false negative values returned by the function.
min.overlap	Numeric. Controls the minimum amount of overlap required for a detection and a reference sound for it to be counted as true positive. Default is 0.5. Overlap is measured as intersection over union.

Details

This function takes a a reference data frame or 'selection_table' ('X') and the output of `template_correlator` and estimates the detection performance for different detection parameter combinations. This is done by comparing the position in time of the detection to those of the reference selections. 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 `template_detector`. Supported file formats:'.wav', '.mp3', '.flac' and '.wac'.

Value

A data frame in which each row shows the result of a detection job for each cutoff value, including the following diagnostic metrics:

- `true.positives`: number of sound events in 'reference' that correspond to any detection. 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 sound events in 'reference'. In a perfect detection routine it should be 0.
- `false.negatives`: number of sound events in 'reference' that were not detected (not found in 'detection'. In a perfect detection routine it should be 0.
- `splits`: number of detections overlapping reference sounds that also overlap with other detections. In a perfect detection routine it should be 0.
- `merges`: number of sound events in 'detection' that overlap with more than one sound event in 'reference'. In a perfect detection routine it should be 0.
- `recall`: Proportion of sound events in 'reference' that were detected. In a perfect detection routine it should be 1.
- `precision`: Proportion of detections that correspond to sound events 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., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

`optimize_energy_detector`, `template_correlator`, `template_detector`

Examples

```
{
# Save sound files to temporary working directory
data("lbh1", "lbh2", "lbh_reference")
tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

# template for the second sound file in 'lbh_reference'
templ <- lbh_reference[11, ]

# generate template correlations
tc <- template_correlator(templates = templ, path = tempdir(),
files = "lbh2.wav")

# using 2 threshold
optimize_template_detector(template.correlations = tc, reference =
lbh_reference[lbh_reference$sound.files == "lbh2.wav", ],
threshold = c(0.2, 0.5))

# using several thresholds
optimize_template_detector(template.correlations = tc,
reference = lbh_reference[lbh_reference$sound.files == "lbh2.wav", ],
 threshold = seq(0.5, 0.9, by = 0.05))

 # template for the first and second sound file in 'lbh_reference'
 templ <- lbh_reference[c(1, 11), ]

 # generate template correlations
 tc <- template_correlator(templates = templ, path = tempdir(),
 files = c("lbh1.wav", "lbh2.wav"))

optimize_template_detector(template.correlations = tc, reference =
  lbh_reference, threshold = seq(0.5, 0.7, by = 0.1))

 # showing diagnostics by sound file
 optimize_template_detector(template.correlations = tc, reference =
 lbh_reference,
 threshold = seq(0.5, 0.7, by = 0.1), by.sound.file = TRUE)
}
```

---

plot_detection *Plot detection and reference annotations*

Description

plot_detection evaluates the performance of a sound event detection procedure comparing the output selection table to a reference selection table

Usage

```
plot_detection(reference, detection, mid.point = FALSE, size = 20, positions = c(1, 2))
```

Arguments

reference	Data frame or 'selection.table' (following the warbleR package format) with the reference selections (start and end of the sound events) 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". **It must contain the reference selections that will be used for detection optimization**.
detection	Data frame or 'selection.table' with the detections (start and end of the sound events) that will be compared against the 'reference' selections. Must contained at least the following columns: "sound.files", "selec", "start" and "end". It can contain data for additional sound files not found in 'references'. In this case the routine assumes that no sound events are found in those files, so detection from those files are all false positives.
mid.point	Logical argument to control if each annotations is shown as a rectangle with fix width center at the mid point of the time position (if TRUE) or the true time range of the annotations is used (if FALSE, default). 'mid.point' can be useful to make visible annotations in very long sound files that would otherwise look to thin.
size	Numeric. Controls the size of the rectangles if mid.point = TRUE. Default is 20.
positions	Numeric. Controls the vertical position of the rectangles representing anotations. Default is c(1, 2). This can be used to get reference and detection annotations closer in the vertical axis. Note that the height of rectangles is 0.5.

Details

The function helps to visualize the match between reference and detection annotations by plotting them next to each other as rectangles along the time axis. If the annotations contain data for several sound files each sound file will be plotted in its own panel. The plot can be further modify by users using regular ggplot syntax.

Value

A ggplot graph (i.e. an object of class "ggplot").

