Package ‘warbleR’

October 12, 2022

Type Package

Title Streamline Bioacoustic Analysis

Version 1.1.27

Date 2022-02-22

Description Functions aiming to facilitate the analysis of the structure of animal acoustic signals in 'R'. ‘warbleR’ makes use of the basic sound analysis tools from the package 'see-wave', and offers new tools for acoustic structure analysis. The main features of the package are the use of loops to apply tasks through acoustic signals referenced in a selection (annotation) table and the production of spectrograms in image files that allow to organize data and verify acoustic analyzes. The package offers functions to explore, organize and manipulate multiple sound files, explore and download 'Xeno-Canto' recordings, detect signals automatically, create spectrograms of complete recordings or individual signals, run different measures of acoustic signal structure, evaluate the performance of measurement methods, catalog signals, characterize different structural levels in acoustic signals, run statistical analysis of duet coordination and consolidate databases and annotation tables, among others.

License GPL (>= 2)

Imports dtw, fftw, graphics, grDevices, monitoR, parallel, pbapply, R Curl, rjson, stats, utils, methods, Rcpp, knitr, crayon, bioacoustics, stringi

LinkingTo Rcpp

Depends R (>= 3.2.1), tuneR, seewave (>= 2.0.1), NatureSounds

LazyData TRUE

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

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

NeedsCompilation yes

Suggests ggplot2, rmarkdown, jpeg, ape, soundgen, wavethresh, png, pracma, Sim.DiffProc, maps, leaflet, kableExtra

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.1.1

Encoding UTF-8
R topics documented:

auto_detec ......................................................... 3
catalog ............................................................. 6
catalog2pdf .......................................................... 11
check_sound_files .................................................. 13
color_spectro ....................................................... 14
compare_methods ................................................... 17
comp_matrix ......................................................... 21
consolidate .......................................................... 22
cross_correlation ................................................... 24
duration_sound_files .............................................. 27
evelope ............................................................... 28
filter_sels ............................................................ 29
find_annotations ..................................................... 31
find_clipping ......................................................... 32
find_peaks ............................................................. 33
fix_extended_selection_table .................................... 35
fix_wavs .............................................................. 36
freq_DTW ............................................................. 37
freq_range ........................................................... 40
freq_range_detec .................................................... 43
freq_ts ............................................................... 45
full_spectrogram2pdf ............................................. 48
full_spectrograms .................................................. 49
gaps ................................................................. 52
image_to_wave ...................................................... 53
inflections ............................................................ 55
info_sound_files .................................................... 56
is_extended_selection_table ...................................... 58
is_selection_table .................................................. 59
lbh_selec_table ..................................................... 60
lbh_selec_table2 ..................................................... 61
map_xc .............................................................. 62
mfcc_stats ............................................................ 63
move_imgs ........................................................... 65
mp32wav ............................................................. 67
multi_DTW ........................................................... 68
auto_detec automatically detects the start and end of vocalizations in sound files based on amplitude, duration, and frequency range attributes.

| auto_detec | auto_detec automatically detects the start and end of vocalizations in sound files based on amplitude, duration, and frequency range attributes. |

Index

| auto_detec | auto_detec automatically detects the start and end of vocalizations in sound files based on amplitude, duration, and frequency range attributes. |

Description

auto_detec automatically detects the start and end of vocalizations in sound files based on amplitude, duration, and frequency range attributes.
Usage

auto_detec(X = NULL, wl = 512, threshold = 15, parallel = 1, power = 1, output = 'data.frame', thinning = 1, path = NULL, pb = TRUE, ssmooth = 0, bp = NULL, flist = NULL, hold.time = 0, mindur = NULL, maxdur = NULL, envt = NULL, msmooth = NULL,osci = NULL, xl = NULL, picsize = NULL, res = NULL, flim = NULL, ls = NULL, sxrow = NULL, rows = NULL, redo = NULL, img = NULL, it = NULL, set = NULL, smadj = NULL, pal = NULL, fast.spec = NULL)

Arguments

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 provided the detection will be conducted only within the selections in 'X'. Alternatively, an 'autodetec.output' object can be input. These objects are also generated by this function when output = "list". If so the detection runs much faster as envelopes have been already calculated.

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.

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

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).

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).

output
Character string indicating if the output should be a 'data.frame' with the detections (default) or a list (of class 'autodetec.output') containing both 1) the detections and 2) the amplitude envelopes (time vs amplitude) for each sound file. The list can be input into full_spectrograms to explore detections and associated amplitude envelopes.

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.

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

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).
auto_detec

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

flist
character vector or factor indicating the subset of files that will be analyzed. Ignored if X is provided.

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 into a single selection). Default is 0.

mindur
Numeric vector of length 1 giving the shortest duration (in seconds) of the signals to be detected. It removes signals below that threshold.

maxdur
Numeric vector of length 1 giving the longest duration (in seconds) of the signals to be detected. It removes signals above that threshold.

envt
DEPRECATED.

msmooth
DEPRECATED.

osci
DEPRECATED.

xl
DEPRECATED.

picsize
DEPRECATED.

res
DEPRECATED.

flim
DEPRECATED.

ls
DEPRECATED.

sxrow
DEPRECATED.

rows
DEPRECATED.

redo
DEPRECATED.

img
DEPRECATED.

it
DEPRECATED.

set
DEPRECATED.

smadj
DEPRECATED.

pal
DEPRECATED.

fast.spec
DEPRECATED.

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
A data frame containing the start and end of each signal by sound file and selection number. If 'output = "list"' then a list including 1) a detection data frame, 2) amplitude envelopes and 3) parameters will be return. An additional column 'org.selec' is added when 'X' is provided (so detection can be traced back to the selections in 'X').
catalog

Create catalog of vocal signals

description

catalog produces spectrograms of selections (signals) split into multiple rows and columns.

Usage

catalog(X, flim = NULL, nrow = 4, ncol = 3, same.time.scale = TRUE,
collevels = seq(-40, 0, 1), ovlp = 50, parallel = 1, mar = 0.05, prop.mar = NULL,
lab.mar = 1, w1 = 512, wn = "hanning", gr = FALSE, pal = reverse.gray.colors.2,
it = "jpeg", path = NULL, pb = TRUE, fast.spec = FALSE, res = 100,
orientation = "v", labels = c("sound.files", "selec"), height = NULL,
width = NULL, tags = NULL, tag.pal = list(temp.colors, heat.colors, topo.colors),
legend = 3, cex = 1, leg.wd = 1, img.suffix = NULL, img.prefix = NULL,
tag.widths = c(1, 1), hatching = 0, breaks = c(5, 5), group.tag = NULL,
spec.mar = 0, spec.bg = "white", max.group.cols = NULL, sub.legend = FALSE,
rm.axes = FALSE, title = NULL, by.row = TRUE, box = TRUE)
Arguments

X  'selection_table', 'extended_selection_table' or data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end). Default is NULL.

flim  A numeric vector of length 2 indicating the highest and lowest frequency limits (kHz) of the spectrogram, as in \texttt{spectro}. Default is NULL.

nrow  A numeric vector of length 1. Specifies number of rows. Default is 4.

ncol  A numeric vector of length 1. Specifies number of columns. Default is 3.

same.time.scale  Logical. Controls if all spectrograms are in the same time scale (i.e. have the same duration).

collevels  A numeric vector of length 3. Specifies levels to partition the amplitude range of the spectrogram (in dB). The more levels the higher the resolution of the spectrogram. Default is seq(-40, 0, 1). seq(-115, 0, 1) will produces spectrograms similar to other acoustic analysis software packages.

ovlp  Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in \texttt{spectro}. Default is 50. High values of ovlp slow down the function but produce more accurate selection limits (when X is provided).

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).

mar  Numeric vector of length 1. Specifies the margins (in seconds) adjacent to the start and end points of selections, delineating spectrogram limits. Default is 0.05.

prop.mar  Numeric vector of length 1. Specifies the margins adjacent to the start and end points of selections as a proportion of the duration of the signal. If provided 'mar' argument is ignored. Default is NULL. Useful when having high variation in signal duration. Ignored if same.time.scale = FALSE.

lab.mar  Numeric vector of length 1. Specifies the space allocated to labels and tags (the upper margin). Default is 1.

wl  A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.

wn  Character vector of length 1 specifying the window function name. See \texttt{ftwindow} for name options. Default is "hanning".

gr  Logical vector to add grid to spectrogram. Default is FALSE.

pal  Color palette function for spectrogram. Default is reverse.gray.colors.2. See \texttt{spectro} for more palettes. Palettes as \texttt{gray.2} may work better when \texttt{fast.spec} = TRUE.

it  A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".

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

pb  Logical argument to control progress bar. Default is TRUE.
Logical. If TRUE then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors gray.1, gray.2, gray.3, topo.1 and rainbow.1 (which should be imported from the package monitoR) seem to work better with ’fast.spec’ spectrograms. Palette colors gray.1, gray.2, gray.3 offer decreasing darkness levels.

Numeric argument of length 1. Controls image resolution. Default is 100 (faster) although 300 is recommended for publication/presentation quality. Note that high resolution produce significantly bigger image files. This could be problematic when creating pdf files using catalog.

String. Indicates whether a letter page size image is produced in vertical (’v’ option) or horizontal orientation (’h’ option). Note that width and height can also be specified.

String vector. Provides the column names that will be used as labels above the corresponding spectrograms.

Numeric. Single value (in inches) indicating the height of the output image files. Default is 11 for vertical orientation.

Numeric. Single value (in inches) indicating the width of the output image files. Default is 8.5 for vertical orientation.

String vector. Provides the column names that will be used for the color tagging legend above. Tags can also be numeric. Continuous variables would be break down in 10 color classes.

List of color palette function for tags. Should be of length 1, 2 or 3. Default is list(temp.colors, heat.colors, topo.colors).

A numeric vector of length 1 controlling a legend for color tags is added. Ignored if no tags are provided. Four values are allowed:

- 0: No label
- 1: Label for the first color tag
- 2: Label for the second color tag
- 3: Labels both color tags

Default is 3. Currently no legend can be set for group tags. Use labels instead.

A numeric vector of length 1 giving the amount by which text (including labels and axis) should be magnified. Default is 1.

Numeric. Controls the width of the legend column. Default is 1.

A character vector of length 1 with a suffix (label) to add at the end of the names of image files. Default is NULL (no suffix). Useful to label catalogs from different individuals, species or sites.

A character vector of length 1 with a prefix (label) to add at the beginning of the names of image files. Default is NULL (no prefix). Useful to label catalogs from different individuals, species or sites and ensure they will be grouped together when sorted by file name.

A numeric vector of length 2 to control the relative width of the color tags (when 2 tags are provided).
hatching

A numeric vector of length 1 controlling cross-hatching is used for color tags. Several cross-hatching patterns are used to make tags with similar colors more distinguishable. Four values are allowed:

- 0: No cross-hatching
- 1: Cross-hatching the first color tag
- 2: Cross-hatching the second color tag
- 3: Cross-hatching both color tags

breaks

Numeric vector of length 1 or 2 controlling the number of intervals in which a numeric tag will be divided. The numbers control the first and second tags respectively. Ignored if tags are not numeric. Default is c(5, 5).

group.tag

Character vector of length 1 indicating the column name to be used to color the empty plot areas around the spectrograms. If provided selections that belong to the same tag level are clumped together in the catalog (the 'X' data frame is sorted by that column). This tags cannot be included in the legend so it would be better to use the label field to identify the different levels.

spec.mar

Numeric vector of length 1 to add space at the top, left and right sides of the spectrogram. Useful to better display the grouping of selections when 'group.tag' is provided. Internally applied for setting 'mar' using `par`.

spec.bg

Character vector of length 1 to control the background color of the spectrogram. Default is 'white'. Ignored if `group.tag = NULL`.

max.group.cols

Numeric vector of length 1 indicating the number of different colors that will be used for group tags (see 'group.tag' argument). If provided (and the number is smaller than the number of levels in the 'group.tag' column) the colors will be recycled, although ensuring that adjacent groups do not share the same color. Useful when the 'group.tag' has many levels and the colors assigned become very similar. Default is NULL.

sub.legend

Logical. If TRUE then only the levels present on each page are shown in the legend. Default is FALSE.

rm.axes

Logical. If TRUE frequency and time axes are excluded. Default is FALSE.

title

Character vector of length 1 to set the title of catalogs.

by.row

Logical. If TRUE (default) catalogs are filled by rows.

box

Logical. If TRUE (default) a box is drawn around spectrograms and corresponding labels and tags.

Details

This function aims to simplify the visual exploration of multiple vocalizations. The function plots a matrix of spectrograms from a selection table. Spectrograms can be labeled or color tagged to facilitate exploring variation related to a parameter of interest (e.g. location, song type). A legend will be added to help match colors with tag levels (if legend is > 0). Different color palettes can be used for each tag. Numeric tags are split in intervals (the number of intervals can be controlled with break argument). The width and height can also be adjusted to fit more column and/or rows. This files can be put together in a single pdf file with `catalog2pdf`. We recommend using low resolution (~60-100) and smaller dimensions (width & height < 10) if aiming to generate pdfs (otherwise pdfs could be pretty big).
Catalog

Value

Image files with spectrograms of whole sound files in the working directory. Multiple pages can be returned, depending on the length of each sound file.

Author(s)

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

References


See Also

blog post on catalogs, blog post on customizing catalogs, catalog2pdf

Examples

```r
## Not run:
# save sound file examples
data(list = c("Phae.long1", "Phae.long2","lbh_selec_table"))
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"))

catalog(X = lbh_selec_table, flim = c(1, 10), nrow = 4, ncol = 2, same.time.scale = T, ovlp = 90, parallel = 1, mar = 0.01, wl = 200, gr = FALSE, orientation = "v", labels = c("sound.files", "selec"), legend = 0, path = tempdir())

different time scales and tag palette
catalog(X = lbh_selec_table, flim = c(1, 10), nrow = 4, ncol = 2, same.time.scale = F, ovlp = 90, parallel = 1, mar = 0.01, wl = 200, orientation = "v", labels = c("sound.files", "selec"), legend = 0, tag.pal = list(terrain.colors), path = tempdir())

#adding tags and changing spectro palette
catalog(X = lbh_selec_table, flim = c(1, 10), nrow = 4, ncol = 2, same.time.scale = F, ovlp = 90, parallel = 1, mar = 0.01, wl = 200, pal = reverse.heat.colors, orientation = "v", labels = c("sound.files", "selec"), legend = 1, tag.pal = list(terrain.colors), tags = "sound.files", path = tempdir())

#create a bigger selection table
X <- rbind(lbh_selec_table, lbh_selec_table, lbh_selec_table, lbh_selec_table)
X <- rbind(X, X)

#create some simulated labels
```

X$songtype <- sample(letters[13:15], nrow(X), replace = T)
X$indiv <- sample(letters[1:12], nrow(X), replace = T)

# 12 columns in 5 rows, 2 tags
catalog(X = X, flim = c(1, 10), nrow = 5, ncol = 12, same.time.scale = F,
    ovlp = 90, parallel = 1, mar = 0.01, w1 = 200,
    orientation = "v", labels = c("sound.files", "selec"), legend = 3,
    collevels = seq(-65, 0, 5), tag.pal = list(terrain.colors), tags = c("songtype", "indiv"),
    path = tempdir())

# with legend
catalog(X = X, flim = c(1, 10), nrow = 5, ncol = 12, same.time.scale = F,
    ovlp = 90, parallel = 1, mar = 0.01, w1 = 200, gr = FALSE,
    orientation = "v", labels = c("sound.files", "selec"), legend = 3,
    width = 20, collevels = seq(-65, 0, 5), tag.pal = list(terrain.colors),
    tags = c("songtype", "indiv"),
    path = tempdir())

# horizontal orientation
catalog(X = X, flim = c(1, 10), nrow = 5, ncol = 12, same.time.scale = F,
    ovlp = 90, parallel = 1, mar = 0.01, w1 = 200, gr = FALSE,
    orientation = "h", labels = c("sound.files", "selec"), legend = 3,
    width = 20, collevels = seq(-65, 0, 5), tag.pal = list(terrain.colors),
    tags = c("songtype", "indiv"),
    path = tempdir())

catalog2pdf(keep.img = TRUE, overwrite = FALSE, parallel = 1, path = NULL,
    pb = TRUE, by.img.suffix = FALSE, ...)

Arguments

keep.img Logical argument. Indicates whether jpeg files should be kept (default) or remove.
overwrite Logical argument. If TRUE all jpeg pdf will be produced again when code is rerun. If FALSE only the ones missing will be produced. Default is FALSE.

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).

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

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

by.img.suffix Logical. If TRUE catalogs with the same image suffix will be put together in a single pdf (so one pdf per image suffix in the catalog images). Default is FALSE (i.e. no suffix).

... Additional arguments to be passed to the internal pdf creating function pdf for customizing output.

Details

The function combines catalog images in .jpeg format from the catalog function into pdfs. Images must be saved in .jpeg format. Note that using lower resolution and smaller dimension (width and height) when creating catalogs will substantially decrease the size of pdf files (which could be pretty big).

Author(s)

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

References


See Also

full_spectrogram2pdf, blog post on catalogs

Examples

## Not run:
# save sound file examples
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"))
catalog(X = lbh_selec_table, nrow = 2, ncol = 4)

# now create single pdf removing jpeg
catalog2pdf(keep.img = FALSE, path = tempdir())

# check this floder
tempdir()

## End(Not run)
check_sound_files  

Description

check_sound_files checks whether sound files can be read by subsequent functions.

Usage

check_sound_files(X = NULL, path = NULL)

Arguments

- **X**: Optional, 'selection_table' object or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. If provided the function also returns the smallest number of samples from the listed selections, which limits the minimum window length (wl argument in other functions) that can be used in batch analyses. This could be useful for avoiding errors in downstream functions (e.g. spectro_analysis).
- **path**: Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.

Details

This function checks if sound files in the working directory can be read. Users must set the working directory where they wish to check sound files beforehand. If X is provided it also returns the smallest number of samples from the selections listed in X (if all files can be read). Note that corrupt files can be fixed using fix_wavs (‘sox’ must be installed to be able to run this function). The function is intended for a "quick and dirty" check of the sound files in a selections data frame. For a more thorough analysis see check_sels.

Value

If all sound files are ok, returns message "All files can be read". Otherwise returns the names of the corrupted sound files.

Author(s)

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

References

See Also

color_spectro

tailor_sels

color_spectro

Highlight spectrogram regions

description

color_spectro highlights spectrogram regions specified by users

Usage

color_spectro(wave, wl = 512, wn = "hanning", ovlp = 70,
dB = "max0", collevels = NULL, selec.col = "red2", col.clm = NULL,
base.col = "black", bg.col = "white", strength = 1,
cexlab = 1, cexaxis = 1, tlab = "Time (s)", flab = "Frequency (kHz)",
title = NULL, axisX = TRUE, axisY = TRUE, flim = NULL,
rm.zero = FALSE, X = NULL, fast.spec = FALSE, t.mar = NULL,
f.mar = NULL, interactive = NULL, add = FALSE)

Arguments

wave A 'wave' object produced by readWave or similar functions.
wl A numeric vector of length 1 specifying the window length of the spectrogram. Default is 512.
wn Character vector of length 1 specifying window name. Default is "hanning". See function ftwindow for more options.
ovlp Numeric vector of length 1 specifying the percent overlap between two consecutive windows, as in spectro. Default is 70.
dB     Character vector of length 1 controlling the amplitude weights as in spectro. Default is 'max0'.
collevels Numeric. Levels used to partition amplitude range as in spectro. Default is NULL.
selec.col Character vector of length 1 specifying the color to be used to highlight selection. See 'col.clm' for specifying unique colors for each selection. Default is 'red2'. Ignored if 'col.clm' and 'X' are provided.
col.clm Character vector of length 1 indicating the name of the column in 'X' that contains the color names for each selection. Ignored if X == NULL or interactive != NULL. Default is NULL.
base.col Character vector of length 1 specifying the color of the background spectrogram. Default is 'black'.
bg.col Character vector of length 1 specifying the background color for both base and highlighted spectrograms. Default is 'white'.
strength Numeric vector of length 1 controlling the strength of the highlighting color (actually how many times it is repeated in the internal color palette). Must be a positive integer. Default is 1.
cexlab Numeric vector of length 1 specifying the relative size of axis labels. See spectro. Default is 1.
cexaxis Numeric vector of length 1 specifying the relative size of axis. See spectro. Default is 1.
tlab Character vector of length 1 specifying the label of the time axis.
flab Character vector of length 1 specifying the label of the frequency axis.
title Logical argument to add a title to individual spectrograms. Default is TRUE.
axisX Logical to control whether time axis is plotted. Default is TRUE.
axisY Logical to control whether frequency axis is plotted. Default is TRUE.
flim A numeric vector of length 2 for the frequency limit (in kHz) of the spectrogram, as in spectro. Default is NULL.
rm.zero Logical indicated if the 0 at the start of the time axis should be removed. Default is FALSE.
X Optional. Data frame containing columns for start and end time of signals ('start' and 'end') and low and high frequency ('bottom.freq' and 'top.freq').
fast.spec Logical. If TRUE then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors gray.1, gray.2, gray.3, topo.1 and rainbow.1 (which should be imported from the package monitoR) seem to work better with 'fast' spectrograms. Palette colors gray.1, gray.2, gray.3 offer decreasing darkness levels.
t.mar Numeric vector of length 1. Specifies the margins adjacent to the start and end points to be added when highlighting selection. Default is NULL.
f.mar Numeric vector of length 1. Specifies the margins adjacent to the low and high frequencies to be added when highlighting selection. Default is NULL.
interactive  Numeric. Allow user to interactively select the signals to be highlighted by clicking on the graphic device. Users must select the opposite corners of a square delimiting the spectrogram region to be highlighted. Controls the number of signals that users would be able to select (2 clicks per signal).

add       Logical. If TRUE new highlighting can be applied to the current plot (which means that the function with add = FALSE should be run first). Default is FALSE.

Details
This function highlights regions of the spectrogram with different colors. The regions to be highlighted can be provided in a selection table (as the example data 'lbh_selec_table') or interactively ('interactive' argument).

Value
A plot is produced in the graphic device.

Author(s)
Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>) and Grace Smith Vidaurre

References

See Also
track_freq_contour for creating spectrograms to visualize frequency measurements by spectro_analysis, snr_spectrograms for creating spectrograms to optimize noise margins used in sig2noise

Other spectrogram creators: freq_DTW(), multi_DTW(), phylo_spectro(), snr_spectrograms(), spectrograms(), track_freq_contour()

Examples
## Not run:
data(list = c("Phae.long1", "lbh_selec_table"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) #save sound files

# subset selection table
st <- lbh_selec_table[lbh_selec_table$sound.files == "Phae.long1.wav",]

# read wave file as an R object
sgnl <- tuneR::readWave(file.path(tempdir(), st$sound.files[1]))

# create color column
st$colors <- c("red2", "blue", "green")

# highlight selections
color_spectro(wave = sgnl, wl = 300, ovlp = 90, flim = c(1, 8.6), collevels = seq(-40, 0, 5),
compare_methods

Assessing the performance of acoustic distance measurements

Description

compare_methods creates graphs to visually assess performance of acoustic distance measurements

Usage

```r
compare_methods(X = NULL, flim = NULL, bp = NULL, mar = 0.1, wl = 512, ovlp = 90,
res = 150, n = 10, length.out = 30,
methods = NULL,
it = "jpeg", parallel = 1, path = NULL, sp = NULL, custom1 = NULL,
custom2 = NULL, pb = TRUE, grid = TRUE, clip.edges = TRUE,
threshold = 15, na.rm = FALSE, scale = FALSE, pal = reverse.gray.colors.2,
img = TRUE, ...)
```

Arguments

- **X**: 'selection_table' object or data frame with results from `auto_detec` function, or any data frame with columns for sound file name (`sound.files`), selection number (`selec`), and start and end time of signal (`start` and `end`). Default `NULL`.
- **flim**: A numeric vector of length 2 for the frequency limit in kHz of the spectrogram, as in `spectro`. Default is `NULL`.
- **bp**: numeric vector of length 2 giving the lower and upper limits of the frequency bandpass filter (in kHz) used in the acoustic distance methods. Default is `NULL`.
- **mar**: Numeric vector of length 1. Specifies plot margins around selection in seconds. Default is 0.1.
- **wl**: A numeric vector of length 1 specifying the window length of the spectrogram and cross-correlation, default is 512.
- **ovlp**: Numeric vector of length 1 specifying the percent overlap between two consecutive windows, as in `spectro`. Default is 90.
- **res**: Numeric argument of length 1. Controls image resolution. Default is 150.
- **n**: Numeric argument of length 1. Defines the number of plots to be produce. Default is 10.
length.out  A character vector of length 1 giving the number of measurements of fundamental or dominant frequency desired (the length of the time series). Default is 30.

methods A character vector of length 2 giving the names of the acoustic distance methods that would be compared. The methods available are:

- XCORR: cross-correlation (cross_correlation function)
- dfDTW: dynamic time warping on dominant frequency contours (freq_DTW function)
- ffDTW: dynamic time warping on fundamental frequency contours (freq_DTW function)
- SP: spectral parameters (spectro_analysis function)
- SPharm: spectral parameters (spectro_analysis function with argument harmonicity = TRUE)
- MFCC: statistical descriptors of Mel frequency cepstral coefficients (mfcc_stats function)

Default NULL.

it  A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".