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

label_spectro, diagnose_detection

Examples

```
{
  # load data
  data("lbh_reference")

  # mid point and regular size
  plot_detection(
    reference = lbh_reference[-14, ],
    detection = lbh_reference[-1, ], mid.point = TRUE
  )

  # mid point and larger size
  plot_detection(
    reference = lbh_reference[-14, ],
    detection = lbh_reference[-1, ], mid.point = TRUE, size = 25
  )

  # true time rectangles
  plot_detection(
    reference = lbh_reference[-14, ],
    detection = lbh_reference[-1, ]
  )

  # use position to make reference and anotations overlap vertically
  plot_detection(
    reference = lbh_reference[-14, ],
    detection = lbh_reference[-1, ], positions = c(1, 1.4)
  )

  # modified using ggplot
  gg_pd <- plot_detection(
    reference = lbh_reference[-14, ],
    detection = lbh_reference[-1, ], positions = c(1, 1.4)
  )

  gg_pd + ggplot2::theme_classic(base_size = 25)
}
```

---

split_acoustic_data *Splits sound files and associated annotations*

Description

split_acoustic_data splits sound files (and corresponding selection tables) in shorter segments

Usage

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

Arguments

path	Directory path where sound files are found. The current working directory is used as default.
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. Supported file formats:'.wav', '.mp3', '.flac' and '.wac'. If not supplied the function will work on all sound files (in the supported format) in 'path'.
cores	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. energy_detector).

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 (original.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. In this case the output data frame contains an additional column, 'split.sels', that inform users whether selections have been split into multiple clips ('split') or not (NA). Sound files in 'path' that are not referenced in 'X' will stil be split.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

cut_sels

Examples

```
{
  # load data and save to temporary working directory
  data("lbh1", "lbh2")
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
  tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

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

  # Check this folder
  tempdir()
}
```

---

summarize_acoustic_data

Summarize information about file format in an acoustic data set

---

Description

summarize_acoustic_data summarizes information about file format in an acoustic data set

Usage

```
summarize_acoustic_data(path = ".", digits = 2)
```

Arguments

path	Character string containing the directory path where the sound files are located. Default is "." (current working directory).
digits	Numeric vector of length 1 with the number of decimals to include. Default is 2.

Details

The function summarizes information about file format in an acoustic data set. It provides information about the number of files, file formats, sampling rates, bit depts, channels, duration and file size (in MB). For file format, sampling rate, bit depth and number of channels the function includes information about the number of files for each format (e.g. '44.1 kHz (2)' means 2 files with a sampling rate of 44.1 kHz).

Value

The function prints a summary of the format of the files in an acoustic data set.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

summarize_reference

Examples

```
{
  # load data and save example files into temporary working directory
  data("lbh1", "lbh2", "lbh_reference")
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
  tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

  # summary across sound files
  summarize_acoustic_data(path = tempdir())
}
```

---

summarize_diagnostic *Summarize detection diagnostics*

Description

summarize_diagnostic summarizes detection diagnostics

Usage

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

Arguments

diagnostic	A data frame with the output of a detection optimization function (diagnose_detection, optimize_energy_detector or optimize_template_detector)
time.diagnostics	Logical argument to control if diagnostics related to the duration of the sound events ("mean.duration.true.positives", "mean.duration.false.positives", "mean.duration.false.negatives" and "proportional.duration.true.positives") are returned (if TRUE). Default is FALSE.
macro.average	Logical argument to control if diagnostics are first calculated for each sound file and then averaged across sound files, which can minimize the effect of unbalanced sample sizes between sound files. If FALSE (default) diagnostics are based on aggregated statistics irrespective of sound files. The following indices can be estimated by macro-averaging: overlap, mean.duration.true.positives, mean.duration.false.positives, mean.duration.false.positives, mean.duration.false.negatives, proportional.duration.true.positives, recall and precision (f.score is always derived from recall and precision). Note that when applying macro-averaging, recall and precision are not derived from the true positive, false positive and false negative values returned by the function.

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. 'splits' and 'merge.positives' are also counted (i.e. counted twice) as 'true.positives'. Therefore "true.positives + false.positives = detections".

Value

A data frame, similar to the output of a detection optimization function (diagnose_detection, optimize_energy_detector, optimize_template_detector) including the following detection performance diagnostics:

- detections: total number of detections
- true.positives: number of sound events in 'reference' that correspond to any detection. 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 (i.e. don't overlap with) any of the sound events in 'reference'. In a perfect detection routine it should be 0.
- false.negatives: number of sound events in 'reference' that were not detected (not found in 'detection'. In a perfect detection routine it should be 0.
- splits: number of detections overlapping reference sounds that also overlap with other detections. In a perfect detection routine it should be 0.
- merges: number of detections that overlap with two or more reference sounds. 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 ms). Only included when time.diagnostics = TRUE.

- `mean.duration.false.negatives`: mean duration of false negatives (in ms). Only included when `time.diagnostics = TRUE`.
- `overlap`: mean intersection over union overlap of true positives.
- `proportional.duration.true.positives`: ratio of duration of true positives to the duration of sound events in 'reference'. In a perfect detection routine it should be 1. Based only on true positives that were not split or merged.
- `duty.cycle`: proportion of a sound file in which sounds were detected. Only included when `time.diagnostics = TRUE` and `path` is supplied. Useful when conducting energy-based detection as a perfect detection can be obtained with a very low amplitude threshold, which will detect everything, but will produce a duty cycle close to 1.
- `recall`: Proportion of sound events in 'reference' that were detected. In a perfect detection routine it should be 1.
- `precision`: Proportion of detections that correspond to sound events in 'reference'. In a perfect detection routine it should be 1.
- `f.score`: Combines recall and precision as the harmonic mean of these two. Provides a single value for evaluating performance. In a perfect detection routine it should be 1.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. Mesaros, A., Heittola, T., & Virtanen, T. (2016). Metrics for polyphonic sound event detection. Applied Sciences, 6(6), 162.

See Also

diagnose_detection

Examples

```
{
  # load example selection tables

  data("lbh_reference")

  # run diagnose_detection() by sound file
  diag <- diagnose_detection(
    reference = lbh_reference,
    detection = lbh_reference[-1, ], by.sound.file = TRUE
  )

  # summarize
  summarize_diagnostic(diagnostic = diag)

  # should be the same as this:
```

```
  diagnose_detection(
    reference = lbh_reference,
    detection = lbh_reference[-1, ], by.sound.file = FALSE
  )
}
```

---

summarize_reference *Summarize temporal and frequency dimensions of annotations and gaps*

Description

summarize_reference summarizes temporal and frequency dimensions of annotations and gaps

Usage

```
summarize_reference(reference, path = NULL, by.sound.file = FALSE,
units = c("ms", "kHz"), digits = 2)
```

Arguments

reference	Data frame or 'selection.table' (following the warbleR package format) with the reference selections (start and end of the sound events) 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". If frequency range columns are included ("bottom.freq" and "top.freq") these are also used to characterize reference selections.
path	Character string containing the directory path where the sound files are located. If supplied then duty cycle and peak frequency features are returned. These features are more helpful for tuning a energy-based detection. Default is NULL.
by.sound.file	Logical argument to control whether features are summarized across sound files (when by.sound.file = FALSE, and more than 1 sound file is included in 'reference') or shown separated by sound file. Default is FALSE.
units	A character vector of length 2 with the units to be used for time and frequency parameters, in that order. Default is c("ms", "kHz"). It can also take 's' and 'Hz'.
digits	Numeric vector of length 1 with the number of decimals to include. Default is 2.

Details

The function extracts quantitative features from reference tables that can inform the range of values to be used in a energy-based detection optimization routine. Features related to selection duration can be used to set the 'max.duration' and 'min.duration' values, frequency related features can inform bandpass values, gap related features inform hold time values and duty cycle can be used to evaluate performance.

Value

The function returns the mean, minimum and maximum duration of selections and gaps (time intervals between selections) and of the number of annotations by sound file. If frequency range columns are included in the reference table (i.e. "bottom.freq" and "top.freq") the minimum bottom frequency ('min.bottom.freq') and the maximum top frequency ('max.top.freq') are also estimated. Finally, if the path to the sound files in 'reference' is supplied the duty cycle (fraction of a sound file corresponding to target sound events) and peak amplitude (highest amplitude in a detection) are also returned. If 'by.sound.file = FALSE' a matrix with features in rows is returned. Otherwise a data frame is returned in which each row correspond to a sound file. By default, time features are returned in 'ms' while frequency features in 'kHz' (but see 'units' argument).

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

optimize_energy_detector, optimize_template_detector

Examples

```
{
  # load data and save example files into temporary working directory
  data("lbh1", "lbh2", "lbh_reference")
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
  tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

  # summary across sound files
  summarize_reference(reference = lbh_reference, path = tempdir())

  # summary across sound files
  summarize_reference(reference = lbh_reference, by.sound.file = TRUE, path = tempdir())
}
```

---

template_correlator *Acoustic templates correlator using time-frequency cross-correlation*

Description

template_correlator estimates templates cross-correlation across multiple sound files.