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).

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

sp DEPRECATED.

custom1 Data frame containing user parameters. The data frame must have 4 columns: the first 2 columns are 'sound.files' and 'selec' columns as in 'X', the other 2 (columns 3 and 4) are 2 numeric columns to be used as the 2 parameters representing custom measurements. If the data has more than 2 parameters try using PCA (i.e. prcomp function) to summarize it in 2 dimensions before using it as an input. Default is NULL.

custom2 Data frame containing user parameters with the same format as 'custom1'. 'custom1' must be provided first. Default is NULL.

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

grid Logical argument to control the presence of a grid on the spectrograms (default is TRUE).

clip.edges Logical argument to control whether edges (start or end of signal) in which amplitude values above the threshold were not detected will be removed when using dfDTW and ffDTW methods. If TRUE this edges will be excluded and signal contour will be calculated on the remaining values. Default is TRUE.

threshold amplitude threshold (%) for dominant and/or fundamental frequency detection when using dfDTW, ffDTW and SP methods. Default is 15.

na.rm Logical. If TRUE all NAs produced when pairwise cross-correlations failed are removed from the results. This means that all selections with at least 1 cross-correlation that failed are excluded in both methods under comparison. Only apply if XCORR is one of the methods being compared.
**compare_methods**

- **scale**
  Logical. If TRUE dominant and/or fundamental frequency values are z-transformed using the `scale` function, which "ignores" differences in absolute frequencies between the signals in order to focus the comparison in the frequency contour, regardless of the pitch of signals. Default is TRUE.

- **pal**
  A color palette function to be used to assign colors in the spectrograms, as in `spectro`. Default is reverse.gray.colors.2.

- **img**
  A logical argument specifying whether an image files would be produced. Default is TRUE.

- **...**
  Additional arguments to be passed to a modified version of `spectro` for customizing graphical output. This includes fast.spec, an argument that speeds up the plotting of spectrograms (see description in `spectrograms`).

**Details**

This function produces graphs with spectrograms from 4 signals in the provided data frame that allow visual inspection of the performance of acoustic distance methods at comparing those signals. The signals are randomly picked up from the provided data frame (X argument). The spectrograms are all plotted with the same frequency and time scales. The function compares 2 methods at a time. The methods available are: cross-correlation (XCORR, from `cross_correlation`), dynamic time warping on dominant frequency time series (dfDTW, from `dtw` applied on `freq_ts` output), dynamic time warping on dominant frequency time series (ffDTW, from `dtw` applied on `freq_ts` output), spectral parameters (SP, from `spectro_analysis`). The graph also contains 2 scatterplots (1 for each method) of the acoustic space of all signals in the input data frame 'X', including the centroid as black dot. The compared selections are randomly picked up from the pool of selections in the input data frame. The argument 'n' defines the number of comparisons (i.e. graphs) to be produced. The acoustic pairwise distance between signals is shown next to the arrows linking them. The font color of a distance value correspond to the font color of the method that generated it, as shown in the scatterplots. Distances are standardized, being 0 the distance of a signal to itself and 1 the farthest pairwise distance in the pool of signals. Principal Component Analysis (`prcomp`) is applied to calculate distances when using spectral parameters (SP) and descriptors of cepstral coefficients (MFCC). In those cases the first 2 PC's are used. Classical Multidimensional Scalling (also known as Principal Coordinates Analysis, (`cmdscale`)) is used for cross-correlation (XCORR) and any dynamic time warping method. The graphs are return as image files in the working directory. The file name contains the methods being compared and the row number of the selections. This function uses internally a modified version of the `spectro` function from seewave package to create spectrograms. Custom data can also be compared against the available methods (or against each other) using the arguments 'custom1' and 'custom2'.

**Value**

Image files with 4 spectrograms of the selection being compared and scatterplots of the acoustic space of all signals in the input data frame 'X'.

**Author(s)**

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>). It uses internally a modified version of the `spectro` function from seewave package to create spectrograms.
References


See Also

blog post on comparing methods

Examples

## Not run:
# Save to temporary working directory
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))
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"))

compare_methods(X = lbh_selec_table, flim = c(0, 10), bp = c(0, 10), mar = 0.1, wl = 300, ovlp = 90, res = 200, n = 10, length.out = 30, methods = c("XCORR", "dfDTW"), parallel = 1, it = "jpeg", path = tempdir())

#remove progress bar
compare_methods(X = lbh_selec_table, flim = c(0, 10), bp = c(0, 10), mar = 0.1, wl = 300, ovlp = 90, res = 200, n = 10, length.out = 30, methods = c("XCORR", "dfDTW"), parallel = 1, it = "jpeg", pb = FALSE, path = tempdir())

#check this folder!
getwd()

#compare SP and XCORR
compare_methods(X = lbh_selec_table, flim = c(0, 10), bp = c(0, 10), mar = 0.1, wl = 300, ovlp = 90, res = 200, n = 10, length.out = 30, methods = c("XCORR", "SP"), parallel = 1, it = "jpeg", path = tempdir())

#compare SP method against dfDTW
compare_methods(X = lbh_selec_table, flim = c(0, 10), bp = c(0, 10), mar = 0.1, wl = 300, ovlp = 90, res = 200, n = 10, length.out = 30, methods = c("dfDTW", "SP"), parallel = 1, it = "jpeg", path = tempdir())

#alternatively we can provide our own SP matrix
Y <- spectro_analysis(lbh_selec_table, path = tempdir())

# selec a subset of variables
Y <- Y[, 1:7]

# PCA
Y <- prcomp(Y[, 3:ncol(Y)])[1]

# add sound files and selec columns
comp_matrix

```r
Y <- data.frame(lbh_selec_table[, c(1, 3)], Y[, 1:2])

compare_methods(X = lbh_selec_table, methods = c("dfDTW"), custom1 = Y, 
path = tempdir())

## End(Not run)
```

---

**comp_matrix**

Example matrix listing selections to be compared by cross_correlation

---

### Description

comp_matrix is a character matrix with 2 columns indicating the selections to be compared (column 1 vs column 2) by cross_correlation.

### Usage

```r
data(comp_matrix)
```

### Format

A data frame with 11 rows and 6 variables:

- **sound.files**: recording names
- **channel**: channel in which signal is found
- **selec**: selection numbers within recording
- **start**: start times of selected signal
- **end**: end times of selected signal
- **bottom.freq**: lower limit of frequency range
- **top.freq**: upper limit of frequency range

### Details

A character matrix with 2 columns indicating the selections to be compared (column 1 vs column 2) by cross_correlation. The first column contain the ID of the selection, which is given by combining the 'sound.files' and 'selec' columns of 'X', separated by '-' (i.e. `paste(X$sound.files, X$selec, sep = "-"))`. The selection id's refer to those on the example data "lbh_selec_table". The second column refers to the sound files in which to search for the templates.

### Source

Marcelo Araya Salas, warbleR
Describes:

`consolidate` changes (sound) files scattered in several directories into a single one.

**Usage**

```r
consolidate(files = NULL, path = NULL, dest.path = NULL, pb = TRUE, file.ext = ".wav$", parallel = 1, save.csv = TRUE, ...)
```

**Arguments**

- **files**: character vector or factor indicating the subset of files that will be analyzed. The files names should include the full file path. Optional.
- **path**: Character string containing the directory path where the sound files are located. "wav.path" set by `warbleR_options` is ignored. If NULL (default) then the current working directory is used.
- **dest.path**: Character string containing the directory path where the sound files will be saved. If NULL (default) then the current working directory is used.
- **pb**: Logical argument to control progress bar. Default is TRUE.
- **file.ext**: Character string defining the file extension (i.e. format) for the files to be consolidated. Default is ".wav$" ignoring case. Several formats can be used: "\.wav$|\.wac$|\.mp3$|\.flac$".
- **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).
- **save.csv**: Logical. Controls whether a data frame containing sound file information is saved in the new directory. Default is TRUE.
- **...**: Additional arguments to be passed to the internal `file.copy` function for customizing file copying.

**Details**

This function allows users to put files scattered in several directories into a single one. By default it works on sound files in ".wav" format but can work with other type of files (for instance ".txt" selection files).

**Value**

All (sound) files are consolidated (copied) to a single directory ("consolidated_files"). The function returns a data frame with each of the files that were copied in a row and the following information:

- `original_dir` the path to the original file
- `old_name` the name of the original file
• new\_name the name of the new file. This will be the same as 'old\_name' if the name was not duplicated (i.e. no files in other directories with the same name).

• file\_size\_bytes size of the file in bytes.

• duplicate indicates whether a file is likely to be duplicated (i.e. if files with the same name were found in other directories). If so it will be labeled as 'possible.dupl', otherwise it will contain NA.

If csv = TRUE (default) a 'file\_names\_info.csv' file with the same information as the output data frame is also saved in the consolidated directory.

Author(s)

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

References


See Also

fix\_wavs for making sound files readable in R

Examples

```r
#
# save wav file examples
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))

# create first folder with 2 sound files
dir.create(file.path(tempdir(), "folder1"))
writeWave(Phae.long1, file.path(tempdir(), "folder1", "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "folder1", "Phae.long2.wav"))

# create second folder with 2 sound files
dir.create(file.path(tempdir(), "folder2"))
writeWave(Phae.long3, file.path(tempdir(), "folder2", "Phae.long3.wav"))
writeWave(Phae.long4, file.path(tempdir(), "folder2", "Phae.long4.wav"))

# consolidate in a single folder
# consolidate(path = tempdir(), dest.path = tempdir())

# check this folder
tempdir()
```
cross_correlation

**Time-frequency cross-correlation**

description

cross_correlation estimates the similarity of two sound waves by means of time-frequency cross-correlation

**Usage**

cross_correlation(X, wl = 512, bp = "pairwise.freq.range", ovlp = 70, dens = NULL, wn = 'hanning', cor.method = "pearson", parallel = 1, path = NULL, pb = TRUE, na.rm = FALSE, cor.mat = NULL, output = "cor.mat", templates = NULL, surveys = NULL, compare.matrix = NULL, type = "fourier", nbands = 40, method = 1)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>'selection_table', 'extended_selection_table' or data frame containing columns for sound files (sound.files), selection number (selec), and start and end time of signal (start and end). All selections must have the same sampling rate.</td>
</tr>
<tr>
<td>wl</td>
<td>A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.</td>
</tr>
<tr>
<td>bp</td>
<td>A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or &quot;pairwise.freq.range&quot; (default) to indicate that values in lowest bottom.freq and highest top.freq columns for the signals involved in a pairwise comparison will be used as bandpass limits.</td>
</tr>
<tr>
<td>ovlp</td>
<td>Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in spectro. Default is 70. High values of ovlp slow down the function but produce more accurate results.</td>
</tr>
<tr>
<td>dens</td>
<td>DEPRECATED.</td>
</tr>
<tr>
<td>wn</td>
<td>A character vector of length 1 specifying the window name as in ftwindow.</td>
</tr>
<tr>
<td>cor.method</td>
<td>A character vector of length 1 specifying the correlation method as in cor.</td>
</tr>
<tr>
<td>parallel</td>
<td>Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).</td>
</tr>
<tr>
<td>path</td>
<td>Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.</td>
</tr>
<tr>
<td>pb</td>
<td>Logical argument to control progress bar. Default is TRUE.</td>
</tr>
<tr>
<td>na.rm</td>
<td>Logical. If TRUE all NAs produced when pairwise cross-correlations failed are removed from the results. This means that all selections with at least 1 cross-correlation that failed are excluded.</td>
</tr>
<tr>
<td>cor.mat</td>
<td>DEPRECATED. Use 'compare.matrix' instead.</td>
</tr>
</tbody>
</table>
cross_correlation

output Character vector of length 1 to determine if only the correlation matrix is returned ('cor.mat', default) or a list ('list') containing 1) the correlation matrix and 2) a data frame with correlation values at each sliding step for each comparison. The list, which is also of class 'xcorr.output', can be used to find detection peaks with find_peaks or to graphically explore detections using lspec.

templates Character vector with the selections in 'X' to be used as templates for cross-correlation detection. To refer to specific selections in 'X' the user must use the format "sound.file-selec" (e.g. "file1.wav-1"). If only the sound file name is included then the entire sound file is used as template.

surveys Character vector with the selections in 'X' to be used as surveys for cross-correlation detection. To refer to specific selections in 'X' the user must use the format "sound.file-selec" (e.g. "file1.wav-1"). If only the sound file name is included then the entire sound file is used as survey.

compare.matrix A character matrix with 2 columns indicating the selections to be compared (column 1 vs column 2). The columns must contain the ID of the selection, which is given by combining the 'sound.files' and 'selec' columns of 'X', separated by '-' (i.e. paste(X$sound.files, X$selec, sep = "-")); Default is NULL. If the 'selec' ID is not supplied then the entire sound file is used instead. If 'compare.matrix' is supplied only those comparisons will be calculated (as opposed to all pairwise comparisons as the default behavior) and the output will be a data frame composed of the supplied matrix and the correspondent cross-correlation values. Note that 'method' is automatically set to 2 (create spectrograms on the fly) when 'compare.matrix' is supplied but can be set back to 1. When supplied 'output' is forced to be 'list'.

type A character vector of length 1 specifying the type of cross-correlation; either "fourier" (i.e. spectrographic cross-correlation using Fourier transform; internally using spectro; default) or "mfcc" (Mel cepstral coefficient spectrogram cross-correlation; internally using melfcc).

nbands Numeric vector of length 1 controlling the number of warped spectral bands to calculate when using type = "mfcc" (see melfcc). Default is 40.

method Numeric vector of length 1 to control the method used to create spectrogram matrices. Two option are available:

- 1: matrices are created first (keeping them internally as a list) and cross-correlation is calculated on a second step. Note that this method may require lots of memory if selection and or sound files are large.

- 2: matrices are created "on the fly" (i.e. at the same time that cross-correlation is calculated). More memory efficient but may require extracting the same matrix several times, which will affect performance. Note that when using this method the function does not check if sound files have the same sampling rate which if not, may produce an error.

Details

This function calculates the pairwise similarity of multiple signals by means of time-frequency cross-correlation. Fourier or Mel frequency cepstral coefficients spectrograms can be used as time-frequency representations of sound. This method "slides" the spectrogram of the shortest selection over the longest one calculating a correlation of the amplitude values at each step. The function runs
pairwise cross-correlations on several signals and returns a list including the correlation statistic for each "sliding" step as well as the maximum (peak) correlation for each pairwise comparison. The correlation matrix could have NA's if some of the pairwise correlation did not work (common when sound files have been modified by band-pass filters).

**Value**

If `output = "cor.mat"` the function returns a matrix with the maximum (peak) correlation for each pairwise comparison (if 'compare.matrix' is not supplied) or the peak correlation for each comparison in the supplied 'compare.matrix'. Otherwise it will return a list that includes 1) a matrix with the maximum correlation for each pairwise comparison ('max.xcorr.matrix') and 2) a data frame with the correlation statistic for each "sliding" step ('scores').

**Author(s)**

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

**References**


**See Also**

mfcc_stats, spectro_analysis, freq_DTW

**Examples**

```r
#load data
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4","lbh_selec_table"))

#save sound files
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"))

# run cross correlation on spectrograms (SPCC)
xcor <- cross_correlation(X = lbh_selec_table, wl = 300, ovlp = 90, path = tempdir())

# run cross correlation on Mel cepstral coefficients (mfccs)
xcor <- cross_correlation(X = lbh_selec_table, wl = 300, ovlp = 90, path = tempdir(), type = "mfcc")

# using the 'compare.matrix' argument to specify pairwise comparisons
# create matrix with ID of signals to compare
```
cmp.mt <- cbind(
paste(lbh_selec_table$sound.files[1:10], lbh_selec_table$selec[1:10], sep = "-"),
paste(lbh_selec_table$sound.files[2:11], lbh_selec_table$selec[2:11], sep = "-"))

# run cross-correlation on the selected pairwise comparisons
xcor <- cross_correlation(X = lbh_selec_table, compare.matrix = cmp.mt,
wl = 300, ovlp = 90, path = tempdir())

---

duration_sound_files  Measure the duration of sound files

**Description**

duration_sound_files measures the duration of sound files

**Usage**

duration_sound_files(files = NULL, path = NULL)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>files</td>
<td>Character vector with the names of the sound files to be measured. The sound files should be in the working directory or in the directory provided in <code>path</code>.</td>
</tr>
<tr>
<td>path</td>
<td>Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.</td>
</tr>
</tbody>
</table>

**Details**

This function returns the duration (in seconds) of sound files.

**Value**

A data frame with the duration (in seconds) of the sound files.

**Author(s)**

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

**References**

envelope

Calculates the absolute amplitude envelope

description
Calculated the absolute amplitude envelope

envelope(x, ssmooth = 0)

Arguments

- **x**: Numeric vector with amplitude values. Required.
- **sssmooth**: Numeric vector of length 1 indicating the size of the sliding window used to smooth envelopes. Default is 0 (no smoothing).

Details

The function calculates the absolute amplitude envelope of an amplitude vector using compiled C code, which is usually several times faster.

Value

An amplitude envelope.

Author(s)

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

References


See Also
e

Examples

```r
{  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"))

duration_sound_files(path = tempdir())  }
filter_sels

Subset selection data frames based on manually filtered image files

Description

filter_sels subsets selection data frames based on image files that have been manually filtered.

Usage

filter_sels(X, path = NULL, lspec = FALSE, img.suffix = NULL, it = "jpeg", incl.wav = TRUE, missing = FALSE, index = FALSE)

Arguments

X object of class 'selection_table', 'extended_selection_table' or data frame with the following columns: 1) "sound.files": name of the .wav files, 2) "sel": number of the selections.

path Character string containing the directory path where the image files are located. If NULL (default) then the current working directory is used. warbleR_options 'wav.path' argument does not apply.

lspec A logical argument indicating if the image files to be use for filtering were produced by the function full_spectrograms. All the image files that correspond to a sound file must be deleted in order to be filtered out.

img.suffix A character vector of length 1 with the suffix (label) at the end of the names of the image files. Default is NULL (i.e. no suffix as in the images produced by spectrograms). Ignored if lspec = TRUE.

it A character vector of length 1 giving the image type ("tiff", "jpeg" or "pdf") Default is "jpeg". Note that pdf files can only be generated by lspec2pdf.

incl.wav Logical. To indicate if sound files extensions are included (TRUE, default) or not in the image file names.

missing Logical. Controls whether the output data frame (or row index if is index = TRUE) contains the selections with images in the working directory (Default, missing = FALSE) or the ones with no image.

index Logical. If TRUE and missing = FALSE the row index for the selections with images in the working directory is returned. If missing = TRUE) then the row index of the ones with no image is returned instead. Default is FALSE.
Details

This function subsets selections (or sound files if lspec is TRUE) listed in a data frame based on the image files from spectrogram-creating functions (e.g. `spectrograms`) in the working directory. Only the selections/sound files with and image in the working directory will remain. This is useful for excluding selections from undesired signals. Note that the image files should be in the working directory (or the directory provided in `path`).

Value

If all sound files are ok, returns message "All files are ok!". Otherwise returns "These file(s) cannot be read" message with names of the corrupted sound files.

Author(s)

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

References


Examples

```r
## Not run:
# save wav file examples
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "lbh_selec_table"))
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"))
spectrograms(lbh_selec_table, flim = c(0, 11), inner.mar = c(4,4.5,2,1), outer.mar = c(4,2,2,1),
picsize = 2, res = 300, cexlab = 2, mar = 0.05, wl = 300, path = tempdir())

#go to the working directory (tempdir()) and delete some images

#filter selection data frame
fmloc <- filter_sels(X = lbh_selec_table, path = tempdir())

#this data frame does not have the selections corresponding to the images that were deleted
fmloc

#now using lspec images
full_spectrograms(sxrow = 2, rows = 8, pal = reverse.heat.colors, wl = 300, ovlp = 10,
path = tempdir())

# go to the working directory (tempdir()) and delete lspec
# images (the ones with several rows of spectrograms)

#filter selection data frame
fmloc2 <- filter_sels(X = lbh_selec_table, lspec = TRUE, path = tempdir())
```
find_annotations

## Obtain annotations from audioblast.org data base

### Description

find_annotations downloads sound file annotations and metadata from audioblast.org.

### Usage

```r
find_annotations(qword, parallel = 1, pb = TRUE, warbler.format = FALSE, download = FALSE, X = NULL, path = NULL)
```

### Arguments

- **qword**: Character vector of length one indicating the scientific name of the species to search for at audioblast’s annotations database. For example, *Phaethornis longirostris*.
- **parallel**: Numeric. Controls whether parallel computing is applied when downloading mp3 files. 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.
- **warbler.format**: Logical. If TRUE columns are renamed using the standard names for a selection table as in ‘warbleR’, frequency limit columns (high and low frequency) in ‘Hz’ are converted to ‘kHz’ (as in warbleR selection tables) and the column names are changed to: ‘sound.files’, ‘selec’, ‘start’, ‘end’, ‘bottom.freq’ and ‘top.freq’. Default is FALSE.
- **download**: Logical argument. If FALSE only the recording file names and associated metadata are downloaded. If TRUE, recordings are also downloaded to the working directory. Default is FALSE.
- **X**: Data frame generated by find_annotations. If supplied the recordings listed in the data frame will be downloaded using the 'recording_url' column information (download argument is automatically set to TRUE). This can be used to select the recordings to be downloaded based on their attributes. Default is NULL.
- **path**: Character string containing the directory path where the sound files will be saved. If NULL (default) then the current working directory is used.

### Details

This function queries for annotations on acoustic media in the open-access online repository audioblast.org.
find_clipping

Value
A data frame with the annotation information.

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

References

See Also
query_xc

Examples
## Not run:
ann <- find_annotations(qword = "Thyroptera")
## End(Not run)

find_clipping

Description
find_clipping gets the proportion of samples that are clipped.

Usage
find_clipping(X, path = NULL, parallel = 1, pb = TRUE)

Arguments
X 'selection_table', 'extended_selection_table' or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "selec": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections.

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

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.
Details

Clipping (i.e. saturation) occurs when an audio signal is amplified above the maximum limit of the recorder. This leads to distortion and a lowering of audio quality. If stereo the mean proportion of both channels is returned. The function assumes specific range values for different bit depths as detailed in `normalize`.

Value

A data frame with the 'sound.files' and 'selec' columns in X plus an additional column ('prop.clipped') indicating the proportion of clipped samples for each row. If sound files are stereo the average proportion of the two channels is returned.

Author(s)

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

References


See Also

`sig2noise`

Examples

```r
{
# load data
data(list = c("Phae.long1", "Phae.long2","lbh_selec_table"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) #save sound files
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))

find_clipping(X = lbh_selec_table[1:5, ], path = tempdir())
}
```

---

**find_peaks**

Find cross-correlation peaks

Description

`find_peaks` find peaks in cross-correlation scores from `cross_correlation`

Usage

```r
find_peaks(xc.output, parallel = 1, cutoff = 0.4, path = NULL, pb = TRUE,
max.peak = FALSE, output = "data.frame")
```
find_peaks

Arguments

xc.output output of `cross_correlation` after setting output = "list".
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).
cutoff Numeric vector of length 1 with a value between 0 and 1 specifying the correlation cutoff for detecting peaks. Default is 0.4.
path Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
pb Logical argument to control progress bar. Default is TRUE.
max.peak Logical argument to control whether only the peak with the highest correlation value is returned (if TRUE; cutoff will be ignored). Default is FALSE.
output Character vector of length 1 to determine if only the detected peaks are returned ("cormat") or a list ("list") containing 1) the peaks and 2) a data frame with correlation values at each sliding step for each comparison. The list, which is also of class 'peaks.output', can be used to graphically explore detections using `full_spectrograms`.

Details

This function finds cross-correlation peaks along signals (analogous to `findPeaks`).

Value

The function returns a data frame with time and correlation score for the detected peaks.

Author(s)

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

References


See Also

`auto_detec`, `findPeaks`

Examples

```
{  
  # load data
  data(list = c("Phae.long4", "Phae.long2", "lbh_selec_table2", "comp_matrix"))

  # save sound files
  writeWave(Phae.long4, file.path(tempdir(), "Phae.long4.wav"))
}
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))

# run cross-correlation
xc.output <- cross_correlation(X = lbh_selec_table2, output = "list",
compare.matrix = comp_matrix, path = tempdir())

# find peaks
pks <- find_peaks(xc.output = xc.output, path = tempdir())

fix_extended_selection_table

Fix extended selection tables

Description

fix_extended_selection_table fixes extended selection tables that have lost their attributes

Usage

fix_extended_selection_table(X, Y, to.by.song = FALSE)

Arguments

X an object of class 'selection_table' or data frame that contains columns for sound
file name (sound.files), selection number (selec), and start and end time of signal
(start and end).

Y an object of class 'extended_selection_table'

to.by.song Logical argument to control if the attributes are formatted to a match a 'by.song'
extended selection table. This is required when 'X' is created by collapsing
an Y by song (see 'by.song' argument in selection_table). Mostly needed
internally by some warbleR functions.

Author(s)

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

Examples

{data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))

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"))

# create extended selection table
ext_st <- selection_table(lbh_selec_table, extended = TRUE, confirm.extended = FALSE,
path = tempdir() 

# remove attributes  
st <- as.data.frame(ext_st)  

# check class  
class(st)  

# fix selection table  
st <- fix_extended_selection_table(X = st, Y = ext_st)  

# check class  
class(st)  
}

---

**fix_wavs**  
*Fix .wav files to allow importing them into R*

**Description**  

fix_wavs fixes sound files in .wav format so they can be imported into R.

**Usage**  

```r  
fix_wavs(checksels = NULL, files = NULL, samp.rate = NULL, bit.depth = NULL,  
path = NULL, mono = FALSE)  
```

**Arguments**  

- **checksels**: Data frame with results from `check_sels`. Default is `NULL`. If both 'checksels' and 'files' are `NULL` then all files in 'path' are converted.
- **files**: Character vector with the names of the wav files to fix. Default is `NULL`. If both 'checksels' and 'files' are `NULL` then all files in 'path' are converted.
- **samp.rate**: Numeric vector of length 1 with the sampling rate (in kHz) for output files. Default is `NULL` (remain unchanged).
- **bit.depth**: Numeric vector of length 1 with the dynamic interval (i.e. bit depth) for output files. Default is `NULL` (remain unchanged).
- **path**: Character string containing the directory path where the sound files are located. If `NULL` (default) then the current working directory is used.
- **mono**: Logical indicating if stereo (2 channel) files should be converted to mono (1 channel). Default is `NULL` (remain unchanged).
freq_DTW

Details

This function aims to simplify the process of converting sound files that cannot be imported into R and/or homogenizing sound files. Problematic files can be determined using check_wavs or check_sels. The check_sels output can be directly input using the argument 'checksels'. Alternatively a vector of file names to be "fixed" can be provided (argument 'files'). If neither of those 2 are provided the function will convert all sound files in the working directory to the specified sample rate/bit depth. Files are saved in a new directory ('converted_sound_files'). Internally the function calls SOX (if 'sox = TRUE', SOX must be installed). If both 'checksels' and 'files' are NULL then all files in 'path' are converted.

Value

A folder inside the working directory (or path provided) all 'converted_sound_files', containing sound files in a format that can be imported in R.

Author(s)

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

References


Examples

```r
## Not run:
# Load example files and save to temporary working directory

cat(tempdir())

## End(Not run)
```

freq_DTW  

**Acoustic dissimilarity using dynamic time warping on dominant frequency contours**

Description

freq_DTW calculates acoustic dissimilarity of frequency contours using dynamic time warping. Internally it applies the dtwDist function from the dtw package.
Usage

freq_DTW(X = NULL, type = "dominant", wl = 512, wl.freq = 512, length.out = 20, 
wn = "hanning", ovlp = 70, bp = NULL, threshold = 15, threshold.time = NULL, 
threshold.freq = NULL, img = TRUE, parallel = 1, path = NULL, ts.df = NULL, 
img.suffix = "dfDTW", pb = TRUE, cclip.edges = TRUE, window.type = "none", 
open.end = FALSE, scale = FALSE, frange.detec = FALSE, fsmooth = 0.1, 
adjust.wl = TRUE, ...)  

Arguments

X  
object of class 'selection_table', 'extended_selection_table' or data frame containing columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end). The output of auto_detec can be used as the input data frame.

type  
Character string to determine the type of contour to be detected. Three options are available, "dominant" (default), "fundamental" and "entropy".

wl  
A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.

wl.freq  
A numeric vector of length 1 specifying the window length of the spectrogram for measurements on the frequency spectrum. Default is 512. Higher values would provide more accurate measurements.

length.out  
A numeric vector of length 1 giving the number of measurements of frequency desired (the length of the time series).

wn  
Character vector of length 1 specifying window name. Default is "hanning". See function ftwindow for more options.

ovlp  
Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in spectro. Default is 70.

bp  
A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz). Default is NULL.

threshold  
amplitude threshold (%) for frequency detection. Default is 15.

threshold.time  
amplitude threshold (%) for the time domain. Use for frequency detection. If NULL (default) then the 'threshold' value is used.

threshold.freq  
amplitude threshold (%) for the frequency domain. Use for frequency range detection from the spectrum (see 'frange.detec'). If NULL (default) then the 'threshold' value is used.

img  
Logical argument. If FALSE, image files are not produced. Default is TRUE.

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).

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

ts.df  
Optional. Data frame with frequency contour time series of signals to be compared. If provided "X" is ignored.

img.suffix  
A character vector of length 1 with a suffix (label) to add at the end of the names of image files. Default is NULL.
freq_DTW

pb Logical argument to control progress bar. Default is TRUE.
clip.edges Logical argument to control whether edges (start or end of signal) in which amplitude values above the threshold were not detected will be removed. If TRUE (default) this edges will be excluded and contours will be calculated on the remaining values. Note that DTW cannot be applied if missing values (e.i. when amplitude is not detected).
window.type dtw windowing control parameter. Character: "none", "itakura", or a function (see dtw).
open.end dtw control parameter. Performs open-ended alignments (see dtw).
scale Logical. If TRUE frequency values are z-transformed using the scale function, which "ignores" differences in absolute frequencies between the signals in order to focus the comparison in the frequency contour, regardless of the pitch of signals. Default is TRUE.
frange.detec DEPRECATED.
frange.detec DEPRECATED.
fsmooth A numeric vector of length 1 to smooth the frequency spectrum with a mean sliding window (in kHz) used for frequency range detection (when frange.detec = TRUE). This help to average amplitude "hills" to minimize the effect of amplitude modulation. Default is 0.1.
adjust.wl Logical. If TRUE 'wl' (window length) is reset to be lower than the number of samples in a selection if the number of samples is less than 'wl'. Default is TRUE.
... Additional arguments to be passed to track_freq_contour for customizing graphical output.

Details
This function extracts the dominant frequency values as a time series and then calculates the pairwise acoustic dissimilarity using dynamic time warping. The function uses the approx function to interpolate values between dominant frequency measures. If `img` is TRUE the function also produces image files with the spectrograms of the signals listed in the input data frame showing the location of the dominant frequencies.

Value
A matrix with the pairwise dissimilarity values. If img is FALSE it also produces image files with the spectrograms of the signals listed in the input data frame showing the location of the dominant frequencies.

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

References
freq_range

Detect frequency range iteratively

Description

freq_range detect frequency range iteratively from signals in a selection table.

Usage

freq_range(X, wl = 512, it = "jpeg", line = TRUE, fsmooth = 0.1, threshold = 10,
dB.threshold = NULL, wn = "hanning", flim = NULL, bp = NULL,
propwidth = FALSE, xl = 1, picsize = 1, res = 100, fast.spec = FALSE, ovlp = 50,
pal = reverse.gray.colors.2, parallel = 1, widths = c(2, 1), main = NULL,
img = TRUE, mar = 0.05, path = NULL, pb = TRUE, impute = FALSE)

Arguments

X object of class 'selection_table', 'extended_selection_table' or data frame with
the following columns: 1) "sound.files": name of the sound files, 2) "sel": num-
ber of the selections, 3) "start": start time of selections, 4) "end": end time of
selections. The output of auto_detect can also be used as the input data frame.
freq_range

wl  A numeric vector of length 1 specifying the window length of the spectrogram, default is 512. This is used for calculating the frequency spectrum (using meanspec) and producing the spectrogram (using spectro, if img = TRUE).

it  A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".

line  Logical argument to add red lines (or box if bottom.freq and top.freq columns are provided) at start and end times of selection. Default is TRUE.

fsmooth  A numeric vector of length 1 to smooth the frequency spectrum with a mean sliding window in kHz. This helps to average amplitude "hills" to minimize the effect of amplitude modulation. Default is 0.1.

threshold  Amplitude threshold (%) for frequency range detection. The frequency range (not the cumulative amplitude) is represented as percentage (100% = highest amplitude). Default is 10. Ignored if dB.threshold is supplied.

dB.threshold  Amplitude threshold for frequency range detection (in dB). The value indicates the decrease in dB in relation to the highest amplitude (e.g. the peak frequency) in which range will be detected. For instance a dB.threshold = 20 means that the amplitude threshold would be 20 dB below the highest amplitude. If provided 'threshold' is ignored. Default is NULL. Note that the power spectrum is normalized when using a dB scale, so it looks different than the one produced when no dB scale is used (e.g. when using 'threshold' argument).

wn  Character vector of length 1 specifying window name. Default is "hanning". See function ftwindow for more options. This is used for calculating the frequency spectrum (using meanspec) and producing the spectrogram (using spectro, if img = TRUE).

flim  A numeric vector of length 2 for the frequency limit of the spectrogram (in kHz), as in spectro. Default is NULL.

bp  A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or "frange" to indicate that values in 'bottom.freq' and 'top.freq' columns will be used as bandpass limits. Default is c(0, 22).

propwidth  Logical argument to scale the width of spectrogram proportionally to duration of the selected call. Default is FALSE.

xl  Numeric vector of length 1. A constant by which to scale spectrogram width. Default is 1.

csize  Numeric argument of length 1. Controls relative size of spectrogram. Default is 1.

res  Numeric argument of length 1. Controls image resolution. Default is 100 (faster) although 300 - 400 is recommended for publication/presentation quality.

fast.spec  Logical. If TRUE then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors gray.1, gray.2, gray.3, topo.1 and rainbow.1 (which should be imported from the package monitoR) seem to work better with 'fast.spec' spectrograms. Palette colors gray.1, gray.2, gray.3 offer decreasing darkness levels.
freq_range

ovlp
Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in spectro. Default is 50. This is used for calculating the frequency spectrum (using meanspec) and producing the spectrogram (using spectro, if img = TRUE).

pal
Color palette function for spectrogram. Default is reverse.gray.colors.2. See spectro for more palettes. Palettes as gray.2 may work better when fast.spec = TRUE.

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).

widths
Numeric vector of length 2 to control the relative widths of the spectro (first element) and spectrum (second element).

main
Character vector of length 1 specifying the img title. Default is NULL.

img
Logical. Controls whether a plot is produced. Default is TRUE.

mar
Numeric vector of length 1. Specifies the margins adjacent to the selections to set spectrogram limits. Default is 0.05.

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

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

impute
Logical. If TRUE then missing range values are imputed with the corresponding bandpass value (hence ignored when bp = NULL). Default is FALSE.

Details
This function aims to automatize the detection of frequency ranges. The frequency range is calculated as follows:

- bottom.freq = the start frequency of the amplitude 'hill' containing the highest amplitude at the given threshold.
- top.freq = the end frequency of the amplitude 'hill' containing the highest amplitude at the given threshold.

If img = TRUE a graph including a spectrogram and a frequency spectrum is generated for each selection (saved as an image file in the working directory). The graph would include gray areas in the frequency ranges excluded by the bandpass ('bp' argument), dotted lines highlighting the detected range. The function freq_range_detec is used internally.

Value
The original data frame with an additional 2 columns for low and high frequency values. A plot is produced in the working directory if img = TRUE (see details).

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


See Also

freq_range_detec, auto_detec

Examples

```
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))
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"))

freq_range(X = lbh_selec_table, wl = 112, fsmooth = 1, threshold = 13, widths = c(4, 1),
img = TRUE, pb = TRUE, it = "tiff", line = TRUE, mar = 0.1, bp = c(1,10.5),
flim = c(0, 11), path = tempdir())
```

freq_range_detec

**Detect frequency range on wave objects**

**Description**

freq_range_detec detects the frequency range of acoustic signals on wave objects.

**Usage**

freq_range_detec(wave, wl = 512, fsmooth = 0.1, threshold = 10, dB.threshold = NULL, wn = "hanning", flim = NULL, bp = NULL, fast.spec = FALSE, ovlp = 50, pal = reverse(gray.colors.2, widths = c(2, 1), main = NULL, plot = TRUE, all.detec = FALSE)

**Arguments**

- `wave`  
  A 'wave' object produced by `readWave` or similar functions.
- `wl`  
  A numeric vector of length 1 specifying the window length of the spectrogram, default is 512. This is used for calculating the frequency spectrum (using `meanspec`) and producing the spectrogram (using `spectro`, if `plot = TRUE`).
- `fsmooth`  
  A numeric vector of length 1 to smooth the frequency spectrum with a mean sliding window in kHz. This help to average amplitude "hills" to minimize the effect of amplitude modulation. Default is 0.1.
freq_range_detec

### Arguments

- **threshold**: Amplitude threshold (%) for frequency range detection. The frequency range (not the cumulative amplitude) is represented as percentage (100% = highest amplitude). Default is 10. Ignored if 'dB.threshold' is supplied.

- **dB.threshold**: Amplitude threshold for frequency range detection (in dB). The value indicates the decrease in dB in relation to the highest amplitude (e.g. the peak frequency) in which range will be detected. For instance a dB.threshold = 20 means that the amplitude threshold would be 20 dB below the highest amplitude. If provided 'threshold' is ignored. Default is NULL. Note that the power spectrum is normalized when using a dB scale, so it looks different than the one produced when no dB scale is used (e.g. when using 'threshold' argument).

- **wn**: Character vector of length 1 specifying window name. Default is "hanning". See function `ftwindow` for more options. This is used for calculating the frequency spectrum (using `meanspec`) and producing the spectrogram (using `spectro`, if `plot = TRUE`).

- **flim**: A numeric vector of length 2 for the frequency limit of the spectrogram (in kHz), as in `spectro`. Default is NULL.

- **bp**: A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or "frange" to indicate that values in 'bottom.freq' and 'top.freq' columns will be used as bandpass limits. Default is NULL.

- **fast.spec**: Logical. If TRUE then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors `gray.1`, `gray.2`, `gray.3`, `topo.1` and `rainbow.1` (which should be imported from the package monitoR) seem to work better with 'fast.spec' spectrograms. Palette colors `gray.1`, `gray.2`, `gray.3` offer decreasing darkness levels.

- **ovlp**: Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in `spectro`. Default is 50. This is used for calculating the frequency spectrum (using `meanspec`) and producing the spectrogram (using `spectro`, if `plot = TRUE`).

- **pal**: Color palette function for spectrogram. Default is `reverse.gray.colors.2`. See `spectro` for more palettes. Palettes as `gray.2` may work better when `fast.spec = TRUE`.

- **widths**: Numeric vector of length 2 to control the relative widths of the spectro (first element) and spectrum (second element).

- **main**: Character vector of length 1 specifying the plot title. Default is NULL.

- **plot**: Logical. Controls whether an image file is produced for each selection (in the working directory). Default is TRUE.

- **all.detec**: Logical. If TRUE returns the start and end of all detected amplitude "hills". Otherwise only the range is returned. Default is FALSE.

### Details

This function aims to automatize the detection of frequency ranges. The frequency range is calculated as follows:

1. The frequency range is defined as the range in which the amplitude is above a certain threshold (either in percentage or in dB).
2. The amplitude threshold is calculated as a percentage of the highest amplitude in the signal. If the dB.threshold is provided, the amplitude threshold is calculated as the decrease in dB relative to the highest amplitude.
3. The window function is used to calculate the frequency spectrum.
4. The spectrogram is produced using a bandpass filter, if provided.
5. The spectrogram's limits are defined by the flim and bp arguments.
6. The overlap between windows is controlled by the ovlp argument.
7. The color palette for the spectrogram is defined by the pal argument.
8. The relative widths of the spectro and spectrum are controlled by the widths argument.
9. The plot title is defined by the main argument.
10. An image file is produced for each selection, if plot is TRUE.
11. The start and end of all detected amplitude "hills" are returned, if all.detec is TRUE.
freq_ts

- bottom.freq = the start frequency of the amplitude 'hill' containing the highest amplitude at the given threshold.
- top.freq = the end frequency of the amplitude 'hill' containing the highest amplitude at the given threshold.

If plot = TRUE a graph including a spectrogram and a frequency spectrum is produced in the graphic device. The graph would include gray areas in the frequency ranges excluded by the bandpass ('bp' argument), dotted lines highlighting the detected range.

Value

A data frame with 2 columns for low and high frequency values. A plot is produced (in the graphic device) if plot = TRUE (see details).

Author(s)

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

References


See Also

frange, autodetect

Examples

{
  data(tico)
  freq_range_detec(wave = tico, wl = 512, fsmooth = 0.01, threshold = 1, bp = c(2, 8),
                   widths = c(4, 2))

  data(sheep)
  freq_range_detec(wave = sheep, wl = 512, fsmooth = 0.2, threshold = 50, bp = c(0.3, 1),
                   flim = c(0, 1.5), pal = reverse.heat.colors, main = "sheep")
}

freq_ts

Extract frequency contours as time series

Description

freq_ts extracts the fundamental frequency values as a time series.
Usage

freq_ts(X, type = "dominant", wl = 512, length.out = 20, wn = "hanning",
ovlp = 70, bp = NULL, threshold = 15, img = TRUE, parallel = 1, path = NULL,
img.suffix = "frequency.ts", pb = TRUE, clip.edges = FALSE, leglab = "frequency.ts",
track.harm = FALSE, raw.contour = FALSE, adjust.wl = TRUE,
ff.method = "seewave", entropy.range = c(2, 10), ...)

Arguments

X object of class 'selection_table', 'extended_selection_table' or data frame containing columns for sound file name (sound.files), selection number (select), and start and end time of signal (start and end).

type Character string to determine the type of contour to be detected. Three options are available, "dominant" (default), "fundamental" and "entropy".

wl A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.

length.out A numeric vector of length 1 giving the number of measurements of fundamental frequency desired (the length of the time series).

wn Character vector of length 1 specifying window name. Default is "hanning". See function ftwindow for more options.

ovlp Numeric vector of length 1 specifying % overlap between two consecutive windows, as in spectro. Default is 70.

bp A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz). Default is NULL.

threshold amplitude threshold (%) for fundamental frequency detection. Default is 15.

img Logical argument. If FALSE, image files are not produced. Default is TRUE.

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).

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

img.suffix A character vector of length 1 with a suffix (label) to add at the end of the names of image files.

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

clip.edges Logical argument to control whether edges (start or end of signal) in which amplitude values above the threshold were not detected will be removed. If TRUE this edges will be excluded and signal contour will be calculated on the remaining values. Default is FALSE. # @param leglab A character vector of length 1 or 2 containing the label(s) of the frequency contour legend in the output image.

leglab A character vector of length 1 or 2 containing the label(s) of the frequency contour legend in the output image.

track.harm Logical. If TRUE warbleR’s track_harmonic function is used to track dominant frequency contours. Otherwise seewave’s dfreq is used by default. Default is FALSE.
freq_ts

raw.contour Logical. If TRUE then a list with the original contours (i.e. without interpolating values to make all contours of equal length) is returned (and no images are produced).

adjust.wl Logical. If TRUE 'wl' (window length) is reset to be lower than the number of samples in a selection if the number of samples is less than 'wl'. Default is TRUE. Used only for dominant frequency detection.

ff.method Character. Selects the method used to detect fundamental frequency contours. Either 'tuneR' (using FF) or 'seewave' (using fund). Default is 'seewave'. 'tuneR' performs faster (and seems to be more accurate) than 'seewave'.

entropy.range Numeric vector of length 2. Range of frequency in which to display the entropy values on the spectrogram (when img = TRUE). Default is c(2, 10). Negative values can be used in order to stretch more the range.

Additional arguments to be passed to track_freq_contour. for customizing graphical output.

Details

This function extracts the dominant frequency, fundamental frequency or spectral entropy contours as time series. The function uses the approx function to interpolate values between frequency measures. If there are no frequencies above the amplitude threshold (for dominant and fundamental) at the beginning or end of the signals then NAs will be generated. On the other hand, if there are no frequencies above the amplitude threshold in between signal segments in which amplitude was detected then the values of this adjacent segments will be interpolated to fill out the missing values (e.g. no NAs in between detected amplitude segments).

Value

A data frame with the fundamental frequency values measured across the signals. If img is TRUE it also produces image files with the spectrograms of the signals listed in the input data frame showing the location of the fundamental frequencies (see track_freq_contour description for more details).

Author(s)

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

See Also

sig2noise, track_freq_contour, freq_ts, freq_DTW

Examples

{
  #load data
  data(list = c("Phae.long1", "Phae.long2","lbh_selec_table"))
  writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))  #save sound files
  writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))  #save sound files
  # run function with dominant frequency

freq_ts(X = lbh_selec_table, length.out = 30, flim = c(1, 12), bp = c(2, 9),
wl = 300, pb = FALSE, path = tempdir())

# note a NA in the row 4 column 3 (dfreq-1)
# this can be removed by clipping edges (removing NAs at the start and/or end
# when no freq was detected)

freq_ts(X = lbh_selec_table, length.out = 30, flim = c(1, 12), bp = c(2, 9),
wl = 300, pb = FALSE, clip.edges = TRUE, path = tempdir())

# run function with fundamental frequency
freq_ts(lbh_selec_table, type = "fundamental", length.out = 50,
flim = c(1, 12), bp = c(2, 9), wl = 300, path = tempdir())

# run function with spectral entropy
# without clip edges
freq_ts(X = lbh_selec_table, type = "entropy", threshold = 10,
clip.edges = FALSE, length.out = 10, sp.en.range = c(-25, 10), path = tempdir(),
img = FALSE)

# with clip edges and length.out 10
freq_ts(X = lbh_selec_table, type = "entropy", threshold = 10, bp = c(2, 12),
clip.edges = TRUE, length.out = 10, path = tempdir(), img = FALSE)

full_spectrogram2pdf  full_spectrogram2pdf combines full_spectrograms images in .jpeg format to a single pdf file.

Description

full_spectrogram2pdf combines full_spectrograms images in .jpeg format to a single pdf file.

Usage

full_spectrogram2pdf(keep.img = TRUE, overwrite = FALSE,
parallel = 1, path = NULL, pb = TRUE)

Arguments

keep.img Logical argument. Indicates whether jpeg files should be kept (default) or remove. (including sound file and page number) should be magnified. Default is 1.

overwrite Logical argument. If TRUE all jpeg pdf will be produced again when code is rerun. If FALSE only the ones missing will be produced. Default is FALSE.

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).

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

pb Logical argument to control progress bar. Default is TRUE.
**Details**

The function combines spectrograms for complete sound files from the `full_spectrograms` function into a single pdf (for each sound file).

**Author(s)**

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

**References**


**See Also**

`full_spectrograms`, `catalog2pdf`, blog post on spectrogram pdfs

**Examples**

```r
## Not run:
# save sound file examples
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"))

full_spectrograms(sxrow = 2, rows = 8, pal = reverse.heat.colors, wl = 300,
it = "jpeg", path = tempdir())

#now create single pdf removing jpeg
full_spectrogram2pdf(keep.img = FALSE, path = tempdir())

# check this folder
  tempdir()

## End(Not run)
```

---

**full_spectrograms**  
*Create long spectrograms of whole sound files*

**Description**

`full_spectrograms` produces image files with spectrograms of whole sound files split into multiple rows.
Usage

full_spectrograms(X = NULL, flim = NULL, sxrow = 5, rows = 10, collevels = seq(-40, 0, 1), ovlp = 50, parallel = 1, wl = 512, gr = FALSE, pal = reverse.gray.colors.2, cex = 1, it = "jpeg", flist = NULL, overwrite = TRUE, path = NULL, pb = TRUE, fast.spec = FALSE, labels = "selec", horizontal = FALSE, song = NULL, suffix = NULL, dest.path = NULL, ...)

Arguments

X 'selection_table' object or any data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end). If given, a transparent box is plotted around each selection and the selections are labeled with the selection number (and selection comment, if available). Default is NULL. Alternatively, it can also take the output of cross_correlation or auto_detect (when 'output' is a 'list', see cross_correlation or auto_detect). If supplied a secondary row is displayed under each spectrogram showing the detection (either cross-correlation scores or wave envelopes) values across time.

flim A numeric vector of length 2 indicating the highest and lowest frequency limits (kHz) of the spectrogram, as in spectro. Default is NULL.

sxrow A numeric vector of length 1. Specifies seconds of spectrogram per row. Default is 5.

rows A numeric vector of length 1. Specifies number of rows per image file. Default is 10.

collevels A numeric vector of length 3. Specifies levels to partition the amplitude range of the spectrogram (in dB). The more levels the higher the resolution of the spectrogram. Default is seq(-40, 0, 1).

ovlp Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in spectro. Default is 50. High values of ovlp slow down the function but produce more accurate selection limits (when X is provided).

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).

wl A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.

gr Logical argument to add grid to spectrogram. Default is FALSE.

pal Color palette function for spectrogram. Default is reverse.gray.colors.2. See spectro for more palettes.

cex A numeric vector of length 1 giving the amount by which text (including sound file and page number) should be magnified. Default is 1.

it A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".

flist character vector or factor indicating the subset of files that will be analyzed. Ignored if X is provided.

overwrite Logical argument. If TRUE all selections will be analyzed again when code is rerun. If FALSE only the selections that do not have a image file in the working directory will be analyzed. Default is FALSE.
**full_spectrograms**

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

- **pb**: Logical argument to control progress bar. Default is TRUE.