Usage

```
template_correlator(templates, files = NULL, hop.size = 11.6, wl = NULL, ovlp = 0,
wn ='hanning', cor.method = "pearson", cores = 1, path = ".",
pb = TRUE, type = "fourier", fbtype = "mel", ...)
```

Arguments

templates	'selection_table', 'extended_selection_table' (warbleR package's formats, see selection_table) or data frame with time and frequency information of the sound event(s) to be used as templates (1 template per row). The object must contain columns for sound files (sound.files), selection number (selec), and start and end time of sound event (start and end). If frequency range columns are included ('bottom.freq' and 'top.freq', in kHz) the correlation will be run on those frequency ranges. All templates must have the same sampling rate and both templates and 'files' (in which to find templates) must also have the same sampling rate.
files	Character vector with name of the files in which to run the cross-correlation with the supplied template(s). Supported file formats:'.wav', '.mp3', '.flac' and '.wac'. If not supplied the function will work on all sound files (in the supported formats) in 'path'.
hop.size	A numeric vector of length 1 specifying the time window duration (in ms). Default is 11.6 ms, which is equivalent to 512 wl for a 44.1 kHz sampling rate. Ignored if 'wl' is supplied.
wl	A numeric vector of length 1 specifying the window length of the spectrogram. Default is NULL. If supplied, 'hop.size' is ignored.
ovlp	Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in spectro. Default is 0. High values of ovlp slow down the function but may produce more accurate results.
wn	A character vector of length 1 specifying the window name as in ftwindow.
cor.method	A character vector of length 1 specifying the correlation method as in cor.
cores	Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).
path	Character string containing the directory path where the sound files are located. The current working directory is used as default.
pb	Logical argument to control progress bar. Default is TRUE.
type	A character vector of length 1 specifying the type of cross-correlation: "fourier" (i.e. spectrographic cross-correlation using Fourier transform; internally using spectro; default), "mfcc" (auditory scale coefficient matrix cross-correlation; internally using melfcc) or "mel-auditory" (cross-correlation of auditory spectrum, i.e. spectrum after transformation to an auditory scale; internally using melfcc). The argument 'fbtype' controls the auditory scale to be used. Note that the last 2 methods have not been widely used in this context so can be regarded as experimental.
fbtype	Character vector indicating the auditory frequency scale to use: "mel", "bark", "htkmel", "fcmel".

... Additional arguments to be passed to melfcc for further customization when using auditory scales.

Details

This function calculates the similarity of acoustic templates across sound files by means of time-frequency cross-correlation. Fourier spectrograms or time-frequency representations from auditory scales (including cepstral coefficients) can be used. Several templates can be run over several sound files. Note that template-based detection is divided in two steps: template correlation (using this function) and template detection (or peak detection as it infers detection based on peak correlation scores, using the function template_detector). So the output of this function (and object of 'template_correlations') must be input into template_detector for inferring sound event occurrences. optimize_template_detector can be used to optimize template detection.

Value

The function returns an object of class 'template_correlations' which is a list with the correlation scores for each combination of templates and files. 'template_correlations' objects must be used to infer sound event occurrences using template_detector or to graphically explore template correlations across sound files using full_spectrograms.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

Khanna H., Gaunt S.L.L. & McCallum D.A. (1997). Digital spectrographic cross-correlation: tests of recall. Bioacoustics 7(3): 209-234.

Lyon, R. H., & Ordubadi, A. (1982). Use of cepstra in acoustical signal analysis. Journal of Mechanical Design, 104(2), 303-306.