- **fast.spec**: Logical. If TRUE then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors gray.1, gray.2, gray.3, topo.1 and rainbow.1 (which should be imported from the package monitoR) seem to work better with 'fast' spectrograms. Palette colors gray.1, gray.2, gray.3 offer decreasing darkness levels.

- **labels**: Character string with the name of the column(s) for selection labeling. Default is 'selec'. Set to NULL to remove labels.

- **horizontal**: Logical. Controls if the images are produced as horizontal or vertical pages. Default is FALSE.

- **song**: Character string with the name of the column to used as a label a for higher organization level in the song (similar to 'song_colm' in song_analysis). If supplied then lines above the selections belonging to the same 'song' are plotted. Ignored if 'X' is not provided.

- **suffix**: Character vector of length 1. Suffix for the output image file (to be added at the end of the default file name). Default is NULL.

- **dest.path**: Character string containing the directory path where the image files will be saved. If NULL (default) then the folder containing the sound files will be used instead.

- **...**: Additional arguments for image formatting. It accepts 'width', 'height' (which will overwrite 'horizontal') and 'res' as in png.

**Details**

The function creates spectrograms for complete sound files, printing the name of the sound files and the "page" number (p1-p2...) at the upper right corner of the image files. If 'X' is supplied, the function delimits and labels the selections. This function aims to facilitate visual inspection of multiple files as well as visual classification of vocalization units and the analysis of animal vocal sequences.

**Value**

- image files with spectrograms of whole sound files in the working directory. Multiple pages can be returned, depending on the length of each sound file.

**Author(s)**

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

**References**

See Also

full_spectrogram2pdf, catalog2pdf, cross_correlation, auto_detec blog post on spectrogram pdfs

Examples

## Not run:

```r
# save sound file examples to temporary working directory
data(list = c("Phae.long1", "Phae.long2","lbh_selec_table"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))

full_spectrograms(sxrow = 2, rows = 8, pal = reverse.heat.colors, wl = 300,
path = tempdir())

# including selections
full_spectrograms(sxrow = 2, rows = 8, X = lbh_selec_table,
pal = reverse.heat.colors, overwrite = TRUE, wl = 300, path = tempdir())

#check this folder
# tempdir()
## End(Not run)
```

### gaps

#### Gap duration

gaps measures gap duration

**Usage**

gaps(X = NULL, by = "sound.files", parallel = 1, pb = TRUE)

**Arguments**

- **X**: 'selection_table', 'extended_selection_table' (created 'by.song') or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "selec": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections.
- **by**: Character vector with column names. Controls the levels at which gaps will be measured. "sound.files" must always be included.
- **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.
Details

The function measures the time intervals (i.e. gaps) between selections. The gap for a given selection is calculated as the time interval to the selection immediately after. Hence, there is no gap for the last selection in a sound file (or level determined by the 'by' argument). Note that sound files are not required.

Value

A data frame identical to that supplied in 'X', with an additional column ('gaps') with the duration of the time interval between selections.

Author(s)

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

References


See Also

inflections, song_analysis.

Examples

{
  # get warbleR sound file examples
  data(list = "lbh_selec_table")

  # get gaps
  gaps(X = lbh_selec_table)
}

image_to_wave

Convert images into wave objects

Description

image_to_wave converts images in 'png' format into wave objects using the inverse Fourier transformation.

Usage

image_to_wave(file, duration = 1, samp.rate = 44.1, bit.depth = 16, flim = c(0, samp.rate / 2), plot = TRUE)
Arguments

- **file**: Character with the name of image file to be converted. File must be in `png` format.
- **duration**: duration of the output wave object (in s).
- **samp.rate**: Numeric vector of length 1 indicating the sampling rate of the output wave object (in kHz). Default is 44.1.
- **bit.depth**: Numeric vector of length 1 with the dynamic interval (i.e. bit depth) for output files. Default is 16.
- **flim**: Numeric vector of length 2 indicating the highest and lowest frequency limits (kHz) in which the image would be located. Default is `c(0, samp.rate / 2)`.
- **plot**: Logical argument to control if image is plotted after being imported into R.

Details

This function converts images in `png` format into wave objects using the inverse Fourier transformation.

Value

- A single wave object.

Author(s)

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

References


Examples

```r
{  
  ### create image with text to use in the spectrogram  
  # remove margins of plot  
  par(mar = rep(0, 4))  
  
  # empty plot  
  plot(0, type='n', axes = FALSE, ann = FALSE, xlim = c(0, 1), ylim = c(0, 1))  

  # text to include  
  text <- "warbleR"  

  # add text  
  text(x = 0.5, y = 0.5, labels = text, cex = 11, font = 1)  

  # save image in temporary directory  
  dev2bitmap(file.path(tempdir(), "temp-img.png"), type = "pngmono", res = 30)  
}
# read it
wv <- image_to_wave(file = file.path(tempdir(), "temp-img.png"), plot = TRUE, flim = c(1, 12))

# output wave object
# wv

### plot it
# reset margins
par(mar = c(5, 4, 4, 2) + 0.1)

# plot spectrogram
# spectro(wave = wv, scale = FALSE, collevels = seq(-30, 0, 5),
# palette = reverse.terrain.colors, ovlp = 90, grid = FALSE, flim = c(2, 11))

---

**inflections**  
*Count number of inflections in a frequency contour*

Description

*inflections* counts the number of inflections in a frequency contour (or any time series)

Usage

```r
inflections(X = NULL, parallel = 1, pb = TRUE)
```

Arguments

- **X**  
  data frame with the columns for "sound.files" (sound file name), "selec" (unique identifier for each selection) and columns for each of the frequency values of the contours. No other columns should be included.

- **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.

Details

The function counts the number of inflections in a frequency contour.

Value

A data frame with 3 columns: "sound.files", "selec" and "infls" (number of inflections).

Author(s)

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


See Also

freq_ts, track_freq_contour, gaps

Examples

```r
# get warbleR sound file examples
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))
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"))

# measure frequency contours
dom.freq.ts <- freq_ts(X = lbh_selec_table, path = tempdir())

# get number of inflections
inflections(X = dom.freq.ts)
```

---

**info_sound_files**

*Get sound file parameter information*

**Description**

info_sound_files is a wrapper for selection_table that returns sound file information

**Usage**

```r
info_sound_files(path = NULL, parallel = 1, pb = TRUE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.</td>
</tr>
<tr>
<td>parallel</td>
<td>Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).</td>
</tr>
<tr>
<td>pb</td>
<td>Logical argument to control progress bar and messages. Default is TRUE.</td>
</tr>
</tbody>
</table>
info_sound_files

Details

This function is a wrapper for selection_table that returns a data frame with the following descriptive parameters for each sound file in the working directory (or 'path'):

- **duration**: duration of selection in seconds
- **sample.rate**: sampling rate in kHz
- **channels**: number of channels
- **bits**: bit depth
- **wav.size**: sound file size in MB
- **samples**: number of samples in the sound file

Value

A data frame with descriptive information about the sound files in the working directory (or 'path'). See "details".

Author(s)

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

References


See Also

fix_wavs, selection_table & check_sels

Examples

```r
{  
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "1bh_selec_table"))  
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"))  

# get info  
info_sound_files(path = tempdir())  
}
```
is_extended_selection_table

Class 'extended_selection_table': selection table containing wave objects

Description

Class for selections of signals in sound files and corresponding wave objects

Usage

is_extended_selection_table(x)

Arguments

x R object

Details

An object of class extended_selection_table created by selection_table is a list with the following elements:

- selections: data frame containing the frequency/time coordinates of the selections, sound file names, and any additional information
- check.results: results of the checks on data consistency using check_sels
- wave.objects: list of wave objects corresponding to each selection
- by.song: a list with 1) a logical argument defining if the 'extended_selection_table' was created 'by song' and 2) the name of the song column (see selection_table)

Value

A logical argument indicating whether the object class is 'extended_selection_table'

Author(s)

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

See Also

selection_table, selection_table Check if object is of class "extended_selection_table"
is_extended_selection_table Check if the object belongs to the class "extended_selection_table"
selection_table:is_selection_table
Examples
{
  data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))

  is_extended_selection_table(lbh_selec_table)

  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"))

  st <- selection_table(lbh_selec_table, extended = TRUE, confirm.extended = FALSE, path = tempdir())

  is_extended_selection_table(st)

  class(st)
}

---

is_selection_table  Class 'selection_table': double-checked frequency/time coordinates of
selections

Description
Class for selections of signals in sound files

Usage
is_selection_table(x)

Arguments
x R object.

Details
An object of class selection_table created by selection_table is a list with the following elements:

- selections: data frame containing the frequency/time coordinates of the selections, sound
  file names, and any additional information
- check.results: results of the checks on data consistency using check_sels

Value
A logical argument indicating whether the object class is 'selection_table'

Author(s)
Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)
See Also

- `selection_table` Check if object is of class "selection_table"
- `is_selection_table` Check if the object belongs to the class "selection_table"
- `selection_table`

Examples

```r
{  
  # load data
  data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))

  is_selection_table(lbh_selec_table)

  # save wave files in temporary directory
  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"))

  st <- selection_table(lbh_selec_table, path = tempdir())

  is_selection_table(st)
  class(st)
}
```

`lbh_selec_table`  
Example data frame of selections (i.e. `selection_table`).

Description

`lbh_selec_table` alternative name for `selec.table`. `selec.table` will be deprecated in future versions.

Usage

```r
data(lbh_selec_table)
```

Format

A data frame with 11 rows and 7 variables:

- `sound.files` recording names
- `channel` channel in which signal is found
- `selec` selection numbers within recording
- `start` start times of selected signal
- `end` end times of selected signal
- `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 example sound files included in this package. Same data than 'selec_table'. 'selec_table' will be removed in future package version.

Source

Marcelo Araya-Salas, warbleR

---

**lbh_selec_table2**  
*Example data frame of selections (i.e. selection table).*

**Description**

lbh_selec_table2 is a data frame containing the start, end, low and high frequency of 2 selections. Mostly to be used as an example in `find_peaks`.

**Usage**

data(lbh_selec_table2)

**Format**

A data frame with 11 rows and 7 variables:

- `sound.files` recording names
- `channel` channel in which signal is found
- `selec` selection numbers within recording
- `start` start times of selected signal
- `end` end times of selected signal
- `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 example sound files included in this package. Similar than 'lbh_selec_table', but it contains only 2 selections.

**Source**

Marcelo Araya-Salas, warbleR
map_xc  

Maps of 'Xeno-Canto' recordings by species

Description

map_xc creates maps to visualize the geographic spread of 'Xeno-Canto' recordings.

Usage

map_xc(X, img = TRUE, it = "jpeg", res = 100, labels = FALSE, path = NULL, leaflet.map = FALSE, leaflet.cluster = FALSE)

Arguments

- **X**: Data frame output from `query_xc`.
- **img**: A logical argument specifying whether an image file of each species map should be returned, default is TRUE.
- **it**: A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".
- **res**: Numeric argument of length 1. Controls image resolution. Default is 100 (faster) although 300 - 400 is recommended for publication/presentation quality.
- **labels**: A logical argument defining whether dots depicting recording locations are labeled. If TRUE then the Recording_ID is used as label.
- **path**: Character string with the directory path where the image files will be saved. If NULL (default) then the current working directory is used. Ignored if img = FALSE.
- **leaflet.map**: Logical to control whether the package 'leaflet' is used for displaying the maps. 'leaflet' maps are interactive and display information about recordings and links to the Xeno-Canto website. If TRUE a single map is displayed regardless of the number of species and all other image related arguments are ignored. Default is FALSE. The hovering label shows the species scientific name (or the subspecies if only 1 species is present in 'X'). Note that colors will be recycled if more after 18 species (or subspecies).
- **leaflet.cluster**: Logical to control if icons are clustered by locality (as in Xeno-Canto maps). Default is FALSE.

Details

This function creates maps for visualizing the geographic spread of recordings from the open-access online repository Xeno-Canto. The function takes the output of `query_xc` as input. Maps can be displayed in the graphic device (or Viewer if 'leaflet.map = TRUE') or saved as images in the working directory. Note that only recordings with geographic coordinates are displayed.
**Value**

A map of 'Xeno-Canto' recordings per species (image file), or a faceted plot of species map(s) in the active graphic device.

**Author(s)**

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>) and Grace Smith Vidaurre

**References**


**Examples**

```r
## Not run:
# search in xeno-canto
X <- query_xc("Phaethornis anthophilus", download = FALSE)

# create image in R graphic device
map_xc(X, img = FALSE)

# create leaflet map
map_xc(X, leaflet.map = TRUE)

## End(Not run)
```

---

**mfcc_stats**

*Calculate descriptive statistics on Mel-frequency cepstral coefficients*

**Description**

mfcc_stats calculates descriptive statistics on Mel-frequency cepstral coefficients and its derivatives.

**Usage**

```r
mfcc_stats(X, ovlp = 50, wl = 512, bp = 'frange', path = NULL, numcep = 25, nbands = 40, parallel = 1, pb = TRUE, ...)
```

**Arguments**

X

'selection_table', 'extended_selection_table' or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. The output of auto_detect can be used as the input data frame.
ovlp  Numeric vector of length 1 specifying % of overlap between two consecutive windows. Internally this is used to set the 'hoptime' argument in melfcc. Default is 50.
wl   A numeric vector of length 1 specifying the spectrogram window length. Default is 512. See 'wl.freq' for setting windows length independently in the frequency domain.
bp   A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or "frange" (default) to indicate that values in minimum of 'bottom.freq' and maximum of 'top.freq' columns will be used as bandpass limits.
path Character string containing the directory path where the sound files are located.
umcep Numeric vector of length 1 controlling the number of cepstra to return (see melfcc).
nbands Numeric vector of length 1 controlling the number of warped spectral bands to use (see melfcc). Default is 40.
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.
... Additional parameters to be passed to melfcc.

Details

The function calculates descriptive statistics on Mel-frequency cepstral coefficients (MFCCs) for each of the signals (rows) in a selection data frame. The descriptive statistics are: minimum, maximum, mean, median, skewness, kurtosis and variance. It also returns the mean and variance for the first and second derivatives of the coefficients. These parameters are commonly used in acoustic signal processing and detection (e.g. Salamon et al 2014).

Value

A data frame containing the descriptive statistics for each of the Mel-frequency cepstral coefficients (set by 'numcep' argument). See details.

Author(s)

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

References


move_imgs

See Also

fix_wavs, remove_silence, spectro_analysis

Examples

```r
{  
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))  
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"))  
  
  # run function  
  mel_st <- mfcc_stats(X = lbh_selec_table, pb = FALSE, path = tempdir())  
  head(mel_st)  
  
  # measure 12 coefficients  
  mel_st12 <- mfcc_stats(X = lbh_selec_table, numcep = 12, pb = FALSE, path = tempdir())  
  head(mel_st12)  
}
```

move_imgs

Move/copy image files between directories

Description

move_imgs moves/copies image files created by warbleR between directories (folders).

Usage

```r
move_imgs(from = NULL, to = NULL, it = "all", cut = TRUE,  
overwrite = FALSE, create.folder = TRUE, folder.name = "image_files",  
parallel = 1, pb = TRUE)
```

Arguments

- `from`: Directory path where image files to be copied are found. If NULL (default) then the current working directory is used.
- `to`: Directory path where image files will be copied to.
- `it`: A character vector of length 1 giving the image type to be used. "all", "tiff", "jpeg" and "pdf" are admitted ("all" includes all the rest). Default is "all".
- `cut`: Logical. Determines if files are removed from the original location after being copied (cut) or not (just copied). Default is TRUE.
- `overwrite`: Logical. Determines if files that already exist in the destination directory should be overwritten. Default is FALSE.
create.folder Logical. Determines if files are moved to a new folder (which is named with the "folder.name" argument). Ignored if ‘to’ is provided. Default is TRUE.

folder.name Character string with the name of the new folder where the files will be copied to. Ignored if ‘to’ is provided. Default is "image_files".

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 aims to simplify the manipulation of the image files generated by many of the warbleR function. It copies/cuts files between directories.

Author(s)

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

References


See Also

gfiltersels

Other data manipulation: open_wd(), split_sound_files()

Examples

{
  # load data
data("Cryp.soui")
writeWave(Cryp.soui, file.path(tempdir(), "Cryp.soui.wav")) # save sound files

  # autodetect location of signals
ad <- autodetect(threshold = 6, bp = c(1, 3), mindur = 1.2, maxdur = 3, ssmooth = 600, wl = 300, flist = "Cryp.soui.wav", path = tempdir())

  # track dominant frequency graphs with freq range detection
track_freq_contour(X = ad[!is.na(ad$start),], flim = c(0, 5), ovlp = 90, it = "tiff", bp = c(1, 3), contour = "df", wl = 300, frange = TRUE, path = tempdir())

  # create folder to move image files
dir.create(file.path(tempdir(), "imgs"))

  # copy files
move_imgs(cut = FALSE, from = tempdir(), to = file.path(tempdir(), "imgs"))

  # cut files
move_imgs(cut = TRUE, from = tempdir(),}
mp32wav

Convert .mp3 files to .wav

Description

mp32wav converts several .mp3 files in working directory to .wav format

Usage

mp32wav(samp.rate = NULL, parallel = 1, path = NULL, dest.path = NULL, bit.depth = 16, pb = TRUE, overwrite = FALSE)

Arguments

samp.rate Sampling rate in kHz at which the .wav files should be written. If not provided the sample rate of the original .mp3 file is used. THIS FEATURE IS CURRENTLY NOT AVAILABLE. However, downsampling can be done after .mp3’s have been converted using the fix_wavs function (which uses SOX instead). Default is NULL (e.g. keep original sampling rate).

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).

path Character string containing the directory path where the .mp3 files are located. If NULL (default) then the current working directory is used.

dest.path Character string containing the directory path where the .wav files will be saved. If NULL (default) then the folder containing the sound files will be used.

bit.depth Character string containing the units to be used for amplitude normalization. Check normalize for details. Default is 16.

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

overwrite Logical. Control whether a .wav sound file that is already in the working directory should be overwritten.

Details

The function will convert all mp3 files in working directory or 'path' supplied to wav format. bioacoustics package must be installed when changing sampling rates (i.e. if 'samp.rate' is supplied). Note that sound files are normalized using normalize so they can be written by writeWave.

convert all .mp3 files in working directory to .wav format. Function used internally to read .mp3 files (readMP3) sometimes crashes.
multi_DTW

Value
.
.wav files saved in the working directory with same name as original mp3 files.

Author(s)
Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>) and Grace Smith Vidaurre

References

Examples
```r
## Not run:
# download mp3 files from xeno-canto
query_xc(qword = "Phaethornis aethopygus", download = TRUE, path = tempdir())

# Convert all files to .wav format
mp32wav(path = tempdir(), dest.path = tempdir())

#check this folder!!
tempdir()

## End(Not run)
```

---

**multi_DTW**

A wrapper on `dtwDist` for comparing multivariate contours

**Description**

`multi_DTW` is a wrapper on `dtwDist` that simplify applying dynamic time warping on multivariate contours.

**Usage**

```r
multi_DTW(ts.df1 = NULL, ts.df2 = NULL, pb = TRUE, parallel = 1, window.type = "none", open.end = FALSE, scale = FALSE, dist.mat = TRUE, ...)
```

**Arguments**

- `ts.df1` Optional. Data frame with frequency contour time series of signals to be compared.
- `ts.df2` Optional. Data frame with frequency contour time series of signals to be compared.
- `pb` Logical argument to control progress bar. Default is TRUE. Note that progress bar is only used when parallel = 1.
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). Not available in Windows OS.

window.type dtw windowing control parameter. Character: "none", "itakura", or a function (see dtw).

open.end dtw control parameter. Performs open-ended alignments (see dtw).

scale Logical. If TRUE dominant frequency values are z-transformed using the scale function, which "ignores" differences in absolute frequencies between the signals in order to focus the comparison in the frequency contour, regardless of the pitch of signals. Default is TRUE.

dist.mat Logical controlling whether a distance matrix (TRUE, default) or a tabular data frame (FALSE) is returned.

... Additional arguments to be passed to track_freq_contour for customizing graphical output.

Details

This function extracts the dominant frequency values as a time series and then calculates the pairwise acoustic dissimilarity using dynamic time warping. The function uses the approx function to interpolate values between dominant frequency measures. If 'img' is TRUE the function also produces image files with the spectrograms of the signals listed in the input data frame showing the location of the dominant frequencies.

Value

A matrix with the pairwise dissimilarity values. If img is FALSE it also produces image files with the spectrograms of the signals listed in the input data frame showing the location of the dominant frequencies.

Author(s)

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

References


See Also

freq_ts, blog post on DTW similarity

Other spectrogram creators: color_spectro(), freq_DTW(), phylo_spectro(), snr_spectrograms(), spectrograms(), track_freq_contour()
new_function_names

## Description

A data frame containing the old and new names for warbleR functions.

## Usage

```r
data(new_function_names)
```

## Format

An object of class data.frame with 34 rows and 2 columns.

## Author(s)

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

| open_wd | Open working directory |

**Description**

open_wd opens the working directory in the default file browser.

**Usage**

`open_wd(path = getwd(), verbose = TRUE)`

**Arguments**

- `path` Directory path to be opened. By default it’s the working directory. 'wav.path' set by `warbleR_options` is ignored in this case.
- `verbose` Logical to control whether the 'path' is printed in the console. Default is `TRUE`.

**Details**

The function opens the working directory using the default file browser and prints the working directory in the R console. This function aims to simplify the manipulation of sound files and other files produced by many of the `warbleR` function.

**Author(s)**

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

**References**


**See Also**

- `move_imgs`
- Other data manipulation: `move_imgs()`, `split_sound_files()`

**Examples**

```r
{
  open_wd()
}
```
optimize_auto_detec  
Optimize the detection of signals based on a-priori detections

Description
Optimize the detection of signals based on a-priori detections

Usage
optimize_auto_detec(X, Y, threshold = 10, power = 1, wl = 512, ssmooth = 0, hold.time = 0, mindur = NULL, maxdur = NULL, thinning = 1, parallel = 1, pb = TRUE, by.sound.file = FALSE, bp = NULL, path = NULL, previous.output = NULL)

Arguments
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). It should contain the selections that will be used for detection optimization.

Y Optional. An object of class 'autodetec.output' (generated by auto_detec) in which to optimize detections. Must refer to the same sound files as in 'X'. Default is (NULL).

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

mindur 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.

maxdur 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.**

**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 to control if diagnostics are calculated for each sound file independently (`TRUE`) or for all sound files combined (`FALSE`, default).

**bp**

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

**path**

Character string containing the directory path where the sound files are located. If `NULL` (default) then the current working directory is used. Only needed if `Y` is not supplied.

**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' ('X') and the output of a `auto_detec` routine ('Y') 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 in 'X'. The function returns several diagnostic metrics to allow user to determine which parameter values provide a detection that more closely matches the selections in 'X'. Those parameters can be later used for performing a more efficient detection using `auto_detec`.

**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 'X'. 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 'X'.
- **false.positives**: number of detections that don’t match any of the signals referenced in 'X'. 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 'X' 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 'X'. In a perfect detection routine it should be 1.

• **sensitivity**: Proportion of signals referenced in 'X' that were detected. In a perfect detection routine it should be 1.

• **specificity**: Proportion of detections that correspond to signals referenced in 'X' that were detected. In a perfect detection routine it should be 1.

**Author(s)**

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

**References**


**Examples**

```r
{  
  # Save to temporary working directory  
  data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))  
  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"))  
  
  # run auto_detec with thinning  
  ad <- auto_detec(output = "list", thinning = 1 / 10, ssSmooth = 300, path = tempdir())  
  optimize_auto_detec(X = lbh_selec_table, Y = ad, threshold = c(5, 10, 15), path = tempdir())  
}
```

**overlapping_sels**

Find overlapping selections

**Description**

overlapping_sels finds which selections overlap in time within a given sound file.

**Usage**

```r
overlapping_sels(X, index = FALSE, pb = TRUE, max.ovlp = 0, relabel = FALSE,  
drop = FALSE, priority = NULL, priority.col = NULL, unique.labs = NULL,  
indx.row = FALSE, parallel = 1, verbose = TRUE)
```
overlapping_sels

Arguments

- **X**: 'selection_table' object or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "selec": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections.

- **index**: Logical. Indicates if only the index of the overlapping selections would be returned. Default is FALSE.

- **pb**: Logical argument to control progress bar and messages. Default is TRUE.

- **max.ovlp**: Numeric vector of length 1 specifying the maximum overlap allowed (in seconds). Default is 0.

- **relabel**: Logical. If TRUE then selection names ('selec' column) are reset within each sound files. Default is FALSE.

- **drop**: Logical. If TRUE, when 2 or more selections overlap the function will remove all but one of the overlapping selection. Default is FALSE.

- **priority**: Character vector. Controls the priority criteria used for removing overlapped selections. It must list the levels of the column used to determine priority (argument priority.col) in the desired priority order. Default is NULL.

- **priority.col**: Character vector of length 1 with the name of the column use to determine the priority of overlapped selections. Default is NULL.

- **unique.labs**: DEPRECATED.

- **indx.row**: Logical. If TRUE then a character column with the indices of all selections that overlapped with each selection is added to the output data frame (if index = TRUE). For instance, if the selections in rows 1,2 and 3 all overlapped with each other, the 'indx.row' value would be "1/2/3" for all. However, if selection 3 only overlaps with 2 but not with 1, then it returns, "1/2" for row 1, "1/2/3" for row 2, and "2/3" for row 3. Default is FALSE.

- **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).

- **verbose**: Logical to control if messages are printed to the console.

Details

This function detects selections within a selection table that overlap in time. Selections must be listed in a data frame similar to lbh_selec_table. Note that row names are set to 1:nrow(X).

Value

A data frame with the columns in X plus an additional column ('ovlp.sels') indicating which selections overlap. The ones with the same number overlap with each other. If drop = TRUE only the non-overlapping selections are return. If 2 or more selections overlap only the first one is kept. The arguments 'priority' and 'priority.col' can be used to modified the criterium for dropping overlapping selections.

Author(s)

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


See Also

filtersels, lbh_selec_table

Examples

```
# no overlap
overlapping_sels(X = lbh_selec_table)

# modified lbh_selec_table to make the first and second selection overlap
Y <- lbh_selec_table
Y$end[4] <- 1.5
overlapping_sels(X = Y)

# drop overlapping
overlapping_sels(X = Y, drop = TRUE)

# get index instead
overlapping_sels(X = Y, index = TRUE)
```

**phylo_spectro**

Add spectrograms onto phylogenetic trees

Description

phylo_spectro Add spectrograms to the tips of an objects of class phylo.

Usage

```
phylo_spectro(X, tree, type = "phylogram", par.mar = rep(1, 4),
size = 1, offset = 0, path = NULL, ladder = NULL, horizontal = TRUE, ...)
```

Arguments

- **X** 'selection_table', 'extended_selection_table' or data frame containing columns for sound file name (sound.files), selection number (selec), and start and end time of signals (start and end). 'top.freq' and 'bottom.freq' columns are optional. In addition, the data frame must include the column 'tip.label' that contains the names of the tip labels found in the tree (e.g. `tree$tip.label`). This column is used to match rows and tip labels. If using an 'extended_selection_table' the sound files are not required (see selection_table).
Object of class 'phylo' (i.e. a phylogenetic tree). Ultrametric trees may produce better results. If NULL (default) then the current working directory is used. Tip labels must match the names provided in the 'tip.label' column in 'X' (see `X` argument).

Character string of length 1 specifying the type of phylogeny to be drawn (as in `plot.phylo`). Only 'phylogram' (default) and 'fan' are allowed.

Numeric vector with 4 elements, default is `rep(1, 4)`. Specifies the number of lines in inner plot margins where axis labels fall, with form c(bottom, left, top, right). See `par`. See 'inner.par' argument for controlling spectrogram margins.

Numeric vector of length 1 controlling the relative size of spectrograms. Higher numbers increase the height of spectrograms. Default is 1. Numbers between range c(>0, Inf) are allowed.

Numeric vector of length 1 controlling the space between tips and spectrograms. Default is 0.

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

Character string controlling whether the phylogeny is ladderized (i.e. the internal structure of the tree is reorganized to get the ladderized effect when plotted). Only 'left' of 'right' values are accepted. Default is NULL (no ladderization). See `ladderize` for more details.

Logical. Controls whether spectrograms in a fan phylogeny are placed in a horizontal position `FALSE` or in the same angle as the tree tips. Currently only horizontal spectrograms are available.

Additional arguments to be passed to the internal spectrogram creating function (`spectrograms`) or phylogeny plotting function (`plot.phylo`) for customizing graphical output. Only rightwards phylogenies can be plotted.

The function adds the spectrograms of sounds annotated in a selection table ('X' argument) onto the tips of a phylogenetic tree. The 'tip.label' column in 'X' is used to match spectrograms and tree tips. The function uses internally the `plot.phylo` function to plot the tree and the `spectrograms` function to create the spectrograms. Arguments for both of these functions can be provided for further customization.

A phylogenetic tree with spectrograms on tree tips is plotted in the current graphical device.

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

See Also

spectrograms.plot.phylo

Other spectrogram creators: color_spectro(), freq_DTW(), multi_DTW(), snr_spectrograms(), spectrograms(), track_freq_contour()

Examples

{

    # First set empty folder
    # save example sound files
    data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "lbh_selec_table"))
    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"))

    # set spectrogram options (can be done at the phylo_spectro() function too)
    warbleR_options(wl = 200, ovlp = 90, flim = "frange", wav.path = tempdir())

    # subset example selection table
    X <- lbh_selec_table[1:8, ]

    # create random tree (need ape to be installed)
    set.seed(1)
    tree <- ape::rtree(nrow(X))

    # Force tree to be ultrametric
    tree <- ape::chronoMPL(tree)

    # add tip label column to example selection table (just for the sake of the example)
    X$tip.label <- tree$tip.label

    # print phylogram with spectros
    phylo_spectro(X = X, tree = tree, par.mar = c(0, 0, 0, 8), size = 2)

    # no margin in spectrograms and showing tip labels (higher offset)
    phylo_spectro(X = X, tree = tree, offset = 0.1, par.mar = c(0, 0, 0, 6),
                  inner.mar = rep(0, 4), size = 2)

    # print fan tree and no margin in spectrograms
    phylo_spectro(X = X, tree = tree, offset = 0.6, par.mar = rep(3, 4),
                  inner.mar = rep(0, 4), size = 2, type = "fan", show.tip.label = FALSE)

    # changing edge color and width
    phylo_spectro(X = X, tree = tree, offset = 0.2, par.mar = rep(3, 4),
                  inner.mar = rep(0, 4), size = 2, type = "fan", show.tip.label = FALSE, edge.color = "red", edge.width = 2)

    # plotting a tree representing cross-correlation distances
    xcorr_mat <- cross_correlation(X, bp = c(1, 10))
plot_coordination <- ape::chronoMPL(ape::as.phylo(hclust(as.dist(1 - xcorr_mat))))

X$tip.label <- xc.tree$tip.label

phylo_spectro(X = X, tree = xc.tree, offset = 0.03, par.mar = rep(3, 4),
inner.mar = rep(0, 4), size = 0.3, type = "fan", show.tip.label = FALSE,
edge.color = "red", edge.width = 2)
}

plot_coordination  Coordinated singing graphs

Description

plot_coordination creates graphs of coordinated singing and highlights the signals that overlap in time. The signals are represented by polygons of different colors.

Usage

plot_coordination(X, only.coor = FALSE, ovlp = TRUE, xl = 1, res = 80, it = "jpeg",
img = TRUE, tlim = NULL, pb = TRUE)

Arguments

X Data frame containing columns for singing event (sing.event), individual (indiv), and start and end time of signal (start and end).

only.coor Logical. If TRUE only the segment in which both individuals are singing is included (solo singing is removed). Default is FALSE.

ovlp Logical. If TRUE the vocalizations that overlap in time are highlighted. Default is TRUE.

xl Numeric vector of length 1, a constant by which to scale spectrogram width. Default is 1.

res Numeric argument of length 1. Controls image resolution. Default is 80.

it A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".

img Logical argument. If FALSE, image files are not produced and the graphs are shown in the current graphic device. Default is TRUE.

tlim Numeric vector of length 2 indicating the start and end time of the coordinated singing events to be displayed in the graphs.

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

Details

This function provides visualization for coordination of acoustic signals. Signals are shown as polygon across a time axis. It also shows which signals overlap, the amount of overlap, and highlights the individual responsible for the overlap using a color code. The width of the polygons depicting the time of overlap.
Value

The function returns a list of graphs, one for each singing event in the input data frame. The graphs can be plotted by simply calling the list. If `img` is TRUE then the graphs are also saved in the working directory as files.

Author(s)

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

References


Examples

```r
## Not run:
# load simulate singing events (see data documentation)
data(sim_coor_sing)

# make plot_coordination in graphic device format
cgs <- plot_coordination(X = sim_coor_sing, ovlp = TRUE, only.coor = FALSE, img = FALSE)
cgs
## End(Not run)
```

query_xc

### Access ‘Xeno-Canto’ recordings and metadata

Description

query_xc downloads recordings and metadata from Xeno-Canto.

Usage

```r
query_xc(qword, download = FALSE, X = NULL, file.name = c("Genus", "Specific_epithet"), parallel = 1, path = NULL, pb = TRUE)
```

Arguments

- `qword` Character vector of length one indicating the genus, or genus and species, to query ‘Xeno-Canto’ database. For example, *Phaethornis* or *Phaethornis longirostris*. More complex queries can be done by using search terms that follow the xeno-canto advance query syntax. This syntax uses tags to search within a particular aspect of the recordings (e.g. country, location, sound type). Tags are
of the form tag:searchterm’. For instance, 'type:song' will search for all recordings in which the sound type description contains the word 'song'. Several tags can be included in the same query. The query "phaethornis cnt:belize" will only return results for birds in the genus Phaethornis that were recorded in Belize. See Xeno-Canto’s search help for a full description and see examples below for queries using terms with more than one word.

download Logical argument. If FALSE only the recording file names and associated metadata are downloaded. If TRUE, recordings are also downloaded to the working directory as .mp3 files. Default is FALSE. Note that if the recording is already in the working directory (as when the downloading process has been interrupted) it will be skipped. Hence, resuming downloading processes will not start from scratch.

X Data frame with a 'Recording_ID' column and any other column listed in the file.name argument. Only the recordings listed in the data frame will be download (download argument is automatically set to TRUE). This can be used to select the recordings to be downloaded based on their attributes.

file.name Character vector indicating the tags (or column names) to be included in the sound file names (if download = TRUE). Several tags can be included. If NULL only the ‘Xeno-Canto’ recording identification number (“Recording_ID") is used. Default is c("Genus", "Specific epithet"). Note that recording id is always used (whether or not is listed by users) to avoid duplicated names.

parallel Numeric. Controls whether parallel computing is applied when downloading mp3 files. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing). Applied both when getting metadata and downloading files.

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

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

Details

This function queries for avian vocalization recordings in the open-access online repository Xeno-Canto. It can return recordings metadata or download the associated sound files. Complex queries can be done by using search terms that follow the xeno-canto advance query syntax (check "qword" argument description). Files are double-checked after downloading and "empty" files are re-downloaded. File downloading process can be interrupted and resume later as long as the working directory is the same. Maps of recording coordinates can be produced using map_xc.

Value

If X is not provided the function returns a data frame with the following recording information: recording ID, Genus, Specific epithet, Subspecies, English name, Recordist, Country, Locality, Latitude, Longitude, Vocalization type, Audio file, License, URL, Quality, Time, Date. Sound files in .mp3 format are downloaded into the working directory if download = TRUE or if X is provided; a column indicating the names of the downloaded files is included in the output data frame.

Author(s)

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


See Also

map_xc, blog post on accessing Xeno-Canto recordings

Examples

```r
## Not run:
# search without downloading
df1 <- query_xc(qword = 'Phaethornis anthophilus', download = FALSE)
View(df1)

# downloading files
query_xc(qword = 'Phaethornis anthophilus', download = TRUE, path = tempdir())

# check this folder
tempdir()

## search using xeno-canto advance query ###
orth.pap <- query_xc(qword = 'gen:orthonyx cnt:papua loc:tari', download = FALSE)

# download file using the output data frame as input
query_xc(X = orth.pap, path = tempdir())

# use quotes for queries with more than 1 word (e.g. Costa Rica), note that the
# single quotes are used for the whole 'qword' and double quotes for the 2-word term inside
# Phaeochroa genus in Costa Rica
phae.cr <- query_xc(qword = 'gen:phaeochroa cnt:"costa rica"', download = FALSE)

# several terms can be searched for in the same field
# search for all female songs in sound type
femsong <- query_xc(qword = 'type:song type:female', download = FALSE)

## End(Not run)
```

**read_sound_file**

An extended version of read_wave that reads several sound file formats and files from selection tables

**Description**

read_sound_file reads several sound file formats as well as files referenced in selection tables
read_sound_file

Usage

read_sound_file(X, index, from = X$start[index], to = X$end[index],
channel = X$channel[index], header = FALSE, path = NULL)

Arguments

X 'data.frame', 'selection_table' or 'extended_selection_table' containing columns for sound file name (sound.files), selection number (selec), and start and end time of signals (start and end). Alternatively, the name of a sound file or URL address to sound file can be provided. The function can read sound files in 'wav', 'mp3', 'flac' and 'wac' format. The file name can contain the directory path. 'top.freq' and 'bottom.freq' columns are optional. Default is NULL.

index Index of the selection in 'X' that will be read. Ignored if 'X' is NULL.

from Where to start reading, in seconds. Default is X$start[index].
to Where to stop reading, in seconds. Default is X$end[index].

channel Channel to be read from sound file (1 = left, 2 = right, or higher number for multichannel waves). Default is X$channel[index]. If a 'channel' column does not exist it will read the first channel.

header If TRUE, only the header information of the Wave object is returned, otherwise (the default) the whole Wave object.

path Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used. If 'X' refers to a sound file including its directory 'path' is ignored.

Details

The function is a wrapper for readWave that read sound files, including those referenced in selection tables. It is also used internally by warbleR functions to read wave objects from extended selection tables (see selection_table for details). Note that reading 'flac' files requires creating a temporary copy in 'wav' format, which can be particularly slow for long files.

Value

An object of class "Wave".

Author(s)

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

References

### Examples

```
## Not run:
# write wave files with lower case file extension
data(list = c("Phae.long1"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))

# read from selection table
read_sound_file(X = lbh_selec_table, index = 1, path = tempdir())

# from extended selection table
library(NatureSounds)
read_sound_file(X = lbh.est, index = 1)

# read from selection table
read_sound_file(X = lbh_selec_table, index = 1, path = tempdir())

# read WAV
filepath <- system.file("extdata", "recording.wav", package = "bioacoustics")
read_sound_file(filepath)

# read MP3
filepath <- system.file("extdata", "recording.mp3", package = "bioacoustics")
read_sound_file(filepath)

# read WAC
filepath <- system.file("extdata", "recording_20170716_230503.wac", package = "bioacoustics")
read_sound_file(filepath, from = 0, to = 0.2)

# URL file
read_sound_file(X = 'https://www.xeno-canto.org/513948/download')
```

## End(Not run)

---

**read_wave**  
A wrapper for tuneR's `readWave` function that reads sound files listed within selection tables

### Description

`read_wave` is a wrapper for tuneR’s `readWave` function that reads sound files listed in data frames and selection tables.

### Usage

```
read_wave(X, index, from = X$start[index], to = X$end[index], channel = NULL, 
header = FALSE, path = NULL)
```
Arguments

- **X**: 'data.frame', 'selection_table' or 'extended_selection_table' containing columns for sound file name (sound.files), selection number (selec), and start and end time of signals (start and end). Alternatively, the name of a '.wav' file or URL address to a '.wav' or '.mp3' file can be provided. The file name can contain the directory path. 'top.freq' and 'bottom.freq' columns are optional. Default is NULL.

- **index**: Index of the selection in 'X' that will be read. Ignored if 'X' is NULL.

- **from**: Where to start reading, in seconds. Default is X$start[index].

- **to**: Where to stop reading, in seconds. Default is X$end[index]. Inf can be used for reading the entire sound file (when 'X' is a sound file name).

- **channel**: Channel to be read from sound file (1 = left, 2 = right, or higher number for multichannel waves). If NULL (default) or higher than the number of channels in a wave then the first channel is used. Only applies to '.wav' files in local directories.

- **header**: If TRUE, only the header information of the Wave object is returned, otherwise (the default) the whole Wave object.

- **path**: Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used. If 'X' refers to a sound file including its directory 'path' is ignored.

Details

The function is a wrapper for readWave that read sound files listed within selection tables. It is also used internally by warbleR functions to read wave objects from extended selection tables (see selection_table for details).

Value

An object of class "Wave".

Author(s)

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

References


Examples

```r
# write wave files with lower case file extension
data(list = c("Phae.long1"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))

# read from selection table
```
```r
read_wave(X = lbh_selec_table, index = 1, path = tempdir())

# from extended selection table
library(NatureSounds)
read_wave(X = lbh.est, index = 1)

# read WAV
filepath <- system.file("extdata", "recording.wav", package = "bioacoustics")
read_wave(filepath)

# read MP3
filepath <- system.file("extdata", "recording.mp3", package = "bioacoustics")
read_wave(filepath)

# URL file
read_wave(X = "https://www.xeno-canto.org/513948/download")

remove_channels(files = NULL, channels, path = NULL, parallel = 1, pb = TRUE)
```

### Description

remove_channels remove channels in wave files

### Usage

```r
remove_channels(files = NULL, channels, path = NULL, parallel = 1, pb = TRUE)
```

### Arguments

- **files**
  - Character vector indicating the files that will be analyzed. If not provided, optional, then all wave files in the working directory (or path) will be processed.
- **channels**
  - Numeric vector indicating the index (or channel number) for the channels that will be kept (left = 1, right = 2; 3 to inf for multichannel sound files).
- **path**
  - Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
- **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.

### Details

The function removes channels from wave files. It works on regular and multichannel wave files. Converted files are saved in a new directory ("converted_sound_files") and original files are not modified.
Value

Sound files that have been converted are saved in the new folder "converted_sound_files". If 'img = TRUE' then spectrogram images highlighting the silence segments that were removed are also saved.

Author(s)

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

References


See Also

fix_wavs, info_wavs,

Examples

```r
# save sound file examples
data("Phae.long1")
Phae.long1.2 <- stereo(Phae.long1, Phae.long1)
writeWave(Phae.long1.2, file.path(tempdir(), "Phae.long1.2.wav"))
remove_channels(channels = 1, path = tempdir())
#check this floder
tempdir()
```

remove_silence

Remove silence in wave files

Description

remove_silence Removes silences in wave files

Usage

```r
remove_silence(path = NULL, min.sil.dur = 2, img = TRUE, it = "jpeg", flim = NULL, files = NULL, flist = NULL, parallel = 1, pb = TRUE, downsample = TRUE)
```
remove_silence

Arguments

- **path**: Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
- **min.sil.dur**: Numeric. Controls the minimum duration of silence segments that would be removed.
- **img**: Logical argument. If FALSE, image files are not produced. Default is TRUE.
- **it**: A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".
- **flim**: A numeric vector of length 2 indicating the highest and lowest frequency limits (kHz) of the spectrogram as in spectro. Default is NULL. Ignored if ‘img = FALSE’.
- **files**: character vector or factor indicating the subset of files that will be analyzed. If not provided then all wave files in the working directory (or path) will be processed.
- **flist**: DEPRECATED. Please use ‘files’ instead.
- **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.
- **downsample**: Logical indicating whether files are downsampled to a 5000 kHz sampling rate. This can be used to speed up the process, but will make the function ignore sound/noise above 2500 kHz. Default is TRUE.

Details

The function removes silence segments (i.e. segments with very low amplitude values) from wave files.

Value

Sound files for which silence segments have been removed are saved in the new folder "silence-removed_files" in .wav format. If ‘img = TRUE’ then spectrogram images highlighting the silence segments that were removed are also saved.

Author(s)

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

References


See Also

fix_wavs, auto_detec
Examples

```
# save sound file examples
data(list = c("Phae.long1", "Phae.long2","1bh_selec_table"))
sil <- silence(samp.rate = 22500, duration = 3, xunit = "time")

wv1 <- pastew(pastew(Phae.long1, sil, f = 22500, output = "Wave"),
Phae.long2, f = 22500, output = "Wave")

#check silence in between amplitude peaks
env(wv1)

#save wave file
writeWave(object = wv1, filename = file.path(tempdir(), "wv1.wav"),
extensible = FALSE)

#remove silence
# remove_silence(files = "wv1.wav", pb = FALSE, path = tempdir())

#check this floder
tempdir()
```

rename_est_waves

Rename wave objects and associated metadata in extended selection tables

**Description**

rename_est_waves rename wave objects and associated metadata in extended selection tables

**Usage**

```r
rename_est_waves(X, new.sound.files, new.selec = NULL)
```

**Arguments**

- **X** object of class `extended_selection_table`.
- **new.sound.files** Character vector of length equals to the number of rows in `X`. Specifies the new names to be used for wave objects and sound file column. Note that this will rename wave objects and associated attributes and data in `X`. Must be provided and must contain unique labels for each row if the extended selection table was created by element (see `selection_table`). If created by song, then a single name for each sound file should be supplied.
rename_est_waves

new.selec Numeric or character vector of length equals to the number of rows in 'X' to specify the 'selec' column labels. Default is NULL. If not provided the 'selec' column is kept unchanged. Note that the combination of 'sound.files' and 'selec' columns must produce unique IDs for each selection (row).

Details
This function allows users to change the names of 'sound.files' and 'selec' columns in extended selection tables. These names can become very long after manipulations used to produce extended tables.

Value
An extended selection table with rename sound files names in data frame and attributes. The function adds columns with the previous sound file names (and 'selec' if provided).

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

References

See Also
Other extended selection table manipulation: resample_est_waves()

Examples
{
  data("lbh.est")

  # order by sound file name
  lbh.est <- lbh.est[order(lbh.est$sound.files),]

  # create new sound file name
  nsf <- sapply(strsplit(lbh.est$sound.files, ".wav", fixed = TRUE), ".[1]"

  slc <- vector(length = nrow(lbh.est))
  slc[1] <- 1
  for(i in 2:length(slc))
    if (nsf[i - 1] == nsf[i]) slc[i] <- slc[i - 1] + 1 else slc[i] <- 1
  nsf <- paste(nsf, slc, sep = ".")

  # rename sound files
  Y <- rename_est_waves(X = lbh.est, new.sound.files = nsf)
}
selec.table

Alternative name for lbh_selec_table

Description

selec.table alternative name for lbh_selec_table. selec.table will be deprecated in future versions.

Usage

selec.table

Format

An object of class data.frame with 11 rows and 9 columns.

Details

Selection table of long-billed hermit songs

Source

Marcelo Araya Salas, warbleR

selection_table

Create 'selection_table' and 'extended_selection_table' objects

Description

selection_table converts data frames into an object of classes 'selection_table' or 'extended_selection_table'.

Usage

selection_table(X, max.dur = 10, path = NULL, whole.recs = FALSE, extended = FALSE, confirm.extended = TRUE, mar = 0.1, by.song = NULL, pb = TRUE, parallel = 1, verbose = TRUE, ...)

Arguments

**X**
data frame with the following columns: 1) "sound.files": name of the .wav files, 2) "select": unique selection identifier (within a sound file), 3) "start": start time and 4) "end": end time of selections. Columns for 'top.freq', 'bottom.freq' and 'channel' are optional. Note that, when 'channel' is not provided the first channel (i.e. left channel) would be used by default. Frequency parameters (including top and bottom frequency) should be provided in kHz. Alternatively, a 'selection_table' class object can be input.

**max.dur**
the maximum duration of expected for a selection (i.e. end - start). If surpassed then an error message will be generated. Useful for detecting errors in selection tables.

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

**whole.recs**
Logical. If TRUE the function will create a selection table for all sound files in the working directory (or "path") with 'start = 0' and 'end = duration_wavs()'. Default is if FALSE. Note that this will not create a extended selection table. If provided 'X' is ignored.

**extended**
Logical. If TRUE, the function will create an object of class 'extended_selection_table' which included the wave objects of the selections as an additional attribute ('wave.objects') to the data set. This is and self-contained format that does not require the original sound files for running most acoustic analysis in *warbleR*. This can largely facilitate the storing and sharing of (bio)acoustic data. Default is if FALSE. An extended selection table won’t be created if there is any issue with the selection. See 'details'.

**confirm.extended**
Logical. If TRUE then the size of the 'extended_selection_table' will be estimated and the user will be asked for confirmation (in the console) before proceeding. Ignored if 'extended' is FALSE. This is used to prevent generating objects too big to be dealt with by R. See 'details' for more information about extended selection table size.

**mar**
Numeric vector of length 1 specifying the margins (in seconds) adjacent to the start and end points of the selections when creating extended selection tables. Default is 0.1. Ignored if 'extended' is FALSE.

**by.song**
Character string with the column name containing song labels. If provided a wave object containing for all selection belonging to a single song would be saved in the extended selection table (hence only applicable for extended selection tables). Note that the function assumes that song labels are not repeated within a sound file. If NULL (default), wave objects are created for each selection (e.g. by selection). Ignored if extended = FALSE.

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

**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).

**verbose**
Logical argument to control if summary messages are printed to the console. Default is TRUE.

**...**
Additional arguments to be passed to *check.sels* for customizing checking routine.
Details

This function creates an object of class 'selection_table' or 'extended_selection_table' (if extended = TRUE, see below). First, the function checks:

1) if the selections listed in the data frame correspond to .wav files in the working directory
2) if the sound files can be read and if so,
3) if the start and end time of the selections are found within the duration of the sound files

If no errors are found the a selection table or extended selection table will be generated. Note that the sound files should be in the working directory (or the directory provided in 'path'). This is useful for avoiding errors in downstream functions (e.g. spectro_analysis, cross_correlation, catalog, freq_DTW). Note also that corrupt files can be fixed using fix_wavs ('sox' must be installed to be able to run this function). The 'selection_table' class can be input in subsequent functions.