See Also

energy_detector, template_detector, optimize_template_detector

Examples

```
{
  # load example data
  data("lbh1", "lbh2", "lbh_reference")

  # save sound files
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
  tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

  # create template
```

```
templ <- lbh_reference[4, ]
templ2 <- warbleR::selection_table(templ,
  extended = TRUE,
  path = tempdir()
)

# fourier spectrogram
(tc_fr <- template_correlator(templates = templ, path = tempdir(), type = "fourier"))

# mel auditory spectrograms
(tc_ma <- template_correlator(templates = templ, path = tempdir(), type = "mel-auditory"))

# mfcc spectrograms
(tc_mfcc <- template_correlator(templates = templ, path = tempdir(), type = "mfcc"))

# similar results (but no exactly the same) are found with the 3 methods
# these are the correlation of the correlation vectors
# fourier vs mel-auditory
cor(
  tc_fr$`lbh2.wav-4/lbh2.wav`$correlation.scores,
  tc_ma$`lbh2.wav-4/lbh2.wav`$correlation.scores
)

# fourier vs mfcc
cor(
  tc_fr$`lbh2.wav-4/lbh2.wav`$correlation.scores,
  tc_mfcc$`lbh2.wav-4/lbh2.wav`$correlation.scores
)

# mel-auditory vs mfcc
cor(
  tc_ma$`lbh2.wav-4/lbh2.wav`$correlation.scores,
  tc_mfcc$`lbh2.wav-4/lbh2.wav`$correlation.scores
)

# using an extended selection table
templ_est <- warbleR::selection_table(templ,
  extended = TRUE,
  path = tempdir()
)

tc_fr_est <- template_correlator(templates = templ_est, path = tempdir(), type = "fourier")

# produces the same result as templates in a regular data frame
cor(
  tc_fr$`lbh2.wav-4/lbh2.wav`$correlation.scores,
  tc_fr_est$`lbh2.wav_4-1/lbh2.wav`$correlation.scores
)
}
```

`template_detector` *Acoustic template detection from time-frequency cross-correlations*

Description

`template_detector` find sound event occurrences in cross-correlation vectors from `template_correlator`

Usage

```
template_detector(template.correlations, cores = 1, threshold, pb = TRUE,
 verbose = TRUE)
```

Arguments

`template.correlations`	object of class 'template_correlations' generated by `template_correlator` containing the correlation score vectors.
`cores`	Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).
`threshold`	Numeric vector of length 1 with a value between 0 and 1 specifying the correlation threshold for detecting sound event occurrences (i.e. correlation peaks). Must be supplied. Correlation scores are forced to between 0 and 1 (by converting negative scores to 0). 0 and 1 represent the lowest and highest similarity to the template respectively.
`pb`	Logical argument to control progress bar. Default is `TRUE`.
`verbose`	Logical argument to control if some summary messages are printed to the console.

Details

This function infers sound events occurrences from cross-correlation scores along sound files. Correlation scores must be generated first using `template_correlator`. The output is a data frame (or selection table if sound files are still found in the original path supplied to `template_correlator`, using the warbleR package's format, see `selection_table`) containing the start and end of the detected sound events as well as the cross-correlation score ('scores' column) for each detection. **Note that the detected sounds are assumed to have the same duration as the template, so their start and end correspond to the correlation peak position +/- half the template duration**.

Value

The function returns a 'selection_table' (warbleR package's formats, see `selection_table`) or data frame (if sound files can't be found) with the start and end and correlation score for the detected sound events.

Author(s)

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

References

Araya-Salas, M., Smith-Vidaurre, G., Chaverri, G., Brenes, J. C., Chirino, F., Elizondo-Calvo, J., & Rico-Guevara, A. 2022. ohun: an R package for diagnosing and optimizing automatic sound event detection. BioRxiv, 2022.12.13.520253. https://doi.org/10.1101/2022.12.13.520253

See Also

energy_detector, template_correlator, optimize_template_detector

Examples

```
{
  # load example data
  data("lbh1", "lbh2", "lbh_reference")

  # save sound files
  tuneR::writeWave(lbh1, file.path(tempdir(), "lbh1.wav"))
  tuneR::writeWave(lbh2, file.path(tempdir(), "lbh2.wav"))

  # template for the first sound file in 'lbh_reference'
  templ1 <- lbh_reference[1, ]

  # generate template correlations
  tc <- template_correlator(templates = templ1, path = tempdir(), files = "lbh1.wav")

  # template detection
  td <- template_detector(template.correlations = tc, threshold = 0.4)

  # diagnose detection
  diagnose_detection(
    reference =
      lbh_reference[lbh_reference$sound.files == "lbh1.wav", ],
    detection = td
  )

  # template for the second and third sound file in 'lbh_reference'
  # which have similar song types
  templ2 <- lbh_reference[4, ]

  # generate template correlations
  tc <- template_correlator(
    templates = templ2, path = tempdir(),
    files = c("lbh1.wav", "lbh2.wav")
  )

  # template detection
  td <- template_detector(template.correlations = tc, threshold = 0.3)

  # diagnose detection
  diagnose_detection(reference = lbh_reference, detection = td)
}
```

Index