When extended = TRUE the function will generate an object of class 'extended_selection_table' which will also contains the wave objects for each of the selections in the data frame. This transforms selection tables into self-contained objects as they no longer need the original sound files to run acoustic analysis. This can largely facilitate the storing and sharing of (bio)acoustic data. Extended selection table size will be a function of the number of selections nrow(X), sampling rate, selection duration and margin duration. As a guide, a selection table with 1000 selections similar to the ones in 'lbh_selec_table' (mean duration ~0.15 seconds) at 22.5 kHz sampling rate and the default margin (mar = 0.1) will generate a extended selection table of ~31 MB (~310 MB for a 10000 rows selection table). You can check the size of the output extended selection table with the object.size function. Note that extended selection table created 'by.song' could be considerable larger.

Value

An object of class selection_table which includes the original data frame plus the following additional attributes:

1) A data frame with the output of check_sels run on the input data frame. If a extended selection table is created it will also include the original values in the input data frame for each selections. This are used by downstream warbleR functions to improve efficiency and avoid errors due to missing or mislabeled data, or selection out of the ranges of the original sound files.

2) A list indicating if the selection table has been created by song (see 'by.song argument).

3) If a extended selection table is created a list containing the wave objects for each selection (or song if 'by.song').

Author(s)

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

References

sig2noise

Measure signal-to-noise ratio

Description

sig2noise measures signal-to-noise ratio across multiple files.

Usage

sig2noise(X, mar, parallel = 1, path = NULL, pb = TRUE, type = 1, eq.dur = FALSE, in.dB = TRUE, before = FALSE, lim.dB = TRUE, bp = NULL, wl = 10)
Arguments

**X**

object of class 'selection_table', 'extended_selection_table' or any data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end).

**mar**

numeric vector of length 1. Specifies the margins adjacent to the start and end points of selection over which to measure noise.

**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). It can also be set globally using the 'parallel' option (see warbleR_options).

**path**

Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used. It can also be set globally using the 'wav.path' option (see warbleR_options).

**pb**

Logical argument to control if progress bar is shown. Default is TRUE. It can also be set globally using the 'pb' option (see warbleR_options).

**type**

Numeric. Determine the formula to be used to calculate the signal-to-noise ratio (S = signal, N = background noise):

- 1: ratio of S mean amplitude envelope to N mean amplitude envelope \( \frac{\text{mean}(\text{env}(S))}{\text{mean}(\text{env}(N))} \)
- 2: ratio of S amplitude envelope RMS (root mean square) to N amplitude envelope RMS \( \frac{\text{rms}(\text{env}(S))}{\text{rms}(\text{env}(N))} \)
- 3: ratio of the difference between S amplitude envelope RMS and N amplitude envelope RMS to N amplitude envelope RMS \( \frac{\text{rms}(\text{env}(S)) - \text{rms}(\text{env}(N))}{\text{rms}(\text{env}(N))} \)

**eq.dur**

Logical. Controls whether the noise segment that is measured has the same duration than the signal (if TRUE, default FALSE). If TRUE then 'mar' argument is ignored.

**in.dB**

Logical. Controls whether the signal-to-noise ratio is returned in decibels (20*log10(SNR)). Default is TRUE.

**before**

Logical. If TRUE noise is only measured right before the signal (instead of before and after). Default is FALSE.

**lim.dB**

Logical. If TRUE the lowest signal-to-noise would be limited to -40 dB (if in.dB = TRUE). This would remove NA's that can be produced when noise segments have a higher amplitude than the signal itself. Default is TRUE.

**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 length of the spectrogram for applying bandpass. Default is 10. Ignored if bp = NULL. It can also be set globally using the 'wl' option (see warbleR_options). Note that lower values will increase time resolution, which is more important for signal-to-noise ratio calculations.

Details

Signal-to-noise ratio (SNR) is a measure of the level of a desired signal compared to background noise. The function divides the mean amplitude of the signal by the mean amplitude of the background noise adjacent to the signal. A general margin to apply before and after the acoustic signal...
must be specified. Setting margins for individual signals that have been previously clipped from larger files may take some optimization, as for calls within a larger file that are irregularly separated. When margins overlap with another acoustic signal nearby, the signal-to-noise ratio (SNR) will be inaccurate. Any SNR less than or equal to one suggests background noise is equal to or overpowering the acoustic signal. snr_spectrograms can be used to troubleshoot different noise margins.

Value

Data frame similar to auto_detect output, but also includes a new variable with the signal-to-noise values.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>) and Grace Smith Vidaurre

References


Examples

```r
{
  data(list = c("Phae.long1","lbh_selec_table"))
  writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) #save sound files

  # specifying the correct margin is important
  # use snr_spectrograms to troubleshoot margins for sound files
  sig2noise(lbh_selec_table[grep("Phae.long1", lbh_selec_table$sound.files), ], mar = 0.2, path = tempdir())

  # this smaller margin doesn't overlap neighboring signals
  sig2noise(lbh_selec_table[grep("Phae.long1", lbh_selec_table$sound.files), ], mar = 0.1, path = tempdir())
}
```

### simulate_songs

**Simulate animal vocalizations**

**Description**

simulate_songs simulate animal vocalizations in a wave object under brownian motion frequency drift.
simulate_songs

Usage

```r
simulate_songs(n = 1, durs = 0.2, harms = 3, harm.amps = c(1, 0.5, 0.2), am.amps = 1,
  gaps = 0.1, freqs = 5, samp.rate = 44.1, sig2 = 0.5,
  steps = 10, bgn = 0.5, seed = NULL, diff.fun = "GBM",
  fin = 0.1, fout = 0.2, shape = "linear", selec.table = FALSE,
  file.name = NULL, path = NULL)
```

Arguments

- `n`: Number of song subunits (e.g. elements). Default is 1.
- `durs`: Numeric vector with the duration of subunits in seconds. It should either be a single value (which would be used for all subunits) or a vector of length `n`.
- `harms`: Numeric vector of length 1 specifying the number of harmonics to simulate. 1 indicates that only the fundamental frequency harmonic will be simulated.
- `harm.amps`: Numeric vector with the relative amplitude of each of the harmonics (including the fundamental frequency).
- `am.amps`: Numeric vector with the relative amplitude for each step (see 'step' argument) to simulate amplitude modulation (only applied to the fundamental frequency). Should have the same length as the number of steps. Default is 1 (no amplitude modulation). If supplied ‘fin’ and ‘fout’ are ignored.
- `gaps`: Numeric vector with the duration of gaps (silence between subunits) in seconds. It should either be a single value (which would be used for all subunits) or a vector of length `n + 1`.
- `freqs`: Numeric vector with the initial frequency of the subunits (and ending frequency if `diff.fun == "BB"`) in kHz. It should either be a single value (which would be used for all subunits) or a vector of length `n`.
- `samp.rate`: Numeric vector of length 1. Sets the sampling frequency of the wave object (in kHz). Default is 44.1.
- `sig2`: Numeric vector defining the sigma value of the brownian motion model. It should either be a single value (which would be used for all subunits) or a vector of length `n + 1`. Higher values will produce faster frequency modulations. Ignored if `diff.fun == "BB"`. Default is 0.1. Check the ‘BB’ function in the Sim.DiffProc package for more details.
- `steps`: Numeric vector of length 1. Controls the mean number of segments in which each song subunit is split during the brownian motion process. If not all subunits have the same duration, longer units will be split in more steps (although the average duration subunit will have the predefined number of steps). Default is 10.
- `bgn`: Numeric vector of length 1 indicating the background noise level. 0 means no additional noise will 1 means noise at the same amplitude than the song subunits. Default is 0.5.
- `seed`: Numeric vector of length 1. This allows users to get the same results in different runs (using `set.seed` internally). Default is NULL.
diff.fun  Character vector of length 1 controlling the function used to simulate the brownian motion process of frequency drift across time. Only "BB", "GBM" and "pure.tone" are accepted at this time. Check the 'BB' function in the Sim.DiffProc package for more details.

fin  Numeric vector of length 1 setting the proportion of the sub-unit to fade-in amplitude (value between 0 and 1). Default is 0.1. Note that 'fin' + 'fout' cannot be higher than 1.

fout  Numeric vector of length 1 setting the proportion of the sub-unit to fade-out amplitude (value between 0 and 1). Default is 0.2. Note that 'fin' + 'fout' cannot be higher than 1.

shape  Character string of length 1 controlling the shape of in and out amplitude fading of the song sub-units ('fin' and 'fout'). "linear" (default), "exp" (exponential), and "cos" (cosine) are currently allowed.

selec.table  Logical. If TRUE a data frame containing the start/end time, and bottom/top frequency of the sub-units is also returned and the wave object is returned. In that case the function returns a list with the selection table and the wave object. In addition, a '.wav' file is saved in the working directory. Default is FALSE.

file.name  Character string for naming the '.wav' file. Ignored if 'selec.table' is FALSE. If not provided the date-time stamp will be used.

path  Character string containing the directory path where the sound files are located. Ignored if 'selec.table' is FALSE. If NULL (default) then the current working directory is used.

Details

This functions uses a geometric (diff.fun == "GBM") or Brownian bridge (diff.fun == "BB") motion stochastic process to simulate modulation in animal vocalizations (i.e. frequency traces across time). The function can also simulate pure tones (diff.fun == "pure.tone", 'sig2' is ignored). Several song subunits (e.g. elements) can be simulated as well as the corresponding harmonics.

Value

A wave object containing the simulated songs. If 'selec.table' is TRUE the function saves the wave object as a '.wav' sound file in the working directory (or 'path') and returns a list including 1) a selection table with the start/end time, and bottom/top frequency of the sub-units and 2) the wave object.

Author(s)

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

References


See Also

query_xc for for downloading bird vocalizations from an online repository.
Examples

```r
## Not run:
# simulate a song with 3 elements and no harmonics
sm_sng <- simulate_songs(n = 3, harms = 1)

# plot spectro
seewave::spectro(sm_sng)

# simulate a song with 5 elements and 2 extra harmonics
sm_sng2 <- simulate_songs(n = 5, harms = 3)

# plot spectrogram
seewave::spectro(sm_sng2)

# six pure tones with frequency ranging form 4 to 6 and returning selection table
sm_sng <- simulate_songs(n = 6, harms = 1, seed = 1, diff.fun = "pure.tone",
                         freqs = seq(4, 6, length.out = 6), selec.table = TRUE,
                         path = tempdir())

# plot spectro
seewave::spectro(sm_sng$wave, flim = c(2, 8))

# selection table
sm_sng$selec.table

## End(Not run)
```

---

**sim_coor_sing**  
Simulated coordinated singing events.

Description

**sim_coor_sing** contains selections of simulated interactive singing events. The simulated events use the mean and standard deviation of real lekking *Phaethornis longirostris* (Long-billed Hermit hummingbird) songs and intervals between songs (e.g. gaps). Three events are simulated: overlapping signals (ovlp), alternating signals (altern) and non-synchronized signals (uncoor).

Usage

```r
data(sim_coor_sing)
```

Format

**sim_coor_sing**  
Simulated coordinated singing events that overlap and do not overlap most of the time, for use with test_coordination
**Description**

`snn_spectrograms` creates spectrograms to visualize margins over which background noise will be measured by `sig2noise`.

**Usage**

```r
snn_spectrograms(X, wl = 512, flim = NULL, wn = "hanning", ovlp = 70,
inner.mar = c(5, 4, 4, 2), outer.mar = c(0, 0, 0, 0), picsize = 1,
res = 100, cexlab = 1, title = TRUE, before = FALSE, eq.dur = FALSE,
propwidth= FALSE, xl = 1, osci = FALSE, gr = FALSE, sc = FALSE, mar = 0.2,
lnrmar = 0.1, it = "jpeg", parallel = 1, path = NULL, pb = TRUE)
```

**Arguments**

- `X`: 'selection_table', 'extended_selection_table' or any data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end).
- `wl`: A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.
- `flim`: A numeric vector of length 2 for the frequency limit in kHz of the spectrogram, as in `spectro`. Default is NULL.
- `wn`: Character vector of length 1 specifying window name. Default is "hanning". See function `ftwindow` for more options.
- `ovlp`: Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in `spectro`. Default is 70.
- `inner.mar`: Numeric vector with 4 elements, default is c(5,4,4,2). Specifies number of lines in inner plot margins where axis labels fall, with form c(bottom, left, top, right). See `par`.
- `outer.mar`: Numeric vector with 4 elements, default is c(0,0,0,0). Specifies number of lines in outer plot margins beyond axis labels, with form c(bottom, left, top, right). See `par`.
- `picsize`: Numeric argument of length 1, controls relative size of spectrogram. Default is 1.
- `res`: Numeric argument of length 1 that controls image resolution. Default is 100 (faster) although 300 - 400 is recommended for publication/presentation quality.
- `cexlab`: Numeric vector of length 1 specifying relative size of axis labels. See `spectro`.
- `title`: Logical argument to add a title to individual spectrograms. Default is TRUE.
- `before`: Logical. If TRUE noise is only measured right before the signal (instead of before and after). Default is FALSE.
eq.dur Logical. Controls whether the noise segment that is measured has the same duration than the signal (if TRUE, default FALSE). If TRUE then ‘snrmar’ argument is ignored.

propwidth Logical argument to scale the width of spectrogram proportionally to duration of the selected call. Default is FALSE.

xl Numeric vector of length 1, a constant by which to scale spectrogram width if propwidth = TRUE. Default is 1.

osci Logical argument to add an oscillogram underneath spectrogram, as in spectro. Default is FALSE.

gr Logical argument to add grid to spectrogram. Default is FALSE.

sc Logical argument to add amplitude scale to spectrogram, default is FALSE.

mar Numeric vector of length 1. Specifies the margins adjacent to the start and end points of the selections to define spectrogram limits. Default is 0.2. If snrmar is larger than mar, then mar is set to be equal to snrmar.

snrmar Numeric vector of length 1. Specifies the margins adjacent to the start and end points of the selections where noise will be measured. Default is 0.1.

it A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".

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).

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

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

Details

This function can be used to test different margins to facilitate accurate SNR measurements when using sig2noise down the line. Setting margins for individual calls that have been previously clipped from larger files may take some optimization, as for calls within a larger file that are irregularly separated. Setting inner.mar to c(4,4,5,2,1) and outer.mar to c(4,2,2,1) works well when picsize = 2 or 3. Title font size, inner.mar and outer.mar (from mar and oma in par) don’t work well when osci or sc = TRUE, this may take some optimization by the user.

Value

Spectrograms per selection marked with margins where background noise will be measured.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>) and Grace Smith Vidaurre

References

See Also

`track_freq_contour` for creating spectrograms to visualize frequency measurements by `spectro_analysis`, `spectrograms` for creating spectrograms

Other spectrogram creators: `color_spectro()`, `freq_DTW()`, `multi_DTW()`, `phylo_spectro()`, `spectrograms()`, `track_freq_contour()`

Examples

```r
## Not run:
data(list = c("Phae.long1", "Phae.long2", "lbh_selec_table"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) # save sound files
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))

# make Phae.long1 and Phae.long2 spectrograms
# snrmar needs to be smaller before moving on to sig2noise()
snr_spectrograms(lbh_selec_table, flim = c(0, 14), inner.mar = c(4, 4.5, 2, 1),
outer.mar = c(4, 2, 2, 1), picsize = 2, res = 300, cexlab = 2, mar = 0.2,
snrmar = 0.1, it = "jpeg", w1 = 300, path = tempdir())

# make only Phae.long1 spectrograms
# snrmar now doesn't overlap neighboring signals

snr_spectrograms(lbh_selec_table[grepl(c("Phae.long1"), lbh_selec_table$sound.files), ],
flim = c(3, 14), inner.mar = c(4, 4.5, 2, 1), outer.mar = c(4, 2, 2, 1),
picsize = 2, res = 300, cexlab = 2, mar = 0.2, snrmar = 0.01, w1 = 300,
path = tempdir())

# check this folder!
tempdir()

## End(Not run)
```

### song_analysis

`song_analysis` **Calculates acoustic parameters at the song level**

**Description**

`song_analysis` calculates descriptive statistics of songs or other higher levels of organization in the signals.

**Usage**

```r
song_analysis(X = NULL, song_colm = "song", mean_colm = NULL, min_colm = NULL,
max_colm = NULL, elm_colm = NULL, elm_fun = NULL, sd = FALSE, parallel = 1,
na.rm = FALSE, weight = NULL)
```
**Arguments**

- **X**
  'selection_table', 'extended_selection_table' (created by.song) or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "selec": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections.

- **song_colm**
  Character string with the column name containing song labels. It can be used to label any hierarchical level at which parameters need to be calculated (e.g. syllables, phrases). Note that the function assumes that song labels are not repeated within a sound file.

- **mean_colm**
  Numeric vector with the index of the columns that will be averaged. If NULL the mean of all numeric columns in 'X' is returned.

- **min_colm**
  Character vector with the name(s) of the columns for which the minimum value is needed. Default is NULL.

- **max_colm**
  Character vector with the name(s) of the columns for which the maximum value is needed. Default is NULL.

- **elm_colm**
  Character vector with the name(s) of the columns identifying the element labels (i.e. element types). If supplied 'unq.elms' and 'mean.elm.count' are returned. Default is NULL.

- **elm_fun**
  Function to be applied to the sequence of elements composing a song. Default is NULL. Ignored if 'elm_colm' is not supplied. The name of the column containing the function's output is "elm_fun".

- **sd**
  Logical value indicating whether standard deviation is also returned for variables in which averages are reported. Default is FALSE.

- **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.

- **na.rm**
  Logical value indicating whether 'NA' values should be ignored for calculations.

- **weight**
  Character vector defining 1 or more numeric vectors to weight average measurements (i.e. song parameters). Names of numeric columns in 'X' can also be used. See **weighted.mean** for more details. Default is NULL (unweighted average).

**Details**

The function calculates average or extreme values of acoustic parameters of elements in a song or other level of organization in the signals.

**Value**

A data frame similar to the input 'X' data frame, but in this case each row corresponds to a single song. The data frame contains the mean or extreme values for numeric columns for each song. Columns that will be averaged can be defined with 'mean_colm' (otherwise all numeric columns are used). Columns can be weighted by other columns in the data set (e.g. duration, frequency range). In addition, the function returns the following song level parameters:

- **elm.duration**: mean length of elements (in s)
• song.duration: length of song (in s)
• num.elms: number of elements (or song units)
• start: start time of song (in s)
• end: end time of song (in s)
• bottom.freq: lowest 'bottom.freq' from all song elements (in kHz)
• top.freq: highest 'top.freq' from all song elements (in kHz)
• freq.range: difference between song’s 'top.freq' and 'bottom.freq' (in kHz)
• song.rate: number of elements per second (NA if only 1 element). Calculated as the number of elements in the 'song' divided by the duration of the song. In this case song duration is calculated as the time between the start of the first element and the start of the last element, which provides a rate that is less affected by the duration of individual elements. Note that this calculation is different than that from 'song.duration’ above.
• gapduration: average length of gaps (i.e. silences) in between elements (in s, NA if only 1 element)
• elm.types: number of element types (i.e. number of unique types, only if 'elm_colm' is supplied)
• mean.elm.count: mean number of times element types are found (only if 'elm_colm' is supplied)

This function assumes that song labels are not repeated within a sound file.

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

References

See Also
spectro_analysis

Examples
{
  # get warbleR sound file examples
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))
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"))

  # add a 'song' column
lbh_selec_table$song <- c("song1", "song1", "song1", "song2",
  "song2", "song3", "song3", "song3", "song4", "song4", "song4")

  # measure acoustic parameters
sp <- spectro_analysis(lbh_selec_table[1:8, ], bp = c(1, 11), 300, fast = TRUE, path = tempdir())

# add song data
sp <- merge(sp, lbh_selec_table[1:8, ], by = c("sound.files", "selec"))

# calculate song-level parameters for all numeric parameters
song_analysis(X = sp, song_colm = "song", parallel = 1, pb = TRUE)

# calculate song-level parameters selecting parameters with mean_colm
song_analysis(X = sp, song_colm = "song", mean_colm = c("dfrange", "duration"), parallel = 1, pb = TRUE)

# calculate song-level parameters for selecting parameters with mean_colm, max_colm
# and min_colm and weighted by duration
song_analysis(X = sp, weight = "duration", song_colm = "song",
               mean_colm = c("dfrange", "duration"), min_colm = "mindom", max_colm = "maxdom",
               parallel = 1, pb = TRUE)

# with two weights
song_analysis(X = sp, weight = c("duration", "dfrange"), song_colm = "song",
               mean_colm = c("kurt", "sp.ent"), parallel = 1, pb = TRUE)

# with two weights no progress bar
song_analysis(X = sp, weight = c("duration", "dfrange"), song_colm = "song",
               mean_colm = c("kurt", "sp.ent"), parallel = 1, pb = FALSE)

---

sort_colms

**Sort columns in a more intuitive order**

**Description**

sort_colms sorts selection table columns in a more intuitive order.

**Usage**

sort_colms(X)

**Arguments**

- **X**: Data frame containing columns for sound file (sound.files), selection (selec), start and end time of signals ('start' and 'end') and low and high frequency ('bottom.freq' and 'top.freq', optional). See the example data 'lbh_selec_table'.

**Details**

The function returns the data from the input data frame with the most relevant information for acoustic analysis located in the first columns. The priority order for column names is: "sound.files", "channel", "selec", "start", "end", "top.freq", and "bottom.freq".
Value

The same data as in the input data frame but with the most relevant information for acoustic analysis located in the first columns.

Author(s)

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

Examples

library(warbleR)
data("selec.table")

# mess column order
selec.table <- selec.table[, sample(1:ncol(selec.table))]

# check names
names(selec.table)

selec.table <- sort_cols(X = selec.table)

# check names again
names(selec.table)

sound_pressure_level  Measure sound pressure level

Description

sound_pressure_level measures sound pressure level in signals reference in a selection table.

Usage

sound_pressure_level(X, reference = 20, parallel = 1, path = NULL, pb = TRUE,
peak.amplitude = FALSE, wl = 100)

Arguments

X object of class 'selection_table', 'extended_selection_table' or any data frame with columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end).

reference Numeric vector of length 1 indicating the pressure (in µPa) to be used as reference. Alternatively, a character vector with the name of a numeric column containing reference values for each row can be supplied. Default is 20 (µPa).

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). It can also be set globally using the 'parallel' option (see warbleR_options).
**sound_pressure_level**

- **path**
  Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used. It can also be set globally using the 'wav.path' option (see warbleR_options).

- **pb**
  Logical argument to control if progress bar is shown. Default is TRUE. It can also be set globally using the 'pb' option (see warbleR_options).

- **peak.amplitude**
  Logical argument controlling if the sound pressure level across the entire signal is return or only that of the highest amplitude (i.e. peak amplitude) of the signal. Default is FALSE.

- **wl**
  A numeric vector of length 1 specifying the spectrogram window length. Default is 512.

**Details**

Sound pressure level (SPL) is a logarithmic measure of the effective pressure of a sound relative to a reference, so it’s a measure of sound intensity. Note that calibrated measures can be obtained only when the SPL of the environment where recordings were made is used as reference. sig2noise.

**Value**

The object supplied in 'X' with a new variable with the sound pressure level values ('SPL' or 'peak.amplitude' column, see argument 'peak.amplitude') in decibels.

**Author(s)**

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>) and Grace Smith Vidaurre

**References**


**Examples**

```r
{
  data(list = c("Phae.long1", "lbh_selc_table"))
  writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) #save sound files

  spl <- sound_pressure_level(X = lbh_selc_table[grep("Phae.long1", lbh_selc_table$sound.files)], , parallel = 1, pb = TRUE, path = tempdir())
}
```
spectrograms

Spectrograms of selected signals

Description

spectrograms creates spectrograms of signals from selection tables.

Usage

spectrograms(X, wl = 512, flim = "frange", wn = "hanning", pal = reverse.gray.colors.2, ovlp = 70, inner.mar = c(5, 4, 4, 2), outer.mar = c(0, 0, 0, 0), picsize = 1, res = 100, cexlab = 1, propwidth = FALSE, xl = 1, osci = FALSE, gr = FALSE, sc = FALSE, line = TRUE, col = "#07889B", fill = adjustcolor("#07889B", alpha.f = 0.15), lty = 3, mar = 0.05, it = "jpeg", parallel = 1, path = NULL, pb = TRUE, fast.spec = FALSE, by.song = NULL, sel.labels = "selec", title.labels = NULL, dest.path = NULL, ...)

Arguments

X 'selection_table', 'extended_selection_table' or data frame containing columns for sound file name (sound.files), selection number (selec), and start and end time of signals (start and end). 'top.freq' and 'bottom.freq' columns are optional. If using an 'extended_selection_table' the sound files are not required (see selection_table).

wl A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.

flim A numeric vector of length 2 for the frequency limit (in kHz) of the spectrogram, as in spectro. The function also accepts 'frange' (default) which produces spectrograms with a frequency limit around the range of each signal (adding a 1 kHz margin).

wn Character vector of length 1 specifying window name. Default is "hanning". See function ftwindow for more options.

pal A color palette function to be used to assign colors in the plot, as in spectro. Default is reverse.gray.colors.2.

ovlp Numeric vector of length 1 specifying the percent overlap between two consecutive windows, as in spectro. Default is 70.

inner.mar Numeric vector with 4 elements, default is c(5,4,4,2). Specifies number of lines in inner plot margins where axis labels fall, with form c(bottom, left, top, right). See par.

outer.mar Numeric vector with 4 elements, default is c(0,0,0,0). Specifies number of lines in outer plot margins beyond axis labels, with form c(bottom, left, top, right). See par.

picsize Numeric argument of length 1. Controls relative size of spectrogram. Default is 1. Ignored when propwidth is TRUE.

res Numeric argument of length 1. Controls image resolution. Default is 100 (faster) although 300 - 400 is recommended for publication/presentation quality.
**spectrograms**

- **cexlab**: Numeric vector of length 1 specifying the relative size of axis labels. See `spectro`.
- **propwidth**: Logical argument to scale the width of spectrogram proportionally to duration of the selection. Default is FALSE.
- **xl**: Numeric vector of length 1. A constant by which to scale spectrogram width if `propwidth = TRUE`. Default is 1.
- **osci**: Logical argument to add an oscillogram underneath spectrogram, as in `spectro`. Default is FALSE.
- **gr**: Logical argument to add grid to spectrogram. Default is FALSE.
- **sc**: Logical argument to add amplitude scale to spectrogram, default is FALSE.
- **line**: Logical argument to add lines at start and end times of selection (or box if bottom.freq and top.freq columns are provided). Default is TRUE.
- **col**: Color of 'line'. Default is "#07889B".
- **fill**: Fill color of box around selections. Default is `adjustcolor("#07889B", alpha.f = 0.15)`.
- **lty**: Type of 'line' as in `par`. Default is 1.
- **mar**: Numeric vector of length 1. Specifies the margins adjacent to the start and end points of selections, delineating spectrogram limits. Default is 0.05.
- **it**: A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".
- **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).
- **path**: Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
- **pb**: Logical argument to control progress bar. Default is TRUE.
- **fast.spec**: Logical. If TRUE then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors `gray.1`, `gray.2`, `gray.3`, `topo.1` and `rainbow.1` (which should be imported from the package monitoR) seem to work better with 'fast' spectrograms. Palette colors `gray.1`, `gray.2`, `gray.3` offer decreasing darkness levels.
- **by.song**: Character string with the column name containing song labels. If provide a single spectrogram containing all elements for each song will be produce. Note that the function assumes that each song has a unique label within a sound file. If NULL (default), spectrograms are produced for single selections.
- **sel.labels**: Character string with the name of the column(s) for selection labeling. Default is 'selec'. Set to NULL to remove labels.
- **title.labels**: Character string with the name(s) of the column(s) to use as title. Default is NULL (no title). Only sound file and song included if 'by.song' is provided.
- **dest.path**: Character string containing the directory path where the image files will be saved. If NULL (default) then the folder containing the sound files will be used instead.
... Additional arguments to be passed to the internal spectrogram creating function for customizing graphical output. The function is a modified version of \texttt{spectro}, so it takes the same arguments.

Details

This function provides access to batch process of (a modified version of) the \texttt{spectro} function from the \texttt{seewave} package. The function creates spectrograms for visualization of vocalizations. Setting inner.mar to c(4,4,5,2,1) and outer.mar to c(4,2,2,1) works well when picsize = 2 or 3. Title font size, inner.mar and outer.mar (from mar and oma) don’t work well when osci or sc = TRUE, this may take some optimization by the user. Setting ‘fast’ argument to TRUE significantly increases speed, although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels.

Value

Image files containing spectrograms of the signals listed in the input data frame.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>) and Grace Smith Vidaurre

References


See Also

\texttt{track_freq_contour} for creating spectrograms to visualize frequency measurements by \texttt{spectro_analysis}, \texttt{snr_spectrograms} for creating spectrograms to optimize noise margins used in \texttt{sig2noise}

Other spectrogram creators: \texttt{color_spectro()}, \texttt{freq_DTW()}, \texttt{multi_DTW()}, \texttt{phylo_spectro()}, \texttt{snr_spectrograms()}, \texttt{track_freq_contour}()

Examples

{
  # load and save data
  data(list = c("Phae.long1", "Phae.long2","lbh_selec_table"))
  writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) #save sound files
  writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))

  # make spectrograms
  spectrograms(X = lbh_selec_table, flim = c(0, 11), res = 300, mar = 0.05,
  wl = 300, path = tempdir())

  # check this folder
  tempdir()
}
**spectro_analysis**

Measure acoustic parameters in batches of sound files

**Description**

spectro_analysis measures acoustic parameters on acoustic signals for which the start and end times are provided.

**Usage**

```r
spectro_analysis(X, bp = "frange", wl = 512, wl.freq = NULL, threshold = 15,
parallel = 1, fast = TRUE, path = NULL, pb = TRUE, ovlp = 50,
wn = "hanning", fsmooth = 0.1, harmonicity = FALSE, nharmonics = 3, ...)
```

**Arguments**

- **X** `'selection_table', 'extended_selection_table' or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. The output auto_detect can be used as the input data frame.
- **bp** A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or "frange" (default) to indicate that values in bottom.freq and top.freq columns will be used as bandpass limits. Lower limit of bandpass filter is not applied to fundamental frequencies.
- **wl** A numeric vector of length 1 specifying the spectrogram window length. Default is 512. See `wl.freq` for setting windows length independently in the frequency domain.
- **wl.freq** A numeric vector of length 1 specifying the window length of the spectrogram for measurements on the frequency spectrum. Default is 512. Higher values would provide more accurate measurements. Note that this allows to increase measurement precision independently in the time and frequency domain. If NULL (default) then the 'wl' value is used.
- **threshold** amplitude threshold (%) for fundamental frequency and dominant frequency detection. Default is 15.
- **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).
- **fast** Logical. If TRUE (default) then the peakf acoustic parameter (see below) is not computed, which substantially increases performance (~9 times faster). This argument will be removed in future version.
- **path** Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
- **pb** Logical argument to control progress bar and messages. Default is TRUE.
- **ovlp** Numeric vector of length 1 specifying % of overlap between two consecutive windows, used for fundamental frequency (using `fund` or `FF`) and dominant frequency (using `dfreq`). Default is 50.
The function measures 29 acoustic parameters (if `fast = TRUE`) on each selection in the data frame. Most parameters are produced internally by `specprop`, `fpeaks`, `fund`, and `dfreq` from the package seewave and `analyze` from the package soundgen. NAs are produced for fundamental and dominant frequency measures when there are no amplitude values above the threshold. Additional parameters can be provided to the internal function `analyze`, which measures parameters related to harmonicity.

**Value**

Data frame with 'sound.files' and 'selec' as in the input data frame, plus the following acoustic parameters:

- **duration**: length of signal (in s)
- **meanfreq**: mean frequency (in kHz). Mean of frequency spectrum (i.e. weighted average of frequency by amplitude within supplied band pass).
- **sd**: standard deviation of frequency (in kHz).
- **freq.median**: median frequency. The frequency at which the signal is divided in two frequency intervals of equal energy (in kHz)
- **freq.Q25**: first quartile frequency. The frequency at which the signal is divided in two frequency intervals of 25% and 75% energy respectively (in kHz)
- **freq.Q75**: third quartile frequency. The frequency at which the signal is divided in two frequency intervals of 75% and 25% energy respectively (in kHz)
- **freq.IQR**: interquartile frequency range. Frequency range between 'freq.Q25' and 'freq.Q75' (in kHz)
- **time.median**: median time. The time at which the signal is divided in two time intervals of equal energy (in s)
- **time.Q25**: first quartile time. The time at which the signal is divided in two time intervals of 25% and 75% energy respectively (in s). See `acoustat`
- **time.Q75**: third quartile time. The time at which the signal is divided in two time intervals of 75% and 25% energy respectively (in s). See `acoustat`
• time.IQR: interquartile time range. Time range between 'time.Q25' and 'time.Q75' (in s).
  See acoustat
• skew: skewness. Asymmetry of the spectrum (see note in specprop description)
• kurt: kurtosis. Peakedness of the spectrum (see note in specprop description)
• sp.ent: spectral entropy. Energy distribution of the frequency spectrum. Pure tone ~ 0; noisy ~ 1. See sh
• time.ent: time entropy. Energy distribution on the time envelope. Pure tone ~ 0; noisy ~ 1. See th
• entropy: spectrographic entropy. Product of time and spectral entropy sp.ent * time.ent. See H
• sfm: spectral flatness. Similar to sp.ent (Pure tone ~ 0; noisy ~ 1). See sfm
• meandom: average of dominant frequency measured across the acoustic signal
• mindom: minimum of dominant frequency measured across the acoustic signal
• maxdom: maximum of dominant frequency measured across the acoustic signal
• dfrange: range of dominant frequency measured across the acoustic signal
• modindx: modulation index. Calculated as the cumulative absolute difference between adjacent measurements of dominant frequencies divided by the dominant frequency range. 1 means the signal is not modulated.
• startdom: dominant frequency measurement at the start of the signal
• enddom: dominant frequency measurement at the end of the signal
• dfslope: slope of the change in dominant frequency through time ((enddom-startdom)/duration). Units are kHz/s.
• peakf: peak frequency. Frequency with the highest energy. This parameter can take a considerable amount of time to measure. It's only generated if fast = FALSE. It provides a more accurate measure of peak frequency than 'meanpeakf' but can be more easily affected by background noise.
• meanpeakf: mean peak frequency. Frequency with highest energy from the mean frequency spectrum (see meanspec). Typically more consistent than peakf.
• meanfun: average of fundamental frequency measured across the acoustic signal. Only measured if harmonicity = TRUE.
• minfun: minimum fundamental frequency measured across the acoustic signal. Only measured if harmonicity = TRUE.
• maxfun: maximum fundamental frequency measured across the acoustic signal. Only measured if harmonicity = TRUE.
• hn_freq: mean frequency of the 'n' upper harmonics (kHz) (see analyze). Number of harmonics is defined with the argument 'nharmonics'. Only measured if harmonicity = TRUE.
• hn_width: mean bandwidth of the 'n' upper harmonics (kHz) (see analyze). Number of harmonics is defined with the argument 'nharmonics'. Only measured if harmonicity = TRUE.
• harmonics: the amount of energy in upper harmonics, namely the ratio of total spectral power above 1.25 x F0 to the total spectral power below 1.25 x F0 (dB) (see analyze). Number of harmonics is defined with the argument 'nharmonics'. Only measured if harmonicity = TRUE.
• HNR: harmonics-to-noise ratio (dB). A measure of the harmonic content generated by getPitchAutocor. Only measured if harmonicity = TRUE.
split_sound_files

Splits sound files

Description

split_sound_files splits sound files in shorter segments

Usage

```r
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.

Examples

```r
{
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "lbh_selec_table"))
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"))

# measure acoustic parameters
sp_param <- spectro_analysis(X = lbh_selec_table[1:8,], pb = FALSE, path = tempdir())

# measuring peakf
sp_param <- spectro_analysis(X = lbh_selec_table[1:8,], pb = FALSE, fast = FALSE, path = tempdir())

# measuring harmonic-related parameters using progress bar
sp_param <- spectro_analysis(X = lbh_selec_table[1:8,], harmonicity = TRUE, path = tempdir(), ovlp = 0)
}
```
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_detect). The function keeps the
original number of channels in the output clips only for 1- and 2-channel files.

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., & Smith-Vidaurre, G. (2017). warbleR: An R package to streamline analysis of

See Also
cut_sels
Other data manipulation: move_imgs(), open_wd()

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()
}

---

**tailor_sels**  
*Interactive view of spectrograms to tailor selections*

**Description**

`tailor_sels` produces an interactive spectrographic view in which the start/end times and frequency range of acoustic signals listed in a data frame can be adjusted.

**Usage**

```r
tailor_sels(X = NULL, wl = 512, flim = c(0, 22), wn = "hanning", mar = 0.5,
osci = TRUE, pal = reverse.gray.colors.2, ovlp = 70, auto.next = FALSE, pause = 1,
comments = TRUE, path = NULL, frange = TRUE, fast.spec = FALSE, ext.window = TRUE,
width = 15, height = 5, index = NULL, collevels = NULL,
title = c("sound.files", "selec"), ts.df = NULL, col = "#E37222",
alpha = 0.7, auto.contour = FALSE, ...)
```

**Arguments**

- **X**: 'selection_table', 'extended_selection_table' object or data frame with the following columns: 1) "sound.files": name of the .wav files, 2) "selec": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. Notice that, if an output file ("seltailor_output.csv") is found in the working directory it will be given priority over an input data frame.
- **wl**: A numeric vector of length 1 specifying the spectrogram window length. Default is 512.
- **flim**: A numeric vector of length 2 specifying the frequency limit (in kHz) of the spectrogram, as in the function `spectro`. Default is c(0, 22).
- **wn**: A character vector of length 1 specifying the window function (by default "hanning"). See function `ftwindow` for more options.
- **mar**: Numeric vector of length 1. Specifies the margins adjacent to the start and end points of the selections to define spectrogram limits. Default is 0.5.
- **osci**: Logical argument. If TRUE adds a oscillogram whenever the spectrograms are produced with higher resolution (see seltime). Default is TRUE. The external program must be closed before resuming analysis. Default is NULL.
- **pal**: A color palette function to be used to assign colors in the plot, as in `spectro`. Default is reverse-gray.colors.2. See Details.
ovlp  Numeric vector of length 1 specifying the percent overlap between two consecutive windows, as in spectro. Default is 70.

auto.next  Logical argument to control whether the functions moves automatically to the next selection. The time interval before moving to the next selection is controlled by the 'pause' argument. Ignored if ts.df = TRUE.

pause  Numeric vector of length 1. Controls the duration of the waiting period before moving to the next selection (in seconds). Default is 1.

comments  Logical argument specifying if 'sel.comment' (when in data frame) should be included in the title of the spectrograms. Default is TRUE.

path  Character string containing the directory path where the sound files are located.

frange  Logical argument specifying whether limits on frequency range should be recorded. If TRUE (default) time and frequency limits are recorded.

fast.spec  Logical. If TRUE then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors gray.1, gray.2, gray.3, topo.1 and rainbow.1 (which should be imported from the package monitoR) seem to work better with 'fast' spectrograms. Palette colors gray.1, gray.2, gray.3 offer decreasing darkness levels.

ext.window  Logical. If TRUE then an external graphic window is used. Default dimensions can be set using the 'width' and 'height' arguments. Default is TRUE.

width  Numeric of length 1 controlling the width of the external graphic window. Ignored if ext.window = FALSE. Default is 15.

height  Numeric of length 1 controlling the height of the external graphic window. Ignored if ext.window = FALSE. Default is 5.

index  Numeric vector indicating which selections (rows) of 'X' should be tailored. Default is NULL. Ignored when the process is resumed. This can be useful when combined with filtersels output (see 'index' argument in filtersels).

collevels  Numeric. Set of levels used to partition the amplitude range (see spectro).

title  Character vector with the names of the columns to be included in the title for each selection.

ts.df  Optional. Data frame with frequency contour time series of signals to be tailored. If provided then 'autonext' is set to FALSE. Default is NULL. The data frame must include the 'sound.files' and 'selec' columns for the same selections included in 'X'.

col  Character vector defining the color of the points when 'ts.df' is provided. Default is "#E37222" (orange).

alpha  Numeric of length one to adjust transparency of points when adjusting frequency contours.

auto.contour  Logical. If TRUE contours are displayed automatically (without having to click on 'contour'). Note that adjusting the selection box (frequency/time limits) won’t be available. Default is FALSE. Ignored if 'ts.df' is not provided.

Additional arguments to be passed to the internal spectrogram creating function for customizing graphical output. The function is a modified version of spectro, so it takes the same arguments.
Details

This function produces an interactive spectrographic view in which users can select new time/frequency coordinates the selections. 4 "buttons" are provided at the upper right side of the spectrogram that allow to stop the analysis (stop symbol, a solid rectangle), go to the next sound file ("»"), return to the previous selection ("«") or delete the current selection ("X"). An additional "button" to tailored frequency contour is shown when 'ts.df' is provided. The button contains a symbol with a 4 point contour. When a unit has been selected, the function plots dotted lines in the start and end of the selection in the spectrogram (or a box if frange = TRUE). Only the last selection is kept for each selection that is adjusted. The function produces a .csv file (seltailor_output.csv) with the same information than the input data frame, except for the new time coordinates, plus a new column (X$tailored) indicating if the selection has been tailored. The file is saved in the working directory and is updated every time the user moves into the next sound file ("»") or stop the process (stop "button"). It also return the same data frame as and object in the R environment. If no selection is made (by clicking on "»") the original time/frequency coordinates are kept. When resuming the process (after 'stop' and re-running the function in the same working directory), the function will continue working on the selections that have not been analyzed. When deleting a file (X button) an orange "X" when returning to that selection. If X is used again the selection is recovered. The function also displays a progress bar right on top of the spectrogram. The zoom can be adjusted by setting the mar argument. To fix contours a data.frame containing the 'sound.files' and 'selec' columns as in 'X' as well as the frequency values at each contour step must be provided. The function plots points corresponding to the time/frequency coordinates of each element of the contour. Clicking on the spectrogram will substitute the frequency value of the points. The contour point closest in time to the "click" will be replaced by the frequency value of the "click".

Value

data frame similar to X with the and a .csv file saved in the working directory with start and end time of selections.

Author(s)

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

References


Examples

```r
# Not run:
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))
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"))
tailor_sels(X = lbh_selec_table, flim = c(1,12), wl = 300, auto.next = TRUE, path = tempdir())
# Read output .csv file
```
test_coordination

Randomization test for singing coordination

Description

Monte Carlo randomization test to assess the statistical significance of overlapping or alternating singing (or any other simultaneously occurring behavior).

Usage

test_coordination(X, iterations = 1000, ovlp.method = "count", randomization = "keep.gaps", less.than.chance = TRUE, parallel = 1, pb = TRUE, rm.incomp = FALSE, cutoff = 2, rm.solo = FALSE)

Arguments

X Data frame containing columns for singing event (sing.event), individual (indiv), and start and end time of signal (start and end).

iterations number of iterations for shuffling and calculation of the expected number of overlaps. Default is 1000.

ovlp.method Character string defining the method to measure the amount of overlap. Two methods are accepted: ‘count’ and ‘duration’. As the name suggests, the ‘count’ method will count the number of overlapping signals while ‘duration’ will measure the total duration (in s) in which signals overlap. Default is ‘count’.

randomization Character string defining the procedure for signal randomization. Three methods are available:

• keep.gaps the position of both signals and gaps (i.e. intervals between signals) are randomized. Default.
• sample.gaps gaps are simulated using a lognormal distribution with mean and standard deviation derived from the observed gaps. Signal position is randomized.
• keep.song.order only the position of gaps is randomized.

More details in Masco et al. (2015).

less.than.chance Logical. If TRUE the test evaluates whether overlaps occur less often than expected by chance. If FALSE the opposite pattern is evaluated (whether overlaps occur more often than expected by chance). Default is TRUE.

seltailor.df <- read.csv(file.path(tempdir(), "seltailor_output.csv"))
seltailor.df

# check this directory for .csv file after stopping function
tempdir()

## End(Not run)
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.

rm.incomp

Logical. If TRUE removes the events that don’t have 2 interacting individuals. Default is FALSE.

cutoff

Numeric. Determines the minimum number of signals per individual in a singing event. Events not meeting this criterium are removed. Default is 2. Note that randomization tests are not reliable with very small sample sizes. Ideally 10 or more signals per individual should be available in each singing event.

rm.solo

Logical. Controls if signals that are not alternated at the start or end of the sequence are removed (if TRUE). For instance, the sequence of signals A-A-A-B-A-B-A-B-B-B (in which A and B represent different individuals, as in the 'indiv' column) would be subset to A-B-A-B-A-B. Default is FALSE.

Details

This function calculates the probability of finding an equal or more extreme amount of song overlap (higher or lower) in a coordinated singing event (or any pair-coordinated behavior). The function shuffles the sequences of signals and silence-between-signals for both individuals to produce a null distribution of overlaps expected by chance. The observed overlaps is compared to this expected value. The p-values are calculated as the proportion of random expected values that were lower (or higher) than the observed value. All procedures described in Masco et al. (2015) are implemented. In addition, either the number (ovlp.method = "count") or the total duration (ovlp.method = "duration") in which signals overlap can be used for estimating the overall degree of overlap. The function runs one test for each singing event in the input data frame. This function assumes that there are no overlaps between signals belonging to the same individual. See Masco et al. (2015) for recommendations on randomization procedures for specific signal structures.

Value

A data frame with the following columns:

- sing.event: singing event ID
- obs.overlap: observed amount of overlap (counts or total duration, depending on overlap method, see 'ovlp.method' argument)
- mean.random.ovlp: mean amount of overlap expected by chance
- p.value: p value
- coor.score: coordination score (sensu Araya-Salas et al. 2017), calculated as:

\[
\frac{(\text{obs.overlap} - \text{mean.random.ovlp})}{\text{mean.random.ovlp}}
\]

Positive values indicate a tendency to overlap while negative values indicate a tendency to alternate. NA values will be returned when events cannot be randomized (e.g. too few signals).

Author(s)

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

Spectrograms with frequency measurements

Description

track_freq_contour creates spectrograms to visualize dominant and fundamental frequency measurements (contours)
track_freq_contour

Usage

track_freq_contour(X, wl = 512, wl.freq = 512, flim = NULL, wn = "hanning", pal = reverse.gray.colors.2, ovlp = 70, inner.mar = c(5, 4, 4, 2), outer.mar = c(0, 0, 0, 0), picsize = 1, res = 100, cexlab = 1, title = TRUE, propwidth = FALSE, xl = 1, osci = FALSE, gr = FALSE, sc = FALSE, bp = NULL, cex = c(0.6, 1), threshold = 15, threshold.freq = NULL, contour = "both", col = c("#E37222B3", "#07889BB3"), pch = c(21, 24), mar = 0.05, lpos = "topright", it = "jpeg", parallel = 1, path = NULL, img.suffix = NULL, custom.contour = NULL, pb = TRUE, type = "p", leglab = c("Ffreq", "Dfreq"), col.alpha = 0.6, line = TRUE, fast.spec = FALSE, ff.method = "seewave", frange.detec = FALSE, fsmooth = 0.1, widths = c(2, 1), freq.continuity = NULL, clip.edges = 2, track.harm = FALSE, ...)

Arguments

X object of class 'selection_table', 'extended_selection_table' or data frame containing columns for sound file name (sound.files), selection number (select), and start and end time of signal (start and end). The output auto_detec can also be used as the input data frame.

wl A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.

wl.freq A numeric vector of length 1 specifying the window length of the spectrogram for measurements on the frequency spectrum. Default is 512. Higher values would provide more accurate measurements.

flim A numeric vector of length 2 for the frequency limit of the spectrogram (in kHz), as in spectro. Default is NULL.

wn Character vector of length 1 specifying window name. Default is "hanning". See function ftwindow for more options.

col A color palette function to be used to assign colors in the plot, as in spectro. Default is reverse.gray.colors.2.

ovlp Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in spectro. Default is 70.

inner.mar Numeric vector with 4 elements, default is c(5,4,4,2). Specifies number of lines in inner plot margins where axis labels fall, with form c(bottom, left, top, right). See par.

outer.mar Numeric vector with 4 elements, default is c(0,0,0,0). Specifies number of lines in outer plot margins beyond axis labels, with form c(bottom, left, top, right). See par.

picsize Numeric argument of length 1. Controls relative size of spectrogram. Default is 1.

res Numeric argument of length 1. Controls image resolution. Default is 100 (faster) although 300 - 400 is recommended for publication/presentation quality.

cexlab Numeric vector of length 1 specifying the relative size of axis labels. See spectro.

title Logical argument to add a title to individual spectrograms. Default is TRUE.
propwidth Logical argument to scale the width of spectrogram proportionally to duration of the selected call. Default is FALSE.
xl Numeric vector of length 1. A constant by which to scale spectrogram width. Default is 1.
osci Logical argument to add an oscillogram underneath spectrogram, as in spectro. Default is FALSE.
gr Logical argument to add grid to spectrogram. Default is FALSE.
sc Logical argument to add amplitude scale to spectrogram, default is FALSE.
bp A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or "frange" to indicate that values in bottom.freq and top.freq columns will be used as bandpass limits. Default is NULL.
cex Numeric vector of length 2, specifies relative size of points plotted for frequency measurements and legend font/points, respectively. See spectro.
threshold amplitude threshold (%) for fundamental and dominant frequency detection as well as frequency range from the spectrum (see 'frange.detec'). Default is 15. WILL BE DEPRECATED. Use 'threshold.time' and 'threshold.freq' instead.
threshold.time amplitude threshold (%) for the time domain. Use for fundamental and dominant frequency detection. If NULL (default) then the 'threshold' value is used.
threshold.freq amplitude threshold (%) for the frequency domain. Use for frequency range detection from the spectrum (see 'frange.detec'). If NULL (default) then the 'threshold' value is used.
contour Character vector, one of "df", "ff" or "both", specifying whether the dominant or fundamental frequencies or both should be plotted. Default is "both".
col Vector of length 1 or 2 specifying colors of points plotted to mark fundamental and dominant frequency measurements respectively (if both are plotted). Default is c("#E37222B3", "#07889BB3"). Extreme values (lowest and highest) are highlighted in yellow.
pch Numeric vector of length 1 or 2 specifying plotting characters for the frequency measurements. Default is c(21, 24).
mar Numeric vector of length 1. Specifies the margins adjacent to the selections to set spectrogram limits. Default is 0.05.
lpos Character vector of length 1 or numeric vector of length 2, specifying position of legend. If the former, any keyword accepted by xy.coords can be used (see below). If the latter, the first value will be the x coordinate and the second value the y coordinate for the legend’s position. Default is "topright".
it A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted. Default is "jpeg".
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).
path Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
img.suffix A character vector of length 1 with a suffix (label) to add at the end of the names of image files. Default is NULL.
track_freq_contour

A data frame with frequency contours for exactly the same sound files and selection as in X. The frequency values are assumed to be equally spaced in between the start and end of the signal. The first 2 columns of the data frame should contain the 'sound.files' and 'selec' columns and should be identical to the corresponding columns in X (same order).

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

type A character vector of length 1 indicating the type of frequency contour plot to be drawn. Possible types are "p" for points, "l" for lines and "b" for both.

leglab A character vector of length 1 or 2 containing the label(s) of the frequency contour legend in the output image.

col.alpha A numeric vector of length 1 within [0,1] indicating how transparent the lines/points should be.

line Logical argument to add red lines (or box if bottom.freq and top.freq columns are provided) at start and end times of selection. Default is `TRUE`.

fast.spec Logical. If `TRUE` then image function is used internally to create spectrograms, which substantially increases performance (much faster), although some options become unavailable, as collevels, and sc (amplitude scale). This option is indicated for signals with high background noise levels. Palette colors `gray.1`, `gray.2`, `gray.3`, `topo.1` and `rainbow.1` (which should be imported from the package monitoR) seem to work better with 'fast' spectrograms. Palette colors `gray.1`, `gray.2`, `gray.3` offer decreasing darkness levels.

ff.method Character. Selects the method used to calculate the fundamental frequency. Either 'tuneR' (using `FF`) or 'seewave' (using `fund`). Default is 'seewave'. 'tuneR' performs faster (and seems to be more accurate) than 'seewave'.

frange.detec Logical. Controls whether frequency range of signal is automatically detected using the `freq_range_detec` function. If so, the range is used as the bandpass filter (overwriting 'bp' argument). Default is `FALSE`.

fsmooth A numeric vector of length 1 to smooth the frequency spectrum with a mean sliding window (in kHz) used for frequency range detection (when `frange.detec = TRUE`). This help to average amplitude "hills" to minimize the effect of amplitude modulation. Default is 0.1.

widths Numeric vector of length 2 to control the relative widths of the spectro (first element) and spectrum (second element, (when `frange.detec = TRUE`)).

freq.continuity Numeric vector of length 1 to control whether dominant frequency detections outliers (i.e that differ from the frequency of the detections right before and after) would be removed. Should be given in kHz. Default is `NULL`.

clip.edges Integer vector of length 1 to control if how many 'frequency-wise discontinuous' detection would be remove at the start and end of signals (see 'freq.continuity' argument). Default is 2. Ignored if `freq.continuity = NULL`.

track.harm Logical to control if `track_harmonic` or a modified version of `dfreq` is used for dominant frequency detection. Default is `FALSE` (use `dfreq`).

... Additional arguments to be passed to the internal spectrogram creating function for customizing graphical output. The function is a modified version of `spectro`, so it takes the same arguments.
**Details**

This function provides visualization of frequency measurements as the ones made by `spectro_analysis`, `freq_ts` and `freq_DTW`. Frequency measures can be made by the function or input by the user (see 'custom.contour' argument). If `frange = TRUE` the function uses `freq_range_detection` to detect the frequency range. In this case the graphical output includes a frequency spectrum showing the detection threshold. Extreme values (lowest and highest) are highlighted in yellow. Note that, unlike other `warbleR` functions that measure frequency contours, `track_freq_contour` do not interpolate frequency values.

**Value**

Spectrograms of the signals listed in the input data frame showing the location of the dominant and fundamental frequencies.

**Author(s)**

Grace Smith Vidaurre and Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

**References**


**See Also**

`spectrograms` for creating spectrograms from selections, `snr_spectrograms` for creating spectrograms to optimize noise margins used in `sig2noise`.

Other spectrogram creators: `color_spectro()`, `freq_DTW()`, `multi_DTW()`, `phylo_spectro()`, `snr_spectrograms()`, `spectrograms()`

**Examples**

```r
{
#load data
data("Cryp.soui")
writeWave(Cryp.soui, file.path(tempdir(), "Cryp.soui.wav")) #save sound files

#autodetect location of signals
ad <- auto_detection(threshold = 6, bp = c(1, 3), mindur = 1.2, flim = c(0, 5),
maxdur = 3, img = FALSE, ssmooth = 600, wl = 300, flist = "Cryp.soui.wav",
path = tempdir())

#track dominant frequency graphs with freq range detection
track_freq_contour(X = ad[!is.na(ad$start),], flim = c(0, 5), ovlp = 90,
it = "tiff", bp = c(1, 3), contour = "df", wl = 300, frange = TRUE,
path = tempdir())

#using users frequency data (custom.contour argument)
#first get contours using freq_ts
df <- freq_ts(X = ad[!is.na(ad$start),], flim = c(0, 5), ovlp = 90, img = FALSE,

```
bp = c(1, 3), wl = 300, path = tempdir())

# now input the freq_ts output into track_freq_contour
track_freq_contour(X = ad[!is.na(ad$start),], custom.contour = df, flim = c(0, 5), ovlp = 90,
                   it = "tiff", path = tempdir())

# Check this folder
tempdir()

#track both frequencies
track_freq_contour(X = ad[!is.na(ad$start),], flim = c(0, 5), ovlp = 90,
                   it = "tiff", bp = c(1, 3), contour = "both", wl = 300, path = tempdir())
}

---

**try_na**  
*Wrapper for "try" function*

**Description**

*try_na* silly wrapper for *try* function that returns an NA if an error is found. TO BE DEPRECATED IN FUTURE VERSIONS.

**Usage**

```r
try_na(expr, silent = TRUE, outFile)
```

**Arguments**

- **expr**: An R expression to try.
- **silent**: Logical to control whether the report of error messages is suppressed. Default is TRUE.
- **outFile**: A connection, or a character string naming the file to print to (via cat(*, file = outFile)); used only if silent is false, as by default.

**Details**

This is a silly wrapper on *try* that returns an ‘NA’ if any error occurs during the evaluation of a expression. See *try* for details.

**Value**

Returns an ‘NA’ if any error occurs during the evaluation of a expression. If not, it will return the result of the evaluation.

**Author(s)**

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


Examples

```
{  
  # try a function that does not exists to produce an error
  try_na(crazy78(12))

  # try a real function (no error)
  try_na(mean(1:5))
}
```

---

`tweak_spectro`  
*Plot a mosaic of spectrograms with varying display parameters*

**Description**

`tweak_spectro` plots a mosaic of spectrograms with varying display parameters to facilitate selection of display parameters.

**Usage**

`tweak_spectro(X, length.out = 5, ovlp = 90, wl = c(100, 1000), wn = "hanning", collev.min = -40, pal = "reverse.gray.colors.2", path = NULL, rm.axes = TRUE, ...)`

**Arguments**

- `X` object of class 'selection_table', 'extended_selection_table' or data frame with a single row and columns for sound file name (sound.files), selection number (selec), and start and end time of signal (start and end). Default is NULL.
- `length.out` Numeric vector of length 1 controlling the number of sublevels of the numeric arguments for which a range has been provided. Ranges are allowed for 'ovlp', 'wl', and 'collev.min' arguments.
- `ovlp` Numeric vector of length 1 or 2 specifying % of overlap (or lower/upper values the desired range) between two consecutive windows, as in `spectro`. Default is 90.
- `wl` A numeric vector of length 1 or 2 specifying the window length (length 1) or the lower and upper range limits of the desired window length range (length 2) for creating spectrograms. Default is c(100, 1000).
- `wn` Character vector specifying the window function names to be used. Several names can be provided. See `ftwindow` for name options. Default is "hanning". If "all", then all window functions available are used.
collev.min  A (negative) numeric vector of length 1 or 2. Determines the first argument to use in 'collevels' for the internal spectrogram creating function. This replaces the first element in the 'collevels' as in spectro. Note that 'collevels' is not available in this function tweak_spectro.

pal  Color palette function for spectrogram. Default is "reverse.gray.colors.2". Several palettes can be provided in a character vector. Note that, contrary to other warbleR and seeewave functions, the palette must be provided as character string rather than as a function. See spectro for more palettes.

path  Character string containing the directory path where the sound file are located.

rm.axes  Logical. If TRUE frequency and time axes are excluded. Default is TRUE.

...  Additional arguments to be passed to catalog function for customizing graphical output. Check out catalog for more details.

Details

This functions aims to simplify the selection of spectrogram parameters. The function plots, for a single selection, a mosaic of spectrograms with varying display parameters. For numeric arguments the upper and lower limits of a range can be provided. The following arguments accept can have varying values:

- **wl**: Windows length (numeric range)
- **ovlp**: Overlap (numeric range)
- **collev.min**: Minimum value of the color levels (numeric range)
- **wn**: window function names (character)
- **pal**: palette (character)

Outputs are similar to those of catalog. The output image files can be put together in a single pdf file with catalog2pdf. We recommend using low resolution (~60-100) and smaller dimensions (width & height < 10) if aiming to generate pdfs (otherwise pdfs could be pretty big).

Value

Image files with spectrograms of whole sound files in the working directory. Multiple pages can be returned, depending on the length of each sound file.

Author(s)

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

See Also

blog post on creating catalogs, blog post on customizing catalogs, catalog2pdf
Examples

## Not run:
# Save to temporary working directory

data(list = c("Phae.long1", "lbh_selec_table"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))

# variable collevels
tweak_spectro(X = lbh_selec_table, wl = 164, ovlp = c(90), wn = c("flattop"),
length.out = 16, nrow = 4, ncol = 4, width = 20, height = 11.3, rm.axes = TRUE,
cex = 1, box = F, collev.min = c(-20, -150), path = tempdir())

# variable overlap and wn
tweak_spectro(X = lbh_selec_table, wl = 164, ovlp = c(50, 90),
wn = c("hanning", "hamming", "rectangle", "bartlett", "blackman", "flattop"),
length.out = 7, nrow = 6, ncol = 7, width = 20, height = 11.3, rm.axes = TRUE,
cex = 1, box = F), path = tempdir())

# variable wl and wn
tweak_spectro(X = lbh_selec_table, wl = c(100, 1000), ovlp = c(50, 90), wn = "all",
length.out = 5, nrow = 10, ncol = 14, width = 20, height = 11.3, rm.axes = TRUE,
cex = 0.7, path = tempdir())

# variable wl, collev.min and wn
tweak_spectro(X = lbh_selec_table, wl = c(100, 1000), ovlp = 90,
wn = c("hanning", "hamming", "rectangle"), collev.min = c(-110, -25),
length.out = 3, nrow = 10, ncol = 14, width = 20, height = 11.3, rm.axes = TRUE,
cex = 0.7, path = tempdir())

# variable wl, wn and pal
tweak_spectro(X = lbh_selec_table, wl = c(100, 1000), ovlp = 90,
wn = c("hanning", "hamming", "rectangle"),
pal = c("reverse.gray.colors.2", "reverse.topo.colors", "reverse.terrain.colors", "reverse.cm.colors"),
length.out = 4, nrow = 5, ncol = 10, width = 20, height = 11.3, rm.axes = TRUE, cex = 0.7, lab.mar = 2, path = tempdir())

# wl, wn and pal
tweak_spectro(X = lbh_selec_table, wl = c(100, 1000), ovlp = 90,
wn = c("hanning", "hamming", "rectangle"),
pal = c("reverse.gray.colors.2", "reverse.topo.colors", "reverse.terrain.colors", "reverse.cm.colors"),
length.out = 4, nrow = 5, ncol = 10, width = 20, height = 11.3, rm.axes = TRUE,
cex = 0.7, group.tag = "wn", spec.mar = 0.4, lab.mar = 0.8, box = FALSE,
tag.pal = list(reverse.cm.colors), path = tempdir())

check this floder
tempdir()

## End(Not run)
warbleR

warbleR: A package to streamline bioacoustic analysis

Description

warbleR is intended to facilitate the analysis of the structure of animal acoustic signals in R. Users can collect open-access avian recordings or enter their own data into a workflow that facilitates spectrographic visualization and measurement of acoustic parameters. warbleR makes use of the fundamental sound analysis tools of the seewave package, and offers new tools for acoustic structure analysis. These tools are available for batch analysis of acoustic signals.

Details

The main features of the package are:

- The use of loops to apply tasks through acoustic signals referenced in a selection table
- The production of images in the working folder with spectrograms that allow to organize data and verify acoustic analyzes

The package offers functions to:

- Explore and download Xeno Canto recordings
- Explore, organize and manipulate multiple sound files
- Detect signals automatically (in frequency and time)
- Create spectrograms of complete recordings or individual signals
- Run different measures of acoustic signal structure
- Evaluate the performance of measurement methods
- Catalog signals
- Characterize different structural levels in acoustic signals
- Statistical analysis of duet coordination
- Consolidate databases and annotation tables

Most of the functions allow the parallelization of tasks, which distributes the tasks among several processors to improve computational efficiency. Tools to evaluate the performance of the analysis at each step are also available. In addition, warbleR satisfies the need for rigorous open source bioacoustic analysis, which facilitates opportunities for use in research and innovation of additional custom analyzes.

The warbleR package offers three overarching categories of functions:

License: GPL (>= 2)

Obtaining animal vocalization data

query_xc: Download recordings and/or metadata from 'Xeno-Canto'
find_annotations: Obtain annotations from audioblast.org data base
sim_songs: Simulate animal vocalizations
Managing sound files

- **read_wave**: Read wave files into ‘wave’ objects
- **read_sound_file**: Read sound files into ‘wave’ objects
- **selection_table**: Create ‘selection_table’ class objects
- **mp32wav**: Convert several .mp3 files in working directory to .wav format
- **check_sels**: Check whether selections can be read by subsequent functions
- **check_wavs**: Check whether .wav files can be read by subsequent functions and the minimum windows length ("wl" argument) that can be used
- **fix_wavs**: Fix .wav files so they can be read by other functions
- **split_wavs**: Split .wav files in several sound files
- **resample_est_waves**: Resample wave objects in extended selection tables
- **duration_wavs**: Determine the duration of sound files
- **cut_sels**: Cut selections from a selection table into individual sound files
- **remove_silence**: Remove silence segments from wave files
- **remove_channels**: Remove channels in wave files
- **consolidate**: Consolidate sound files into a single folder
- **selection_table**: Create double-checked and self-contained selection tables
- **fix_extended_selection_table**: Fix attributes of extended selection tables

Exploring/analyzing signal structure

- **auto_detec**: Automatically detect start and end of acoustic signals
- **tailor_sels**: Interactive view of spectrograms to tailor start and end of selections
- **sig2noise**: Measure signal-to-noise ratio across multiple files
- **track_freq_contour**: Create spectrograms to visualize frequency measurements
- **filter_sels**: Filter selection data frames based on filtered image files
- **freq_range**: Detect frequency range iteratively from signals in a selection table
- **freq_range_detec**: Detect frequency range in a Wave object
- **spectro_analysis**: Measure acoustic parameters on selected acoustic signals
- **mfcc_stats**: Calculate descriptive statistics on Mel-frequency cepstral coefficients
- **song_analysis**: Measure acoustic parameters at higher levels of organization
- **cross_correlation**: Pairwise cross-correlation of multiple signals
- **gaps**: Measures gap duration
- **freq_ts**: Extract frequency contours the signal as a time series
- **find_peaks**: Find peaks in cross-correlation scores from **cross_correlation**
- **freq_DTW**: Calculate acoustic dissimilarity using dynamic time warping on frequency contours
- **wpd_features**: Measure wavelet packet decomposition features
**warbleR_options**

**compare_methods:** Produce graphs to visually assess performance of acoustic distance measurements

**test_coordination:** Assess statistical significance of singing coordination

**overlapping_sels:** Find selections that overlap in time within a given sound file

**track_harmonic:** Track harmonic frequency contour

**Graphical outputs**

**map_xc:** Create maps to visualize the geographic spread of 'Xeno-Canto' recordings

**catalog:** Produce a vocalization catalog with spectrograms in and array with several rows and columns

**catalog2pdf:** Combine catalog images to single pdf files

**plot_coordination:** Create graphs of coordinated singing

**color_spectro:** Highlight spectrogram regions

**full_spectrograms:** Produce spectrograms of whole recordings split into multiple rows

**full_spectrogram2pdf:** Combine lspec images to single pdf files

**spectrograms:** Create spectrograms of selections

**snr_spectrograms:** Create spectrograms to visualize margins over which noise will be measured by sig2noise

**phylo_spectro:** Add spectrograms onto phylogenetic trees

**tweak_spectro:** Visually inspect effect of different settings for creating (and improving) spectrograms

**Author(s)**

Marcelo Araya-Salas & Grace Smith Vidaurre

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

---

**Description**

`warbleR_options` sets global parameters for warbleR functions

**Usage**

`warbleR_options(reset = FALSE, ...)`

**Arguments**

- `reset` Logical. If `TRUE` then all global parameters are removed. Default is `FALSE`.
- `...` Arguments in "parameter = value" form, or a list of tagged values. The tags (i.e. parameters) must come from the list of parameters described below.
Details

The function aims to simplify the use of parameters that apply to many warbleR functions (i.e. global parameters) by setting a default value that will be used to any function in downstream analyses. Tags that are set with warbleR_options will be used by the functions that share those arguments. However, if an argument is set within a function call it will overwrite the values set by warbleR_options. Hence, the functions remain 'flexible' as their parameters can also be modified 'on the fly'. The following tags are available:

- **bp**: Numeric vector of length 2 giving the lower and upper limits of a frequency bandpass filter (in kHz).
- **collevels**: A numeric vector of length 3. Specifies levels to partition the amplitude range of the spectrogram (in dB) as in spectro. The more levels the higher the resolution of the spectrogram. The lower the first value the darker the spectrograms.
- **flim**: A numeric vector of length 2 for the frequency limit in kHz of the spectrogram, as in spectro.
- **it**: A character vector of length 1 giving the image type to be used. Currently only "tiff" and "jpeg" are admitted.
- **osci**: Logical argument to add an oscillogram underneath spectrogram, as in spectro.
- **pal**: A color palette function to be used to assign colors in the plot, as in spectro.
- **parallel**: Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used in iterative functions.
- **pb**: Logical argument to control whether progress bar is used.
- **res**: Numeric argument of length 1. Controls image resolution in all image creating functions.
- **wav.path**: Character string containing the directory path where the sound files are located. Used as ‘path’ in all functions in which sound files are read.
- **wl**: A numeric vector of length 1 specifying the window length for creating spectrogram (either for plotting or for measuring spectrogram parameters).
- **wn**: Character vector of length 1 specifying the window name for creating spectrogram (either for plotting or for measuring spectrogram parameters). See function ftwindow for options.

Value

When parameters are set by warbleR_options, their former values are returned in an invisible named list. Such a list can be passed as an argument to pboptions to restore the parameter values. If the function is called with no arguments the current option values are printed.

Author(s)

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

Examples

```
# load data and save in temporary working directory
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "Ibh_selec_table"))
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"))

# sig2noise with progress bar (by default is TRUE)
a <- sig2noise(X = lbh_selec_table, mar = 0.1, path = tempdir())

# set progress bar to FALSE with warbleR_options
warbleR_options(pb = FALSE, path = tempdir())

# sig2noise without progress bar
a <- sig2noise(X = lbh_selec_table, mar = 0.1)

# sig2noise with progress bar by setting it within the function call (overwritting options)
a <- sig2noise(X = lbh_selec_table, pb = TRUE, mar = 0.1)

# sig2noise without progress bar using warbleR_options setting again
a <- sig2noise(X = lbh_selec_table, mar = 0.1)
}

---

**wav_2_flac**

Convert .wav files to .flac

**Description**

wav_2_flac converts several .wav files to .flac compressed lossless format

**Usage**

```r
wav_2_flac(files = NULL, path = NULL, overwrite = FALSE, pb = TRUE, parallel = 1)
```

**Arguments**

- **files**: character vector with the names of files to be converted. If NULL all files in the working directory (or 'path' if supplied) are converted.
- **path**: Character string containing the directory path where the .wav files are located. If NULL (default) then the current working directory is used.
- **overwrite**: Logical. Control whether a .flac sound file that is already in the working directory should be overwritten.
- **pb**: Logical argument to control if progress bar is shown. Default is TRUE. It can also be set globally using the `pb` option (see warbleR_options).
- **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). It can also be set globally using the 'parallel' option (see warbleR_options).
wpd_features

Measure wavelet packet decomposition features (EXPERIMENTAL)

Description

wpd_features Measure wavelet packet decomposition features.

Usage

wpd_features(X, normalize = TRUE, threshold1 = 6, threshold2 = 0.5, path = NULL, pb = TRUE, parallel = 1)
Arguments

- X: object of class 'selection_table', 'extended_selection_table' or data frame with the following columns: 1) "sound.files": name of the sound files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. The output of auto_detect can also be used as the input data frame.
- normalize: Logical to determine if features are normalized by signal duration.
- threshold1: Threshold (%) for wavelet coefficient detection. Equivalent to denominator of equation 6 in Selin et al (2007). Must be a value between 0 and 1.
- path: Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
- pb: Logical argument to control progress bar and messages. Default is TRUE.
- 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).

Details

Measures wavelet packet decomposition features. STILL IN DEVELOPMENT. USE IT UNDER YOUR OWN RISK.

Author(s)

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

References


See Also

mfcc_stats, mfcc_stats

Examples

```r
{  
data(list = c("Phae.long1", "Phae.long2", "lbh_selec_table"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))

  # not normalize
wpd_features(lbh_selec_table[1:5, ], threshold2 = 0.3, nor = FALSE)
}
```
Index

* data manipulation
  move_imgs, 65
  open_wd, 71
  split_sound_files, 114
* datasets
  comp_matrix, 21
  lbh_selec_table, 60
  lbh_selec_table2, 61
  new_function_names, 70
  selec.table, 91
  sim_coor_sing, 99
* extended selection table manipulation
  rename_est_waves, 89
* sound file manipulation
  consolidate, 22
* spectrogram creators
  color_spectro, 14
  freq_DTW, 37
  multi_DTW, 68
  phylo_spectro, 76
  snr_spectrograms, 100
  spectrograms, 108
  track_freq_contour, 121
acoustat, 112, 113
analyze, 112, 113
approx, 4, 39, 47, 69, 73
auto_detec, 3, 17, 34, 38, 40, 43, 50, 52, 63, 72, 73, 88, 96, 111, 115, 122, 131, 136
autodetec, 45
catalog, 6, 8, 11, 12, 93, 128, 132
catalog2pdf, 9, 10, 11, 19, 52, 128, 132
check_sels, 13, 14, 36, 37, 57–59, 92, 93, 131
check_sound_files, 13
check_wavs, 37, 94, 131
cmdscale, 19
color_spectro, 14, 40, 69, 78, 102, 110, 125, 132
cmp_matrix, 21
compare_methods, 17, 132
consolidate, 22, 131
cor, 24
cross_correlation, 18, 19, 21, 24, 33, 34, 50, 52, 93, 131
cut_sels, 115, 131
dfreq, 46, 111, 112, 124
dtw, 19, 39, 69
dtwDist, 37, 68
duration_sound_files, 27
duration_wavs, 131
env, 28
evelope, 28
FF, 47, 111, 124
ffilter, 4, 72
file.copy, 22
filter_sels, 29, 131
filtersels, 66, 76, 117
find_annotations, 31, 31, 130
find_clipping, 32
find_peaks, 25, 33, 61, 131
findPeaks, 34
fix_extended_selection_table, 35, 131
fix_wavs, 13, 23, 36, 57, 65, 67, 87, 88, 93, 131
fpeaks, 112
frange, 45
freq_DTW, 16, 18, 26, 37, 47, 69, 78, 93, 102, 110, 125, 131
freq_range, 40, 131
freq_range_detec, 42, 43, 43, 124, 125, 131
freq_ts, 19, 40, 45, 47, 56, 69, 125, 131
ftwindow, 7, 14, 24, 38, 41, 44, 46, 100, 108, 112, 116, 122, 127, 133
full_spectrogram2pdf, 12, 48, 52, 132
full_spectrograms, 4, 29, 34, 48, 49, 49, 132
warbleR, 65, 66, 70, 71, 92, 130
warbleR_options, 22, 29, 71, 95, 106, 107, 132, 134
wav2flac, 135
wav_2_flac, 134
weighted.mean, 103
wpd_features, 131, 135
writeWave, 67, 135